ADCS 2009

Proceedings of the Fourteenth Australasian Document Computing Symposium

4 December 2009

Edited by
Judy Kay, Paul Thomas, and Andrew Trotman

Technical report TR 645
School of Information Technologies, University of Sydney

Proceedings of the Fourteenth Australasian Document Computing Symposium

University of New South Wales, Sydney, NSW
4 December 2009

Published by
School of Information Technologies, University of Sydney

Editors

Judy Kay
Paul Thomas
Andrew Trotman

ISBN: 978-1-74210-171-2
http://es.csiro.au/adcs2009

Proceedings of the Fourteenth Australasian Document Computing Symposium

University of New South Wales, Sydney, NSW
4 December 2009

Chairs' preface

These proceedings contain the papers of the Fourteenth Australasian Document Computing Symposium hosted by HCSNet at the University of New South Wales, Sydney.

The varied long and short papers, as well as David Traum's and Mark Sanderson's plenaries, are indicative of the wide breadth of research in the Australasian document computing community and the wide scope for application.

The quality of submissions was once again high this year. Of the 32 papers submitted (26 full and 6 short), 10 were accepted for presentation at the symposium (28%) and 11 were accepted as posters (31%). All submissions received at least two anonymous reviews by experts in the area, and several received three reviews. Dual submissions were explicitly prohibited.

The members of the program committee and the extra reviewers deserve special thanks for their effort, especially given the very tight turnaround needed for this year's symposium. We would also like to thank HCSNet for its support of ADCS, which freed us from worrying about most of the logisitics.

The ADCS community has contributed many good papers this year, but as before the symposium's greatest benefit may be the opportunity it provides for researchers and practitioners to meet and share ideas. We hope you enjoy it.

Symposium chair

Judy Kay	University of Sydney	Australia

Programme co-chairs

Andrew Trotman	University of Otago	New Zealand
Paul Thomas	CSIRO	Australia

Programme committee

Alexander Krumpholz	CSIRO/ANU	Australia
Alistair Moffat	University of Melbourne	Australia
Andrew Turpin	RMIT University	Australia
Christopher Lueg	University of Tasmania	Australia
David Hawking	Funnelback	Australia
Falk Scholer	RMIT University	Australia
Gitesh Raikundalia	Victoria University	Australia
James Thom	RMIT University	Australia
Judy Kay	University of Sydney	Australia
Nathan Rountree	University of Otago	New Zealand
Peter Bruza	QUT	Australia
Ross Wilkinson	Australian National Data Service,	Australia
Sally Jo Cunningham	University of Waikato	New Zealand
Shlomo Geva	QUT	Australia
Timothy Jones	ANU	Australia
Tom Rowlands	CSIRO/ANU	Australia
Vo Anh	University of Melbourne	Australia
William Webber	University of Melbourne	Australia
Yun Sing Koh	AUT	New Zealand

Additional reviewers

Cécile Paris	CSIRO	Australia
Richard O'Keefe	University of Otago	New Zealand
Stephen Wan	CSIRO	Australia

ADCS steering committee

Alistair Moffat	University of Melbourne	Australia
Andrew Trotman	University of Otago	New Zealand
Andrew Turpin	RMIT University	Australia
David Hawking	Funnelback	Australia
James Thom	RMIT University	Australia
Judy Kay	University of Sydney	Australia
Justin Zobel	University of Melbourne	Australia
Paul Thomas	CSIRO	Australia
Peter Bruza	QUT	Australia
Ross Wilkinson	Australian National Data Service,	Australia
Shlomo Geva	QUT	Australia

Contents

Is this document relevant? Errr it'll do

Mark Sanderson
University of Sheffield
m.sanderson@shef.ac.uk

Abstract *Evaluation of search engines is a critical topic in the field of information retrieval. Doing evaluation well allows researchers to quickly and efficiently understand if their new algorithms are a valuable contribution or if they need to go back to the drawing board. The modern methods used for evaluation developed by organizations such as TREC in the US have their origins in research that started in the early 1950s. Almost all of the core components of modern testing environments, known as test collections, were present in that early work. Potential problems with the design of these collections were described in a series of publications in the 1960s, but the criticisms were largely ignored. However, in the past decade a series of results were published showing potentially catastrophic problems with a test collection's "ability" to predict the way that users will work with searching systems. A number of research teams showed that users given a good system (as measured on a test collection) searched no more effectively than users given one that was bad.*

In this talk, I will briefly outline the history of search evaluation, before detailing the work finding problems with test collections. I will then describe some pioneering but relatively overlooked research that pointed out that the key problem for researchers isn't the question of how to measure searching systems accurately, the problem is how to accurately measure people.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

Collaborative Filtering Recommender Systems based on Popular Tags

Huizhi Liang *Yue Xu* *Yuefeng Li* *Richi Nayak*

School of Information Technology
Queensland University of Technology
Queensland, QLD 4001, Australia

oklianghuizi@gmail.com *yue.xu@qut.edu.au* *y2.li@qut.edu.au* *r.nayak@qut.edu.au*

Abstract *The social tags in web 2.0 are becoming another important information source to profile users' interests and preferences for making personalized recommendations. However, the uncontrolled vocabulary causes a lot of problems to profile users accurately, such as ambiguity, synonyms, misspelling, low information sharing etc. To solve these problems, this paper proposes to use popular tags to represent the actual topics of tags, the content of items, and also the topic interests of users. A novel user profiling approach is proposed in this paper that first identifies popular tags, then represents users' original tags using the popular tags, finally generates users' topic interests based on the popular tags. A collaborative filtering based recommender system has been developed that builds the user profile using the proposed approach. The user profile generated using the proposed approach can represent user interests more accurately and the information sharing among users in the profile is also increased. Consequently the neighborhood of a user, which plays a crucial role in collaborative filtering based recommenders, can be much more accurately determined. The experimental results based on real world data obtained from Amazon.com show that the proposed approach outperforms other approaches.*

Keywords Information Retrieval, recommender systems, social tags, web 2.0

1 Introduction

Collaborative tagging is a new means to organize and share information resources or items on the web such as web pages, books, music tracks, people and academic papers etc. Due to the simplicity, effectiveness and being independent of the contents of items, social tags have been used in various web applications including social web page bookmarking site del.icio.us, academic paper sharing website CiteULike, and electronic commerce website Amazon.com.

A social tag is a piece of brief textural information given by users explicitly and proactively to describe and group items, thus it implies user's interests or preferences information. Therefore, the social tag information can be used to profile user's interested and preferred topics to improve personalized searching [1], generate user and item clusters [2], and make personalized recommendations [3] etc. However, as the tag terms are chosen by users freely (i.e., uncontrolled vocabularies), social tags suffer from many difficulties such as ambiguity in the meaning of and differences between terms, a proliferation of synonyms, varying levels of specificity, meaningless symbols, and lack of guidance on syntax and slight variations of spelling and phrasing [4]. These problems cause inaccurate user profiling and low information sharing among users, and also bring challenges to generate proper neighborhood for making item recommendations and consequently result in low recommendation performances. Therefore, a crucial problem in applying user tagging information to user profiling is to represent the semantic meanings of the tags.

Popular tags refer to the tags that are used by many users to collect items. Those popular tags are factual tags [5] that often capture the tagged items' content related information or topics while those tags that have low popularity are often irrelevant to the content of the tagged items or meaningless to other users, or even misspelled [5]. For one item, the popularity of using a tag to classify the item reflects the degree of common understanding to the tag and the item. High popularity means that the majority of the users think this item can be described by the tag. Thus, the popular tags reflect the common viewpoint of users or the "wisdom of crowds" [6] in the classification or descriptions of this item. Therefore, we argue that the popular tags can be used to describe the topics of the tagged items. For each user, the original tags and the collected items represent the user's personal viewpoint of item classifications and collections. In a tag, a set of items are grouped together according to the user's viewpoint. The actual topics of the tag can be described by the frequent topics of the collected items.

Proceedings of the 14th Australasian Document Computing
Symposium, Sydney, Australia, 4 December 2009.

As we just mentioned above, the major topics of each item can be represented by its popular tags, thus the popular tags of the collected items in a tag can be used to represent that tag's actual topics. Since the user's personal viewpoint of the classifications of the collected items are still kept while the original tag terms are converted to popular tags that shared by many users, the user information sharing will be improved.

In this paper, we propose to use popular tags to represent the topics of items, tags, and users' interests to solve the problems of inaccurate user profiling and low information sharing caused by the free-style vocabularies of social tags. In Section 2, the related work will be briefly reviewed. Then, the proposed collaborative filtering recommendation approach based on popular social tags will be discussed in details in Section 3. In this section, the definitions and the selection of popular social tags will be discussed firstly. Then, the approaches of representing items and tags with popular social tags will be presented. Followed by the user profiling, neighborhood formation, and recommendation generation approaches, the experimental results and evaluations will be discussed in Section 4. Finally, the conclusions will be given in Section 5.

2 Related Work

Recommender systems have been an active research area for more than a decade, and many different techniques and systems with distinct strength have been developed. Recommender systems can be broadly classified into three categories: content-based recommender systems, collaborative filtering or social filtering based recommender systems and hybrid recommender systems [7]. Because of the advantages of using similar users' recommendation and independent with the contents of items, the collaborative filtering based recommender systems have been widely used. Typically, users' explicit numeric ratings towards items are used to represent users' interests and preferences to find similar users or similar content items to make recommendations. However, because users' explicit rating information is not always available, the recommendation techniques based on user's implicit ratings have drawn more and more attention recently.

Besides the web log analysis of users' usage information such as click stream, browse history and purchase record etc., users' textural information such as tags, blogs, reviews in web 2.0 becomes an important implicit rating information source to profile users' interests and preferences to make recommendations [10]. Currently, the researches about tags in recommender systems are mainly focused on how to recommend tags to users such as using the co-occurrence of tags [2] and association rules [10] etc. Not so much work has been done on the item recommendation. Although there are some recent work which discusses about integrating tag information with content based recommender systems [11], extending the user-item matrix to user-item-tag matrix to make collaborative filtering item recommendation [12], combining uses' explicit rating with the predicted users' preferences for items based on their inferred preferences for tags [16] etc, more advanced approaches of how to exploit tags to improve the performances of item recommendations are still in demand.

More recently, the semantic meaning of social tags has become one important research question. The research of Sen etc. [5] suggests that the factual tags are more likely to be reused by different users. The work of Suchanek etc. [15] shows that popular tags are more semantically meaningful than unpopular tags. And, the research of Bischoff etc. [4] shows that not all tags are useful for searching and those tags related to the content information of items are more useful. These findings support this research. To solve the difficulties caused by the uncontrolled vocabularies of social tags, some approaches have been discussed to get the actual semantics of tags such as combining the content keywords with tags [10], using dictionaries to annotate tags [6], and contextualizing tags [17] etc. Different from these approaches, this paper proposes to use popular tags generated from the collected items to represent the semantic meanings of tags.

3 The Proposed Approach

3.1 Definitions

To describe the proposed approach, we define some key concepts and entities used in this paper as below. In this paper, tags and social tags are interchangeably used.

- **Users:** $U = \{u_1, u_2, \dots, u_n\}$ contains all users in an online community who have used tags to organize items.
- **Items or (Products, Resources):** $P = \{p_1, p_2, \dots, p_m\}$ contains all items tagged by users in U. Items could be any type of online information resources or products in an online community such as web pages, videos, music tracks, photos, academic papers, books etc. Each item p can be described by a set of tags contributed by different users.
- **Topics:** contain items' content related information such as content topics, genres, locations, attributes. For example, "globalization" is a topic that describes items' content information, "comedy" is a topic that describes items' genre information, and "Shakespeare" is a topic that describes the attribute of author information.
- **Social Tags:** $T = \{t_1, t_2, \dots, t_l\}$ contains all tags used by the users in U.
- **Popular social tags:** $C = \{c_1, c_2, \dots, c_q\}$ contains a set of popular social tags. Popular social tags are

tags that are used by at least θ users, where θ is a threshold. The selection of popular social tags is discussed in the followed Section.

3.2 The Selection of Popular Social Tags

Through tagging, the users, items and tags form a three dimensional relationship [12]. Based on tags, items are aggregated together if they are collected under the same tag by different users and also users are grouped together if they have used the same tag. Usually, the global popularity of a tag can be measured by the number of users that have used this tag.

Let $u(t_i)$ be the set of users who have used the tag $t_i \epsilon T$, $u(t_i, p_j)$ be the set of users who have used t_i for the item $p_j \epsilon P$, $u(t_i) = \{u(t_i, p_j) | p_j \epsilon P(t_i)\}$, where $P(t_i)$ is the set of items collected under tag t_i and $P(t_i) \subseteq P$. The global popularity of t_i can be measured by $|u(t_i)|$ which is the number of users that have used tag t_i, and the local popularity of t_i for the item p_j can be measured by $|u(t_i, p_j)|$. If we choose popular tags only based on the global popularity, some important tags that have high local popularities but relatively low global popularities (i.e., the tags that only have one kind of meaning and are used by a small number of users for tagging some particular items) will be missed out. Moreover, because a tag can have multiple meanings and users may have different understandings to the tags, some tags will have high global popularities but low local popularities such as subjective tags (i.e., "funny"). But because of the high global popularity, those tags will be incorrectly selected.

To select those popular tags that can well represent the item topics, we define the global popularity of a tag based on its maximum local popularity. Let $O(t_i)$ be the global popularity of the tag t_i, $O(t_i) = \max_{p_j \epsilon P(t_i)}\{|u(t_i, p_j)|\}$. Thus, let θ be a threshold, any tag t_i with $O(t_i) > \theta$ will be selected as a popular social tag.

Theoretically, the threshold θ can be any positive numbers. However, since $O(t_i)$ is the maximum local popularity of t_i for its collected items, if θ is too large, the number of popular tags will be small, and there might be some items which are not tagged by any of those selected popular tags. On the other hand, each item collects a set of tags that have been used by different users to tag this item. Let $T(p_j)$ be the collected tag set of p_j, $\max_{t_i \epsilon T(p_j)}\{|u(t_i, p_j)|\}$ is the maximum local popularity of the tags in $T(p_j)$ for item p_j. Apparently, if $\theta > \max_{t_i \epsilon T(p_j)}\{|u(t_i, p_j)|\}$, then all the tags of item p_j will be excluded which will result in no popular tags to describe the topics of p_j. To avoid this situation, we define an upper boundary for the threshold θ. Let $\lambda = min_{p_j \epsilon P}\{\max_{t_i \epsilon T(p_j)}\{|u(t_i, p_j)|\}\}$. If $\theta \leq \lambda$, then each item can be guaranteed to have at least one popular tag to describe it. Therefore, the popular social tag set C also can be denoted as:

$$C = \{t_i | O(t_i) \geq \theta, t_i \in T, \lambda \geq \theta > 0\}, C \subseteq T.$$

3.3 Item and Tag Representations

The selected popular tags are used to represent items' major topics and the actual topics of each user's tags.

Item Representation

Traditionally, the item classifications or descriptions are given by experts using a set of standard and controlled vocabulary as well as a hierarchical structure representing the semantic relationships among the topics to describe the topics of the items such as item taxonomy and ontology. In web 2.0, harnessing the collaborative work of thousands or millions of web users, the aggregated tags contributed by different users form the item classifications or descriptions from the viewpoint of users or folksonomy [13]. For each item p_j, the set of tags used by users to tag p_j, denoted as $T(p_j)$, and the number of users for each tag in $T(p_j)$ form the item description of item p_j, which is defined as below.

Definition 1 (Item Description): Let p_j be an item, the item description of p_j is defined as the set of social tags for p_j and their numbers of being used to tag the item p_j, which is denoted as $D(p_j) = \{(t_i, O(t_i, p_j)) | t_i \in T(p_j), O(t_i, p_j) > 0\}$, where $O(t_i, p_j)$ is the number of users that use the tag t_i to tag the item p_j and $O(t_i, p_j) = |u(t_i, p_j)|$.

An example of item description is shown in Figure 1. The book "*The World is Flat"* is described by 10 tags such as "globalization", "economics", "business" etc. and their user numbers.

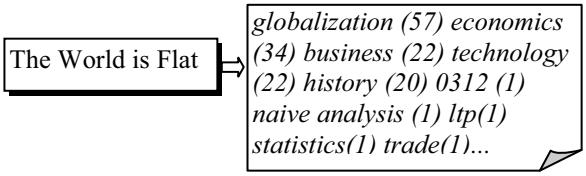


Figure 1: An example of item description formed by social tags.

Different from the item descriptions or classifications provided by experts, the item descriptions formed by social tags contain a lot of noise, which brings challenges for the organizing, sharing and retrieval of items. However, an advantage provided by the item descriptions formed by social tags is that the item description $D(p_j)$ records the user number of each tag for p_j or the local popularity of each tag for p_j. This feature can be used to find the major topics of items and filter out the noise. For example, in Figure 1, we can see that 57 users use the tag "globalization" to classify the book "The World is Flat", which is the most frequently used tag to tag this book, and the term "globalization" is indeed the actual

major topic of this book. Moreover, the tag "0312" only has one user, and it doesn't reveal any information in terms of the topics of the book. Removing the unpopular tags such as "0312" won't reduce the coverage of the remaining tags to represent the topics of the book but the noise. Therefore, we propose to use the selected popular tags to represent the items.

Definition 2 (Item Representation) Let p_j be an item, $C = \{c_1, c_2, \dots, c_q\}$ be the set of popular tags, the representation of p_j is defined as a set of popular social tags along with their frequencies as described below:
$IR(p_j) = \{(c_x, f(p_j, c_x)) | c_x \in C, f(p_j, c_x) > 0\}$,
$f(p_j, c_x) = O(c_x, p_j) / \sum_{c_y \in C} O(c_y, p_j)$, where $f(p_j, c_x)$ is the frequency of c_x for p_j, $f(p_j, c_x) \in [0,1]$ and $\sum_{c_x \in C} f(p_j, c_x) = 1$.

The frequency $f(p_j, c_x)$ represents the degree of item p_j belonging to c_x. For a given set of popular tags C with size q, i.e.,$|C| = q$,the topics of each item $p_j \in P$ can be represented by a vector $\overrightarrow{b_j} = (b_{j,1}, b_{j,2}, \dots, b_{j,x}, \dots, b_{j,|C|})$, where $b_{j,x} = f(p_j, c_x)$. Thus, for each item p_j, its topic representation becomes:
$\overrightarrow{b_j} = (b_{j,1}, b_{j,2}, \dots, b_{j,x}, \dots, b_{j,|C|})$

Tag Representation

As mentioned in Introduction, since the unrestricted nature of tagging, social tags contain a lot of noise and suffer some problems such as semantic ambiguity and a lot of synonyms etc., which brings challenges to make use of social tags to profile users' interests accurately.

Although not all tags are meaningful to other users or can be used to represent the topics, for each user, his/her own tags and items collected with those tags reflect that user's personal viewpoint of classification of the collected items. Thus, each tag used by a user is useful for profiling that user no matter how popular this tag is. In a tag, a set of items are grouped together according to a user's viewpoint, therefore, the frequent topics of these items can be used to represent the actual topics of the tag. Since the major topics of each item can be represented by its popular tags, the frequent popular tags of the collected items in a tag can be used to represent that tag's actual covered or related topics.

Definition 3 (Tag Representation): Let t be a tag used by user u, $C = \{c_1, c_2, \dots, c_q\}$ be the set of popular tags, the representation of t is defined as a set of weighted popular social tags as described below:
$TR(t,u) = \{(c_x, w(c_x,t,u)) | c_x \in C, w(c_x,t,u)) > 0\}$, where $w(c_x,t,u)$ is the weight of c_x, $w(c_x,t,u) \in [0,1]$, $\sum_{c_x \in C} w(c_x,t,u) = 1$.

The weight of c_x or $w(c_x,t,u)$ can be measured through calculating the total frequency of c_x for all the items collected in the tag t by the user u. Since the number of items in different tags may be different, we normalize $w(c_x,t,u)$ with the number of items in the tag t of u. Let $P(t,u)$ denote the set of items that are collected or classified to the tag t by user u, then the weight of c_x can be calculated as below:
$w(c_x,t,u) = \frac{1}{|P(t,u)|} \sum_{p_j \in P(t,u)} f(p_j, c_x)$, where $f(p_j, c_x)$ is the frequency of c_x for the item p_j in the tag t, as shown in Definition 2, $f(p_j, c_x) = O(c_x, p_j) / \sum_{c_y \in C} O(c_y, p_j)$.
Apparently, the tag representation $TR(t,u)$ *is* generated based on the items collected in the tag t by the user u. That means, $TR(t,u)$ still reflects the personal viewpoint of the user u about the item classifications or collections. Thus, each user's viewpoint of classifying his/her items is still kept while a set of popular tags are obtained to represent each tag term's semantic meaning. For different users, the representations for the same tag can be different. On the other hand, for different users, the representations for different tags can be the same or similar. Even though the tag terms are freely chosen by individual users, by representing each tag using a set of popular tags, all tags become comparable since all of them are represented using the same set of terms (i.e., popular tags). With the popular tag representation, those unpopular tags that often cause confusions and noises become understandable by other users according to the understanding to their corresponding popular tag representation. For those popular tags, their tag representations reveal other related popular tags, very often, these popular tags themselves have high weight in their tag representation. Since each tag is represented by a set of popular tags which provides the ground for comparison, this approach can help to solve the problems caused by the free style vocabulary of tags such as tag synonyms which means some different tags have the same meaning, semantic ambiguity of tags which means one tag has different meanings for different users, and spelling variations etc.

3.3 User Profile Generation

User profile is used to describe user's interests and preferences information. Usually, a user-item rating matrix is used in collaborative filtering based recommender systems to profile users' interests, which are used to find similar users through calculating the similarity of item ratings or the overlaps of item sets [14]. With the tag information, users can be described with the matrix (user, (tag, item)), where (tag, item) is a sub matrix representing the relationship between the tag set and item set of each user. Binary values "1" and "0" are used to specify whether a tag or an item has been used or tagged by a user or not. Through calculating the overlaps of tags and items or each user's sub relationship of tags and items, neighborhood can be

formed to do collaborative filtering to recommend items to a target user [12][3].

As mentioned before, the free-style vocabulary of tags causes a lot of noise in tags which resulted in inaccurate user profiles and incorrect neighbors. Moreover, because of the long tails of items and tags, the size of the matrix is very big and the overlaps of commonly used tags and tagged items are very low, which makes it difficult to find similar users through calculating the overlaps of tags and items. To solve these problems, we propose to profile users' interests to topics by using a set of popular tags and convert the binary matrix (user, (tag, item)) into a much smaller sized user-topics matrix. The popular tags will be used to represent each user's interested topics and numeric scores will be used to represent how much the user are interested in these topics.

Definition 4 (User Profile): Let u_i be a user, $C = \{c_1, c_2, \dots, c_q\}$ be the set of popular tags, the user profile of u_i is defined as a $|C|$-sized vector with scores reflecting user's interests to the popular tags, which is donated as $\overrightarrow{v_i} = (v_{i,1}, v_{i,2}, \dots, v_{i,x}, \dots, v_{i,|C|}) = (sc(u_i, c_1), sc(u_i, c_2), \dots, sc(u_i, c_x), \dots, sc(u_i, c_q))$. $sc(u_i, c_x)$ is the score to $v_{i,x}$ that represents the degree of u_i's interests to the popular tag c_x.

A matrix $\vec{v}$ with size $|U| \times |C|$, can be used to represent the user profiles for all users in U. Each row $\overrightarrow{v_i}$ in the matrix $\vec{v}$ represents the user profile of user u_i. In order to facilitate the similarity measure of any two users, user-wise normalization is applied. We suppose each $u_i \epsilon U$ has the same total interest score N and $\sum_{c_x \in C} sc(u_i, c_x) = N$, where N is the normalization factor, which can be any positive number. Thus, $sc(u_i, c_x) \in [0, N]$.

To calculate each user's topic interest degree $sc(u_j, c_x)$, firstly, we calculate the user's interest distribution for his/her own original tags. Let $T_i = \{t_{i,1}, t_{i,k}, \dots, t_{i,a}\}$ be the tag set of u_i, $t_{i,1}, t_{i,k}, \dots, t_{i,a} \epsilon T$, $s(t_{i,k})$ be the score to measure how much u_i is interested in $t_{i,k}$, then the score vector $(s(t_{i,1}), s(t_{i,k}), \dots, s(t_{i,a}))$ will represent u_i's interest distribution over his/her own tags, $\sum_{k=1}^{a} s(t_{i,k}) = N$.

A common sense is that, if a user is more interested in a tag or topic, usually the user may collect more items under that tag or about that topic. That means, the number of items in a tag is an important indicator about how much the user is interested in the tag. Let $|P(t_{i,k}, u_i)|$ denote the number of items in the tag $t_{i,k}$ used by user u_i, we use the proportion of $|P(t_{i,k}, u_i)|$ to the total number of items in all tags of u_i to measure the user's interest degree to the tag $t_{i,k}$. Thus, $sc(t_{i,k})$ can be calculated as shown as follows:

$$s(t_{i,k}) = N \cdot \frac{|P(t_{i,k}, u_i)|}{\sum_{k=1}^{a} |P(t_{i,k}, u_i)|} \quad (1)$$

By using Equation 1, we can obtain the user-tag matrix that describes tag interests of all the users. As discussed before, a tag can be represented with a set of popular social tags derived from the collected items with that tag. We can calculate the score of user u_i to topic c_x in each tag $t_{i,k}$ denoted as $c_{x,k}$ for the user u_i, shown as below:

$$sc(u_i, c_{x,k}) = s(t_{i,k}) \cdot w(c_{x,k}, t_{i,k}, u_i), x = 1..q, k = 1..a \quad (2)$$

The user's interest score to the topic c_x, $sc'(u_i, c_x)$, is calculated by summing up the user's interests to the topic in all his tags:

$$sc(u_i, c_x) = \sum_{k=1}^{a} sc(u_i, c_{x,k}) \quad (3)$$

With Equation 3, users' interest distributions over their own original tags are converted to users' interest distributions over the topics of items that are represented by the popular tags. Using this user profiling approach, the noise of social tags can be greatly removed while each user's personal viewpoint of classifications or collections will still remain. Moreover, since the size of the converted matrix is much smaller than the size of the matrix (user, (tag, item)), the information sharing among different users can be improved as well.

3.4 Neighborhood Formation

Neighborhood formation is to generate a set of like-minded peers for a target user. Forming a neighborhood for a target user $u_i \in U$ with standard "best-*K*-neighbors" technique involves computing the distances between u_i and all other users and selecting the top K neighbors with shortest distances to u_i. Based on user profiles, the similarity of users can be calculated through various proximity measures. Pearson correlation and cosine similarity are widely used to calculate the similarity based on numeric values.

Based on the user profiles discussed above, for any two users u_i and u_j with profile v_i and v_j, the Pearson correlation is used to calculate the similarity, which is defined as below:

$$sim(u_i, u_j) = \frac{\sum_{y=1}^{q} (v_{i,y} - \overline{v_i}) \cdot (v_{j,y} - \overline{v_j})}{\sum_{y=1}^{q} (v_{i,y} - \overline{v_i})^2 \cdot \sum_{y=1}^{q} (v_{j,y} - \overline{v_j})^2)} \quad (4)$$

Using the similarity measure approach, we can generate the neighborhood of the target user u_i, which includes K nearest neighbour users who have similar topic interests with u_i. The neighbourhood of u_i, is denoted as:

$$\check{N}(u_i) = \{u_j | u_j \epsilon \, maxK\{sim(u_i, u_j)\}, u_j \epsilon U$$

where *maxK {}* is to get the top K values.

3.5 Recommendation Generation

For each target user u_i, a set of candidate items will be generated from the items tagged by u_i 's neighbourhood formed based on the similarity of users, which is denoted as $\check{C}(u_i)$, $\check{C}(u_i) = \{p_k | p_k \epsilon P(u_j), u_j \epsilon \check{N}(u_i), p_k \notin P(u_i)\}$, where $P(u_j)$

is the item set of user u_j . With the typical collaborative filtering approach, those items that have been collected by the nearest neighbors will be recommended to the target user.

As discussed in Section 3.2, the aggregated social tags describe the content information of items and the topics of each item can be represented by popular social tags. Thus, we propose to combine the content information of items formed by popular social tags with the typical collaborative filtering approach to generate recommendations. Those items that not only have been collected by the nearest neighbors but also have the most similar topics to the target user's interests will be recommended to the target user, which makes the proposed recommendation generation approach actually get the benefits of the content based recommendation approaches [8].

For each candidate item $p_k \epsilon \check{C}(u_i)$, let $\check{N}(u_i, p_k)$ be the set of users in $\check{N}(u_i)$ who have tagged the item p_k, the prediction score of how much u_i may be interested in p_k is calculated in terms of the aspects of how similar those users who have the item p_k and how similar the item's topics with u_i's topic interest.

With Equation 4, the similarity of two users can be measured. Similarly, the Pearson correlation is used to calculate the similarity of the topic interests of user u_i and the topics of the candidate item p_k, which is denoted as below:

$$sim(u_i, p_k) = \frac{\sum_{y=1}^{q}(v_{i,y} - \overline{v_i}) \cdot (b_{k,y} - \overline{b_k})}{\sum_{y=1}^{q}(v_{i,y} - \overline{v_i})^2 \cdot \sum_{y=1}^{q}(b_{k,y} - \overline{b_k})^2)} \quad (5)$$

Thus, the prediction score denoted as $A(u_i, p_k)$ can be calculated with Equation 6.

$$A(u_i, p_k) = \frac{sim(u_i, p_k) \cdot \sum_{u_j \epsilon \check{N}(u_i, p_k)} sim(u_i, u_j)}{|\check{N}(u_i, p_k)|} \quad (6)$$

The top *N* items with larger prediction scores will be recommended to the target user u_i.

4 Experiments and Evaluations

4.1 Experiment setup

We conducted the experiments using the dataset obtained from Amazon.com. The dataset was crawled from amazon.com on April, 2008. The items of the dataset are books. To avoid too sparse, in pre-processing, we removed the books that are only tagged by one user. The final dataset comprises 5177 users, 37120 tags, 31724 books and 242496 records.

The precision and recall are used to evaluate the recommendation performance. The whole dataset is split into a training dataset and a test dataset with 5-folded and the split percentage is 80% for the training dataset and 20% for the test dataset, respectively. Because our purpose is to recommend books to users, the test dataset only contain users' books information. Each record in the test dataset consists of the books that are tagged by one user. The training dataset, which is used to build user profiles, contains users' books and corresponding tags information as well. For each user in the test dataset, the top N items will be recommended to the user. If any item in the recommendation list is in the target user's testing set, then the item is counted as a hit.

4.2 Parameterization

The global popularities of tags are shown in Figure 2. We can see that the user number of tags follows the power law distribution, which means that a small number of tags are used by a large number of users while a large number of tags are only used by a small number of users. Among 37120 tags, there are about 67% tags (i.e., 25006 tags) which are only used by one user.

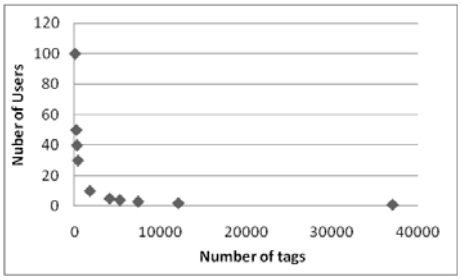


Figure 2: The distribution of social tags.

After calculating the local popularity of each tag for each item, we get λ=2. Thus, we set θ=2. To evaluate the effectiveness of the selected popular tag set, we compared the top 5 precision and recall results of the threshold θ=2 with the results of θ =1, θ =3, θ =4, and θ =5. With threshold θ =1, 37120 tags are selected, which is the whole tag set. Thus, each item was represented with all the tags. Different from the *Topic-Tag* approach, each tag was represented with the selected tags. With threshold θ =2, 12214 tags are selected. When threshold θ =3, 7428 tags were selected and there were 1188 books that have no selected tags describes them. With threshold θ =4, 5297 tags were selected and there were 1668 books that have no selected tags describes them. With threshold θ =5, 4104 tags were selected and there were 2452 books that have no selected tags describes them. The top 5 precision and recall results with different threshold are shown in Figure 3.

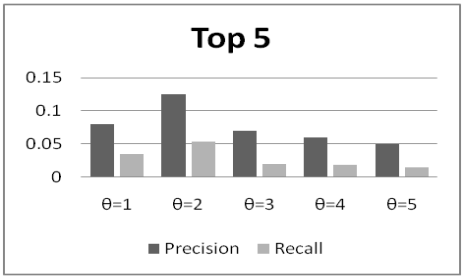


Figure 3. The top 5 precision and recall evaluation results with different threshold θ values.

From the results of Figure 3, we can see the results of θ =2 was better than other values. Thus, the popular tags can be used to represent the topics of items and tags. And, since some books may don't have any selected tags describing their topics when the threshold is too high, the results are worse.

4.3 Comparison

To evaluate the effectiveness of the proposed approach, we compared the precision and recall of the recommended top N items produced by the following approaches:

- ***Topic-PopularTag approach***. This is the proposed approach that uses the popular tag to represent items' topics, tags' actual topics and users' topic interests.
- ***Topic-Tag approach.*** This approach uses users' interest distribution to their original tags to make recommendation. Different from *Topic-PopularTag* approach, this approach only uses the users' original tags to profile users and doesn't include the tag representations.
- ***Singular Value Decomposition (SVD).*** This is a wildly used approach to reduce the dimensions of a matrix and reduce noise. In this paper, the standard SVD based recommendation approach [8] was implemented based on the user-tag matrix.
- ***Tso-Sutter's approach***. This approach is proposed by Tso-Sutter that uses two derived binary matrixes user-item, user-tag to make recommendation [9], which is an extended standard collaborative filtering approach.
- ***Liang's approach.*** This approach is proposed by Liang that uses three derived binary matrixes user-item, user-tag to tag-item sub matrix to make recommendation [12], which is an extended standard collaborative filtering approach.
- ***Standard CF approach.*** This is the standard collaborative filtering (CF) approach [14] that uses the implicit item ratings or the binary matrix user-item only. This is the baseline approach.

We compared the proposed approach that has the threshold θ =2 with other state of art approaches, the precision and recall results are shown in Figure 4 and Figure 5.

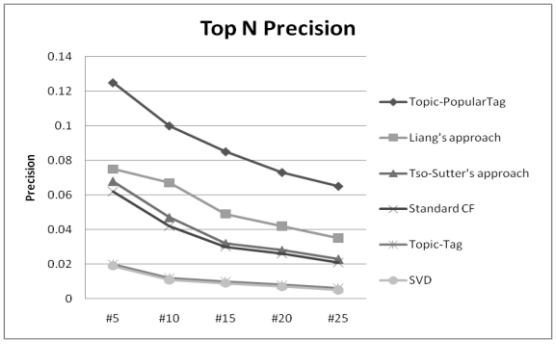


Figure 4: Precision evaluation results.

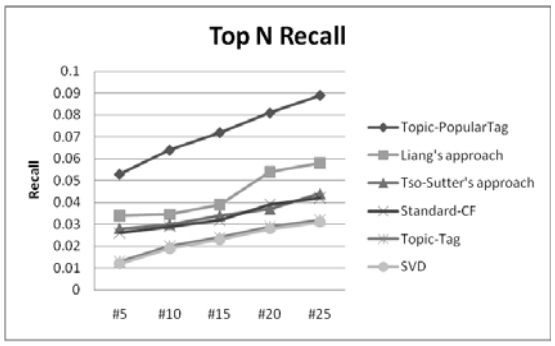


Figure 5: Recall evaluation results.

4.4 Discussions

From the experimental results, we can see that the proposed approach outperformed the other approaches, which means the proposed collaborative filtering approach based on popular social tags is effective. Since the dataset is very sparse (i.e., the average number of items that each user has is about 12.6), the overall precision and recall values are low. The approach *Topic-Tag* approach performed the worst, which means that although tags implies users' interests and preferences information, since the social tags contains a lot of noise, it's inaccurate to profile users with their original tags directly. The comparison between the approaches of *Tso-Sutter* and *Liang* and the *Standard CF* approach shows that social tags are helpful to improve the user profiling accuracy when the social tags are used together with the users' collected items. Moreover, the comparison between the proposed *Topic-PopularTag* approach and the *SVD* approach suggests that the proposed approach performs better than the traditional dimension reduction approach. The proposed approach not only reduce the dimension through using a much smaller sized user-topic matrix to profile users but also significantly improves the accuracy of user profiling and information sharing through representing the personal or unpopular tags with a set of popular tags.

5. Conclusions

In this paper, we propose a collaborative filtering approach that combines each user's personal viewpoint of the classifications of items and the common viewpoint of many users about the classifications of items to make personalized item recommendation. The popular tags are used to represent items' major topics, tags' actual covered or related topics and users' topic interests. Moreover, a user profiling approach that converts users' interest distribution for their own original tags to users' interest distribution for topics that are represented with the popular tags are proposed to improve user profiling accuracy and information sharing. Also, we propose a recommendation generation approach that incorporates the item content

information formed by the collaborative working of tagging to generate recommended items that are not only have been collected by most similar users but also have the most similar topics with the target user's interests.

The experiments show that the proposed approach outperforms other approaches. Since the social tags can be used to describe any types of items or resources, this research can be used to recommend various kinds of items to users, which provides possible solutions to the recommendation of those items that the traditional collaborative filtering approaches or content based approaches fail to work well such as people. Moreover, this research made a contribution to the improvement of information sharing, organization and retrieval of online tagging systems as well as the improvement of the recommendation performances of traditional recommender systems (i.e., in e-commerce websites) through incorporating this new type of user information in web 2.0.

References

[1] Bao, S., Wu, X., Fei, B., Xue, G., Su, Z. and Yu, Y., "Optimizing Web Search Using Social Annotations", In *Proc. of WWW'07*, 2007, pp. 501-510.

[2] Li, X., Guo, L., and Zhao, Y. E., "Tag-based social interest discovery", In *Proc. of WWW'08*, 2008, pp. 675-684.

[3] Tso-Sutter, K.H.L., Marinho, L.B. and Schmidt-Thieme, L., "Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms", In *Proc. of Applied Computing*, 2008, pp. 1995-1999.

[4] Bischoff, K., Firan, C. S., Nejdl, W., Paiu, R., "Can All Tags be Used for Search?", In *Proc. of CIKM'08*, 2008, pp. 193-202.

[5] Sen, S., S. Lam, A. Rashid, D. Cosley, D. Frankowski, J.Osterhouse, M. Harper, and J. Riedl., "Tagging, communities, vocabulary, evolution", In *Proc. of CSCW '06*, 2006, pp. 181-190.

[6] What Is Web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[7] Burke, R., "Hybrid Recommender Systems: Survey and Experiments", *User Modeling and User-Adapted Interaction*, 12(2002), pp. 331-370.

[8] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. "Application of Dimensionality Reduction in Recommender System — A Case Study." In *Proc.of WebKDD'00, 2000*.

[9] K.H.L. Tso-Sutter, L.B. Marinho and L.Schmidt-Thieme, "Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms", In *Proc. Applied Computing'*08, 2008, pp.1995-1999.

[10] Heymann, P., Ramage, D., and Garcia-Molina, H., "Social tag prediction", In *Proc. of SIGIR'08,* 2008, pp. 531–538.

[11] Gemmis, M. de, Lops, P., Semeraro, G., and Basile, P., "Integrating tags in a semantic content-based recommender", In *Proc. of the 2008 ACM conference on Recommender systems,* 2008, pp. 163-170.

[12] Liang, H., Xu, Y., Li, Y., and Nayak, R., "Collaborative Filtering Recommender Systems Using Tag Information", In *Proc. of The 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI-08) Workshops*, 2008, pp. 59-62.

[13] Al-Khalifa, H.S. and Davis, H. C., "Exploring the Value of Folksonomies for Creating Semantic Metadata", *International Journal on Semantic Web and Information Systems*, 3,1 (2007), pp. 13-39.

[14] Shardanand, U. and Maes,P., "Social Information Filtering: Algorithms for Automating 'Word of Mouth'", In *Proc. of SIGCHI*, 1995, pp. 210 -217.

[15] Suchanek, F. M., Vojnovi´c, M., Gunawardena D., "Social tags: Meaning and Suggestions", In *Proc.of CIKM'08,* 2008, pp. 223-232

[16] Sen, S., Vig, J., Riedl, J., "Tagommenders: Connecting Users to Items through Tags", In *Proc. of WWW'09*, 2009, pp. 671-680

[17] Au Yeung, C. M., Gibbins, N. and Shadbolt, N., "Contextualizing Tags in Collaborative Tagging Systems", In *Proc. of the 20th ACM Conference on Hypertext and Hypermedia*, 2009.

External Evaluation of Topic Models

David Newman *Sarvnaz Karimi* *Lawrence Cavedon*
NICTA and The University of Melbourne
Parkville, Victoria 3010, Australia
{*david.newman, sarvnaz.karimi, lawrence.cavedon*}*@nicta.com.au*

Abstract *Topic models can learn topics that are highly interpretable, semantically-coherent and can be used similarly to subject headings. But sometimes learned topics are lists of words that do not convey much useful information. We propose models that score the usefulness of topics, including a model that computes a score based on pointwise mutual information (PMI) of pairs of words in a topic. Our PMI score, computed using word-pair co-occurrence statistics from external data sources, has relatively good agreement with human scoring. We also show that the ability to identify less useful topics can improve the results of a topic-based document similarity metric.*

Keywords Topic Modeling, Evaluation, Document Similarity, Natural Language Processing, Information Retrieval

1 Introduction

Topic models are unsupervised probabilistic models for document collections, and are generally regarded as the state-of-the-art for extracting course-grained semantic information from collections of text documents. The extracted semantic content is useful for a variety of applications including automatic categorization and faceted browsing. The topic model technique learns a set of thematic topics from words that tend to co-occur in documents. The technique assigns a small number of topics to each document, and those topics can then be used to explain and retrieve documents. However this explanation of a document is only useful if we can understand what is meant by a given topic.

Since the introduction of the original topic model approach [Blei et al., 2003, Griffiths and Steyvers, 2004], many researchers have modified and extended topic modeling in a variety of ways. However, there has been less effort on understanding the semantic nature of topics learned by topic models. While the list of the most likely (i.e. important) words in a topic provides good transparency to defining a topic, how can humans best interpret and understand the gist of a topic? Some researchers have started to address this problem, including Mei et al. [2007] who looked at the problem of automatic assignment of a short label for a topic, and Griffiths and Steyvers [2006] who applied topic models to word sense distinction tasks. Wallach et al. [2009] proposed methods for evaluating topic models, but they focused on the statistics of the model, not the meaning of individual topics.

The challenge of helping a user understand a discovered topic is exacerbated by the variable semantic quality of topics produced by a topic model. Certain types of document collections, for example collections of abstracts of research papers, produce mostly high-quality interpretable topics which have clear semantic meaning. However, the broader class of document collections — for example emails, blogs, news articles and books — tend to produce a wider mix of topics. The novelty of our work is targetting this challenge by focusing on evaluation of topics using their degree of usefulness to humans.

In this work we first ask humans to decide whether individual learned topics are useful or not (we define what is meant by useful). We then propose models that use external text data sources, such as Wikipedia or Google hits, to predict human judgements. Finally, we show how an assessment of useful and useless topics can improve the outcome of a document similarity task.

2 Topic Modeling

The topic model — also known as *latent Dirichlet allocation* or *discrete principal component analysis (PCA)* — is a Bayesian graphical model for text document collections represented by bags-of-words (see Blei et al. [2003], Griffiths and Steyvers [2004], Buntine and Jakulin [2004]). In a topic model, each document in the collection of D documents is modeled as a multinomial distribution over T topics, where each topic is a multinomial distribution over W words. Typically, only a small number of words are important (have high likelihood) in each topic, and only a small number of topics are present in each document.

The collapsed Gibbs [Geman and Geman, 1984] sampled topic model simultaneously learns the topics and the mixture of topics in documents by iteratively sampling the topic assignment z to every word in every document, using the Gibbs sampling update

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

$$p(z_{id} = t | x_{id} = w, \mathbf{z}^{\neg id}) \propto$$
$$\frac{N_{wt}^{\neg id} + \beta}{\sum_w N_{wt}^{\neg id} + W\beta} \quad \frac{N_{td}^{\neg id} + \alpha}{\sum_t N_{td}^{\neg id} + T\alpha},$$

where $z_{id} = t$ is the assignment of the i^{th} word in document d to topic t, $x_{id} = w$ indicates that the current observed word is w, and $\mathbf{z}^{\neg id}$ is the vector of all topic assignments not including the current word. N_{wt} represents integer count arrays (with the subscripts denoting what is counted), and α and β are Dirichlet priors.

The maximum a posterior (MAP) estimates of the topics $p(w|t)$, $t = 1 \ldots T$ and the mixture of topics in documents $p(t|d)$, $d = 1 \ldots D$ are given by

$$p(w|t) = \frac{N_{wt} + \beta}{\sum_w N_{wt} + W\beta},$$
$$p(t|d) = \frac{N_{td} + \alpha}{\sum_t N_{td} + T\alpha}.$$

Pathology of Learned Topics

Despite referring to the distributions $p(w|t)$ as topics, suggesting that they have sensible semantic meaning, they are in fact just statistics that explain count data according to the underlying generative model. To be more explicit, while many learned topics convey information similar to what is conveyed by a subject heading, topics themselves are not subject headings, and they sometimes are not at all related to a subject heading.

Since our focus in this paper is studying and evaluating the wide range of topics learned by topic models, we present examples of less useful topics learned by topic models. Note that these topics are not simply artifacts from one particular model started from some particular random initialization – they are stable features present in the data that can be repeatedly learned from different models, hyperparameter settings and random initializations. The following list shows an illustrative selection of less useful topics:

- north south carolina korea korean southern kim daewoo government country million flag thoreau economic war ... *This topic has associated Carolina with Korea via the words north and south.*
- friend thought wanted went knew wasn't love asked guy took remember kid doing couldn't kind ... *This is a typical "prose" style topic often learned from collections of emails, stories or news articles.*
- google domain search public copyright helping querying user automated file accessible publisher commercial legal ... *This is a topic of boilerplate copyright text that occurred in a large subset of a corpus.*
- effect significant increase decrease significantly change resulted measured changes caused ... *This is a topic of comparisons that was learned from a large collection of MEDLINE abstracts.*
- weekend december monday scott wood going camp richard bring miles think tent bike dec pretty ... *This topic includes a combination of several commonly occurring pathologies including lists of names, days of week, and months of year.*

Collections Modeled

We used two document collections: a collection of news articles, and a collection of books. These collections were chosen to produce sets of topics that have more variable quality than one typically observes when topic modeling collections of scientific literature. A collection of $D = 55,000$ news articles was selected from Linguistic Data Corporation's gigaword corpus, and a collection of $D = 12,000$ books was downloaded from the Internet Archive. We refer to these collections as "News Articles" and "Books" throughout the remainder of this paper.

Standard procedures were used to create the bags-of-words for the two collections. After tokenization, and removing stopwords and words that occurred fewer than ten times, we learned topic models of News Articles using $T = 50$ ($T50$) and $T = 200$ ($T200$) topics, and a topic model of Books using $T = 400$ ($T400$) topics. For each topic model, we printed the set of T topics. We define a topic as the list of ten most probable words in the topic. This cutoff at ten words is arbitrary, but it balances between having enough words to convey the meaning of a topic, but not too many words to complicate human judgements or our scoring models.

3 Human Scoring of Topics

We selected 117 topics from News Articles, including all 50 topics from the $T50$ topic model, and 67 selected topics from the $T200$ topic model. We selected 120 topics from the $T400$ topic model of Books. To increase the expected number of useful and useless topics, we pre-scored topics using our scoring models (described later) to select a mix of useful, useless, and in-between topics to make up the sample. We asked nine human subjects to score each of the 237 topics on a 3-point scale where 3="useful" and 1="useless".

We provided a rubric and some guidelines on how to judge whether a topic was useful or useless. In addition to showing several examples of useful and useless topics, we gave the following instructions to people performing the evaluation:

The topics learned by a topic model are usually sensible, meaningful, interpretable and coherent. But some topics learned (while statistically reasonable) are not particularly useful for human use. To evaluate our methods, we would like your judgment on how "useful" some learned topics are. Here, we are purposefully vague about what is "useful" ... it is some combination of coherent, meaningful, interpretable, words are related, subject-heading like, something you could easily label, etc.

Figure 1 shows selected useful and useless topics from News Articles, as scored by nine people. For our purposes, the usefulness of a topic can be thought of as whether one could imagine using the topic in a search interface to retrieve documents about a particular

Selected useful topics (unanimous score=3):
space earth moon science scientist light nasa mission planet mars ...
health disease aids virus vaccine infection hiv cases infected asthma ...
bush campaign party candidate republican mccain political presidential ...
stock market investor fund trading investment firm exchange companies ...
health care insurance patient hospital medical cost medicare coverage ...
car ford vehicle model auto truck engine sport wheel motor ...
cell human animal scientist research gene researcher brain university ...
health drug patient medical doctor hospital care cancer treatment disease ...

Selected useless topics (unanimous score=1):
king bond berry bill ray rate james treas byrd key ...
dog moment hand face love self eye turn young character ...
art budget bos code exp attn review add client sent ...
max crowd hand flag sam white young looked black stood ...
constitution color review coxnet page art photos available budget book ...
category houston filed thompson hearst following bonfire mean tag appear ...
johnson jones miller scott robinson george lawrence murphy mason ...
brook stone steven hewlett packard edge borge nov buck given ...

Figure 1: Selected useful and useless topics from collection of News Articles. Each line represents one topic.

Selected useful topics (unanimous score=3):
steam engine valve cylinder pressure piston boiler air pump pipe ...
furniture chair table cabinet wood leg mahogany piece oak louis ...
building architecture plan churches design architect century erected ...
cathedral church tower choir chapel window built gothic nave transept ...
god worship religion sacred ancient image temple sun earth symbol ...
loom cloth thread warp weaving machine wool cotton yarn mill ...
window nave aisle transept chapel tower arch pointed arches roof ...
cases bladder disease aneurism tumour sac hernia artery ligature pain ...

Selected useless topics (unanimous score=1):
entire finally condition position considered result follow highest greatest ...
aud lie bad pro hut pre able nature led want ...
soon short longer carried rest turned raised filled turn allowed ...
act sense adv person ppr plant sax genus applied dis ...
httle hke hfe hght able turn power lost bring eye ...
soon gave returned replied told appeared arrived received return saw ...
person occasion purpose respect answer short act sort receive rest ...
want look going deal try bad tell sure feel remember ...

Figure 2: Selected useful and useless topics from collection of Books.

subject. An indicator of usefulness is the ease by which one could think of a short label to describe a topic (for example "space exploration" could be a label for the first topic). The useless News Articles topics display little coherence and relatedness, and one would not expect them to be useful as categories or facets in a search interface.

We see similar results in Figure 2, which shows selected useful and useless topics from the Books collection. Again, the useful topics could directly relate to subject headings, and be used in a user interface for browse-by-subject. Note that the useless topics from both collections are not chance artifacts produced by the models, but are in fact stable and robust statistical features in the data sets.

Our human scoring of the 237 topics has high inter-rater reliability, as shown in Figure 3. Each human score has high agreement with the mean of the remaining scores (Pearson correlation coefficient $\rho = 0.78 \ldots 0.81$). In the following sections we present models to predict these human judgements.

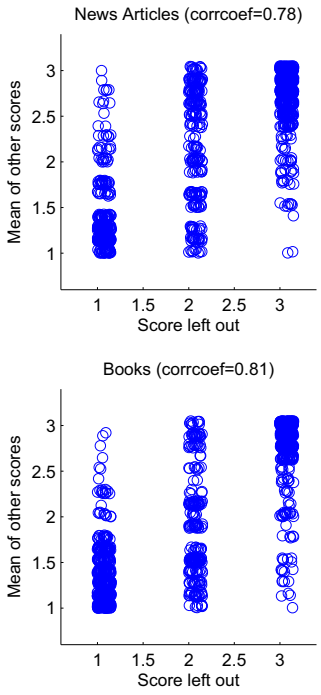


Figure 3: Inter-rater reliability, computed by leave-one-out, showing high agreement between the nine humans.

This inter-rater correlation is an upper bound on how well we can expect our scoring models to perform.

4 Scoring Model I: Pointwise Mutual Information

The intuition behind our first scoring model, pointwise mutual information (PMI) using external data, comes from the observation that occasionally a topic has some odd-words-out in the list of ten words. This leads to the idea of a scoring model based on word association between pairs of words, for all word pairs in a topic. But instead of using the collection itself to measure word association (which could reinforce noise or unusual word statistics), we use a large external text data source to provide *regularization*.

Specifically, we measured co-occurrence of word pairs from two huge external text datasets: all articles from English Wikipedia, and the Google n-grams data set. For Wikipedia we counted a co-occurrence as words w_i and w_j co-occurring in a 10-word window in any article, and for Google n-grams, we counted a co-occurrence as w_i and w_j co-occurring in any of the 5-grams. These co-occurrences are counted over corpora of 1B and 1T words respectively, so they produce reasonably reliable statistics.

We choose pointwise mutual information as the measure of word association, and define the following scoring formula for a topic $\mathbf{w}$:

$$\text{PMI-Score}(\mathbf{w}) = \text{median}\{\text{PMI}(w_i, w_j), ij \in 1 \ldots 10\},$$

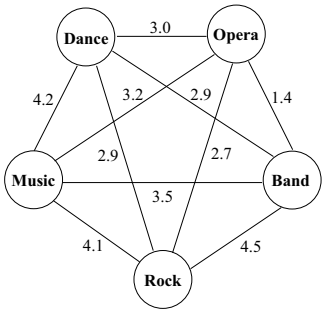


Figure 4: Illustration of pointwise mutual information between word pairs.

$$\text{PMI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)},$$

where the top-ten list of words in a topic is denoted by $\mathbf{w} = (w_1, \ldots, w_{10})$, and we exclude the self PMI case of $i = j$. The PMI-Score for each topic is the median PMI for all pairs of words in a topic (so for a topic defined by the top-10 words, the PMI-Score is the median of 55 PMIs). Note that if two words are statistically independent, then their PMI is zero.

Our PMI-Score is illustrated in Figure 4 for a topic of five words: "music band rock dance opera".[1] Using co-occurrence frequencies from Wikipedia, we see unsurprising high-scoring word pairs, such as PMI(rock,band)=4.5, and PMI(dance,music)=4.2. Some pairs exhibit greater independence, such as PMI(opera,band)=1.4. The PMI-Wiki-Score[2] for this topic is the median of all the PMIs, or PMI-Wiki-Score=3.1.

We see broad agreement between the PMI-Wiki-Score and the human scoring in Figure 5, which shows a scatterplot for all 237 topics. The correlation between the PMI-Wiki-Score and the mean human score is $\rho = 0.72$ for News Articles and $\rho = 0.73$ for Books (we define correlation ρ as the Pearson correlation coefficient). This correlation is relatively high given that the inter-rater-correlation is only slightly higher at $\rho = 0.78 \ldots 0.81$.

Using the Google 5-grams data instead of English Wikipedia for the external data source produces similar results, shown in Figure 6. In this case, the pointwise mutual information values are computed using word statistics from the 1 billion Google 5-grams instead of 2 million Wikipedia articles. The correlations are in a similar range ($\rho = 0.70 \ldots 0.78$) with a slightly higher correlation of $\rho = 0.78$ for News Articles.

Why does our PMI-Score model agree so well with human scoring of topics? Our intuition is that humans consider associations of pairs of words (or the association between one word and all the other words) to determine the relatedness and usefulness of a topic. This human process is somewhat approximated by the calculation of the PMI-Score.

[1] We illustrate using 5 words instead of 10 for simplicity.

[2] This is the PMI-Score computed using frequency counts from Wikipedia.

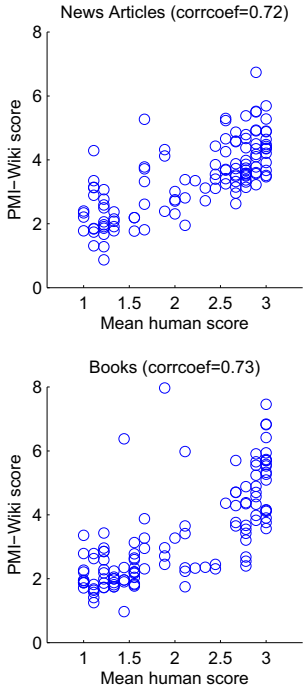


Figure 5: Scatterplot of PMI-Wiki-Score vs. mean human score.

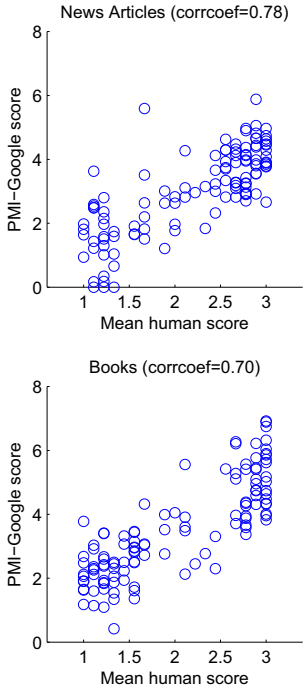


Figure 6: Scatterplot of PMI-Google-Score vs. mean human score.

5 Scoring Model II: Google

In this section we present a second scoring scheme, again based on a large external data source: this time

the entire World Wide Web crawled by Google. We present two scoring formulas that use the Google search engine:

$$\text{Google-titles-match}(\mathbf{w}) = \mathbf{1}\left[w_i = v_j\right],$$

where $i = 1, \ldots, 10$ and $j = 1, \ldots, |V|$, and v_j are all the unique terms mentioned in the titles from the top-100 search results, and $\mathbf{1}$ is the indicator function to count matches; and

$$\text{Google-log-hits}(\mathbf{w}) = \log(\#\text{ results from search for } \mathbf{w}),$$

where $\mathbf{w}$ is the search string "+w_1 +w_2 +w_3 ... +w_{10}". We use the Google advanced search option '+' to search exactly as is and prevent Google from using synonyms. Our intuition is that the mention of topic words in URL titles — or the prevalence of documents that mention all ten words in the topic — may better correlate with a human notion of the usefulness of a topic.

For example, issuing the query to Google: "+space +earth +moon +science +scientist +light +nasa +mission +planet +mars" returns 171,000 results (so Google-log-hits($\mathbf{w}$)=5.2), and the following list shows the titles and URLs of the first 6 results:

1. NASA - STEREO Hunts for Remains of an Ancient Planet near Earth (science.nasa.gov/headlines/y2009/...)
2. NASA - Like Mars, Like Earth (www.nasa.gov/audience/foreducators/k-4/features/...)
3. NASA - Like Mars, Like Earth (www.nasa.gov/audience/forstudents/5-8/features/...)
4. ASP: The Silicon Valley Astronomy Lectures Podcasts (www.astrosociety.org/education/podcast/index.html)
5. NASA calls for ambitious outer solar system mission - space ... (www.newscientist.com/article/...)
6. NASA International Space Station Mission Shuttle Earth Science ... (spacestation-shuttle.blogspot.com/2009/08/...)

The underlined words show mentions of topic words in the URL titles, with the first six titles giving a total of 17 mentions. The top-100 URL titles include a total of 194 matches, so for this topic Google-titles-match($\mathbf{w}$)=194.

We see surprisingly good agreement between the Google-titles-match score and the human scoring in Figure 7 for the News Articles ($\rho = 0.78$), and a lower level of agreement for Books ($\rho = 0.52$). In the PMI-Scores there was no clear pattern of outliers in the scatterplots against the mean human score. However, we see a definite constraint of the Google-titles-match score, where there are many topics that received a high human score, but a low Google-titles-match score. Table 1 shows selected topics having a high human score (useful), but a low Google-titles-match score. The first three topics listed (from News Articles) show different types of problems. The first topic is clearly about cooking, but does not mention the word cooking. Furthermore, it is unlikely that URL titles would include words such as "teaspoon" or "pepper", so we

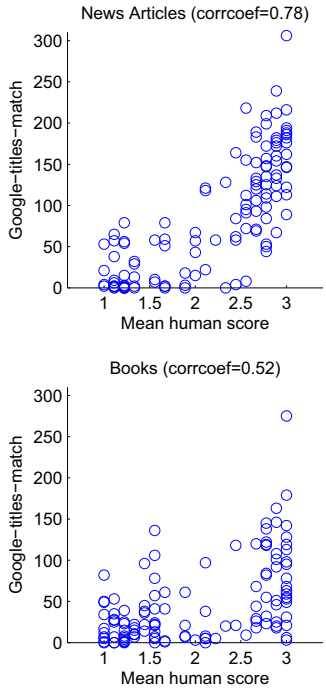


Figure 7: Scatterplot of Google-titles-match score vs. mean human score.

are not surprised that Google-titles-match fails to give this topic a high score. The second topic is mostly about NASA and space exploration, but is polluted by the words "firefighter" and "worcester", which will severely limit the number of results returned. By using the median, the PMI-Score of this topic is less sensitive to these words that don't fit the topic, but the Google-titles-match has less hope of producing a useful list of search results when all ten words are included in the search query. Topics from Books follow, and we see a similar problem to the cooking topic from News Articles, where the words in the topic clearly convey something semantically coherent, but fail to evoke URL titles that mention those general terms.

We see less promising results from our Google-log-hits score, which has relatively low correlation with the mean human scoring ($\rho = -0.09 \ldots 0.49$), as shown in the scatterplots in Figure 8. For this scoring formula we observed the reverse of the problem of Google-titles-match, namely we saw overly favorable scoring of many topics that received a low human score. Table 2 shows selected topics having a low human score (not useful), but a high Google-log-hits score. The topics in this table all exhibit the similar characteristic of all ten words being relatively common words. Consequently there exist many web pages that contain these words (issuing these topics as queries returned between 250,000 and 10,000,000 results). This behavior of Google-log-hits and failure to agree with human scoring (in this case) is relatively easy to understand.

Human	Titles-match	Topic
2.6	8	cup add tablespoon salt pepper teaspoon oil heat sugar pan ...
2.4	4	space nasa moon mission shuttle firefighter astronaut launch worcester rocket ...
2.3	0	oct series braves game yankees league bba met championship red ...
2.9	25	church altar churches stone chapel cathedral vestment service pulpit chancel ...
3.0	6	cases bladder disease aneurism tumour sac hernia artery ligature pain ...
2.8	23	art ancient statues statue marble phidias artist winckelmann pliny image ...
3.0	3	window nave aisle transept chapel tower arch pointed arches roof ...
2.9	18	crop land wheat corn cattle acre grain farmer manure plough ...
2.8	32	account cost item profit balance statement sale credit shown loss ...
2.9	20	pompeii herculaneum room naples painting inscription excavation marble bronze bath ...
3.0	21	window nave choir arch tower churches aisle chapel transept capital ...
3.0	31	drawing draw pencil pen drawn model cast sketches ink outline ...

Table 1: Disagreement between high human scores and low Google-titles-match scores.

Human	log hits	Topic
1.0	5.4	dog moment hand face love self eye turn young character ...
1.2	7.0	change mean different better result number example likely problem possible ...
1.2	6.4	fact change important different example sense mean matter reason women ...
1.1	5.9	friend thought wanted went knew wasn't love asked guy took ...
1.1	5.6	thought feel doesn't guy asked wanted tell friend doing went ...
1.1	6.1	bad doesn't maybe tell let guy mean isn't better ask ...
1.0	6.7	entire finally condition position considered result follow highest greatest fact ...
1.0	6.3	soon short longer carried rest turned raised filled turn allowed ...
1.1	6.1	modern view study turned face detail standing born return spring ...
1.2	6.3	sort deal simple fashion easy exactly call reason shape simply ...
1.1	6.4	proper require care properly required prevent laid making taking allowed ...
1.0	6.7	person occasion purpose respect answer short act sort receive rest ...
1.0	6.1	want look going deal try bad tell sure feel remember ...
1.2	6.3	saw cried looked heard stood asked sat answered began knew ...

Table 2: Disagreement between low human scores and high Google-log-hits scores.

6 Document Similarity

Discovering semantically similar documents in a collection of unstructured text has practical applications, such as search by example. Many studies have been proposed to calculate inter-document similarity since 1950s. For example, Grangier and Bengio [2005] use hyperlinks to score linked documents on the Web higher than unlinked for information retrieval tasks. Kaiser et al. [2009] use Wikipedia to find similar documents for a focused crawler (they also provide a good literature review on recent approaches that use support vector machines, latent semantic analysis (LSA), or explicit semantic analysis). Lee et al. [2005] empirically compare between three categories of binary, count, and LSA similarity models over a small corpus of human judged texts and concluded that evaluation of such models should occur in the context of their applications.

Humans judge two texts to be similar if they share the same concepts or topics [Kaiser et al., 2009]. We use our learned topics from News Articles to find similar documents and compare them against count-based models implemented in a search engine. Our preliminary findings show that if documents contain useless text — words that are not related to the main topic of the text or bear no content, such as advertisements — then they are likely to be mistakenly considered similar using document similarity metrics that rely on term frequencies. Below, we explain our experimental setup and results.

Count-Based Similarity

We used the Okapi BM25 [Walker et al., 1997] ranking function implemented in the Zettair[3] search engine. Similarity scores are based on term frequency and inverse document frequencies in a document collection.

Topic-Based Similarity

A document similarity measure using topics was computed using Hellinger distance. For every pair of documents d_i and d_j in a collection, and a set T of learned topics, Hellinger distance is computed as below:

$$\text{dist}(d_i, d_j) = \frac{1}{2} \sum_{t=1}^{T} \left(\sqrt{p(t|d_i)} - \sqrt{p(t|d_j)} \right)^2 ,$$

$$\text{dist}^*(d_i, d_j) = \frac{1}{2} \sum_{t \in \text{useful}} \left(\sqrt{p(t|d_i)} - \sqrt{p(t|d_j)} \right)^2 ,$$

[3] http://www.seg.rmit.edu.au/zettair/

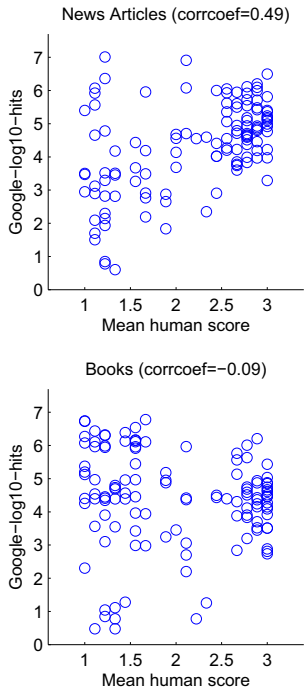


Figure 8: Scatterplot of Google-log-hits score vs. mean human score.

where $p(t|d_i)$ and $p(t|d_j)$ are probabilities of topics in documents i and j. We provide two formulas for Hellinger distance, one based on all topics, and dist* that uses just the "useful" topics.

Experimental Setup

Fifty documents were randomly selected from News Articles based on their proportion of useful and useless topics. An overview of the documents in the collection based on their percentages of useless text is shown in Figure 9. Our aim is to improve document similarity calculations on the right tail of this graph where the documents contain a larger proportion of useless text which could mislead document similarity methods that rely on the frequency of terms. We therefore first extracted those documents that contained at least 30% useful content (based on PMI-Wiki-Score) and at least 40% non-content text. We then calculated the similarity scores of 50 randomly selected documents from this subset with other documents in the collection. For count-based methods, we used each of these 50 full documents as queries to retrieve a ranked list of similar documents using the Zettair search engine. For the topic-based method, two approaches were used: using all the topics generated for the collection ($T200$), and using useful topics as based on the topics' PMI-Wiki-Score.

In a preliminary experiment, a human judge was presented with original documents and the top most similar document (Top-1) extracted by each method. The human judge was not aware of the order of methods which the documents were retrieved. A simple binary

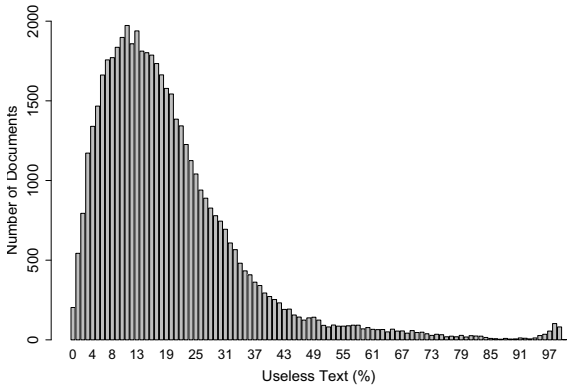


Figure 9: Number of documents versus proportion of usefuless content. 4.3% of documents have more than 50% useless text and 16.4% have more than 30% useless text.

scoring of *similar* or *not-similar* was used. The criteria for similarity was the overall subject of the documents, for example, both being about a specific sport. For 32 of 50 cases (64%), all methods successfully resulted in documents judged to be similar by the human judge. In only one case did Okapi outperform both topic-based methods. Using the useful-topics metric (dist*) led to 94% accuracy against similarity judgements; all topics (dist) was 88% accurate; Okapi was 70% accurate. Also, the overlap between the ranked outputs of the two systems, Okapi and useful topics, was very low: 30% in Top-1 overlapped (the documents were the same for the both systems).

Figure 10 shows an illustrative example where using topic modeling, in particular using good topics (i.e. dist*), outperforms Okapi when the original document contains a large proportion of non-content text.

While he experiments described in this section are limited in scope, they constitute an initial investigation into the task-level effectiveness of topic-based metrics that ignore "useless" topics. We believe that the results indicate that, for texts that contain "noise", identifying the "useful" topics in a topic model has promising applications.

7 Conclusion

Evaluation of topic modeling — the analysis of large sets of unstructured documents and assignment of series of representative words as topics to clusters of documents — has hardly been investigated. In particular, meaning of the topics and human perception of their usefulness had not been studied before. Here, we investigated topic modeling evaluation using external data (Wikipedia documents, Google n-grams, and Google hits), and compared our proposed methods with human judgments on usefulness of the topics. According to our experiments on collections of news articles and books, a scoring method using pointwise mutual information

Original Document

At last! A biography that skips the saint-or-sinner debate. As Dusko Doder and Louise Branson abundantly document, Slobodan Milosevic, almost from the start, epitomized the Balkan-variety bad seed. The child of parents who both committed suicide, Milosevic aligned himself with a woman who hungered for power to avenge the ignominious death of her mother. Milosevic betrayed a college classmate, a mentor of two decades, and his next-door neighbor in lunging to the top of Yugoslavia's diseased post-Tito political leadership. And "Milosevic: Portrait of a Tyrant"...

...

(gm)

FOR WEDNESDAY AMs

Here are today's top news stories from The New York Times News Service for ally at LaSalle University for of Wednesday, Dec. 22: INTERNATIONAL ("i" code) CHINA-INTERNET (Beijing) - With the ambivalent blessing of the Chinese government, locally produced web sites and chat rooms have spread rapidly here in the last two years,... RUSSIA-U.S.-AID (Washington) - The State Department, invoking a seldom-used law, may block a $500 million loan package for Russia's oil sector. By David E. Sanger...

Okapi BM25 (Zettair)

More New Yorkers would vote against Hillary Rodham Clinton as a U.S. Senate candidate than vote for her, a new poll indicates. The survey by the Zogby International polling organization shows the probable Democratic nominee carrying an "unfavorable rating" of 48.4 percent among likely voters, as opposed to her "favorable rating" of 46.3 percent. It marks the first time the potential candidate's statistical negatives have eclipsed her positives in her still-undeclared campaign, pollster John eet, a city councilman in a tight race. "But I hope you of Utica said Tuesday.

...

(gm)

FOR WEDNESDAY AMs

Here are today's top news stories from The New York Times News Service for AMs of Wednesday, Dec. 22: INTERNATIONAL ("i" code) CHINA-INTERNET (Beijing) - With the ambivalent blessing of the Chinese government, locally produced web sites and chat rooms have spread rapidly here in the last two years,... RUSSIA-U.S.-AID (Washington) - The State Department, invoking a seldom-used law, may block a $500 million loan package for Russia's oil sector. By David E. Sanger.

...

All Topics

We may be living in a high-tech era but it still takes a low-tech truck to deliver something you've ordered over the Internet, which is why Forbes magazine picked Atlanta-based United Parcel Service as its "company of the year." "With 157,000 ground vehicles, 610 aircraft and $11 billion invested in technology, UPS moves both atoms and bits," says Forbes in announcing its "platinum list" of "America's best big companies." According to Forbes, UPS's role as a shipper of 6 percent of the nation's gross domestic product makes it "the missing link in the burgeoning world of E-commerce."

...

Story Filed By Cox Newspapers (gm)

Here are the stories New York Times editors are planning for Tuesday, Dec. 28 Page 1. The NYT frontpage advisory, with layout description, will move by 7:30 p.m. ET. The NYT News Service Night

Supervisor is Pat Ryan (888-346-9867). ISRAEL-POLITICS (Jerusalem) - The Shas political party, which represents Sephardic Jews of Middle Eastern and North African descent, announced Monday that it had decided to quit the coalition government of Israeli Prime Minister Ehud Barak.

...

Useful Topics

The Clinton administration, in a move intended to bolster opponents of President Slobodan Milosevic, has agreed to lift economic sanctions on Serbia as soon as there is a free election there, senior administration officials said on Tuesday. The administration had previously vowed that it would not lift the sanctions until Milosevic had been removed from power. But officials calculate that the new strategy should allow the Serbian opposition to increase popular pressure on Milosevic, to call early elections, since holding a free election would mean an end to an oil embargo, an air-travel ban and other sanctions that have weakened an already devastated Serbian economy. Secretary of State Madeleine Albright is expected to make the announcement Wednesday, but it carries a risk: that bickering opposition parties would so fragment the election results that Milosevic might be able to cling to power or, far less likely, that he would win outright in the balloting.

...

Although the constitution of the Yugoslav federation of Serbia and neighboring Montenegro does not grant Milosevic direct power to call new elections, the reality is that his powers are dictatorial

...

Figure 10: An example of top ranked similar documents returned by three methods: Okapi scores generated by Zettair, topic-based similarity using all topics (dist), and topic-based similarity only using useful topics. Using only useful topics (dist*) produces the best result.

on Wikipedia documents and Google n-grams has great potential to distinguish useful (or meaningful) topics from useless ones. This finding is supported by high correlation between our scoring approaches and human judgements on the same topics. We also showed a possible application for distinguished useful topics in extraction of similar documents in a collection.

Acknowledgements NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT centre of Excellence programme. DN has also been supported by a grant from the Institute of Museum and Library Services, and a Google Research Award.
Authors are thankful to Timothy Baldwin for valuable discussions.

References

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

W. L. Buntine and A. Jakulin. Applying discrete PCA in data analysis. In *Proceedings of the 20th Uncertainty in Artificial Intelligence Conference*, pages 59–66, Banff, Canada, 2004.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. volume 6, pages 721–741, November 1984.

D. Grangier and S. Bengio. Inferring document similarity from hyperlinks. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 359–360, Bremen, Germany, 2005.

T. Griffiths and M. Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences*, volume 101, pages 5228–5235, 2004.

T. Griffiths and M. Steyvers. Probabilistic topic models. In *Latent Semantic Analysis: A Road to Meaning*, 2006.

F. Kaiser, H. Schwarz, and M. Jakob. Using Wikipedia-based conceptual contexts to calculate document similarity. In *Proceedings of the 2009 Third International Conference on Digital Society*, pages 322–327, Cancun, Mexico, 2009.

M. D. Lee, B. Pincombe, and M. Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 1254–1259, Mahwah, NJ, 2005.

Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of The 30th International Conference on Knowledge Discovery and Data Mining*, pages 490–499, 2007.

S. Walker, S. Robertson, M. Boughanem, G. Jones, and K. Sparck Jones. Okapi at TREC-6 automatic ad hoc, VLC, routing, filtering and QSDR. In *Proceedings of the 6th Text REtrieval Conference*, pages 125–136, 1997.

H. M. Wallach, I. Murray, R. Salakhutdinov, and D. M. Mimno. Evaluation methods for topic models. In *Proceedings of The 26th International Conference On Machine Learning*, pages 1105–1112, Quebec, Canada, 2009.

Id - Dynamic Views on Static and Dynamic Disassembly Listings

Nicholas Sherlock
Computer Science
University of Otago
Otago 9010 New Zealand
n.sherlock@gmail.com

Andrew Trotman
Computer Science
University of Otago
Otago 9010 New Zealand
andrew@cs.otago.ac.nz

Abstract *Disassemblers are tools which allow software developers and researchers to analyse the machine code of computer programs. Typical disassemblers convert a compiled program into a static disassembly document which lists the machine instructions of the program. Information which would indicate the purpose of routines, such as comments and symbol names, are not present in the compiled program. Researchers must hand-annotate the disassembly in a text editor to record their findings about the purpose of the code.*

Although running programs can change their layout dynamically, the disassembly can only show a snapshot of a program's layout. If a different view of a program is required, the document must be recreated from scratch, making it difficult to preserve user annotations.

In this paper we demonstrate a system which allows a disassembly listing to be refined by user input while retaining user annotations. Users are able to dynamically change the interpretation of the layout of the program in order to effectively analyse programs which can alter their own memory layout. We allow users to combine the independent analysis of several program modules in order to examine the interaction between modules.

By exploring the obsolete "Poly" computer system, we demonstrate that our disassembler can be used to reconstruct and document entire software distributions.

Keywords Digital Libraries, Cognitive Aspects of Documents, Document Workflow

1 Introduction

The rate of computer hardware and software development is increasing exponentially. Five years ago, our desktop computers were all powered by single-core CPUs. Two years ago, they had dual-core CPUs. And today, they are likely to have four or eight cores. The Macintosh series of computers have seen large architectural changes, switching from Motorola CPUs to PowerPCs and finally to Intel x86 CPUs. In successive steps, we have changed our removable storage media from tapes, to 8, 5¼ and 3½ inch floppy disks, to CD-ROMs, DVDs, Blu-ray, and increasingly, removable flash-memory based storage. With each new hardware generation, our old software becomes obsolete and is either rebuilt or abandoned.

This creates a problem for researchers and historians. While design manuals can be scanned and stored accessibly in a digital library, and data can be retrieved from old media with somewhat more effort and expense, storing the software from these old machines in a useful format is an entirely different problem. Performing analysis on software which is stored in the library becomes increasingly difficult with time. This is because a piece of software cannot be used, examined, or understood in isolation. Its behaviour is defined by its interaction with the hardware it was built for. Obsolete hardware becomes progressively more scarce with time. Preserving old hardware by building modern replicas requires an increasingly infeasible amount of effort and resources as microelectronics become more complex.

If an accurate description of the hardware is provided or can be discovered, it can be replaced by a software-based "emulator". An emulator in this context is a program which simulates the action of an old hardware platform on (typically many) modern platforms. In this way, researchers can examine the runtime behaviour of old software without having to perform a costly hardware reconstruction of an old platform.

A second problem is the difficulty of examining the algorithms and implementation details of obsolete software when human-readable source code has been lost or was never provided. It is also a problem for modern software. For example, in order to build a new program which interoperates with an existing program, some knowledge of the original program's internal operation is required. Even if the source code for a program is available, you may still want to examine the machine instructions that the compiler generates to ensure that the generated instruction sequences are correct or efficient. Machine code is a more primitive level of abstraction which can reveal surprising negative performance implications of innocuous-looking high-level code.

If only the machine code that makes up the compiled program is available, it must first be translated

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

into a more abstract form that humans can understand so that it can be analysed. A program which performs this translation is called a "disassembler". Much of the information found in the high-level source code of a program, including comments and the names of variables and routines, is lost in the compilation process. To understand a compiled program, a researcher must recover this lost information. They can achieve this by inspecting the disassembly listing with reference to the behaviour of the running program. They can then share their findings with other researchers by annotating the disassembly.

Several factors make this process difficult. The disassembler's interpretation of the program must be dynamically altered to analyse programs which can change their layout at runtime. In order to change the interpretation of the program, the disassembler must be re-run. This creates a new, independent disassembly document, which makes it difficult to preserve annotations that the researcher has already made.

Even if a program does not change its layout dynamically, the researcher must still frequently change the disassembler's interpretation of the program. This is because the disassembler cannot distinguish code from data in the compiled program with perfect accuracy. Human judgements are required to correct the disassembler's mistakes.

In order to allow researchers to effectively document entire programs, software support is required to assist the user in navigating and imposing structure on large disassembly documents, but this is not provided with a traditional disassembler. Although programs being analysed are often composed of several related modules which can be examined independently, disassemblers typically do not provide any way of linking disassemblies together in order to share information about interacting modules.

In this paper, we will present our disassembly, debugging and emulation system which we used to reconstruct and document the software and hardware of the "Poly" computer system. We will show that our disassembler can solve the problems inherent in documenting the software of the Poly by using it to create a digital Poly software library which researchers will be able to examine long into the future.

2 The Poly computer system

The Poly was a computer system developed in New Zealand in the early 1980s. It was comprised of a server computer called the "Proteus" with a series of fat-client "Poly" machines attached by a token ring network. It was designed to be used in a classroom setting where a teacher would set work on the server computer to be distributed to each student's computer. When the students finished their work, their results would be sent back to the server computer to be saved to disk. The server and the client machines had similar architectures. In one prototype, a client could be turned into a Proteus server with the addition of a disk drive. The computers can be seen in Figure 1.

The Poly never gained much ground in the computer market and few machines were produced. Although the Poly demonstrated innovative technologies and ideas, and is an important part of New Zealand's computing history, little is now known about it. In particular, the Poly's networking capabilities were far ahead of contemporary computers, and it was provided with innovative classroom software to take advantage of those features. But with only a couple of working Polys in existence and little surviving documentation, the exact functionality of the software is largely a mystery.

In order to make the Poly's software available to researchers, we would have to document it in a form that would be useful long after the last Poly stops operating.

3 Disassembly

```
48A6  34 14        PSHS X,B ;Ref from $CD39
48A8  8E 5B 19     LDX #$5B19
48AB  C6 05        LDB #$5
48AD  E7 80        STB ,X+
48AF  35 04        PULS B
48B1  E7 80        STB ,X+
48B3  35 20        PULS Y
48B5  EC A4        LDD ,Y
48B7  ED 84        STD ,X
48B9  8E 5B 19     LDX #$5B19
48BC  10 8E 00 04  LDY #$04
```

Figure 2: A fragment of a disassembly listing

A tool called a "disassembler" examines a program binary (that is, the machine code that the computer will execute, not the source code which is used to generate it) and creates a text file called a disassembly listing. A disassembly listing shows the machine code instruction that appears at each memory address within the program as a human-readable mnemonic code. It also shows the data stored inside the program, such as the text of string literals or numeric literals from the source code.

Figure 2 shows a fragment of a program disassembly for the Poly's Motorola 6809 CPU[8]. The leftmost element is the memory address of the disassembled instruction. Next is a hexadecimal representation of the machine code that the CPU will execute. Finally, a human-readable interpretation of the machine code is displayed. The first part of the instruction is a mnemonic which represents the instruction being performed (for example, PSHS is an instruction to push a value onto the stack). Any arguments to the instruction follow the mnemonic. X, B and other symbols refer to registers on the CPU and values starting with a hash symbol are numeric literals. There is effectively a one-to-one mapping between the machine code and the mnemonic representation shown to the researcher.

(a) Two Poly client machines sit side-by-side

(b) A Proteus server and its CPU and memory board (inset)

Figure 1: The key components of the Poly system

There are two major difficulties in building a useful disassembler program. The primary difficulty is that it is impossible in general to automatically decide which parts of the program binary are data and which parts are code which will be executed. This problem is equivalent to the halting problem[5]. Because of this, disassemblers must sometimes guess where a machine instruction begins in memory and so will make some incorrect guesses. Wrong guesses might identify the beginning of a sequence of instructions at the wrong offset (so that the interpretation of the sequence begins halfway through a machine instruction, generating incorrect output,) or incorrectly identify data as code or vice versa, which hampers correct interpretation of the program. Some code locations can not be identified because their addresses are computed at runtime by the program in a way that the disassembler cannot predict. For example, a program may read the address of the routine to execute from an external file.

The second difficulty is encountered when analysing software that was built for small systems like the Poly. Like many computers of its time, the Poly had more physical memory available than it could simultaneously address. Its CPU's memory address bus is 16-bits wide, allowing it to address 64kB of virtual memory at any one time. The Poly has 128kB of physical RAM plus 8kB of BIOS and memory-mapped peripherals. Software on the Poly dynamically changes the mapping of the 8kB virtual memory pages to the 128 + 8kB physical address space by changing the entries in a memory map.

In Figure 3, a 16-bit virtual memory address is translated into a 17-bit address in physical memory in a series of steps. In "protected" mode (operating system mode), some addresses are directed to hardware and the BIOS. Otherwise, the three most-significant bits of the virtual address are combined with a bank-select bit and used as an index into the programmable memory map. The memory map replaces the three higher bits of the virtual address with four bits of its own, creating a 17-bit address in physical memory.

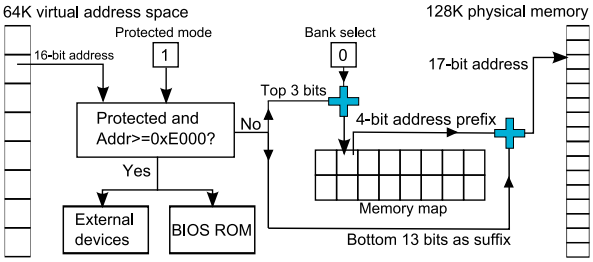


Figure 3: A virtual memory address is translated using the memory map into a physical address

With a traditional disassembler, the researcher would have to disassemble the program once for every memory mapping they wanted to examine, and maintain the different disassembly listings independently, even when information should be shared between them. The same physical memory page can even appear in virtual memory in more than one place simultaneously, making it difficult to manually keep annotations consistent and up to date.

Many small-CPU based systems, including systems of about the Poly's age, use dynamic memory maps to overcome the limitations of restrictively small address spaces. For example, the Apple //e[9], ZX Spectrum 128[1] and Commodore 128[6], which, like the Poly, have 16-bit address busses and can support 128kB or more of memory. Software written for 16-bit operating systems such as MS-DOS on more modern PCs or software for embedded systems also use this technique. To effectively analyse these systems, a new kind of disassembler is required.

4 Background

Disassemblers are available for nearly every platform. Disassembly tools are available in two main contexts: As a static disassembler tool to examine stored programs on disk, or dynamic disassemblers which examine snapshots of running programs.

4.1 Static analysis

The tool "objdump"[2] typifies static disassembly tools. A binary program on disk is provided as input to `objdump`, and the output is a disassembly listing document.

The generated listing may be explored and annotated with a simple text editor, but this approach has two serious disadvantages. Firstly, a text editor treats the disassembly as unstructured text and so can offer very little software support for common annotation operations. It will not offer cross-referencing support, so any references that the code makes to other parts of the program must be followed manually. If the analyst gives a descriptive label to a block of code, that label will not be propagated to the places where the code is called. The analyst cannot effectively experiment with different interpretations for data stored in the program. For example, to reinterpret a number as signed or unsigned will require the researcher to manually convert the number using some other tool, or re-run the disassembler to create an entirely new disassembly. These problems dramatically slow down analysis and make understanding the program much more difficult.

Secondly, a static disassembler cannot always correctly distinguish code from data in the analysed program. For example, the code may include a jump whose target is an address which is computed at runtime. This is common in object-oriented code, where the address of a virtual method must be looked up in an object's virtual address table. In procedural code, this technique is more likely to be used with a jump table—a table of routine addresses that selected from at runtime, often by an equivalent of the "switch" statement in C. Data-flow analysis techniques could be used to discover the targets for some of these computed jumps[4]. For instance, the instruction sequence `LDX #0x5B19 / JMP X` (storing the value `0x5B19` into the register `X`, followed by a jump to the value stored in `X`) is clearly a jump to the location `0x5B19`. Even so, some jumps are computed in a way that no disassembler could possibly understand (e.g. by using data available at runtime which is not present in the image being disassembled, such as data contained in a message received on the network.)

The disassembler might identify a jump with a known target which in fact points to data, not code. If the Poly jumped to that location, it would likely have unexpected results, perhaps crashing. If we assume that the the Poly code does not crash, it is reasonable to assume that it does not take bad jumps. There are at least two possible causes of this situation.

It may be impossible for the flow of execution to ever reach the jump, so the bad jump is never executed in practice. For instance, a program might check the state of a "debugging mode flag", and, based on the value it finds, jump to some logging routine which ended up being cut from the final binary. The debugging mode flag is never set in delivered software so the bad jump is never taken.

The jump may have a definite target, and be taken at runtime, but the target of the jump which is stored in the instruction is overwritten at runtime before the jump is ever called. This is seen on modern architectures. A module of code (such as a Windows dynamic-link library or a Unix shared object) which a program uses may be dynamically loaded at an unpredictable position in its address space. To be able to call routines from the module, the program needs to know their addresses. To achieve this, an "import table" is generated in the application. The import table consists of a series of stubs. The stubs are small routines which contain a jump to an address which is initially some default value (`NULL`). When the library is loaded, the memory locations of its routines are discovered and used to rewrite the code in the import table. To call an imported routine from within the program, a call to the stub is made some time after the library is loaded. Calling the routine before the library is loaded results in undefined behaviour.

In order to correct code which has been misidentified as data, or vice-versa, the user must run the disassembler again with that new information. This produces an entirely independent disassembly listing which must then be manually merged with the listing the user has annotated. This is an error-prone and tedious process. The user is unlikely to want to experiment with different interpretations of a memory address, because each experiment is so costly to run in terms of user effort.

4.2 Dynamic analysis

A debugger like the free tool "gdb"[7] is designed to allow the user to inspect and interact with running programs. If no debugging information or source code is provided which would allow it to show the high-level code that corresponds to the running machine code, it uses an embedded disassembler to show the disassembly of the code that is currently executing.

This approach has several advantages. Code can be distinguished from data with certainty, since the debugger only needs to show the disassembly for instructions which are currently executing or have previously executed. The user can have the debugger interpret any memory location in multiple ways. For example, they could view one location as both an array of integers and an array of characters, and discover that the data only makes sense when interpreted as an array of integers.

The analyst can interact with the running program to see what inputs a piece of code receives, or precisely what action it takes as a result. The running program may be modified by the analyst to explore areas of code that would not normally execute. For example, they can force the code to follow an error-handling branch in order to examine that mechanism, even if they do not know what inputs to the program are needed to cause the error to be triggered in normal execution.

The main disadvantage of this approach is that the user is typically unable to add any annotations to the disassembly. If they discover the purpose of a routine,

they cannot give it a human-readable label which would allow it to be understood the next time it is encountered. Even if annotations are supported, the debugger will not provide any way to save them and load them again later, since it has no expectation that the memory layout of the program will be similar the second time the program is run. Analysis with a debugger is ephemeral, it cannot be effectively used to produce a document which could record the user's findings to be shared with other researchers.

4.3 Interactive disassembly

Traditional disassemblers are frequently used in situations where the disassembly is only useful for a short amount of time, like a single session, and saving annotations is less important. For example, a common task for a traditional disassembler is examining the machine code generated by a procedure in a high-level language to diagnose performance or code generation issues. Since they are typically used to examine a program which is currently in development (and therefore changing dramatically from a machine code perspective), the ability to save annotations is not valuable.

If a disassembly listing is to be modified and examined over an extended period of time (i.e. several analysis sessions), or shared with other people, it must be able to change dynamically as more information is discovered by a human researcher. The researcher will work *with* the disassembler to analyse a program. This is the approach that we decided to take with our own disassembler.

The only interactive disassembler that we are aware of in common usage is IDA[3]. But IDA does not support the dynamic memory model of the Poly. While it supports debugging live code for some targets, it does not integrate with our Poly emulator.

5 Id, the interactive disassembler

To assist our reconstruction of the Poly platform, we developed "Id", an interactive disassembler which supports the Poly's CPU and binary layouts. Id is an application for Windows with a Graphical User Interface. The main pane of Id is the disassembly listing. Surrounding the listing are panels that give extra information about the binary being disassembled. For instance, one panel is a list of all of the symbol names created so far in the image. Id can be seen in Figure 4.

To support the changing layout of programs on the Poly, all of the items that Id identifies in its disassembly are tagged with the physical address that they are stored at, not the virtual memory addresses that they appear at with one possible memory mapping. This allows the user to change the virtual memory map while they are examining the disassembly, and have the disassembly listing change dynamically to reflect the new interpretation of the program's layout. This approach works well with the Poly because code and resources on this platform typically have a fixed location in physical memory. On systems with address spaces larger than the amount of physical memory available, like modern 32- and 64-bit computers, the reverse tends to be true. Programs move around in physical memory but stay in a fixed position in virtual memory.

Initially, no code has been identified in the image (it is all considered to be data). To begin the disassembly process, you must identify the start of a machine instruction in the image. The CPU has to perform the exact same task when the Poly boots. The CPU begins by reading an address from an interrupt table at a fixed location in memory, which is called the "reset vector". This is the location where execution begins after boot. Id begins disassembly at this point. If the instruction at the reset vector is a jump to a different location, Id can follow the jump and recursively identify code there. If the instruction is not a jump, execution will continue to the next location in memory, so Id identifies an instruction there. Following jumps from the entry points defined in the interrupt table identifies much of the code in the image—around 60% for the Poly's BIOS. The remainder of the code is often interrupt handlers whose address is determined at runtime in a fashion that is currently too difficult for Id to discover.

To assist Id, the user can create new entry points (the beginnings of instruction sequences) at any time. Id automatically adds disassembly for those previously-unidentified locations to the disassembly listing. If the disassembler has wrongly identified data as code, the user can convert it back to data.

While the physical representation of the data in the program is known from the disassembly, the information that the data encodes cannot always be inferred automatically. For example, Id knows that the operands of instructions which specify the targets of read or writes to memory locations are memory addresses. It can use syntax colouring to distinguish these addresses from other numeric literals found in the program. When possible, the symbolic name that the user has given to the target address is shown in place of the raw address.

However, operands to instructions which are merely stored into registers or into memory locations have no special meaning defined by the instruction set. Id allows the user to experiment with different interpretations of the data in order to discover what information the data is encoding. For example, Id can interpret data as strings, arrays, addresses, or numeric types of various sizes and formats. As even simple operations such as adding a constant to a number stored in a register can have multiple reasonable meanings, this user-directed assistance is crucial to documenting the purpose of the program. For example, the machine code and effect of subtracting 16 from a number stored in a register is identical to that of adding 65520 (`0xFFF0` in hexadecimal notation). But these two operations suggest very different purposes for the code being disassembled. In the first case, the program may be accessing data that appears immediately before a previously-computed ad-

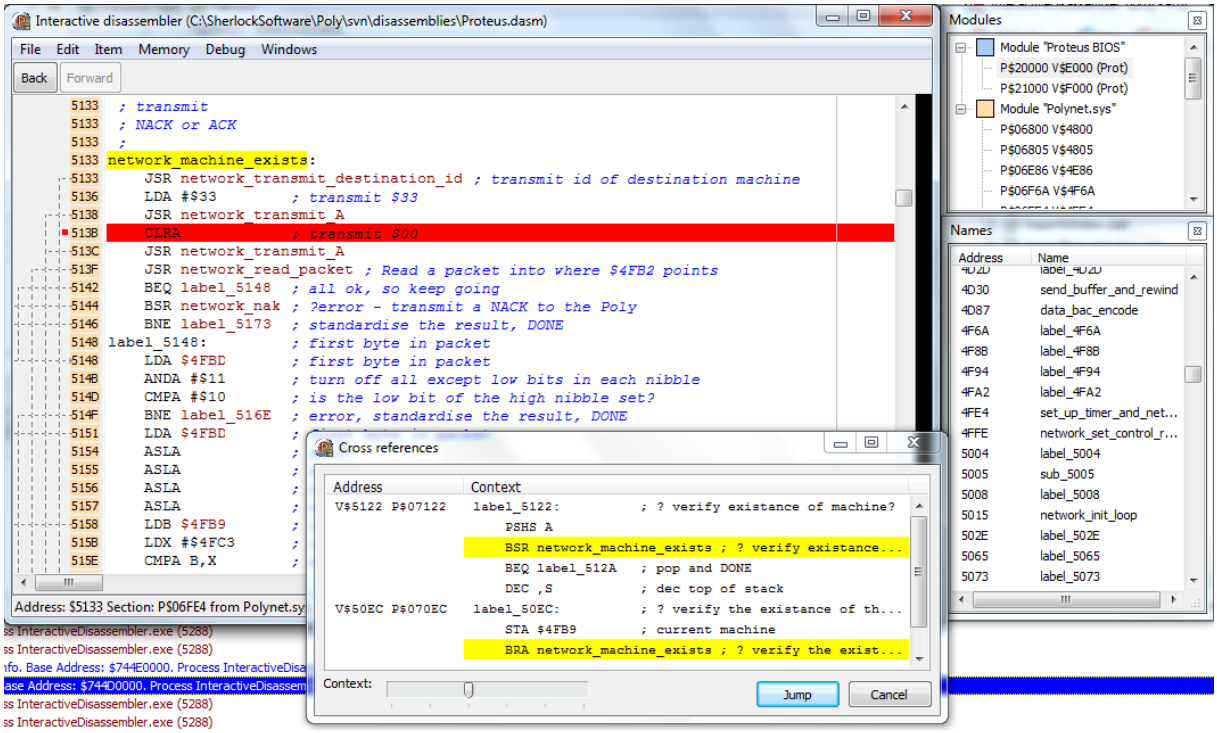


Figure 4: Id's main GUI with the Proteus operating system loaded for disassembly. Andrew Trotman's comments appear after semicolons. All names in the disassembly are user-entered.

dress. This is the pattern expected if the program is iterating over an array of elements whose size is 16-bytes in reverse order. In the second case, the code may be calculating the location of a dynamically-determined field of a structure which is located at the fixed address `0xFFF0`. This is the pattern expected if the program is looking up the current location of one of the installed interrupt routines (as the interrupt-vector table is located at `0xFFF0`). By specifying the signedness of the operand, the user can disambiguate these two cases.

5.1 Annotation

Id is a hypertext document system. The start of each routine or piece of data can be given a title by the user. Id then provides a "names" pane, which lists every title in the program, sorted by module. This structured outline allows the document to be navigated rapidly.

References to memory locations found in the code, such as the targets of jumps or the targets of memory reading instructions, are shown as hyperlinks. The user can double click on the hyperlink to jump directly to its target. If there is a user-supplied title at the target location, it is shown as the anchor text in the disassembly in preference to the target's raw memory address. The user can further describe the purpose of a location by adding an extended comment to the title. This extended comment becomes the default comment which appears automatically at all referral sites. The user can edit the comment at a referral site in order to record the exact way that the target is being used. For example, the researcher might label a memory location as "num_clients", and provide the extended comment "number of Poly clients currently connected" to clarify the meaning of the location. A comment at a referring site which increments this value might then be customised: "record newly-attached client".

As the user adds labels to locations in the memory map, the propogation of these labels to referring sites makes the meaning of previously unexplored code clearer. The analysis process shows a "jigsaw-like" effect. As with a jigsaw, a series of interlinked pieces meet at common interfaces. As the jigsaw is completed, the possible shape and location of the unplaced pieces is futher and further constrained. This implies that the initial analysis—discovering the large-scale structure of the program—is the most difficult phase, with analysis becoming more rapid as the final "pieces" are placed.

Id's hyperlinks are bidirectional. The user can select a title and discover all of the links which point to it by using Id's "cross-references" window. Id helps the user choose interesting referrers for further analysis by showing some context from each incoming link (the closest few instructions and comments). By examining a routine's referrers, the researcher can determine (by trial and error) what kind of inputs will be provided to the routine and what sort of return value is expected in response. Examining referrers is critical to discovering the purpose of a target routine or memory address.

If the memory map changes, the names of the targets of links in the code are updated accordingly. In protected mode, a call to a routine at the address `0xF030` might send instructions to hardware to print

text to the display. But in unprotected mode, a different routine will reside at that address. It could also be a line-printing routine, but it might first copy the user's text to a buffer area before switching to protected mode to call the BIOS's line-printing routine. Id allows calls to these two routines to be distinguished by allowing them to have different names.

Broken hyperlinks, which are links that point to locations which do not currently exist in the disassembly, are highlighted for the user. The presence of a broken link could indicate that the target has not yet been loaded from disk, or that the memory map is expected to change before the link will be accessed by the program. This highlighting is particularly valuable for discovering the connections between loadable modules of code.

While the user can instruct Id to reinterpret a location of the disassembly, the text of the disassembly listing itself cannot be directly edited by the user. This allows Id to ensure that the disassembly listing always corresponds precisely to the machine code. In fact, the listing could be used to reconstruct the original binary program. Id does allow comments to be added to any line of the disassembly as the purpose of routines are discovered. Once a piece of code has been understood, adding a comment allows the purpose of the sometimes confusing assembly to be shared with other researchers.

5.2 Modules

Disassembly will typically be performed on a "module". A module is a set of code and data which is tightly bound together. For instance, one module might be the Poly's BIOS, which is the code that controls the interaction between the hardware and the software. This code is stored in permanent memory chips on the motherboard and appears in the Poly's virtual address space when it enters "protected mode". Another module might be a boot sector read from a floppy disk. Id understands the format of the Poly's file system and programs, so it can simulate the action of the Poly's program loader and load a program from disk as a module into the correct physical memory locations for analysis.

There are substantial cross-references between modules, so the analysis of one module can be used to understand a different module. For example, the BIOS is responsible for loading the boot sector from a floppy disk, then it transfers control to the boot sector program. By disassembling the BIOS, the entry-point of the boot sector's code can be discovered. In a traditional system, the disassembly document of each module is independent so there is no inter-module linking.

To support the analysis of large systems, Id's module system allows multiple disassemblies which were created independently to be composed to appear as one coherent document. For instance, one disassembly might be the *Proteus's* operating system and BIOS. Another disassembly might be the *Poly's* operating system and BIOS. If you created a disassembly of a text editor program, you could choose to either link to the disassembly of the Poly's operating system (to examine the program's effect on the Poly), or link with the Proteus's operating system (to examine the program's effect on the Proteus.) This linking can be easily changed while you are disassembling. The gutter of the disassembly pane is colour-coded to show the module that a line of code belongs to. Cross-module references are dynamically resolved based on the current selection of modules. New information identified about linked modules is automatically used to update those documents when the parent document is saved. This system allows reuse of disassembly information—the information you discover about the operating system while analysing a text editor program is made available when examining the interaction between a database program and the operating system.

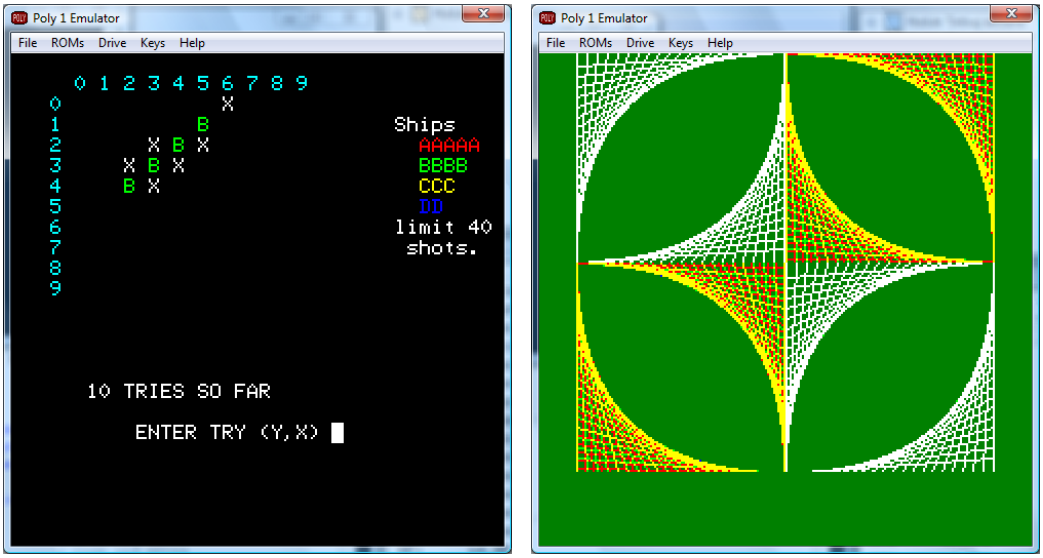
The module system allows the user to document an entire operating system and its attendant application suite as a set of linked disassemblies.

5.3 Debugging

A goal of Id was to unify the capabilities of static and dynamic disassemblers. Id includes a debugger which can attach to a running instance of our Poly emulator. This allows the runtime behaviour of a disassembled program to be examined. It also allows the creation of new disassemblies based on code which is loaded in memory at runtime from unknown sources. For example, the applications on the client Poly machine are loaded from the server across the network. Using the debugger, a copy of a network-loaded application can be dumped from the client for later analysis.

The debugger integrates with the disassembler tightly. For example, the debugger observes the instructions which are executed at runtime in order to discover the location of code in the image which it could not have discovered with a static analysis of the program on disk. The identified code is automatically merged into the document that the user has already created. The user is notified if the newly-identified code is inconsistent with previous disassembly. For example, if a newly-identified instruction lies within a previously-defined instruction (which is vanishingly rare in valid code), the user is asked to decide which interpretation to accept.

By using the debugger, the user can discover cross-references at runtime which cannot be discovered with static analysis. For example, the user can set breakpoints which pause execution when a certain memory location is read to or written from. When the breakpoint is triggered, the user has identified a link which references their memory address. This is particularly useful in debugging the behaviour of hardware devices (which appear at virtual memory addresses in the Poly's memory map).



(a) Emulated text-mode battleships game on the Poly client

(b) Emulated graphics-mode on the Poly client

Figure 5: Emulated Poly machines

In Figure 4, the debugger has been attached to an emulator which is simulating the Proteus server. A breakpoint has been placed in part of a network routine. Execution will automatically pause at that point if the Proteus executes that instruction, allowing the user to inspect the contents of memory and the CPU registers. In Figure 5, two Poly clients with attached debuggers are executing programs delivered by the Proteus server.

6 Conclusions

We used the dynamic document features of our interactive disassembler system, Id, in our analysis and reverse engineering of the Poly. By using it, we were able to investigate the Poly's networking code in order to to solve timing and hardware issues in our emulator. Our Poly emulator is now able to successfully emulate a Poly client attached to an emulated Proteus server.

The disassemblies that we created with Id will form the basis of a online library of information about the Poly's software and hardware. Because Id allows researchers to link and share information between their new disassemblies and our existing disassemblies of the operating system, BIOS, and other system utilities in our library, analysis of the Poly system can be achieved more rapidly and easily than with any other competing disassembly system.

The documents produced all contribute to a long-lasting store of knowledge about the Poly. The disassemblies are in a format that is readable even without using Id. They consist of plain-text, human-readable disassembly listings similar to what Id displays in its main pane, stored inside "zip" compressed archives. This ensures that the information discovered with Id will be accessible long into the future.

Our disassembler's Poly-specific knowledge is segmented from its dynamic document engine, so it is relatively easy to add support for more processors and architectures. This makes Id a very flexible tool for the disassembly and documentation of small systems.

References

[1] Various authors. *128K ZX Spectrum Technical Information*. World Of Spectrum, `http://www.worldofspectrum.org/faq/reference/128kreference.htm`, 2009. Accessed 17th September, 2009.

[2] Inc. Free Software Foundation. *GNU Binutils*. `http://www.gnu.org/software/binutils/`, 2008. Accessed 9 October, 2008.

[3] Ilfak Guilfanov. *IDA Pro Disassembler—multi-processor, windows hosted disassembler and debugger*. Hex-Rays, `http://www.hex-rays.com/idapro/`, 2009. Accessed 17th September, 2009.

[4] Matthew S. Hecht. *Flow Analysis of Computer Programs*. Elsevier Science Inc., New York, NY, USA, 1977.

[5] R. N. Horspool and N. Marovac. An approach to the problem of detranslation of computer programs. *The Computer Journal*, Volume 23, Number 3, pages 223–229, 1980.

[6] Lance Lyon. *Commodore 128 Alive!* Commodore 128, `http://www.commodore128.org`, 2009. Accessed 17th September, 2009.

[7] The GNU Project. *GDB: The GNU project debugger*. `http://www.gnu.org/software/gdb/`, 2009. Accessed 15 September, 2009.

[8] T. Ritter and Boney J. The 6809. *Byte Magazine*, 1979.

[9] Steven Weyhrich. *Apple][History Chap 7*. Apple 2 History, `http://apple2history.org/history/ah07.html`, 2009. Accessed 17th September, 2009.

Interestingness Measures for Multi-Level Association Rules

Gavin Shaw
School of Information Technology
Queensland University of Technology
Brisbane QLD Australia
g4.shaw@student.qut.edu.au

Yue Xu
School of Information Technology
Queensland University of Technology
Brisbane QLD Australia
yue.xu@qut.edu.au

Shlomo Geva
School of Information Technology
Queensland University of Technology
Brisbane QLD Australia
s.geva@qut.edu.au

Abstract *Association rule mining is one technique that is widely used when querying databases, especially those that are transactional, in order to obtain useful associations or correlations among sets of items. Much work has been done focusing on efficiency, effectiveness and redundancy. There has also been a focusing on the quality of rules from single level datasets with many interestingness measures proposed. However, with multi-level datasets now being common there is a lack of interestingness measures developed for multi-level and cross-level rules. Single level measures do not take into account the hierarchy found in a multi-level dataset. This leaves the Support-Confidence approach, which does not consider the hierarchy anyway and has other drawbacks, as one of the few measures available.*

In this paper we propose two approaches which measure multi-level association rules to help evaluate their interestingness. These measures of diversity and peculiarity can be used to help identify those rules from multi-level datasets that are potentially useful.

Keywords Information Retrieval, Interestingness Measures, Association Rules, Multi-Level Datasets

1 Introduction

Association rule mining was first introduced in [1] and since then has become both an important and widespread tool in use. It allows associations between a set of items in large datasets to be discovered and often a huge amount of associations are found. Thus in order for a user to be able to handle the discovered rules it is necessary to be able to screen / measure the rules so that only those that are interesting are presented to the user. This is the role interestingness measures play. In an effort to help discover the interesting rules, work has focused on measuring rules in various ways from both objective and subjective points of view [3] [8]. The most common measure is the support-confidence approach [1] [2] [6], but there are numerous other measures [2] [3] [6] to name a few. All of these measures were proposed for association rules derived from single level or flat datasets, which were most commonly transactional datasets. Today multi-level datasets are more common in many domains. With this increase in usage there is a big demand for techniques to discover multi-level and cross-level association rules and also techniques to measure interestingness of rules derived from multi-level datasets. Some approaches for multi-level and cross-level frequent itemset discovery (the first step in rule mining) have been proposed [4] [5] [10]. However, multi-level datasets are often a source of numerous rules and in fact the rules can be so numerous it can be much more difficult to determine which ones are interesting [1] [2]. Moreover, the existing interestingness measures for single level association rules can not accurately measure the interestingness of multi-level rules since they do not take into consideration the concept of the hierarchical structure that exists in multi-level datasets. In this paper as our contribution we propose measures particularly for asessing the interestingness of multi-level association rules by examining the diversity and distance among rules. These measures can be determined during rule discovery phase for use during post-processing to help users determine the interesting rules. To the authors' best knowledge, this paper is the first attempt to investigate the interestingness measures focused on multi-level datasets.

The paper is organised as follows. Section 2 discusses related work. The theory, background and assumptions behind our proposed interestingness measures are presented in Section 3. Experiments and results are presented in Section 4. Lastly, Section 5 concludes the paper.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

2 Related Work

For as long as association rule mining has been around, there has been a need to determine which rules are interesting. Originally this started with using the concepts of support and confidence [1]. Since then, many more measures have been proposed [2] [3] [6]. The Support-Confidence approach is appealing due to the antimonotonicity property of the support. However, the support component will ignore itemsets with a low support even though these itemsets may generate rules with a high confidence (which is used to indicate the level of interestingness) [6]. Also, the Support-Confidence approach does not necessarily ensure that the rules are truly interesting, especially when the confidence is equal to the marginal frequency of the consequent [6]. Based on this argument, other measures for determing the interestingness of a rule is needed.

Broadly speaking, all of these existing measures fall into three categories; objective based measures (based on the raw data), subjective based (based on the raw data and the user) and semantic based measures (based on the semantic and explanations of the patterns) [3].

In the survey presented in [3] there are nine criteria listed that can be used to determine if a pattern or rule is interesting. These nine criteria are; conciseness, coverage, reliability, peculiarity, diversity, novelty, surprisingness, utility and actionability or applicability. The first five criteria are considered to be objective, with the next two, novelty and surprisingness being considered to be subjective. The final two criteria are considered to be semantic.

Despite all the different measures, studies and works undertaken, there is no widely agreed upon formal definition of what interestingness is in the context of patterns and association rules [3]. More recently several surveys of interestingness measures have been presented [3] [6] [7] [8]. One survey [8] evaluated the strengths and weaknesses of various measures from the point of view of the level or extent of user interaction. Another survey [7] looked at classifying various interestingness measures into five formal and five experimental classes, along with eight evaluation properties. However, all of these surveys result in different outcomes over how useful, suitable etc., an interestingness measure is. Therefore the usefulness of a measure can be considered to be subjective.

All of these measures mentioned above are for rules derived from single level datasets. They work on items on a single level but do not have the capacity for comparing different levels or rules containing items from multiple levels simultaneously. Our research has found that up to now, little work has been done when it comes to interestingness measures for multi-level datasets that can handle items from muliple levels within one rule or rule set.

Here in our work we propose to measure the interestingness of multi-level rules in terms of diversity and peculiarity (also known as distance). These measures were chosen as they are considered to be objective (rely on just the data).

3 Concepts and Calculations of The Proposed Interestingness Measures

In this section we present the key parts of the theory and background and formula behind our proposed measures. We also present the assumptions we have made for our measures.

3.1 Assumptions and Definitions

Here we outline the assumptions we have made. Figure 1 depicts an example of the general structure of a multi-level dataset. As shown, there is a tree-like hierarchical structure to the concepts or items involved in the dataset. Thus items at the bottom are descendant from higher level items. An item at a higher level can contain multiple lower level items.

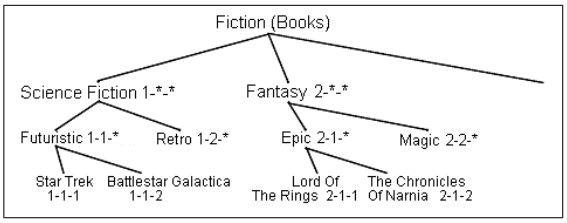


Figure 1: Example of a multi-level dataset.

With this hierarchy we have made the two following assumptions.

1. That each step in the hierarchy tree is of equal length / weight. Thus the step from 1-*-* to 1-1-* is of equal distance to the step from 2-*-* to 2-1-* or 1-1-* to 1-1-1.

2. That the order of sibling items is not important and the order could be changed (along with any descendants) without any effect.

3. That each concept/item has an ancestor concept/item (except for the root) so that no concepts/items or group(s) of concepts/items are isolated from the rest of the hierarchy.

Before presenting our proposed measures we firstly define several terms and formula used.

- n_1 and n_2: represent two items / concepts in the multi-level dataset.
- *ca*: (common ancestor) is the closest item that is an ancestor to both n_1 and n_2.
- *TreeHeight*: is the maximum number of items on a path in the multi-level dataset (not counting root) from the root to a item located at the lowest concept level.
- *h*: represents the entire multi-level dataset hierarchy.

- *Hierarchy level of an item*: the hierarchy level of the root is 1. The hierarchy level of an item in the dataset is larger than the level of its direct parent by 1.
- *Number of Levels Difference*:

$$NLD(x,y) = |\, hierarchy\ level\ of\ x - hierarchy\ level\ of\ y\,| \quad (1)$$

is the number of hierarchy levels difference between items *x* and *y*.

3.2 Diversity

Here we define a diversity measure for multi-level association rules which takes items' structural information into consideration. The diversity defined here is a measure of the difference or distance between the items within a rule, based on their positions in the hierarchy. Two different aspects of the items in a rule are considered to measure the diversity of the rule.

1. Hierarchical relationship distance (HRD) between items.
2. Concept level distance (LD) between items.

We propose that the diversity of a rule can be measured using two different approaches. The first, measures the overall diversity of a rule by combining the items in the antecedent with the items in the consequent into a single set. If the items within this combined itemset are very different, then the rule will have a high overall diversity, regardless of whether the items were from the antecedent or consequent.

Let *R* be a rule with *n* items and D_{OR} denotes the overall diversity of *R*, the diversity of *R* can be determined as follows:

$$D_{OR} = \frac{\alpha_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} HRD(i,j)}{n(n-1)} + \frac{\beta_1 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} LD(i,j)}{n(n-1)} \quad (2)$$

The second, measures the diversity between the items in the antecedent and those in the consequent. Those rules which have a high difference between their antecedent itemsets and consequent itemsets will have a high antecedent-consequent diversity. However, this approach does not consider the difference between items within the antecedent and/or consequent like the overall diversity approach.

Let *R* be a rule $R : A \rightarrow C$, with *n* items in *A* and *m* items in *C* and D_{ACR} denotes the antecedent to consequent diversity of *R*, the diversity of *R* can be determined as follows:

$$D_{ACR} = \frac{\alpha_2 \sum_{i=1}^{n-1} \sum_{j=1}^{m} HRD(i,j)}{n(n-1)} + \frac{\beta_2 \sum_{i=1}^{n-1} \sum_{j=1}^{m} LD(i,j)}{n(n-1)} \quad (3)$$

Where α and β are weighting factors such that $\alpha + \beta = 1$. The values of α and β need to be determined experimentally and for our experiments are both set at 0.5. Equation 2 & 3 consists of two parts, the average hierarchical relationship distance and the concept distance among the items in the rule, respectively. In the following subsections we will define the two aspects in detail.

3.2.1 Hierarchical Relationship Distance

The HRD of two items measures how close two items are in terms of a hierarchical relationship from a common ancestor item (or root). The further apart they are in a hierarchical relation; that is the greater the number of concept levels difference between two items and their common ancestor, the more diverse the two items are and the more diverse the rule will be.

Here for the HRD component, diversity is inversely related to the closeness of items in terms of a hierarchical relationship. The closer the two items are, the less diverse they are. The further / more distant the relationship, the more diverse. For maximum HRD diversity the two items need to have no common ancestor and both be located at the lowest concept level in the dataset.

HRD focuses on measuring the horizontal (or width) distance between two items. Usually the greater the horizontal distance, the greater the distance to a common ancestor and therefore the more diverse the two items are. Due to the second assumption, we can not measure the horizontal distance without also utilising the vertical (height) distance.

Thus to determine the Hierarchical Relationship Distance (HRD) component of the diversity the following is proposed:

$$HRD(n_1,n_2) = \frac{(NLD(n_1,ca) + NLD(n_2,ca))}{2 \times TreeHeight} \quad (4)$$

The Hierarchical Relationship Distance between two items is defined as the ratio between the average number of levels between the two items and their common ancestor and the height of the tree. Thus if two items share a direct parent, the HRD value of the two items becomes the lowest value which is $1/TreeHeight$, while if the two items have no common ancestor or their common ancestor is the root, the HRD values of the two items can score high. Maximum HRD value, which is 1, is achieved when the two items have no common ancestor (or the common ancestor is the root) and both items are at the lowest concept level possible in the hierarchy. If n_1 and n_2 are the same item, then HRD becomes $1/TreeHeight$.

3.2.2 Concept Level Distance

This aspect is based on the hierarchical levels of the two items. The idea is that the more levels between the two items, the more diverse they will be. Thus two items on

the same hierarchy level are not very diverse, but two items on different levels are more diverse as they have different degrees of specificity or abstractness.

LD differs from HRD in that HRD measures the distance from a common ancestor item (or root), whereas LD measures the distance between the two items themselves. LD focuses on measuring the distance between two items in terms of their height (vertical) difference (HRD considers the width (horizontal) distance).

Thus, we propose to use the ratio between the level difference (NLD) of two items and the height of the tree (eg. the maximum level difference) to measure the Level Distance of the two items as defined as follows:

$$LD(n_1, n_2) = \frac{NLD(n_1, n_2)}{(TreeHeight - 1)} \quad (5)$$

This means that two items on the same concept level will have a LD of 0, while an item at the highest concept level and another at the lowest concept lvel will have an LD of 1, as they are as far apart as possible in the given hierarchy.

3.3 Peculiarity

Peculiarity is an objective measure that determines how far away one association rule is from others. The further away the rule is, the more peculiar. It is usually done through the use of a distance measure to determine how far apart rules are from each other. Peculiar rules are usually few in number (often generated from outlying data) and significantly different from the rest of the rule set. It is also possible that these peculiar rules can be interesting as they may be unknown. One proposal for measuring peculiarity is the neighbourhood-based unexpectednedd measure first proposed in [2]. In this proposal it is argued that a rule's interestingness is influenced by the rules that surround it in its neighbourhood.

The measure is based on the idea of determining and mesuring the symmetric difference between two rules, which forms the basis of the distance between them. From this it was proposed [2] that unexpected confidence (where the confidence of a rule *R* is far from the average confidence of the rules in *R's* neighbourhood) and sparsity (where the number of mined rules in a neighbourhood is far less than that of all the potential rules for that neighbourhood) could be determined, measured and used as interestingness measures [2] [3].

This measure [2] for determing the symmetric difference was developed for single level datasets where each item was equally weighted. Thus the mesure is actually a count of the number of items that are not common between the two rules. In a multi-level dataset, each item cannot be regarded as being equal due to the hierarchy. Thus the measure proposed in [2] needs to be enhanced to be useful with these datasets. Here we will present an enhancement as part of our proposed work.

We believe it is possible to take the distance measure presented in [2] and enhance it for multi-level datasets. The original measure is a syntax-based distance metric in the following form:

$$\begin{aligned} P(R_1, R_2) = \quad & \delta_1 \times |(X_1 \cup Y_1)\Theta(X_2 \cup Y_2)| + \\ & \delta_2 \times |X_1\Theta X_2| + \delta_3 \times |Y_1\Theta Y_2| \end{aligned} \quad (6)$$

The Θ operator denotes the symmetric difference between two item sets, thus $X\Theta Y$ is equivalent to $X - Y \cup Y - X$. δ_1, δ_2 and δ_3 are the weighting factors to be applied to different parts of the rule. Equation 6 measures the peculiarity of two rules by a weighted sum of the cardinalities of the symmetric difference between the two rule's antecedents, consequents and the rules themselves.

We propose an enhancement to this measure to allow it to handle a hierarchy. Under the existing measure, every item is unique and therefore none share any kind of 'syntax' similarity. However, we argue that the items 1-*-*-*, 1-1-*-*, 1-1-1-* and 1-1-1-1 (based on Figure 1) all have a relationship with each other. Thus they are not completely different and should have a 'syntax' similarity due to their relation through the dataset's hierarchy.

The greater the $P(R_1, R_2)$ value is, the greater the difference (thus lower similarity) and so the greater the distance between those two rules. Therefore, the further apart the relation is between two items, the greater the difference and distance. Thus if we have,
$R_1 : 1-1-1-* \Rightarrow 1-*-*-*$
$R_2 : 1-1-*-* \Rightarrow 1-*-*-*$
$R_3 : 1-1-1-1 \Rightarrow 1-*-*-*$
We believe that the following should hold; $P(R_1, R_3) < P(R_2, R_3)$ as 1-1-*-* and 1-1-1-1 are further removed from each other than 1-1-1-* and 1-1-1-1.

The difference between any two hierarchically related items / nodes must be less than 1. Thus (for the above rules) $1 > P(R_2, R_3) > P(R_1, R_2) > 0$. In order to achieve this we modify Equation 6 by calculating the diversity of the symmetric difference between two rules instead of the cardinality of the symmetric difference. The cardinality of the symmetric difference measures the difference between two rules in terms of the number of different items in the rules. The diversity of the symmetric difference takes into consideration the hierarchical difference of the items in the symmetric difference to measure the difference of the two rules. We recite Equation 2 in terms of a set of items below, where *S* is a set containing *n* items:

$$\begin{aligned} PD(S) \quad = \quad & \frac{\alpha \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} HRD(i,j)}{n(n-1)} + \\ & \frac{\beta \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} LD(i,j)}{n(n-1)} \end{aligned} \quad (7)$$

Thus the neighbourhood-based distance measure between two rules shown in Equation 6 now becomes;

$$\begin{aligned} PM(R_1, R_2) = \delta_1 \times PD((X_1 \cup Y_1)\Theta(X_2 \cup Y_2)) + \\ \delta_2 \times PD(X_1\Theta X_2) + \delta_3 \times PD(Y_1\Theta Y_2) \end{aligned} \quad (8)$$

Let *RS* be the ruleset of $\{R_1, R_2, ..., R_n\}$ then the average distance of a rule R_i to the ruleset *RS* can be determined by:

$$PM_{ave} = \frac{\sum_{\forall R_j \in RS \text{ and } j \neq i}^{n} PM(R_i, R_j)}{|RS| - 1} \quad (9)$$

4 Experimental Results

In this section we present experimental results of our proposed interestingness measures being used for association rule discovery from a multi-level dataset.

4.1 Dataset and Setup

The dataset used for our experiments is a real world dataset, the BookCrossing dataset (obtained from http://www.informatik.uni-freiburg.de/ cziegler/BX/) [10]. From this dataset we built a multi-level transactional dataset that contains 92,005 user records and 960 leaf items, with 3 concept / hierarchy levels.

To discover the frequent itemsets we use the MLT2_L1 algorithm proposed in [4] [5] with each concept level having its own minimum support. From these frequent itemsets we then derive the frequent closed itemsets and generators using the CLOSE+ algorithm proposed in [9]. From this we then derive the non-redundant association rules using the MinMaxApprox (MMA) rule mining algorithm [9].

4.2 Results

For the experiment we simply use the previously mentioned rule mining algorithm to extract the rules from the multi-level dataset. For this experiment we assign a reducing minimum support threshold to each level. The minimum supports are set to 10% for the first hierarchy level, 7.5% for the second and 5% for the thrid level (the lowest). During the rule extraction process we determine the diversity and peculiarity distance of the rules that meet the confidence threshold. With two measures known for each rule, we are also able to determine the minimum, maximum and average diversity and peculiarity distance for the rule set.

4.2.1 Statistical Analysis

Firstly, we compare the distribution curves of the proposed measures (diversity and distance) against the distribution curves of support and confidence for the rule set. The distribution curves are shown in Figure 2. The value of each measure ranges from 0 to 1. The values of the distance measure are based on the minimum distance (in this case 33,903.7) being equal to 0 and the maximum distance (in this case being 53,862.5) being equal to 1. The range between these two has been uniformly divided into 20 bins.

As Figure 2 shows, the support curve shows that the majority of association rules only have a support of between 0.05 and 0.1. Thus for this dataset distinguishing interesting rules based on their support would be difficult as the vast majority have very similar support values. This would mean the more interesting or important rules would be lost.

The confidence curve shows that the rules are spread out from 0.5 (which is the minimum confidence threshold) up to close to 1. The distribution of rules in this area is fairly consistant and even, ranging from as low as 2,181 rules for 0.95 to 1, to as high as 4,430 rules for 0.85 to 0.9. Using confidence to determine the interesting rules is more practical than support, but still leaves over 2,000 rules in the top bin.

The overall diversity curve shows that the majority of rules (23,665) here have an average overall diversity value of between 0.3 to 0.4. The curve however, also shows that there are some rules which have an overall diveristy value below the majority, in the range of 0.15 to 0.25 and some that are above the majority, in the range of 0.45 up to 0.7. The rules located above the majority are different to the rules that make up the majority and could be of interest as these rules have a high overall diversity.

The antecedent-consequent diversity curve is similar to that of the overall diversity. It has a similar spread of rules, but the antecedent-consequent diversity curve peaks earlier at 0.3 to 0.35 (where as the overall diversity curve peaks at 0.35 to 0.4), with 12,408 rules. The curve then drops down to a low number of rules at 0.45 to 0.5, before peaking again at 0.5 to 0.55, wih 2,564 rules. The shape of this curve with that of the overall diversity seems to show that the two diversity approaches are related. Using the antecedent-consequent diversity allows rules with differing antecedents and consequents to be discovered when support and confidence will not identify them.

Lastly, the distance curve shows the largest spread of rules across a curve. There are rules which have a low distance from the rule set (0 to 0.1 which corresponds to a distance of 33,903.7 to 35,899.56) up to higher distances (such as 0.7 and above which corresponds to a distance of 47,874.88 to 53,862.52). The distance curve peaks at 0.3 to 0.35 (which is a distance of between 39,891.35 and 40,889.29). Using the distance curve to find interesting rules allows those that are close to the ruleset (small distance away) or those that are much further away (greater distance) to be discovered.

Next, we look at the trends of the various measures when compared against the proposed diversity and distance measures.

Figure 3 shows the trend of the average support, average confidence, average antecedent-consequent diversity and average distance values against that of overall diversity. As can be seen the average support remains fairly constant. There is tendancy for the support to increase for those rules with a high overall diversity. Even so, this shows that support does not always agree with overall diversity, so an overall diversity measure can be useful to find a different set of interesting rules.

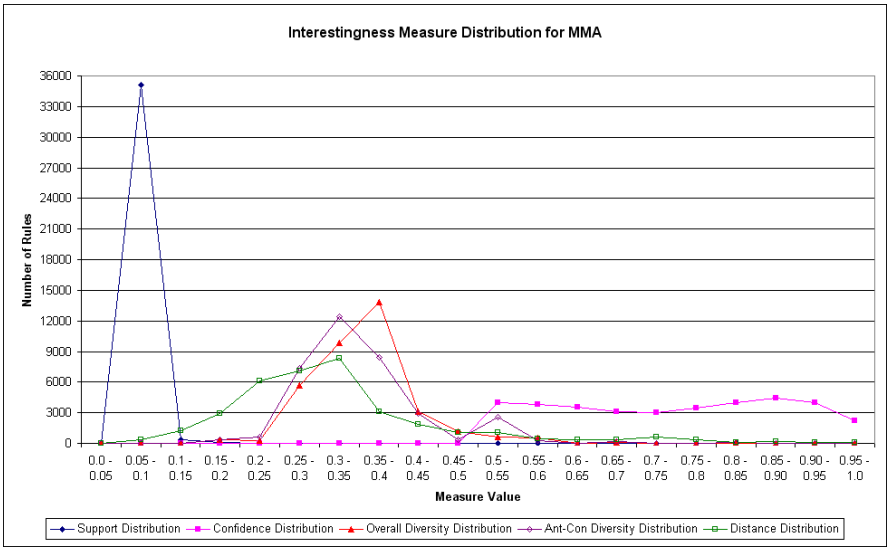


Figure 2: Distribution curves for the proposed interestingness measures, support and confidence.

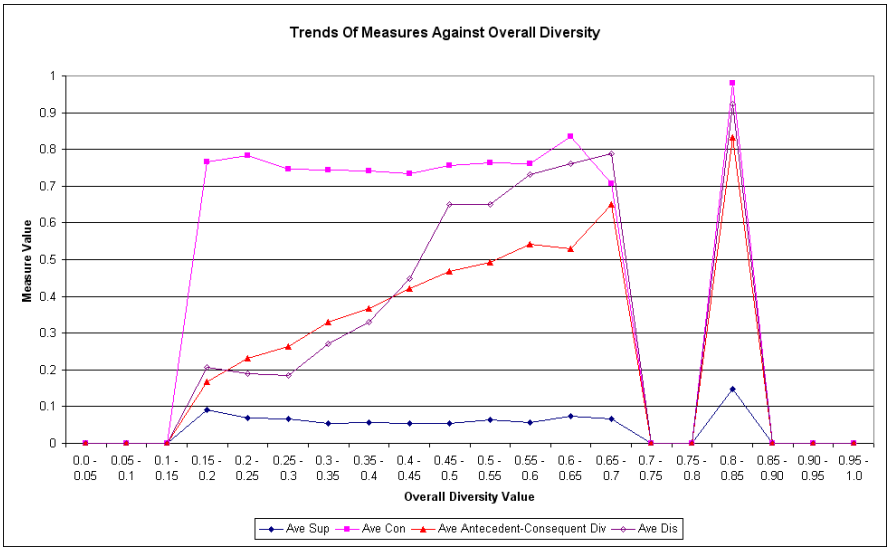


Figure 3: Trends of measures against the proposed overall diversity measure.

The confidence in Figure 3 is also fairly constant (usually varying by less than 0.1) until the end. Again this shows that the confidence will not always discover those rules that are more diverse overall.

The average antecedent-consequent diversity tends to have a consistant upward trend as the overall diversity increases. This shows that both the overall diversity and antecedent-consequent diversity are related/linked (which is not unexpected). It is quite possible that the greatest degree of diversity for a rule comes from comparing the items in the antecedent against those in the consequent and not from comparing the items within just the antecedent and/or consequent.

The distance has an overall upwards trend, although it is not a constant rate nor consistant (as there is a small decrease from 0.2 to 0.3). This, along with the trend of the average overall diversity (which shows a consistant upwards trend as the distance increases) in Figure 5 would indicate that potentially the more overall diverse rules have a higher distance from the rest of the rule set and therefore are further away. This would also imply that those rules with a higher distance are usually more diverse overall as well.

Figure 4 shows the trends of average support, average confidence, average overall diversity and average distance against that of antecedent-consequent diversity. Like in Figure 3, the support remains fairly constant regardless of the antecedent-consequent diversity value.

The confidence tends to decrease as the antecedent-consequent diversity increases, so the more diverse rules will not always be picked up by confidence.

The overall diversity tends to increase as antecedent-consequent diversity increases (similar

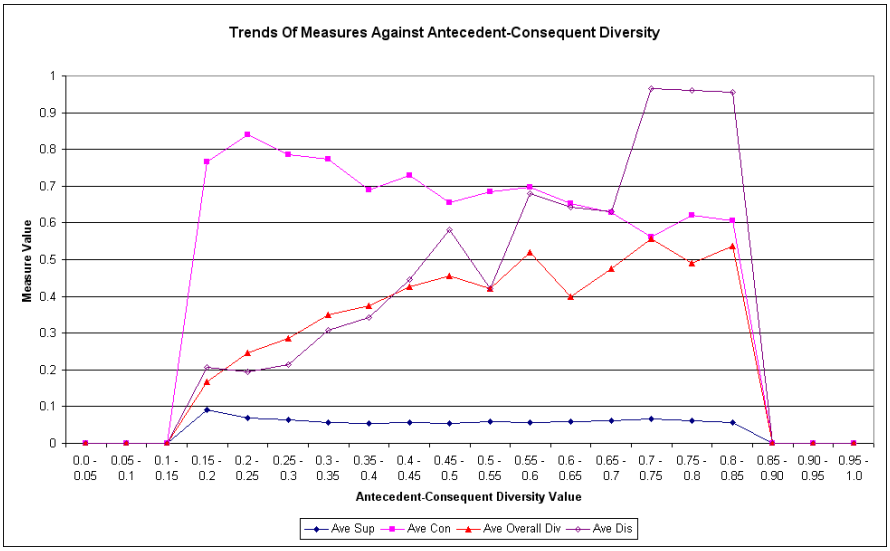


Figure 4: Trends of measures against the proposed antecedent-consequent diversity measure.

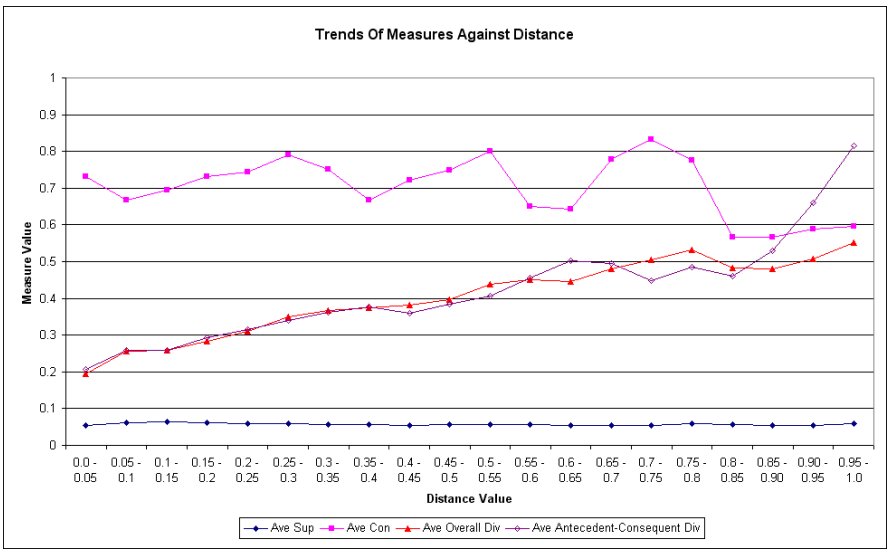


Figure 5: Trends of measures against the proposed distance measure.

to Figure 3). So again the biggest diversity in a rule is often in the difference between the antecedent and consequent.

The distance also tends to increase (gradually at first for lower antecedent-consequent diversity values). There is a big jump in the distance trend when the antecedent-consequent diversity increases from 0.65 to 0.75. The highest distance values are achieved when the antecedent-consequent diversity reaches its highest values (this is also shown in Figure 5).

Figure 5 shows the trend of the average support, average confidence, average overall diversity and average antecedent-consequent diversity values against that of distance. As shown, the support remains very constant regardless of the distance. This shows that support can not be used to discover rules that have a low or high peculiarity distance.

Like the average overall diversity, the average antecedent-consequent diversity also trends upwards as the distance increases. The rate of rise is similar to that of the overall diversity initially at lower distance values, but becomes much steeper at high distance values. This shows that it seems the most distant rules also have the highest diversity between their antecedent and consequent as Figure 5 shows the average antecedent-consequent diversity to be over 0.8 for the rules with the highest distance from the rest of the rule set.

The confidence trend in Figure 5 also shows that confidence will not always discover those rules far away from the rule set (0.8 / 49,870.76 and above), as at these distances the confidence values are at their lowest points. For rules with a low distance value, confidence may also not be the best measure as at these values

(0 / 33,903.7 to 0.1 / 35,899.58) confidence values are not at their highest. The highest confidence value(s) occur when the distance is 47,874.86 to 48,872.82 (0.7 to 0.75).

4.2.2 Examples of Proposed Measures

If we look closer at the discovered rules we find the following examples that show how diversity and peculiarity distance can be useful in identifying potentially interesting rules that would not normally be identified as such. (Note that the hypen breaks the concept levels, while a comma indicates a new item).

Example 1:

R_1=BookClubs-Lit.&Fiction-Pop.Fiction → Subjects-Lit.&Fiction-General

Supp 12.228% Conf 81.5% OverallDiv 0.5

R_2=BookClubs-Lit.&Fiction-Pop.Fiction → Subjects-Mystery&Thrillers

Supp 7.9% Conf 52.67% OverallDiv 0.67

R_1 has a higher support and confidence than R_2, but R_2 has a higher overall diversity. If we used either the support or confidence measure then R_1 would always be chosen as the more interesting rule. However, our proposed overall diversity measure indicates that R_2 is more interesting due to its diversity score, which can be attributed to its more general consequent.

Example 2:

R_3=Subjects-Biographies&Memoirs-General, Subjects-Lit.&Fiction-Authors(A..Z) → BookClubs-Lit.&Fiction

Supp 5.59% Conf 60.9% Ant-ConDiv 0.67

R_3 has low support and reasonbly low confidence, but it has high antecedent-consequent diversity (the average is 0.35). If we use support or confidence this rule will probably not be chosen as interesting as its support value is lower than the average support value for this rule set (5.8%) and its confidence is relatively low and is also lower than the average confidence of the rule set (74.4%). However, if we use antecedent-consequent diversity, then it will be selected as it has a high value. Hence this rule may be of interest because of the diversity between its antecedent and consequent itemsets, which come from different branches of the hierarchy.

Example 3:

R_4=BookClubs, Subjects-Lit.&Fiction-WorldLit. → Subjects-Lit.&Fiction-GenreFiction, Subjects-Mystery&Thrillers

Supp 6.7% Conf 57.7% Dist 50,311.4

R_4 has a noticably higher than average distance and is much further away from the rule set. This may be of interest to a user. But if support and confidence are used, this rule is considered to not be of interest due to their low values.

5 Conclusion

In this paper we have proposed two interestingness measures for association rules derived from multi-level datasets. These proposed interestingness measures are diversity and peculiarity (distance) respectively.

Diversity is a measure that compares items within a rule and peculiarity compares items in two rules to see how different they are.

In our experiments we have shown how diversity and peculiarity distance can be used to identify potentially interesting rules that normally would not be considered as interesting using the traditional support and confidence approach.

Acknowledgements Computational resources and services used in this work were provided by the HPC and Research Support Unit, Queensland University of Technology, Brisbane, Australia.

References

[1] R. Agrawal, T. Imielinski and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In *ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 207–216, Washington D.C., USA, May 1993.

[2] G. Dong and J. Li. Interestingness of Discovered Association Rules in terms of Neighbourhood-Based Unexpectedness. In *Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'98)*, pages 72–86, Melbourne, Australia, April 1998.

[3] L. Geng and H. J. Hamilton. Interestingness Measures for Data Mining: A Survey. *ACM Computing Surveys (CSUR)*, Volume 38, pages 9, 2006.

[4] J. Han and Y. Fu. Discovery of Multiple-Level Association Rules from Large Databases. In *21st International Conference on Very Large Databases (VLDB'95)*, pages 420–431, Zurich, Switzerland, September 1995.

[5] J. Han and Y. Fu. Mining Multiple-Level Association Rules in Large Databases. *IEEE Transactions on Knowledge and Data Engineering*, Volume 11, pages 798–805, 1999.

[6] S. Lallich, O. Teytaud and E. Prudhomme. Association rule interestingness: measure and statistical validation. *Quality Measures in Data Mining*, Volume 43, pages 251–276, 2006.

[7] P. Lenca, B. Vaillant, B. Meyer and S. Lallich. Association rule interestingness: experimental and theoretical studies. *Studies in Computational Intelligence*, Volume 43, pages 51–76, 2007.

[8] K. McGarry. A Survey of Interestingness Measures for Knowledge Discovery. *The Knowledge Engineering Review*, Volume 20, pages 39–61, 2005.

[9] N. Pasquier, R. Taouil, Y. Bastide and G. Stumme. Generating a Condensed Representation for Association Rules. *Journal of Intelligent Information Systems*, Volume 24, pages 29–60, 2005.

[10] C.-N. Ziegler, S. M. McNee, J. A. Konstan and G. Lausen. Improving Recommendation Lists Through Topic Diversification. In *14th International Conference on World Wide Web (WWW'05)*, pages 22–32, Chiba, Japan, May 2005.

Do Users Find Looking at Text More Useful than Visual Representations? A Comparison of Three Search Result Interfaces

Hilal Al Maqbali *Falk Scholer* *James A. Thom* *Mingfang Wu*
School of Computer Science and Information Technology
RMIT University
GPO Box 2476, Melbourne 3001
Victoria, Australia
h.almaqbali@student.rmit.edu.au,{falk.scholer,james.thom,mingfang.wu}@rmit.edu.au

Abstract

The organisation, content and presentation of document surrogates has a substantial impact on the effectiveness of web search result interfaces. Most interfaces include textual information, including for example the document title, URL, and a short query-biased summary of the content. Other interfaces include additional browsing features, such as topic clustering, or thumbnails of the web pages. In this study we analyse three search interfaces, and compare the effectiveness of textual information and additional browsing features. Our analysis indicates that most users spend a substantially larger proportion of time looking at text information, and that those interfaces that focus on text-based representations of document content tend to lead to quicker task completion times for named-page finding search tasks.

Keywords Information Retrieval, User Studies Involving Documents, Web Documents, Eye Tracking

1 Introduction

Search engines are a key tool for supporting users in finding information on the world wide web. These information retrieval systems aim to find relevant documents in response to a user query. While the performance of the underlying ranking function – responsible for identifying good answer resources – is clearly of great importance, the organisation, content and presentation of document surrogates in the search results interface can also have a substantial impact on overall search effectiveness.

One recent study [9] found that only 21% of users found relevant results when querying a search engine, and that 75% were disappointed with the results returned. The way users interact with the search result interface may be one factor in the poor user experience.

This paper analyses three search interfaces that make use of different features including text summaries, clustering information, and visual thumbnail images:

C Carrot2 (`http://www.carrot2.org`),

M Middlespot (`http://www.middlespot.com`), and

N Nexplore (`http://www.nexplore.com`)

A preliminary analysis of overall task completion time with the different interfaces was presented in a previous paper [1]. In this paper, we investigate how much time users spent looking at different regions of the screen, in particular comparing the time spent looking at text surrogates of the result pages with time spent looking at more visual representations. Our analysis indicates that most users find text surrogates to be more useful.

The remainder of this paper is organised as follows. Some related work is presented in Section 2; our experiment design, including the different search interfaces, users, and topics used, is described in Section 3; the results of the experiment are analysed in Section 4; and discussion and conclusions are given in Section 5.

2 Related Work

The presentation of search results influences users' assimilation and guides users to look for the information that is relevant to them. In the past, quite a few studies explored visual presentation of search results [3, 10, 11]. A proper visual representation can communicate some kinds of information much more rapidly and effectively than textual representation. However, visualisation of textually represented information is difficult and challenging [6].

The effectiveness of visual representations largely depends on whether the representation is highly coupled with a search task and on the inherent structure of documents to be presented. Joho and Jose [8] investigated how textual and visual forms of information enabled users to more effectively interact with search answer interfaces in undertaking relevance assessments and reformulating queries.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

Cutrell and Guan [4] found that adding extra contextual information to the document surrogates can improve the effectiveness on search answer interfaces for informational tasks. Hearst and Pedersen [7] is one of many studies that has investigated the effectiveness of clustering search results. Compared with the results of the study described in this paper, where we find the Carrot2 interface that supported clustering to be ineffective when undertaking a navigational task searching for a single correct answer, they found clustering of answers was effective in supporting a user's task that involved finding a set of relevant answer documents.

A previous study by Dziadosz and Chandrasekar [5] had found that the combination of thumbnails and text summary to be more effective for users than either thumbnails or text summaries alone. However, our study suggests that the combination of thumbnails and text is only effective when they are not large, since users mostly look at the text summaries and it is not effective to use too much of the screen real estate on the images of answer pages.

3 Experimental methodology

To investigate the relative attention that users pay to different interface components, we conducted a user study that involved carrying out a series of named-page finding search tasks using a variety of search interfaces.

3.1 User study

Our study was carried out at RMIT University Open Day in August 2009. Subjects participated in the experiment were mostly high school students with an interest in computer science who were visitors to our laboratory. Participants were given a plain language statement outlining the goals of the experiment, the types of tasks to be undertaken, and the data that would be collected. Based on this information, 35 volunteers chose to participate in the experiments. No training was given with the different search interfaces.

Each participant undertook three navigational search tasks (described below), using different search interfaces. Information about visual attention given to the different screen components was collected using a Tobii T60 eye tracker. This non-intrusive device records the gaze position, providing information on fixations and saccades (brief rapid eye movements).

3.2 Search interface features

Our experiment involved users using three different search result interfaces that contained different amounts of surrogate text and visual browse features about answer documents on the result pages.

The three interfaces were selected because they provide a variety of additional novel features, not just a ranked list of text extracts. Carrot2 does not present visual features, however it clusters its search results. In Middlespot, screenshots are presented for the retrieved documents. Nexplore has more visual features such as highlighting of query terms, thumbnails, background colour and highlighting the abstract of the retrieved document when the mouse is moved over it.

Interface Features	C	M	N
Text features	66%	17%	56%
Browse features	19%	75%	7%
Other regions	16%	8%	37%

Table 1: The distribution of interface features.

In this paper, we consider the following areas within each interface page displaying the ranked list of answers:

Surrogate text: Search engines provide surrogates for answer page in the ranked list of answers. This surrogate text may include the URL of the answer, as well as text from the answer web page title, and a synopsis of the answer web page. The surrogate text for the answer documents is in each of the regions marked (1) on the respective answer interfaces: Figure 1 for Carrot2 (accounting for approximately 66% of the screen), Figure 2 for Middlespot (17%), and Figure 3 for Nexplore (56%).

Browse features: The visual browse features for the answer documents are in each of the regions marked (2) on respective answer interfaces. Figure 1 shows the clustering area in Carrot2 which occupies approximately 19% of the screen. Figure 2 shows large images of the answer pages that are displayed in Middlespot and occupying approximately 75% of the screen. Figure 3 shows a much smaller region, approximately 7% of the screen, containing the thumbnails displayed by Nexplore.

Other regions: Each interface also had some other regions, such as banners and the surrounding screen, including a region at the bottom of the screen (not shown in the figures) that contained the topic and some instructions, This accounted for approximately 16% of the screen with the Carrot2 interface, 8% with Middlespot interface, and 37% with Nexplore (since this last interface included a separate area for Wiki Search).

As summarised in Table 1, significant portions of the Carrot2 and Nexplore interfaces are given to surrogate text. The great majority of the Middlespot interface, on the other hand, is occupied by visual browse features.

3.3 Topics

One taxonomic study [2] shows that web search tasks can be classified as informational, transactional or navigational. Navigational tasks are used in our study because we assume that users become more interested in

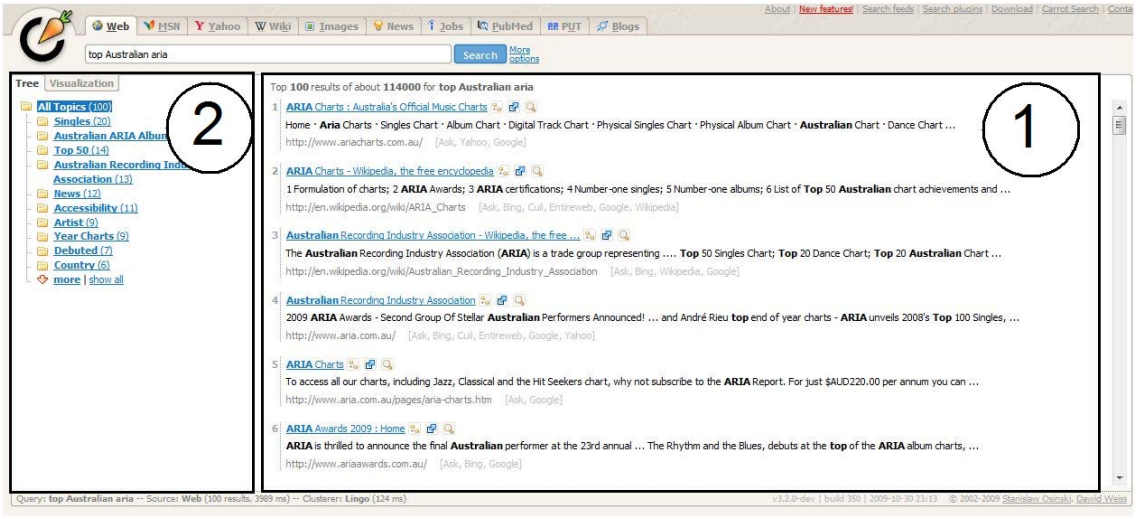


Figure 1: Carrot2 interface (`www.carrot2.org`). Areas marked 1 and 2 indicate Text and Browse features, respectively. Descriptions of the features are provided in the main text.

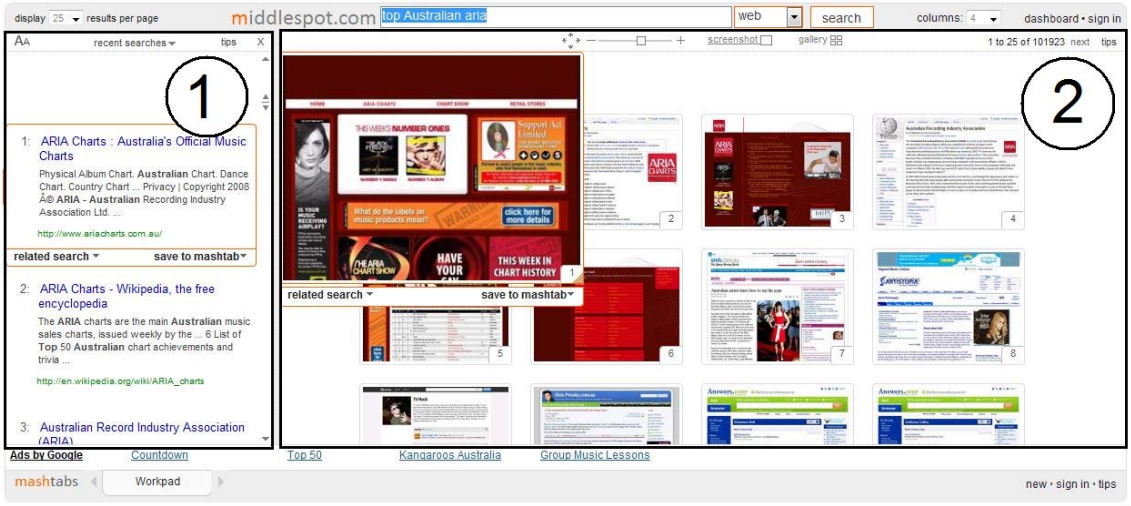


Figure 2: Middlespot interface (`www.middlespot.com`). Descriptions of features are provided in the main text.

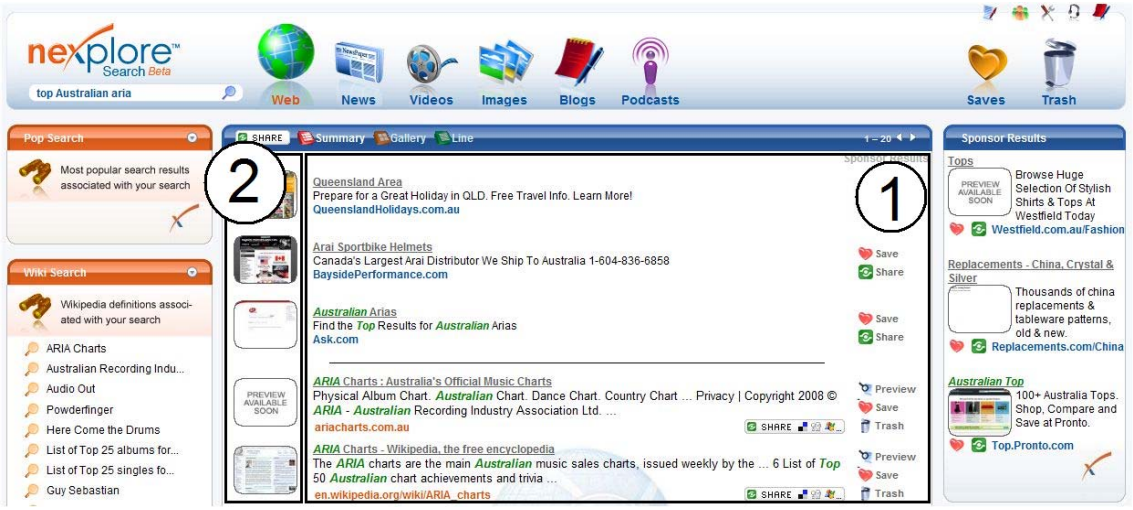


Figure 3: Nexplore interface (`www.nexplore.com`). Descriptions of features are provided in the main text.

Trial	1st task	2nd task	3rd task
1	M- H (4)	C- G (4)	N- A (3)
2	M- G (4)	C- A (4)	N- H (4)
3	M- A (4)	C- H (4)	N- G (3)
4	C- G (3)	N- A (3)	M- H (3)
5	C- A (5)	N- H (5)	M- G (5)
6	C- H (4)	N- G (4)	M- A (3)
7	N- A (3)	M- H (3)	C- G (3)
8	N- H (3)	M- G (3)	C- A (3)
9	N- G (3)	M- A (3)	C- H (3)

Table 2: Experimental design.

using additional web search interface features to get their desired information.

For each interface, users were given a navigational search task, for which they were asked to find a specific single correct answer page for the given topic. The topics were chosen to cover areas that were likely to be of interest to young searchers, and where searchers were unlikely to be hindered due to lack of general knowledge about the domain. The three topics were:

A: Find the ARIA chart of the top 50 music singles in Australia (query terms: `top australia aria`)

G: Find the MSN games website (query term: `msn`)

H: Find the official homepage of the 2009 movie Harry Potter (query terms: `magical potter`)

These topics, and their corresponding answer documents, represent different aspects of navigational searches: the answer for the first topic is a single web page presenting the required (named) information; the second is the hub page for a prime sub-part of the overall MSN website; and, the third is the home page (or index) of an overall website.

After reading a topic, the user would click a "start" button to load the results of issuing the predefined query terms (as indicated above) into one of the three search interfaces. The user could then interact with the search result screen however they wanted to.

We used a latin square experiment design with a block of nine trials varying the order in which topics and interfaces were presented to users, each user was presented with one topic for each interface. Due to some interruptions and other problems, not all combinations were completed exactly the same number of times. Table 2 shows the number of times (in parentheses) each of the different combinations of interface (C, M, N) and topic (A, G, H) were completed as the first, second or third task undertaken by one of the users.

4 Results

We analyse user behaviour when carrying out the three search tasks using the Carrot2, Middlespot and Nexplore interfaces based on the relative attention paid to different interface features, and task completion time.

4.1 Interface features

Different search interface features attract highly variable amounts of user attention. Figure 4 shows the proportions of total viewing time that users spent looking at text, browse and other features for each trial (that is, over all search interfaces and all users). The solid line shows the median time, while the boxes show the 25th to 75th percentiles. Whiskers show the range of the data, with outliers (observations more extreme than 1.5 times the interquartile range). Since the time data is not normally distributed (Shapiro-Wilk, $p < 0.0001$), we analyse multi-level factors using the Kruskal-Wallis test, a non-parametric alternative to ANOVA. Pairswise comparisons are made using the Wilcoxon signed-rank test. The relative times for the different features vary significantly (Kruskal-Wallis, $p < 0.0001$). In particular, users spend significantly more time viewing text features compared to browse features (Wilcoxon, $p < 0.0001$) and other ($p < 0.0001$). The difference in viewing patterns between browse and other is not significant ($p = 0.6504$).

Figure 5 shows the median time (over all search answer interfaces) users spent looking at different regions of the screen, broken down by cases where users identified the correct or incorrect answer document for each search trial. The text region was the area of the screen that users spent most of their time looking at, users found slightly more correct answers if they spent a bit more time in this area; while when users spent more time looking at the visual browse regions these were not effective and could often lead users to the incorrect answers rather than correct answers. Time spent looking at both text and browse regions is significantly different between correct and incorrect answers (Wilcoxon, p = 0.0060 for text regions and p = 0.0303 for browse regions) while the difference is not significant for other areas of the screen (p = 0.7669).

Figure 6 shows the distribution of the proportion of time that users spent viewing different features, split by the three interfaces. For the Carrot2 and Nexplore interfaces, users spent substantially more time viewing the text features. However, for the Middlespot interface, the browse features (in this case, the screenshots of web pages) attracted the greatest proportion of viewing time.

4.2 Task completion time

User task completion performance is evaluated by measuring the time taken to carry out a search task to the user's satisfaction. That is, we measure the time from when the search results screen is displayed to the user, until the time that they indicate that they have found a desired answer (generally, by clicking on the hyperlink in the search results list that they chose as their final answer). This is in contrast to our previous analysis [1], where task completion time was measured by taking the time that the user chose to exit the task (by explicitly pressing F10) as the endpoint. This adds additional variation to the results, since some users spend addi-

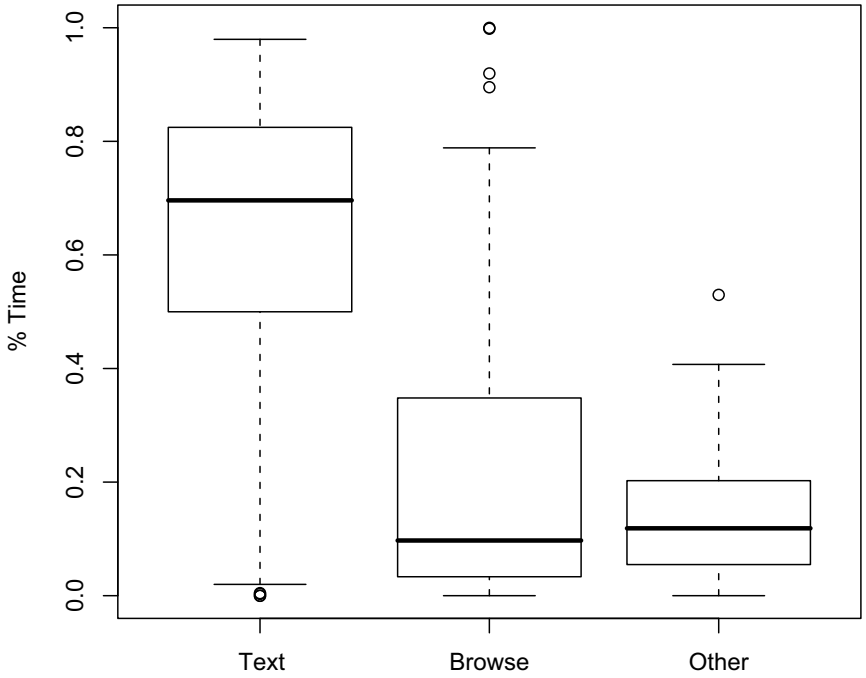


Figure 4: Relative time spent viewing different interface regions.

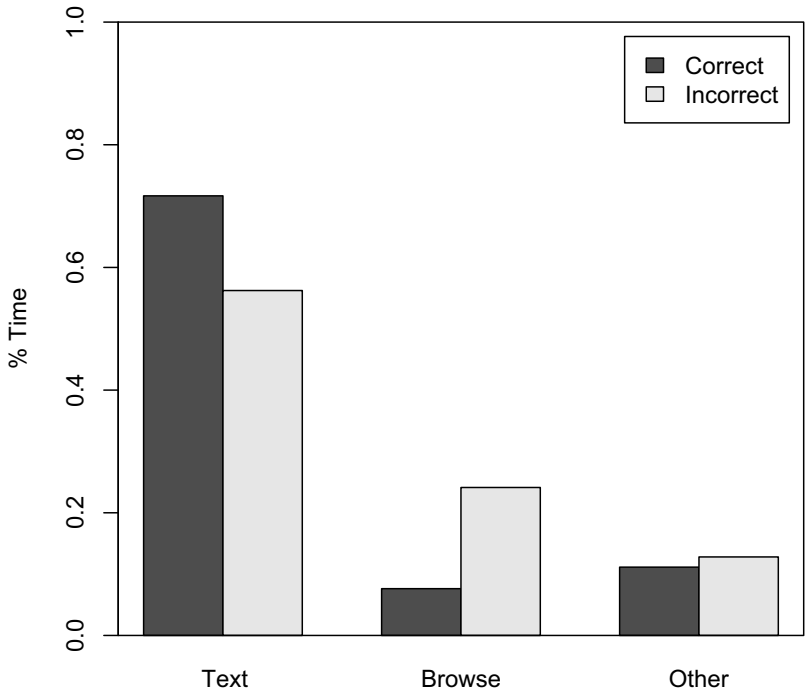


Figure 5: Median proportion of time spent viewing different regions when users found a correct or incorrect answer.

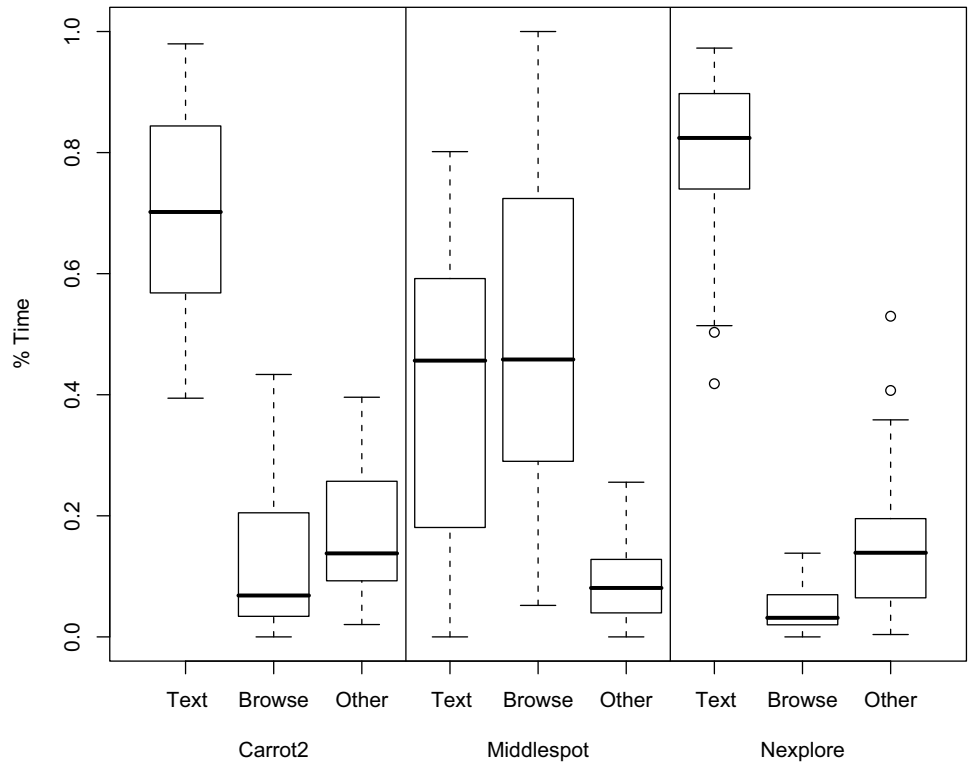


Figure 6: Proportion of time spent viewing different components, by interface.

tional time viewing their chosen answer page, before indicating task completion.

Figure 7 shows the time taken to find an answer, in seconds, for each of the three interfaces. The differences are weakly significant (Kruskal-Wallis, $p = 0.0604$). In particular, the Middlespot and Nexplore differ significantly (Wilcoxon, $p = 0.0129$), while the other pairs do not (Middlespot and Carrot 2, $p = 0.2474$; Nexplore and Carrot2, $p = 0.3225$).

Variation can also be introduced by other sources. The effect of using different search topics was significant (Kruskal-Wallis, $p = 0.0330$). Moreover, because we used real search interfaces and live search results, the rank of the correct answer items in the search results lists of the different interfaces varied somewhat. Although the ranks were similar on average (rank 7.6 for Carrot2, 6.3 for Middlespot, and 6.0 for Nexplore) this did have a significant effect on task completion time (Kruskal-Wallis, $p = 0.0048$). The different users participating in the experiment were not a significant source of variation (Kruskal-Wallis, $p = 0.1227$).

However, this analysis includes all user responses, irrespective of whether the user actually found the correct answer required for the query. We investigate this next.

Answer	Carrot2	Middlespot	Nexplore
Correct	24	18	24
Incorrect	9	14	7

Table 3: Distribution of correct answers by interface.

4.3 Search success

Users were asked to indicate when they felt that they had found the correct answer to the query. However, in many cases users did not in fact identify the correct resource. Table 3 shows the number of incorrect and correct answers found, split by the interface used. The results are strongly indicative of higher success rates with both the Carrot2 and Nexplore interfaces (72.7% and 77.4% of answers are correct, compared to 56.2% for Middlespot). However, the differences are not statistically significant (Fisher, $p = 0.1746$).

We re-analyse the time taken for task completion, using only those trials for which users identified the correct resource in response to the information need. For these responses, the difference between interfaces is greater, and statistically significant (Kruskal-Wallis, $p = 0.0077$). Differences between the interfaces on a pairwise basis are also more pronounced: the median task completion time with Middlespot at 23.71 seconds is significantly longer than that for Carrot2 at 12.81

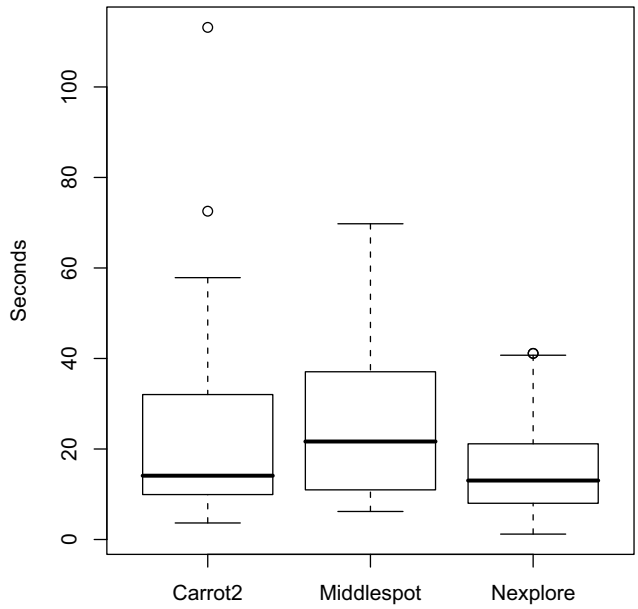


Figure 7: Task completion times by interface.

seconds (Wilcoxon, $p = 0.0112$) and for Nexplore at 12.21 seconds (Wilcoxon, $p = 0.0027$). The difference between Carrot2 and Nexplore is not significant (Wilcoxon, $p = 0.7360$).

Moreover, when considering only those results where users successfully identified correct answers, the effects from topic and user variation are not significant (Kruskal-Wallis, $p = 0.3445$ and 0.2743, respectively). The rank of the answer item only has a weakly significant effect (Kruskal-Wallis, $p = 0.0619$).

5 Discussion and Conclusions

Search result interfaces are an important component of information retrieval systems, and can have substantial impact on overall search task performance. In this paper, we have analysed three publicly available search interfaces, and examined how user attention is split between various features that the search providers make available.

Our analysis has shown that users spend significantly different proportions of time interacting with text, browse and other components of the interfaces. Not surprisingly, these proportions differ between the three interfaces; for Nexplore and Carrot2, text is preferred, while for Middlespot (which presents much less text to the user) browsing features are viewed more.

We have also analysed how task completion time differs between the interfacts, and success rates in identifying correct answers for given informatinon needs. The results show that users spent significantly longer time to interact with the Middlespot interface but found the fewest correct answers. We conclude that, for the navigational search tasks, text features are important in guiding users to finding correct answers quickly.

For the small sample of named-resource finding search tasks, it appears that text information can be vital in supporting users to find the answers that they need. Whether this would also apply to other search tasks, such as informational tasks, will be the subject of future research.

In future work we plan to conduct further user studies over a wider range of tasks. We also plan to investigate the effect of the proportion of screen space that is given over to browsing features as a controlled variable (that is, systematically controlling the proportion).

References

[1] H. Ali [Al Maqbali], F. Scholer, J. A. Thom and M. Wu. User interaction with novel web search interfaces. In *21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group (CHISIG) of the Human Factors and Ergonomics Society of Australia (HFESA)*, 2009.

[2] A. Border. A taxonomy of web search. *ACM SIGIR Forum*, Volume 36, Number 2, pages 3–10, 2002.

[3] S. K. Card, J. D. Mackinlay and B. Shneideman. Morgan Kaufmann Publishers, 1999.

[4] E. Cutrell and Z. Guan. What are you looking for?: an eye-tracking study of information usage in web search. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 407–416, San Jose, California, USA, April–May 2007.

[5] S. Dziadosz and R. Chandrasekar. Do thumbnail previews help users make better relevance decisions about web search results? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference*

on Research and development in information retrieval, pages 365–366, Tampere, Finland, August 2002.

[6] M. Hearst. *User Interfaces and Visualisation*. Addison-Wesley, 1999.

[7] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, Zurich, Switzerland, August 1996.

[8] H. Joho and J. M. Jose. A comparative study of the effectiveness of search result presentation on the web. In *Advances in Information Retrieval, Proceedings of 28th European Conference on IR Research*, pages 302–313, April 2006.

[9] R. S. Rele and A. T. Duchowski. Using eye tracking to evaluate alternative search results interfaces. In *Proceedings of Human Factors and Ergonomics Society Annual Meeting*, pages 1459–1463, 2005.

[10] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of IEEE Symposium on Visual Languages*, 1996.

[11] B. Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *Proceedings of 2008 ACM SIGMOD International Conference on Management of Data*, pages 3–12, Vancouver, Canada, 2008.

Random Indexing K-tree

Christopher M. De Vries *Lance De Vine* *Shlomo Geva*

Faculty of Science and Technology
Queensland University of Technology
Brisbane, Australia

chris@de-vries.id.au *l.devine@qut.edu.au* *s.geva@qut.edu.au*

Abstract *Random Indexing (RI) K-tree is the combination of two algorithms for clustering. Many large scale problems exist in document clustering. RI K-tree scales well with large inputs due to its low complexity. It also exhibits features that are useful for managing a changing collection. Furthermore, it solves previous issues with sparse document vectors when using K-tree. The algorithms and data structures are defined, explained and motivated. Specific modifications to K-tree are made for use with RI. Experiments have been executed to measure quality. The results indicate that RI K-tree improves document cluster quality over the original K-tree algorithm.*

Keywords Random Indexing, K-tree, Dimensionality Reduction, B-tree, Search Tree, Clustering, Document Clustering, Vector Quantization, k-means

1 Introduction

The purpose of this paper is to present and analyse the combination of Random Indexing (RI) with the K-tree algorithm. Both RI and K-tree adapt to changing data and decrease the cost of computationally intensive vector based applications. This combination is particularly suitable to the representation and clustering of very large document collections. Documents are typically represented in vector space as very sparse high dimensional vectors. RI can reduce the dimensionality and sparsity of this representation. In turn, the condensed representation is highly effective when working with K-tree. The paper is focused on determining the effectiveness of using RI with K-tree through experiments and comparative analysis of results.

Sections 2 to 6 discuss K-tree, Random Indexing, Document Representation, Experimental Setup and Experimental results respectively. The paper ends with a conclusion in Section 7.

2 K-tree

K-tree [6, 1] is a height balanced cluster tree. It was first introduced in the context of signal processing by Geva [10]. The algorithm is particularly suitable to clustering of large collections due to its low complexity. It is a hybrid of the B^+-tree and k-means algorithm. The B^+-tree algorithm is modified to work with multi dimensional vectors and k-means is used to perform node splits in the tree. K-tree is also related to Tree Structured Vector Quantization (TSVQ) [9]. TSVQ recursively splits the data set, in a top-down fashion, using k-means. TSVQ does not generally produce balanced trees.

K-tree achieves its efficiency through execution of the high cost k-means step over very small subsets of the data. The number of vectors clustered during any step in the K-tree algorithm is determined by the tree order (usually $\ll 1000$) and it is independent of collection size. It is efficient in updating the collection while maintaining clustering properties through the use of a nearest neighbour search tree that directs new vectors to the appropriate leaf node.

The K-tree forms a hierarchy of clusters. This hierarchy supports multi-granular clustering where generalisation or specialisation is observed as the tree is traversed from a leaf towards the root or vice versa. The granularity of clusters can be decided at run-time by selecting clusters that meet criteria such as distortion or cluster size.

2.1 K-tree and Document Clustering

The K-tree algorithm is well suited to clustering large document collections due to its low time complexity. The time complexity of building K-tree is O(n log n) where n is the number of bytes of data to cluster. This is due to the divide and conquer properties inherent to the search tree. De Vries and Geva [5, 6] investigate the run-time performance and quality of K-tree by comparing results with other INEX submissions and CLUTO [13]. CLUTO is a popular clustering tool kit used in the information retrieval community. K-tree has been compared to k-means, including the CLUTO implementation, and provides comparable quality and a marked increase in run-time performance. However, K-tree forms a hierarchy of clusters and k-means does not. Comparison of the quality of the tree structure will be undertaken in further research. The run-time performance increase of K-tree is most noted when a large number of clusters are required. This is useful in terms of doc-

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

ument clustering because there are a huge number of topics in a typical collection. The on-line and incremental nature of the algorithm is useful for managing changing document collections. Most clustering algorithms are one shot and must be re-run when new data arrives. K-tree adapts as new data arrives and has the low time complexity of O(log n) for insertion of a single document. Additionally, the tree structure also allows for efficient disk based implementations when the size of data sets exceeds that of main memory.

2.2 K-tree Definition

K-tree builds a nearest neighbour search tree over a set of real valued vectors V in d dimensional space.

$$\forall v \in V : v \in \mathbb{R}^d \tag{1}$$

It is inspired by the B$^+$-tree where all data records are stored in leaf nodes. Tree nodes, N, consist of a sequence of (vector, child node) pairs of length l. The tree order, m, restricts the number of vectors stored in any node to between one and m.

$$1 \leq l \leq m \tag{2}$$

$$N = \langle (v_1, c_1), ..., (v_l, c_l) \rangle \tag{3}$$

The tree consists of two types of nodes. Leaf nodes contain the data vectors that were inserted into the tree. Internal nodes contain clusters. A cluster vector is the mean of all data vectors contained in the leaves of all descendant nodes (i.e. the entire cluster sub-tree). This follows the same recursive definition of a B$^+$-tree where each tree is made up of a set of smaller sub-trees. Upon construction of the tree, a nearest neighbour search tree is built in a bottom-up manner by splitting full nodes using k-means [14] where $k = 2$. As the tree depth increases it forms a hierarchy of "clusters of clusters" from the root to the above-leaf level. The above-leaf level contains the finest granularity cluster vectors. Each leaf node stores the data vectors pointed to by the above-leaf level. The efficiency of K-tree stems from the low complexity of the B$^+$-tree algorithm, combined with only ever executing k-means on a relatively small number of vectors, defined by the tree order, and by using a small value of k.

2.3 Modifications to K-tree

The K-tree algorithm was modified for use with RI. This modified version will be referred to as "Modified K-tree" and the original K-tree will be referred to as "Unmodified K-tree".

All the document vectors created by RI are of unit length in the modified K-tree. Therefore, all centroids are normalised to unit length at all times. The k-means used for node splits in K-tree was changed to use randomised seeding and restart if it did not converge within six iterations. The process always converged quickly in our experiments; although it is possible to constrain the number of restarts we did not find this to be necessary.

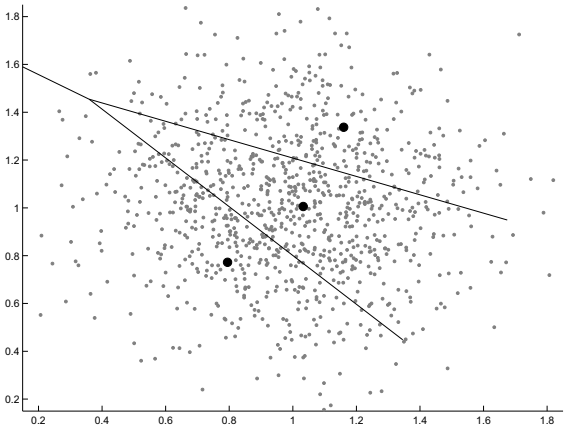


Figure 1: Level 1

The original K-tree algorithm does not modify any of the centroids. They are simply the means of the vectors they represent. The k-means implementation runs to complete convergence and seeds centroids via perturbation of the global mean. To create two seeds the global mean is calculated and then the two seeds are created by moving away from the mean in opposite directions.

2.4 K-tree Example

Figures 1 to 3 are K-tree clusters in two dimensions. 1000 points were drawn from a random normal distribution with a mean of 1.0 and standard deviation of 0.3. The order of the K-tree, m, was 11. The grey dots represent the data set, the black dots represent the centroids and the lines represent the Voronoi tessellation of the centroids. Each of the data points contained within each tile of the tessellation are the nearest neighbours of the centroid and belong to the same cluster. It can be seen that the probability distribution is modelled at different granularities. The top level of the tree is level 1. It is the coarsest grained clustering. In this example it splits the distribution in three. Level 2 is more granular and splits the collection into 19 sub-clusters. The individual clusters in level 2 can only be arrived at through a nearest neighbour association with a parent cluster in level 1 of the tree. Level 3 is the deepest level in the tree consisting of cluster centroids. The 4th level is the data set of vectors that were inserted into the tree.

2.5 Building K-tree

The K-tree is constructed dynamically as data vectors arrive. Initially the tree contains a single empty root node at the leaf level. Vectors are inserted via a nearest neighbour search, terminating at the leaf level. The root of an empty tree is a leaf, so the first m data vectors are stored in the root, at which point the node becomes full. When the $m + 1$ vector arrives the root is split using k-means where $k = 2$, clustering all $m + 1$ vectors into two clusters. The two centroids that result from k-means are then promoted to become the centroids in a new root. The vectors associated with each centroid

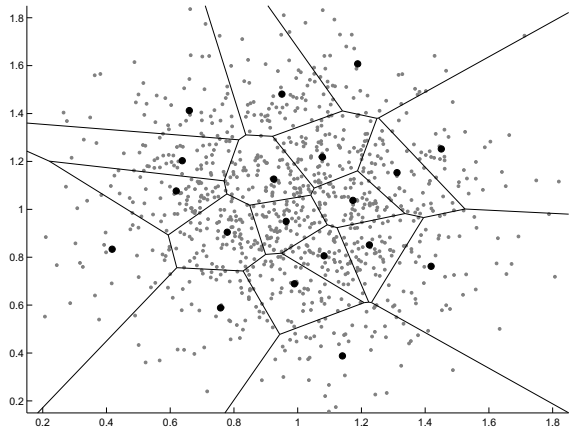


Figure 2: Level 2

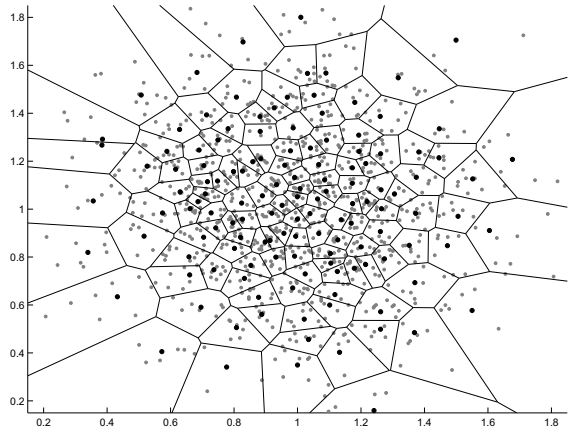


Figure 3: Level 3

are placed into a child node. This promotion process has created a new root and two leaf nodes in the tree. The tree is now two levels deep. Insertion of a new data vector follows a nearest neighbour search to find the closest centroid in the root. The vector is inserted into the associated child. When a new vector is inserted the centroids are updated recursively along the nearest neighbour search path, all the way back to the root node. The propagated means are weighted by the number of data vectors contained beneath them. This ensures that any centroid in K-tree is the mean vector of all the data vectors contained in the associated sub tree. This insertion process continues, splitting leaves when they become full, until the root node itself becomes full. K-means is then run on the root node containing centroids. The vectors in the new root node become centroids of centroids. As the tree grows, internal and leaf nodes are split in the same manner. The process of promotion can potentially propagate to cause a full root node at which point the construction of a new root follows and the tree depth is increased by one. At all times the tree is guaranteed to be height balanced. Although the tree is always height balanced nodes can contain as little as one vector. In this case the tree will contain many more levels than a tree where each node is half full. Figure 4 shows this construction process for a K-tree of order three $(m = 3)$.

2.6 Sparsity and K-tree

K-tree was originally designed to operate with dense vectors. When a sparse representation is used performance degrades even though there is significantly less data to process. The clusters in the top levels of the tree are means of most of the terms in the collection and are not sparse at all. The algorithm updates cluster centres along the insertion path in the tree. Since document vectors have very high dimensionality this becomes a very expensive process.

The medoid K-tree [6] extended the algorithm to use a sparse representation and replace centroids with document examples. This improved run-time performance and decreased memory usage. Unfortunately it decreased quality when using sparse document vectors. The document examples in the root of the tree were almost orthogonal to new documents being inserted. The documents were unlikely to have meaningful overlap in vocabulary.

The approach taken by De Vries and Geva at INEX 2008 [5] is a simple approach to dimensionality reduction or feature selection. It is called TF-IDF culling and it is performed by ranking terms. A rank is calculated by summing all weights for each term. The weights are the BM25 weight for each term in each document. This can also be explained as the sum of the column vector in the document by term matrix. The top n terms with the highest rank are selected, where n is the desired dimensionality. This works particularly well with term occurrences due to the Zipf law distribution of terms [19]. The collection frequency of a term is inversely proportional to its rank according to collection frequency. Most of the term weights are contained in the most frequent terms.

3 Random Indexing

Random Indexing (RI) [18] is an efficient, scalable and incremental approach to the word space model. Word space models use the distribution of terms to create high dimensional document vectors. The directions of these document vectors represent various semantic meanings and contexts.

Latent Semantic Analysis (LSA) [7] is a popular word space model. LSA creates context vectors from a document term occurrence matrix by performing Singular Value Decomposition (SVD). Dimensionality reduction is achieved through projection of the document term occurrence vectors onto the subspace spanned by the vectors with the largest Eigen values in the decomposition. This projection is optimal in the sense that it minimises the variance between the original matrix and the projected matrix. In contrast, Random Indexing first creates random context vectors of lower dimensionality, and then combines them to create a term occurrence matrix in the dimensionally reduced space. Each term

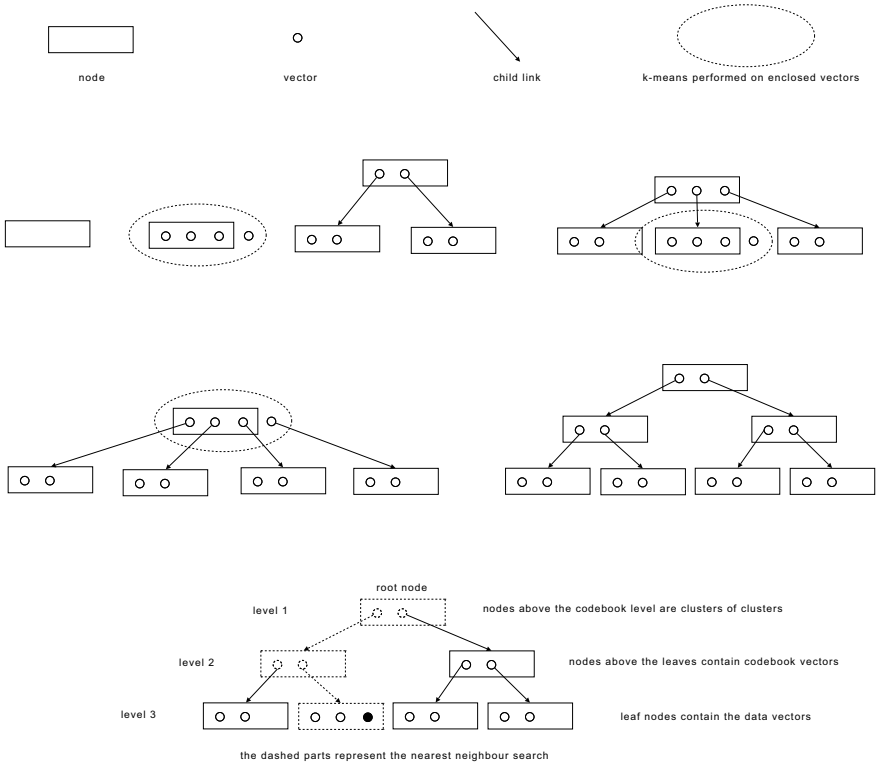


Figure 4: K-tree Construction

in the collection is assigned a random vector, and the document term occurrence vector is then a superposition of all the term random vectors. There is no matrix decomposition and hence the process is efficient.

The RI process is conceptually very different from LSA and does not have the same optimality properties. The context vectors used by RI should optimally be orthogonal. Nearly orthogonal vectors can be used and have been found to perform similarly [4]. These vectors can be drawn from a random Gaussian distribution. The Johnson and Linden-Strauss lemma [11] states that if points are projected into a randomly selected subspace of sufficiently high dimensionality, then the distances between the points are approximately preserved. The same topology that exists in the higher dimensional space is reflected in the lower dimensional randomly selected subspace. Consequently, RI offers low complexity dimensionality reduction while still preserving the topological relationships amongst document vectors.

3.1 Random Indexing Definition

In RI, each dimension in the original space is given a randomly generated index vector. The index vectors are high dimensional, sparse and ternary. Sparsity is controlled via a seed length that specifies the number of randomly selected non-zero dimensions. Ternary vectors consist of randomly distributed +1 and -1 values in the non-zero dimensions.

In the context of document clustering, RI can be viewed as a matrix multiplication of a document by term matrix D and a term by index-vector matrix I. Alternatively, I can be referred to as a random projection matrix. Each row vector in D represents a document, each row vector in I is an index vector, n is the number of documents, t is the number of terms and r is the dimensionality of the reduced spaced. R is the reduced matrix where each row vector represents a document.

$$D_{n \times t} I_{t \times r} = R_{n \times r} \tag{4}$$

RI has several advantages. It can be performed incrementally and on-line as data arrives. Any document can be indexed (i.e. encoded as an RI vector) independently from all other documents in the collection. This eliminates the need to build and store the entire document by term matrix. Additionally, newly encountered dimensions (terms) in the document collection are easily accommodated without having to recalculate the projection of previously encoded documents. In contrast, SVD requires global analysis where the number of documents and terms are fixed. The time complexity of RI is also very attractive. It is linear in the number of terms in a document and independent of collection size.

3.2 Choice of Index Vectors

The index vectors used in RI were chosen to be sparse and ternary. Ternary index vectors for RI were introduced by Achlioptas [2] as being well suited for database environments. The primary concern of sparse index vectors is reducing time and space complexity. Bingham and Mannila [4] run experiments indicating

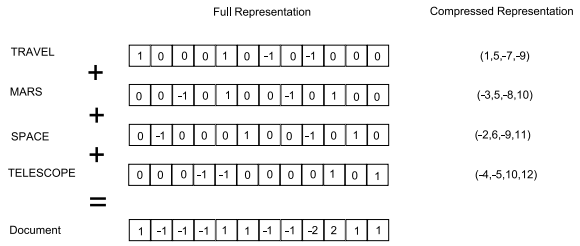


Figure 5: Random Indexing Example

that sparse index vectors do not affect the quality of results. This is not the only choice when creating index vectors. Kanerva [12] introduces binary spatter codes. Plate [15] explores Holographic Reduced Representations that consist of dense vectors with floating point values.

3.3 Random Indexing Example

In practice, to construct a document vector, the document vector is initially set to zero, and then the sparse index vector for each term in the document is added to the document vector. The weight of the added term index vector may be determined by TF-IDF or another weighting scheme. When all terms have been added, the document vector is normalised to unit length. There is no need to explicitly form the random projection matrix in Equation (4) up-front. The random index vectors for each term can be generated and stored as they are first encountered. The fact that each index vector is sparse means that the vectors use less memory to store and are faster to add.

The effect of this approach is that each document will have a particular signature that can be compared with other documents via cosine similarity. The document signature is thus a vector on the unit hyper-sphere.

In the simple scenario in Figure 5 the index vectors for the four words travel, mars, space and telescope, are added to the document vector as they are encountered in the text of the document. Afterwards, the document should be normalised.

The sparse index vectors can be efficiently stored by simply storing the position of the non-zero entries with the sign of the position indicating whether it is one or negative one.

3.4 Random Indexing K-tree

The time complexity of K-tree depends on the length of the document vectors. K-tree insertion incurs two costs, finding the appropriate leaf node for insertion and k-means invocation during node splits. It is therefore desirable to operate with lower dimensional vector representation.

The combination of RI with K-tree is a good fit. Both algorithms operate in an on-line and incremental mode. This allows it to track the distribution of data as it arrives and changes over time. K-tree insertions and deletions allow flexibility when tracking data in volatile and changing collections. Furthermore, K-tree performs best with dense vectors, such as those produced by RI.

4 Document Representation

The INEX 2008 XML Mining collection was used to complete the experiments. It contains 114,366 documents that are a subset of the XML Wikipedia corpus [8]. 15 different categories were provided for the documents.

Document content was represented with BM25 [17]. Stop words were removed and the remaining terms were stemmed using the Porter algorithm [16]. BM25 is determined by term distributions within each document and the entire collection. BM25 works with similar concepts as TF-IDF except that is has two tuning parameters. The BM25 tuning parameters were set to the same values as used for TREC [17], $K1 = 2$ and $b = 0.75$. $K1$ influences the effect of term frequency and b influences document length.

Links were represented using LF-IDF [5]. This resulted in a document-to-document link matrix. If there is a link between documents i and j then a value of one is added to position i, j and j, i in the matrix. If two documents both link to each other a value of two is recorded in their respective vectors. Each row vector of the matrix represents a document as a vector of link frequencies to and from other documents.

The motivation behind this representation is that documents with similar content will link to similar documents. For example, in the current Wikipedia both car manufacturers BMW and Jaguar link to the Automotive Industry document. Link frequencies were weighted with the same Inverse Document Frequency heuristic from TF-IDF. The idea is to decrease the weight of highly frequent links and increase the weight of less frequent links. Links to year documents in the Wikipedia are examples of "stop links" that are weighted down by this heuristic. Unlike term frequencies in TF-IDF the link frequencies in LF-IDF are not normalised. De Vries and Geva [5] found that normalising link frequencies decreased classification performance.

When document and link representations are combined they are both converted to unit vectors and concatenated. Converting each representation to unit vectors ensures that the weights of one representation do not dominate the other. De Vries and Geva [5] found this to be effective for classification.

5 Experimental Setup

Experiments have been run to measure the quality difference between various configurations of K-tree. Section 2.3 describes the modifications made to K-tree. Table 1 lists all the configurations tested.

The following conditions were used when running the experiments.

1. Each K-tree configuration was run a total of 20 times.
2. The documents were inserted in a different random order each time K-tree is built.
3. If RI was used, the index vectors were generated statistically independently each time K-tree was built.
4. For each K-tree built, k-means++ [3] was run 20 times on the codebook vectors to create 15 clusters.
5. All document vectors were unitised after performing dimensionality reduction.

The conditions listed above resulted in 400 measurements for each K-tree configuration. For each of the 20 K-trees built, k-means++ was run 20 times. The repetition of the experiments is to measure the variance caused by the random insertion order into K-tree, the randomised seeding process in k-means in the modified K-tree and the randomised seeding process of k-means++.

Assessment of clustering quality is based on the INEX XML Mining track. The set of 114,366 documents, belonging to 15 classes were used to evaluate clustering quality of INEX submissions. The cluster labels are taken from the Wikipedia itself. K-tree generates clusters in an unsupervised manner, and it is not necessarily going to produce 15 clusters at a particular level in the tree. In order to re-use the INEX test collection, it was necessary to post process the K-tree and to reduce a cluster level in the tree to 15 clusters by using k-means++. Note that this is a low cost operation involving only a small number of vectors, which is not required in an ordinary application. It is done for the sole purpose of producing comparable results with the INEX benchmark data. The same approach was taken at INEX 2008 by De Vries and Geva [5]. For a comparison of entropy and purity to be meaningful they have to be measured on the same number of clusters.

Micro averaged purity and entropy are compared. Micro averaging weights the score of a cluster by its size. Purity and entropy are calculated by comparing the clustering solution to the labels provided. A higher purity score indicates a higher quality solution because the clusters are more pure with respect to the ground truth. A lower entropy score indicates a higher quality solution because there is more order with respect to the ground truth.

6 Experimental Results

Tables 3 to 7 contain results for the K-tree configurations tested listed in Table 1. Table 2 lists the meaning of the symbols used. Figures 6 and 7 are graphical representations of the average micro purity and entropy.

The unmodified K-tree using TF-IDF culling and BM25 had unexpected results as seen in Table 3. The average micro purity and entropy peaked at 400 dimensions. Performing this dimensionality reduction at these lower dimensions had not been performed before. This is an interesting and unexpected result and future experiments will need to determine if the phenomenon occurs in different corpora.

Improvements in micro purity have been tested for significance via t-tests. The null hypothesis is that both results come from the same distribution with the same mean. In this case they are not significantly different. If the null hypothesis is rejected then the difference is statistically significant.

The modifications made to K-tree for use with RI had a significant impact. The unmodified K-tree and modified K-tree were compared. Specifically, configurations B and D, and configurations C and E were tested against each other. All dimensions were compared against each other. The improved performance of the modified K-tree was statistically significant for all dimensions (100 vs 100, 200 vs 200 and so on) with a p-value of 0 or extremely close to 0 ($p < 1 \times 10^{-100}$).

The modified K-tree using RI was tested with two representations. Configurations D and E were tested at all dimensions. The null hypothesis was rejected at all dimensions except 10000. This means that BM25 performed significantly better than the BM25 + LF-IDF representation at all dimensions except 10000. At 10000 dimensions the difference was not considered statistically significant with a p-value of 0.3. The increased performance of this representation in classification did not apply to clustering when using RI. The LF-IDF representation may be interfering with the BM25 representation and approaches such as reducing the weight of LF-IDF in the RI process or performing RI separately on each representation and then concatenating the reduced vectors may improve performance. Running k-means on the full sparse vectors will also indicate if RI is responsible for this. Further experimentation is required to provide more evidence for this result.

The unexpected results in configuration A were tested against the best RI configuration, E. The highest average at 400 dimensions in configuration A was tested against all dimensions in configuration E (400 vs 100, 400 vs 200, 400 vs 400, 400 vs 1000 and so on). The RI K-tree, configuration E, became statistically more significant at 2000 dimensions with a p-value of 1.48×10^{-6} and thus rejected the null hypothesis. For dimensions 4000 through 10000, the performance difference was statistically significant, with a p-value of 0 in all cases. Thus, RI K-tree improves results, even over the unexpected high results of configuration A, by embedding the original 200,000 dimensional term space into at least a 2000 dimension reduced space.

ID	K-tree	Representation
A	Unmodified	TF-IDF Culling, BM25
B	Unmodified	RI, BM25 + LF-IDF
C	Unmodified	RI, BM25
D	Modified	RI, BM25 + LF-IDF
E	Modified	RI, BM25

Table 1: K-tree Test Configurations

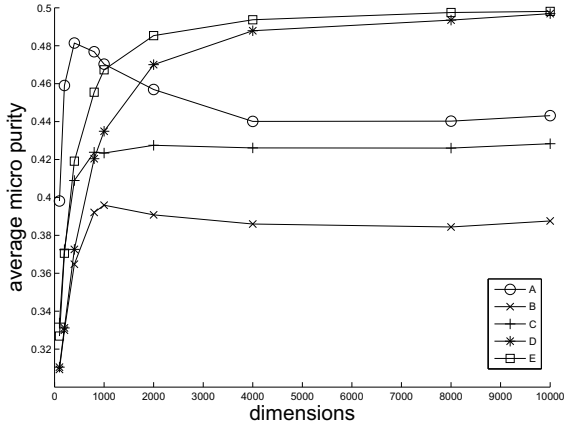


Figure 6: Purity Versus Dimensions

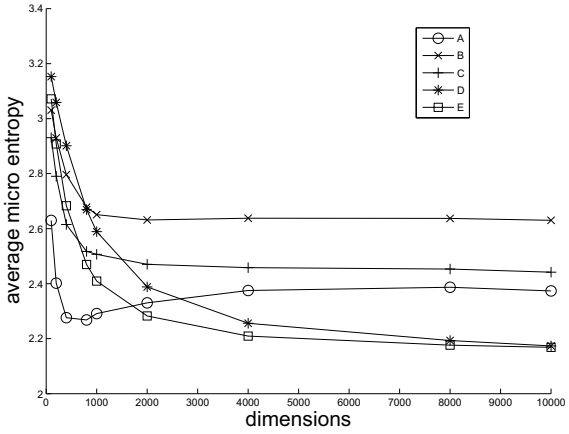


Figure 7: Entropy Versus Dimensions

6.1 INEX Results

The INEX XML Mining track is a collaborative evaluation forum where research teams improve approaches in supervised and unsupervised machine learning with XML documents. Participants make submissions and the evaluation results are later released.

The RI K-tree in configuration E performs on average at a comparable level to the best results submitted to the INEX 2008 XML Mining track. The top two results from the track had a micro purity of 0.49 and 0.50. These are not average scores for the approaches but the best results participants found. The RI K-tree in configuration E had a maximum micro entropy of 0.55. This is 10% greater than the INEX submissions.

Symbol	Meaning
α	Average Micro Entropy
β	Standard Deviation of α
γ	Average Micro Purity
δ	Standard Deviation of γ

Table 2: Symbols for Results

Dimensions	α	β	γ	δ
100	2.6299	0.0194	0.3981	0.0067
200	2.4018	0.0207	0.4590	0.0085
400	2.2762	0.0263	0.4814	0.0093
800	2.2680	0.0481	0.4768	0.0155
1000	2.2911	0.0600	0.4703	0.0192
2000	2.3302	0.0821	0.4569	0.0254
4000	2.3751	0.1103	0.4401	0.0331
8000	2.3868	0.1068	0.4402	0.0300
10000	2.3735	0.1062	0.4431	0.0306

Table 3: A: Unmodified K-tree, TF-IDF Culling, BM25

Dimensions	α	β	γ	δ
100	3.0307	0.0149	0.3093	0.0045
200	2.9295	0.0206	0.3300	0.0079
400	2.7962	0.0379	0.3648	0.0143
800	2.6781	0.0718	0.3921	0.0236
1000	2.6509	0.0842	0.3959	0.0260
2000	2.6315	0.1262	0.3908	0.0345
4000	2.6380	0.1451	0.3860	0.0356
8000	2.6371	0.1571	0.3844	0.0382
10000	2.6302	0.1540	0.3876	0.0385

Table 4: B: Unmodified K-tree, Random Indexing, BM25 + LF-IDF

Dimensions	α	β	γ	δ
100	2.9308	0.0213	0.3337	0.0089
200	2.7902	0.0335	0.3724	0.0126
400	2.6151	0.0417	0.4089	0.0116
800	2.5170	0.0703	0.4238	0.0197
1000	2.5066	0.0858	0.4234	0.0240
2000	2.4701	0.0938	0.4275	0.0258
4000	2.4581	0.0979	0.4261	0.0271
8000	2.4530	0.1139	0.4260	0.0318
10000	2.4417	0.1019	0.4283	0.0283

Table 5: C: Unmodified K-tree, Random Indexing, BM25

Dimensions	α	β	γ	δ
100	3.1527	0.0227	0.3105	0.0047
200	3.0589	0.0266	0.3312	0.0065
400	2.9014	0.0259	0.3726	0.0065
800	2.6690	0.0336	0.4204	0.0085
1000	2.5890	0.0319	0.4349	0.0090
2000	2.3882	0.0428	0.4700	0.0129
4000	2.2558	0.0443	0.4879	0.0144
8000	2.1933	0.0473	0.4935	0.0162
10000	2.1735	0.0496	0.4969	0.0171

Table 6: D: Modified K-tree, Random Indexing, BM25 + LF-IDF

Dimensions	α	β	γ	δ
100	3.0717	0.0263	0.3269	0.0074
200	2.9078	0.0291	0.3706	0.0087
400	2.6832	0.0293	0.4191	0.0077
800	2.4696	0.0350	0.4555	0.0106
1000	2.4093	0.0399	0.4673	0.0115
2000	2.2826	0.0422	0.4853	0.0137
4000	2.2094	0.0416	0.4937	0.0141
8000	2.1764	0.0429	0.4975	0.0149
10000	2.1686	0.0440	0.4981	0.0161

Table 7: E: Modified K-tree, Random Indexing, BM25

7 Conclusion

RI K-tree was introduced as an attractive approach for large scale document clustering. This is the first time RI and K-tree have been combined. The results show that RI K-tree improves quality of clustering results, even over the unexpected results found when using TF-IDF culling. Further experiments are required to determine if the unexpected effect of TF-IDF culling at low dimensions is an anomaly or actually exists in many collections. Additionally, RI K-tree is an efficient and high quality approach to overcome previous problems with sparse representations when using K-tree. Unfortunately the combination of BM25 and LF-IDF representations did not improve results in clustering as they did in earlier classification results.

References

[1] K-tree project page, http://ktree.sourceforge.net. 2009.

[2] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, Volume 66, Number 4, pages 671–687, 2003.

[3] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[4] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, New York, NY, USA, 2001. ACM.

[5] C.M. De Vries and S. Geva. Document clustering with k-tree. *Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers*, pages 420–431, 2009.

[6] C.M. De Vries and S. Geva. K-tree: large scale document clustering. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 718–719, New York, NY, USA, 2009. ACM.

[7] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, Volume 41, Number 6, pages 391–407, 1990.

[8] L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. *SIGIR Forum*, 2006.

[9] A. Gersho and R.M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, 1993.

[10] S. Geva. K-tree: a height balanced tree structured vector quantizer. *Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing X, 2000.*, Volume 1, pages 271–280 vol.1, 2000.

[11] W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, Volume 26, Number 189-206, pages 1–1, 1984.

[12] P. Kanerva. The spatter code for encoding concepts at many levels. In *ICANN94, Proceedings of the International Conference on Artificial Neural Networks*, 1994.

[13] G. Karypis. CLUTO-A Clustering Toolkit. 2002.

[14] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, Volume 28, Number 2, pages 129–137, March 1982.

[15] T.A. Plate. *Distributed representations and nested compositional structure*. Ph.D. thesis, 1994.

[16] M.F. Porter. An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, Volume 40, Number 3, pages 211–218, 2006.

[17] S.E. Robertson and K.S. Jones. Simple, proven approaches to text retrieval. *Update*, 1997.

[18] M. Sahlgren. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*, 2005.

[19] G.K. Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Addison-Wesley Press, 1949.

Modelling Disagreement Between Judges for Information Retrieval System Evaluation

Andrew Turpin *Falk Scholer*
School of Computer Science & IT
RMIT University
GPO Box 2476
Melbourne 3001
{*andrew.turpin,falk.scholer*}*@rmit.edu.au*

Abstract *The batch evaluation of information retrieval systems typically makes use of a testbed consisting of a collection of documents, a set of queries, and for each query, a set of judgements indicating which documents are relevant. This paper presents a probabilistic model for predicting IR system rankings in a batch experiment when using document relevance assessments from different judges, using the precision-at-n family of metrics. In particular, if a new judge agrees with the original judge with an agreement rate of α, then a probability distribution of the difference between the P@n scores of the two systems is derived in terms of α.*

We then examine how the model could be used to predict system performance based on user evaluation of two IR systems, given a previous batch assessment of the two systems together with a measure of the agreement between the users and the judges used to generate the original batch relevance judgements. From the analysis of data collected in previous user experiments, it can be seen that simple agreement (α) between users varies widely between search tasks and information needs. A practical choice of parameters for the model from the available data is therefore difficult. We conclude that gathering agreement rates from users of a live search system requires careful consideration of topic and task effects.

Keywords Information retrieval; Evaluation; User studies

1 Introduction

To test whether one information retrieval system is better than another, researchers either adopt the Cranfield methodology of *batch assessment*, or test their systems with humans in a *user experiment*. The batch assessment methodology requires a collection of documents, a set of queries, and, for each query, a judgment on some or all of the documents indicating whether they are relevant to that query or not. Assessing systems, therefore, is a matter of running each query to get a ranked list of documents, noting which is relevant or not according to the relevance judgements, and summarising the ranked list of relevance values into an overall performance score. The alternative approach requires a group of human users, the designing of a suitable experiment that controls for any biases you may wish to exclude (for example, education or computer literacy), defining an outcome metric (for example, time taken to find a useful answer document), and then measuring how users perform with different retrieval systems.

The batch method is by far the cheapest, easiest, and more repeatable of the two methodologies, and as such has dominated IR research for the last three decades. Recently, however, a series of papers has shown that the two methodologies do not necessarily reach the same conclusions regarding relative system performance. That is, if batch experiments show system A to be better than system B, user experiments may show there is no difference between the systems [1, 2, 5, 6, 7, 11, 13], or that system B is superior [12].

Our recent work has focussed on trying to quantify and rectify this seeming mismatch between the two experimental approaches [9, 10]. A key potential source of mismatch is the different relevance criteria of the judges used to construct the "ground truth" batch judgements, and the users in the user based experiment. Determining the relevance of a document to a query is a complex, multi-faceted task [4]. It often depends on the reason that the relevance judgement is being made, a *task effect*; the query itself, a *topic effect*; and of course the person making the judgement, a *judge effect*. There are many other factors that influence human judgement in general, including motivational biases, preconceptions, salience and availability, and perseverance [8]; these may all have additional effects on the criteria that judges use to decide if a document is relevant or not.

In this paper we develop a probabilistic model of agreement between relevance judges, and derive how this is expected to affect the results of a batch-based evaluation of IR system performance. We then investigate how agreement values could be obtained from a user study, so that the model might be used to transfer the outcomes from a batch experiment to a new user

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

population. Analysis of the data from the user experiments shows that agreement between users is subject to substantial variation from both task and topic effects.

2 Preliminaries

Let a batch evaluation testbed consist of: a set of documents, $D = \{d_1, \ldots, d_{|D|}\}$; N queries, $Q = \{q_1, \ldots, q_N\}$; and for each query-document pair a relevance judgement

$$R(d_i, q) = \begin{cases} 1, & \text{document } d_i \text{ is relevant to query } q, \\ 0, & \text{otherwise.} \end{cases}$$

A *system*, returns a ranked list of m documents $[d_{i_1}, \ldots, d_{i_m}]$ for query q, which are mapped to a vector of relevance judgements in retrieved order $J = [R(d_{i_1}, q), \ldots, R(d_{i_m}, q)]$.

Judgement vectors can be reduced to a single score in various ways, and many different performance metrics have been proposed for representing the overall performance of IR systems. In this paper we use the precision-at-n documents metric, usually written P@n, which is the proportion of the top n documents of the list that are relevant. Formally,

$$\text{P@}n = \frac{1}{n} \sum_{i=1}^{n} J[i].$$

The *score* for a system is the mean P@n over all queries in the corpus. A system with a statistically significantly higher score than another is defined to be superior in the batch mode of system comparison, and is assumed to be superior in the user mode of comparison. This has been shown to be the case for P@1 in user experiments when the measured outcome is "time to save first relevant document" [9], and "satisfaction" [7], and for the tasks and users employed in those studies.

For example, if System B has a score of P@3=0.33, then on average only 1 of the top 3 documents is relevant, while if System A has a score of P@3=1.0, then the top 3 documents are always relevant for all test queries. It is implicitly assumed in IR experimentation that System A is superior to System B.

3 Modelling changes in judges

Using a testbed such as those from the Text REtrieval Conference [16] will yield system rankings that should be comparable with other experiments based on different queries with the same collection [15] or – with suitable standardisation – across different queries and collections [17]. That is, if System A is found to be statistically significantly better than System B when running a batch experiment, then this relationship should in general continue to hold for different queries, and collections. If, however, you kept the same set of documents and queries, but used an alternate relevance judge, so that $R(d, q)$ became $R'(d, q)$, then system rankings may alter. In particular, the P@n score for System B might increase, and the P@n score for System A might decrease, so that B becomes the superior system.

i	$J_A[i]$	$J_B[i]$	δ_i	$J'_A[i]$	$J'_B[i]$	δ'_i
1	1	0	1	0	0	0
2	1	0	1	1	1	0
3	0	0	0	0	1	-1
4	1	1	0	1	1	0
5	1	0	1	0	0	0
		$\Delta(5) = 3/5$			$\Delta'(5) = -1/5$	

Table 1: An example calculation of $\Delta(5)$ and $\Delta'(5)$ when System A has a P@5 value of 0.8 and 0.4 with different judgements, and System B has P@5 of 0.2 and 0.6 respectively.

If we assume that the new judge has some probability of agreeing with the judge used to build the original corpus (independently for any query-document pair), then we could derive a probability distribution of the new scores for System A and B. In turn, this can be used to derive a probability distribution on the difference between the two systems, and we can hypothesise about how transferable system rankings are between judges.

Definition 1 *Let J_A be the relevance vector given by System A for query q using corpus judgements $R(d, q)$, and J_B the relevance vector given by System B. For the same document lists, let J'_A and J'_B be the relevance vector given using judgements $R'(d, q)$ for System A and B respectively.*

We can now define the difference in P@n scores between the systems for query q using either set of relevance judgements, and then derive a probability distribution for that difference based on agreement probabilities between judges.

Definition 2 *For some ranked position $1 \leq i \leq n$, let $\delta_i = J_A[i] - J_B[i]$, and $\delta'_i = J'_A[i] - J'_B[i]$. Then the difference in P@n scores for the systems using either set of relevance judgements is given by $\Delta(n) = \sum_{i=1}^{n} \delta_i / n$ and $\Delta'(n) = \sum_{i=1}^{n} \delta'_i / n$ respectively.*

Table 1 shows an example of how $\Delta(5)$ and $\Delta'(5)$ is calculated. In this instance, using the second set of judgements has decreased System A's superiority to $\Delta'(5) = -0.2$, that is, System B is now apparently better than System A.

Without loss of generality, we will from now assume that System A has a higher P@n score than System B using the corpus judgements J_A and J_B. Thus we are interested in deriving a probability distribution for $\Delta'(n)$, and in particular the probability that $\Delta'(n) \geq 0$; that is, System A remains superior with a new set of judgements.

Definition 3 *Let α_0 be the probability that the new judge agrees with a $R(d, q) = 0$ judgement in the*

$J_A[i]$	$J_B[i]$	$J'_A[i]$	$J'_B[i]$	δ'_i	Probability	Probability$\times\delta'_i$
0	0	0	0	0	$\alpha_0\alpha_0$	0
0	0	0	1	-1	α_0(1-α_0)	-α_0(1-α_0)
0	0	1	0	1	(1-α_0)α_0	α_0(1-α_0)
0	0	1	1	0	(1-α_0)(1-α_0)	0
						$E_{00} = 0$
0	1	0	0	0	α_0(1-α_1)	0
0	1	0	1	-1	$\alpha_0\alpha_1$	-$\alpha_0\alpha_1$
0	1	1	0	1	(1-α_0)(1-α_1)	(1-α_0)(1-α_1)
0	1	1	1	0	(1-α_0)α_1	0
						$E_{01} = 1 - \alpha_0 - \alpha_1$
1	0	0	0	0	(1-α_1)α_0	0
1	0	0	1	-1	(1-α_1)(1-α_0)	-(1-α_0)(1-α_1)
1	0	1	0	1	$\alpha_1\alpha_0$	$\alpha_0\alpha_1$
1	0	1	1	0	α_1(1-α_0)	0
						$E_{10} = \alpha_0 + \alpha_1 - 1$
1	1	0	0	0	(1-α_1)(1-α_1)	0
1	1	0	1	-1	(1-α_1)α_1	-α_1(1-α_1)
1	1	1	0	1	α_1(1-α_1)	α_1(1-α_1)
1	1	1	1	0	$\alpha_1\alpha_1$	0
						$E_{11} = 0$

Table 2: All possible cases for judgement of a document in a ranked list at position i by the corpus and new judges, with their corresponding probabilities. For each possible pair of J_A and J_B values, the expected value of δ'_i, labelled E_x for each x, is computed as the sum of the four entries above it.

corpus, thus $R'(d,q) = 0$, and α_1 be the probability that the new judge agrees with a $R(d,q) = 1$ judgement in the corpus, hence $R'(d,q) = 1$.

For any rank i in the top n documents for a single query, the entries in the relevance vectors for System A and System B for that position is either: $J_A[i] = 0$ and $J_B[i] = 0$, both systems returned an irrelevant document in that position; $J_A[i] = 1$ and $J_B[i] = 1$, both system returned a relevant document in that position; and the two discriminating cases $J_A[i] = 1$ and $J_B[i] = 0$, or $J_A[i] = 0$ and $J_B[i] = 1$. Table 2 shows, for each of these four possible cases, the four possible relevance vector entries at a particular rank i that might result using different judgements ($J'_A[i]$ and $J'_B[i]$). In addition to the δ'_i values for each case, the probability of realising each combination is given in the second last column, which is the product of the appropriate agreement probabilities. For example, in the first row the probability of $J_A[i] = 0$ and $J'_A[i] = 0$ is α_0, and $J_B[i] = 0$ and $J'_B[i] = 0$ is also α_0, so total probability of that event is $\alpha_0\alpha_0$. In the second row, $J_A[i] = J'_A[i] = 0$, but $J_B[i] = 0$ is judged as $J'_B[i] = 1$ with probability $(1 - \alpha_0)$, so the total probability is $\alpha_0(1 - \alpha_0)$. The final column is summed for each of the four possible cases of $J_A[i]$ and $J_B[i]$ to give the expected value of δ'_i for that case, labelled E_{00}, E_{01}, E_{10}, and E_{11} respectively.

Definition 4 *For a given query q and Systems A and B, let c_{00} be the number of rank positions in the top n for query q where $J_A[i] = 0$ and $J_B[i] = 0$, and likewise for c_{10}, c_{01} and c_{11}. That is, $c_{xy} = |\{J_A[i] = x \text{ and } J_B[i] = y, 1 \le i \le n\}|$. Note, $\Delta(n) = (c_{10} - c_{01})/n$.*

For each position in a ranked list, $E_{J_A[i]J_B[i]}$ gives the expected value of δ'_i, and so the expectation of $\Delta'(n)$ can be calculated as:

$$\begin{aligned} E[\Delta'(n)] &= E\left[\sum_{i=1}^{n} \delta'_i/n\right] \\ &= (c_{00}E_{00} + c_{01}E_{01} + c_{10}E_{10} \\ &\quad + c_{11}E_{11})/n \\ &= (1 - \alpha_0 - \alpha_1)(c_{01} - c_{10})/n \\ &= (\alpha_0 + \alpha_1 - 1)\Delta(n) \end{aligned} \quad (1)$$

Intuitively this makes sense. If new judges agree perfectly with the corpus judges, then $\alpha_0 = \alpha_1 = 1$, then $E[\Delta'(n)] = \Delta(n)$: there is no expected difference in the system's scores with either judgement set. If new judges disagree completely with the corpus judges, then $\alpha_0 = \alpha_1 = 0$, then $E[\Delta'(n)] = -\Delta(n)$: that is, the expected system scores are the reverse of the original.

We can also compute the variance of $\Delta'(n)$. Recall that $\text{Var}(X) = E(X^2) - E(X)^2$ by definition, so:

$$
\begin{aligned}
& \text{Var}(\Delta'(n)) \\
= \; & \text{Var}\left(\sum_{i=1}^{n} \delta_i'/n\right) \\
= \; & \sum_{i=1}^{n} \text{Var}(\delta_i')/n^2 \\
= \; & \frac{1}{n^2}\sum_{i=1}^{n}\left(E[(\delta_i')^2] - E[\delta_i']^2\right) \\
= \; & (c_{00}(2\alpha_0(1-\alpha_0)) \\
& +c_{11}(2\alpha_1(1-\alpha_1)) \\
& +(c_{01}+c_{10})(1-\alpha_0-\alpha_1+2\alpha_0\alpha_1) \\
& -(1-\alpha_0-\alpha_1)^2(c_{01}-c_{10})^2)/n^2 \qquad (2)
\end{aligned}
$$

Equations 1 and 2 are for a single query, q, but are easily extended to a score computed over a set of N queries because the P@n metric assigns equal weight to all ranked positions. That is, computing the mean P@n value over the top n documents retrieved for N queries is the same as computing P@Nn for a concatenation of the N $J[1..n]$ relevance vectors for each query. If we use the notation J_i to represent the relevance vector J for query i, and $J_S = J_1[1..n]J_2[1..n]..J_N[1..n]$ to represent the concatenation of the first n elements of all J_i's, then:

$$
\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N}(\text{P@}n \text{ of } J_i) &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n}\sum_{j=1}^{n} J_i[j] \\
&= \frac{1}{Nn}\sum_{k=1}^{nN} J_S[k] \\
&= \text{P@}Nn \text{ of } J_S.
\end{aligned}
$$

Henceforth we will limit our discussions to the single query case for notational convenience.

Equation 2 contains c_{xy} terms, which will alter depending on system, query and judgements. However, if we fix n, or assume the maximum possible separation between systems on the corpus, the equations can be simplified to something immediately useful.

3.1 The P@1 case

When considering P@1, the expression for $\text{Var}(\Delta'(n))$ simplifies to something manageable. As we are interested in the case where System A is better than System B on query q using the corpus judgements, then P@1=1 for System A and for System B, P@1=0. Hence $c_{00} = c_{11} = c_{01} = 0$, $c_{10} = 1$, $n = \Delta(n) = 1$, and

$$
\begin{aligned}
E[\Delta'(n)] &= \alpha_0 + \alpha_1 - 1 \\
\text{Var}(\Delta'(n)) &= \alpha_0 + \alpha_1 - \alpha_0^2 - \alpha_1^2.
\end{aligned}
$$

Assuming $\Delta'(n)$ is normally distributed with mean $E[\Delta'(n)]$ and a standard deviation of $\sqrt{\text{Var}(\Delta'(n))}$, then we can compute $Pr[\Delta'(n) \geq 0]$ which is shown in Figure 1. To be more than 50% confident that a new set of judgements on the corpus will keep System A as superior with the P@1 metric, the sum of α_0 and α_1 must be larger than 1 (approximately). To be 95% confident that System A will remain superior, both agreement probabilities must be over 80%.

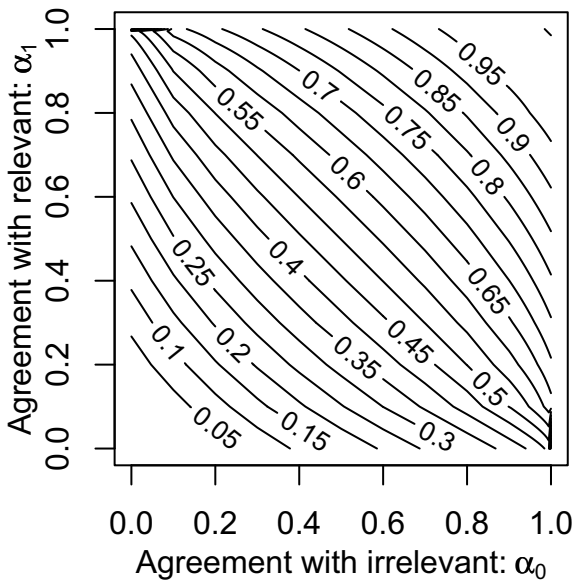


Figure 1: Contour plot of the probability of $\Delta'(1)$ exceeding zero (hence System A remaining superior) with the P@1 score, when the corpus is re-judged by a judge that agrees α_0 and α_1 proportion of the time with the original judge's 0 and 1 judgements, respectively.

3.2 The extreme case

Just as for the P@1 case, assuming P@n=1 for System A and P@n=0 for System B allows simplification of Equations 1 and 2 as all of c_{00}, c_{11} and c_{01} are 0, and $c_{10} = n$.

Thus

$$
\begin{aligned}
E[\Delta'(n)] &= \alpha_0 + \alpha_1 - 1 \\
\text{Var}(\Delta'(n)) &= ((1-\alpha_0-\alpha_1+2\alpha_0\alpha_1) \\
&\quad -(\alpha_0+\alpha_1-1)^2)/n
\end{aligned}
$$

If we assume that $\alpha_0 = \alpha_1 = \alpha$, then we can plot $E[\Delta(n)]$ and a 95% confidence interval as $\pm 1.96\sqrt{\text{Var}(\Delta'(n))}$ for different n values. This is shown in Figure 2.

To be 95% sure that System A remains superior with new judgements, agreement must be at least 90% for P@1 (intersection of dark grey ellipse and the 0 line), 75% for P@5 (intersection of medium grey ellipse and the 0 line), and 70% for P@10 (intersection of light grey ellipse and the 0 line).

3.3 Other cases

It is possible to simplify Equations 1 and 2 for other values of n where System A and System B are not separated extremely, that is, when the gap between System A and System B is less than one: $\Delta(n) < 1$. The technique involves labelling each possible combination of $J_A[i]$ and $J_B[i]$ for all i, but is omitted from this

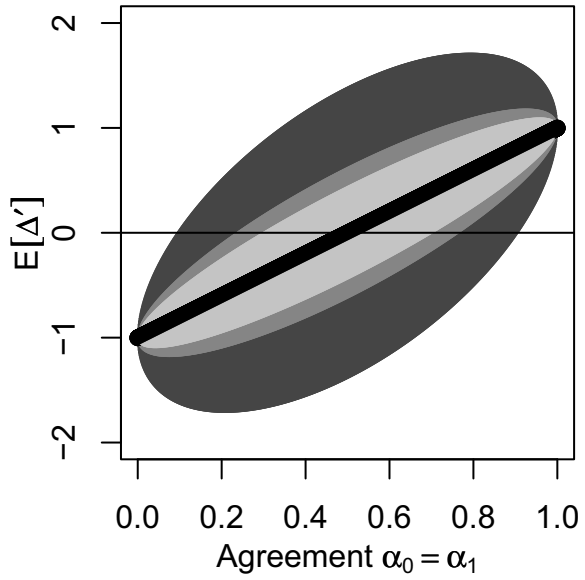


Figure 2: Expected $\Delta'(n)$ values (black) and 95% confidence limits for $n = 1$ (dark grey), $n = 5$ (medium grey) and $n = 10$ (light grey) assuming P@1=1 for System A and P@1=0 for System B.

paper as we concentrate on the P@1 metric in our user studies.

4 Practical considerations

In this section of the paper we turn our attention to an investigation of the likely values α_0 and α_1 when users conduct a web-based search task. In particular, we examine data from one of our previous user studies that involved both document judgements and search-and-click judgements, and see if α values are stable across different topics and tasks for a given pair of users.

4.1 User experiment

Participants for our user study were recruited form RMIT University. All were postgraduate or undergraduate students studying for degrees in computer science and information technology. As a result, most were very familiar with searching for information on the web; in a pre-experiment questionnaire the average user indicated that they search "once or more a day". Experiments were carried out in compliance with the RMIT University Human Research Ethics Committee. 40 users participated in the study; however, three were unable to complete the full experiments, and are therefore excluded from the analysis.

Participants were asked to carry out two tasks: a judging task, and a search task. For both, documents and topics were sourced from the TREC GOV2 collection, a crawl of 426 Gb of data from the `.gov` domain from 2004 [3].

Judging task: For the first task, participants were asked to imagine that they are writing a report, based on a provided information need, and to mark documents that were presented as relevant or not relevant for inclusion in the report. Participants were asked to carry out this task for three TREC topics (numbers 707, 770 and 771); the description and narrative fields of the topics were displayed to users as information needs. Participants were therefore making binary decisions about relevance, when presented with documents that had previously been judged by TREC assessors on a three-point scale (not relevant; relevant; and highly relevant). There was no time constraint for making decisions for the judging task. However, it became clear that carrying out the task for all three topics resulted in severe fatigue effects. The third topic completed by each user is therefore removed from the analysis.

Searching task: Participants also carried out a searching task. Here, when presented with an information need, users were asked to search for and identify a relevant answer document as quickly as possible. Users could enter a single query to a search system, designed to be similar in appearance to popular commercial search engines such as Google, Yahoo! or Bing. Unknown to the user, for each topic they were assigned to a system of a particular quality; that is, the system would return a ranked answer list with a pre-determined P@1 level. For this task, 24 informational topics were chosen from TREC topics 700–850 (topics developed for use with the GOV2 collection). To construct the P@1 controlled lists, judged documents were selected from the two highest-performing runs submitted to the TREC terabyte track in 2004, 2005 and 2006. That is, all documents used in the lists could plausibly be returned in response to the topics by a modern information retrieval system.

After being presented with a search results list, a user could select a document for viewing. They could then take one of two actions: save the document as a relevant answer; or close the document, and return to the results list. In the analysis below, these actions are taken as judgements of the relevance or non-relevance of the document, respectively.

Note that the user studies were not explicitly designed to answer the questions raised in this paper; rather we are retrospectively analysing the data to get insights into likely values of α_0 and α_1. Full details of the user studies are available in previous papers [9, 10].

4.2 Agreement on the judging task

Figure 3 shows the distribution of agreement values between all pairs of users for the judging task. As agreement is not symmetrical [14], each user pair is counted twice, usually with different values. As can be seen, agreement varies anywhere from 100% down to 7.7% for users 14 and 11 on α_0.

Perhaps of more interest is the difference in agreement for any user pair that judged the same two topics. Figure 4 shows that on any two topics, both α_0 and

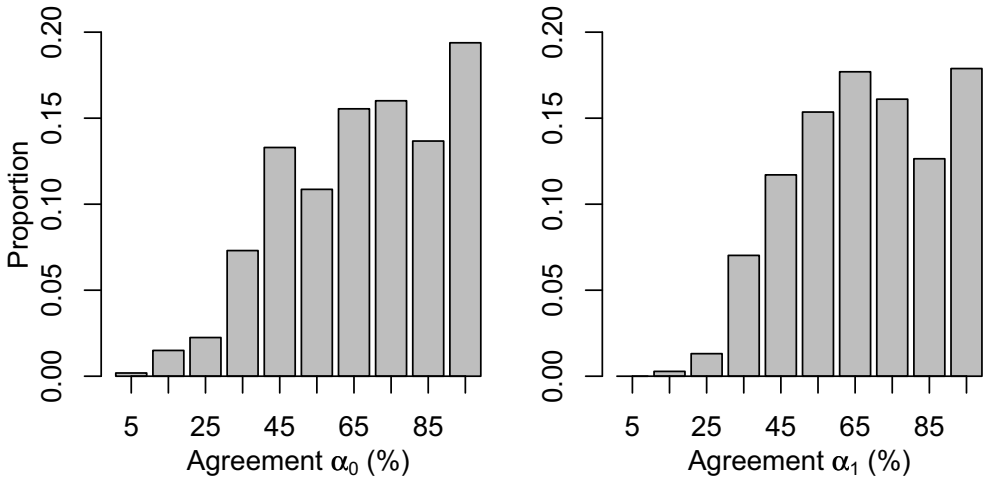


Figure 3: Distribution of agreement amongst all pairs of users on the judging task.

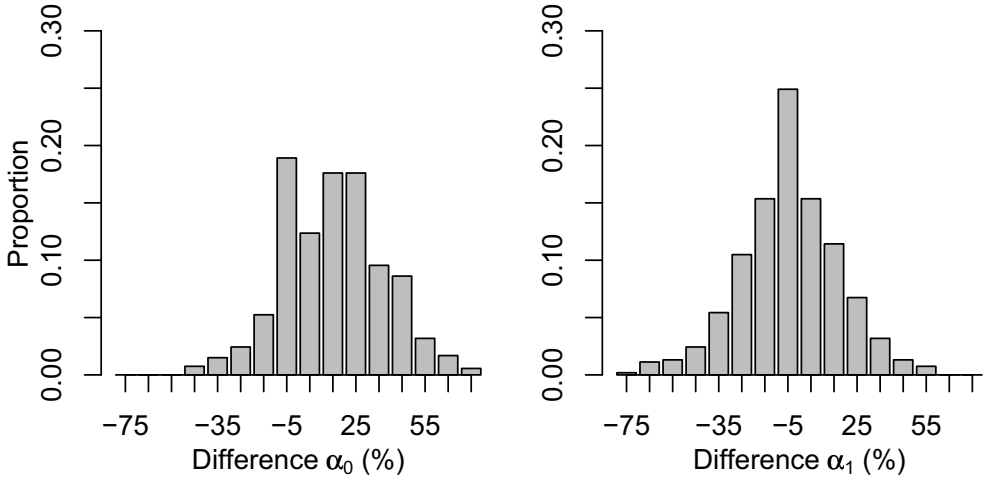


Figure 4: Distribution of the difference in agreement amongst pairs of users on the judging task.

α_1 can vary widely in the judging task. This makes it difficult to choose a representative agreement value for any pair of judges. Note that as we had to remove the third topic judged for each user from the data set, not all pairs of users completed the same two topics. In total 534 of the 1369 pairs are included.

4.3 Agreement on the search task

Figure 5 shows the distribution of agreement values between all pairs of users for the searching task. Here we have taken the event where a user selected a document from the ranked list but did not save it as an "irrelevant" judgement, while the selection and explicit saving of an item is taken as a "relevant" judgement. For any pair of users, we computed α_0 and α_1 over all topic-document pairs that both users selected from the ranked lists for viewing. We only included pairs where at least 6 topic-document pairs were judged as relevant and irrelevant by the first user in the pair, giving 758 user pairs. Again, agreement is not symmetric, and so each pair of users is counted twice, typically with different values. As can be seen, the distribution of agreement values is similar to those for the judging task.

4.4 Agreement across tasks

Figure 6 shows the distribution of the difference in α_0 and α_1 for pairs of users between the searching and judging tasks. Again, the difference across tasks can be large, making it difficult to choose a representative agreement value for any pair of judges/users.

Figure 7 plots each user pair that has an agreement value for both tasks. As is apparent, there is no guarantee that if a pair of users did not agree in the judging task, they will not agree in the search task, and vice versa.

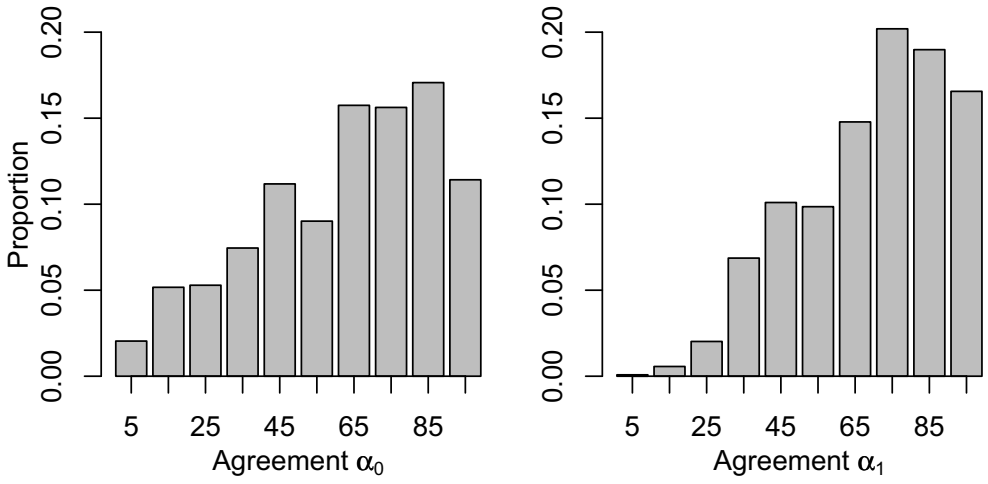


Figure 5: Distribution of agreement amongst all pairs of users on the searching task.

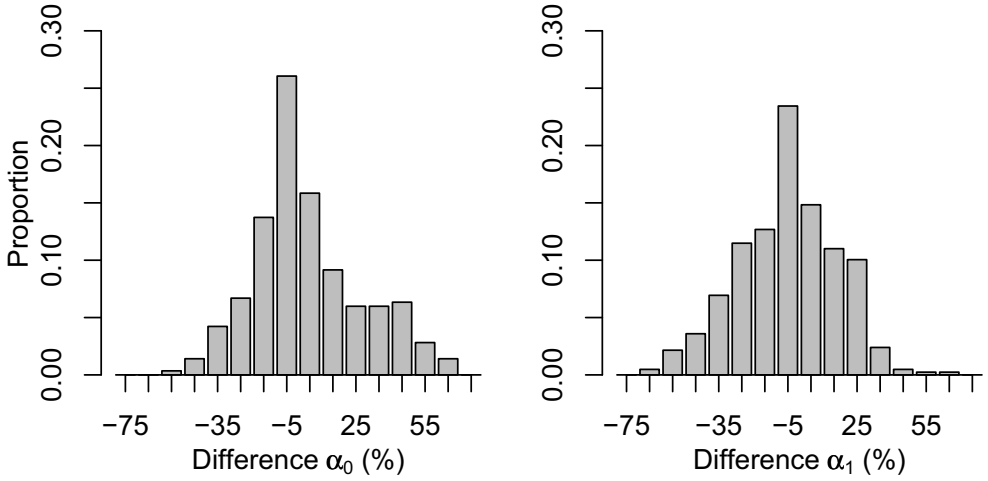


Figure 6: Distribution of the difference in agreement amongst pairs of users on the searching task and judging task.

5 Conclusions

We have presented a simple probabilistic model based on agreement between judges that can predict the effect that altering judges will have on system performance as measured through a batch evaluation experiment. When evaluating performance with P@1, for example, to be 95% confident that one system will remain superior to a second after judges are changed, the agreement between relevance assessments of the judges must be at least 80%.

The model can also be used to assist in selecting metrics. For example, for the P@n family of metrics, it can be seen that the larger the value of n (that is, the more information from the result list that is considered), the lower the required level of agreement between judges to remain confident that the relative system performance will not change. In this paper we have concentrated on the P@n metrics; in future work we plan to extend the approach to other metrics.

Examining the agreement values in one of our user studies has revealed large topic and task effects. That is, for any pair of users, their agreement may alter on different topics or tasks by over 50%. Thus, applying the model presented in Section 3 to predict the effect of changing judges on a corpus requires more sophisticated measuring of α_0 and α_1 than was possible with our available user data. In future work, we plan to investigate controlled experiments for gathering representative agreement values between different users of retrieval systems.

References

[1] Azzah Al-Maskari, Mark Sanderson and Paul Clough. The relationship between IR effectiveness measures and user satisfaction. In *Proceedings of the ACM SIGIR*

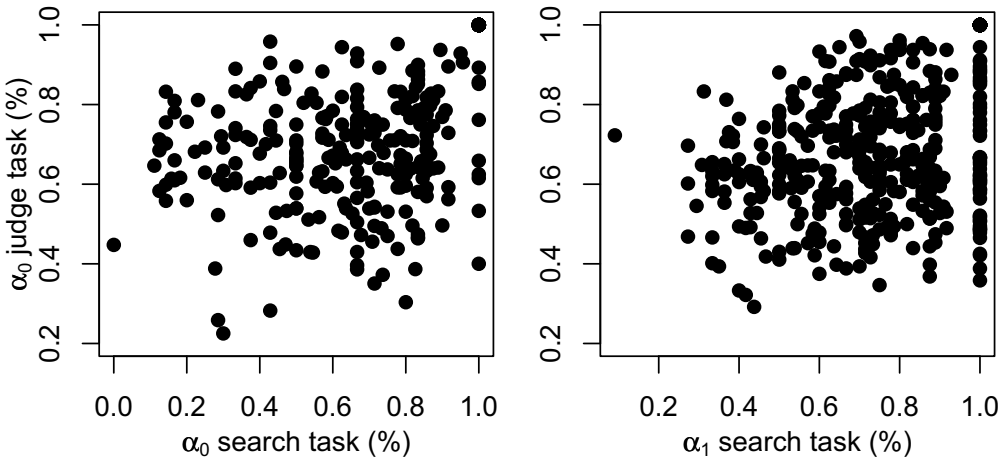


Figure 7: The agreement on each task for each pair of users.

International Conference on Research and Development in Information Retrieval, pages 773–774, Amsterdam, Netherlands, 2007.

[2] James Allan, Ben Carterette and Joshua Lewis. When will information retrieval be "good enough"? In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 433–440, Salvador, Brazil, 2005.

[3] Stefan Büttcher, Charles Clarke and Ian Soboroff. The TREC 2006 terabyte track. In *The Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg, MD, 2007. National Institute of Standards and Technology.

[4] Carlos Cuadra and Robert Katter. The relevance of relevance assessment. In *Proceedings of the American Documentation Institute*, Volume 4, pages 95–99, 1967.

[5] William Hersh, Andrew Turpin, Susan Price, Benjamin Chan, Dale Kraemer, Lynetta Sacherek and Daniel Olson. Do batch and user evaluations give the same results? In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 17–24, Athens, Greece, 2000.

[6] Scott B. Huffman and Michael Hochster. How well does result relevance predict session satisfaction? In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 567–574, Amsterdam, Netherlands, 2007.

[7] Diane Kelly, Xin Fu and Chirag Shah. Effects of rank and precision of search results on users' evaluations of system performance. Technical Report TR-2007-02, University of North Carolina, 2007.

[8] Arie Kruglanski and Icek Ajzen. Bias and error in human judgement. *European Journal of Social Psychology*, Volume 13, pages 1–44, 1983.

[9] Falk Scholer and Andrew Turpin. Metric and relevance mismatch in retrieval evaluation. In *The Fifth Asia Information Retrieval Symposium (AIRS 2009)*, Sapporo, Japan, 2009. To appear.

[10] Falk Scholer, Andrew Turpin and Mingfang Wu. Measuring user relevance criteria. In *The Second International Workshop on Evaluating Information Access (EVIA 2008)*, pages 47–56, Tokyo, Japan, 2008.

[11] Catherine Smith and Paul Kantor. User adaptation: good results from poor systems. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 147–154, Singapore, Singapore, 2008.

[12] Andrew Turpin and William Hersh. Why batch and user evaluations do not give the same results. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 225–231, New Orleans, LA, 2001.

[13] Andrew Turpin and Falk Scholer. User performance versus precision measures for simple web search tasks. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 11–18, Seattle, WA, 2006.

[14] Alexander von Eye and Eun Young Mun. *Analyzing Rater Agreement: Manifest Variable Methods*. Lawrence Erlbaum Associates, 2004.

[15] Ellen M. Voorhees and Chris Buckley. The effect of topic set size on retrieval experiment error. In Micheline Beaulieu, Ricardo Baeza-Yates, Sung Hyon Myaeng and Karlervo Järvelin (editors), *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 316–323, Tampere, Finland, 2002.

[16] Ellen M. Voorhees and Donna K. Harman. *TREC: experiment and evaluation in information retrieval*. MIT Press, 2005.

[17] William Webber, Alistair Moffat and Justin Zobel. Score standardization for inter-collection comparison of retrieval systems. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 51–58, Singapore, Singapore, 2008.

University Student Use of the Wikipedia

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

David Alexander
Department of Computer Science
University of Otago
Dunedin, New Zealand
dalexand@cs.otago.ac.nz

Abstract: *The 2008 proxy log covering all student access to the Wikipedia from the University of Otago is analysed. The log covers 17,635 student users for all 366 days in the year, amounting to over 577,973 user sessions. The analysis shows the Wikipedia is used every hour of the day, but seasonally. Use is low between semesters, rising steadily throughout the semester until it peaks at around exam time. The analysis of the articles that are retrieved as well as an analysis of which links are clicked shows that the Wikipedia is used for study-related purposes. Medical documents are popular reflecting the specialty of the university. The mean Wikipedia session length is about a minute and a half and consists of about three clicks.*

The click graph the users generated is compared to the link graph in the Wikipedia. In about 14% of the user sessions the user has chosen a sub-optimal path from the start of their session to the final document they view. In 33% the path is better than optimal suggesting that users prefer to search than to follow the link-graph. When they do click, they click links in the running text (93.6%) and rarely on "See Also" links (6.4%), but this bias disappears when the frequency of these types of links' occurrence is corrected for.

Several recommendations for changes to the link discovery methodology are made. These changes include using highly viewed articles from the log as test data and using user clicks as user judgements.

Keywords: Information Retrieval, Link Discovery.

1. Introduction

Keeping the link structure up-to-date in a large hypertext collection is difficult. When a new document is added to the collection it is necessary to link from that document to the collection and from the collection to that document. When a document is deleted all links from the collection to the document must be removed. Finally, when a document changes, new links must be added and old links deleted. Deleting links is a mechanical process, but recommending links for new or changing documents is problematic and is an active field of research known as Link Discovery.

Milne & Witten [11] use machine learning to learn links for documents to be added to the Wikipedia. INEX has the Link-the-Wiki track [3] in which the task is to analyse a document (also from the Wikipedia) and to construct an ordered list of links from which a user can choose; Geva [1] and Jenkinson et al. [7] provide the best solutions.

The recent INEX study by Huang et al. [5] raises questions about the validity of the methods of assessment that had been used with all previous solutions to the Link Discovery problem, and therefore the validity of the solutions themselves.

The prior INEX protocol was as follows: A dump of the Wikipedia is taken. From that dump a single document is extracted (the *orphan*). All links between the orphan and the collection are removed. The task is to recommend links for the orphan. Performance is measured relative to the links that were originally in the orphan.

Huang et al. introduced a new protocol to INEX, based on the Cranfield methodology. In this protocol, INEX participants' runs were pooled and manually assessed. Importantly, the links in the original Wikipedia articles were added to the pool. Most importantly, the Wikipedia articles themselves were scored against the pool. Unexpectedly, the Wikipedia articles performed no better than the best submitted runs.

This result suggests that there are many links in the Wikipedia that are not considered relevant to the topic of the articles. The nature of those non-relevant links is not known, but could be studied by analysing the INEX assessments.

This approach would shed light on the nature of relevant and irrelevant links in the Wikipedia and could be used both to help recommend new links and to remove bad links. But a link that is *relevant to the content* of the page may not be *relevant to the information need* of the user. To find *useful* links it is necessary to study how users use links. This raises our research question: *How do users use the Wikipedia link structure?*

To answer this question we studied the log of the University of Otago student web proxy, which all student users of the University computing facilities must pass through, for the 2008 calendar year. From the log we extracted all references to the Wikipedia.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

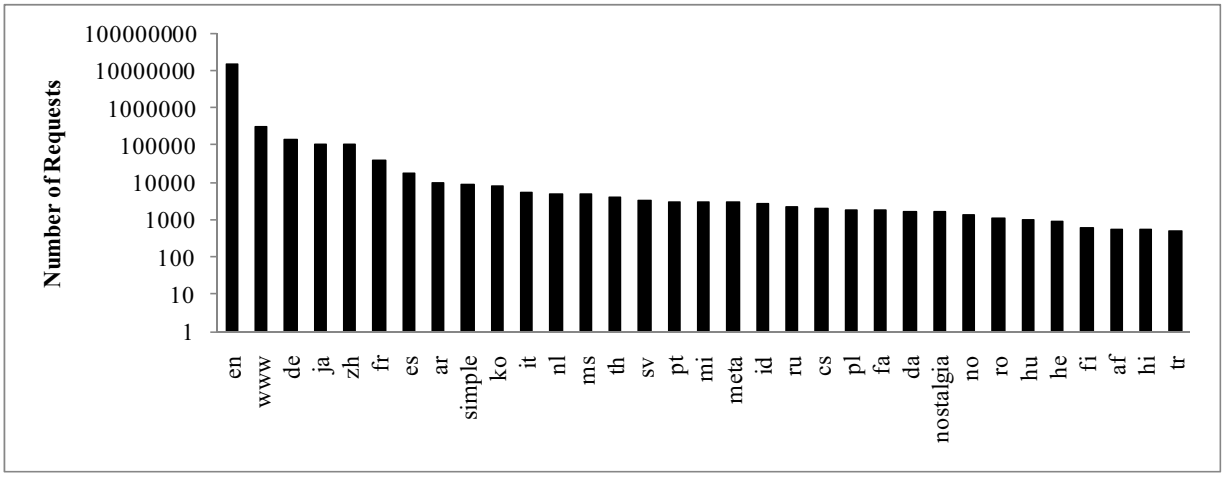


Figure 1: Frequency of use of the versions of the Wikipedia seen more than 500 times in the log. English is the preferred language followed by German, Japanese, Chinese, French, Spanish and so on. The subdomains for language versions of Wikipedia are ISO 639 codes.

Before studying the link-clicking behaviour displayed in the log, we performed a number of preliminary analyses in order to better understand the data, and its applicability to our goal of improving the link structure of Wikipedia. These included examining the request frequency at different times of day and times of year, calculating the length of user sessions, and finding the most commonly-requested pages. The results of these analyses are presented in Sections 3.1 and 3.2

In Section 3.3, the link-clicking behaviour seen in the log is analysed, with particular focus on the question of whether or not the current link graph is being used efficiently. This question is addressed in two ways. The first is to determine the proportion of links clicked on in each article, and to look for patterns in the types of links clicked. The second is to determine whether or not users are reaching their destinations by following links, and if so, whether or not they are doing so in the most efficient way possible.

2. Prior Work

Prior IR research on logs has focused on search engine log analysis. Zhang & Moffat [14], for example, present an analysis of the MSN log while Spink et al. [13] present an analysis of an Excite log.

Internet use by students has previously been studied; however such studies are typically conducted through surveys, for example Metzger et al. [9].

Proxy log use has been limited. Kamps et al. [8] used a (3 month long) New Zealand high school proxy log to validate INEX 2007 results. Their analysis is short. They state: the number of queries; the number of unique queries; the number of clicks in the Wikipedia; the number of queries with Wikipedia clicks; and the number of unique queries with Wikipedia clicks.

There is a growing body of work in link recommendation. Early work [10, 11] conducted outside INEX considers the problem of generating a set of links. INEX considers link discovery to be a recommender task and consequently systems generate a ranked list of results. Geva's solution [1] at INEX is to match the titles of Wikipedia documents against the text of a document, preferring longer titles if several overlap. The Jenkinson et al. solution [7] is based on Itakura & Clarke [6]. They generate a list of all anchors used in the collection along with a list of all documents that are targeted by each anchor text. They rank anchor texts on the frequency with which they occur as links, as a proportion of their overall frequency. They then search for these in the new document and recommend links based on the above frequency. The two approaches perform comparably.

3. Analysis

In this section an analysis of the proxy log is given. The global statistics are presented followed by an analysis of the sessions. Finally the use of the hypertext links is given.

3.1. Global Statistics

The proxy log covers the period from 1st January 2008 to 31st December 2008. It covers 366 days because 2008 was a leap year. The proxy configuration at the university consists of a set of proxies each logging and fulfilling user requests. There were a total of 6 proxy servers and so the analysis is over 2,196 source log files. One of these files (from 30 April 2008) was lost and so the analysis is short by one sixth on that date.

All lines from the log that contained the (case insensitive) word *Wikipedia* were extracted. There were a total of 16,665,418 references in the extracted log, of which 15,696,225 were to the English Wikipedia and 969,193 were to other sites. The references were made by 17,635 students (the university had 20,752 enrolled during 2008). Further fundamental numeric statistics are shown in Table 1.

Table 1: Fundemental statistics of the log

Duration of Log	1/Jan/2008 – 31/Dec/2008
Rows in Log	16,665,418
Rows for English Wikipedia	15,696,225
Users in Log	17,635
Total enrolled students	20,752
Sessions in Log	577,973
Articles Accessed	340,477
Articles in Wikipedia	2,600,000 (approx.)
Wikipedia Subdomains	202 (inc. typos)

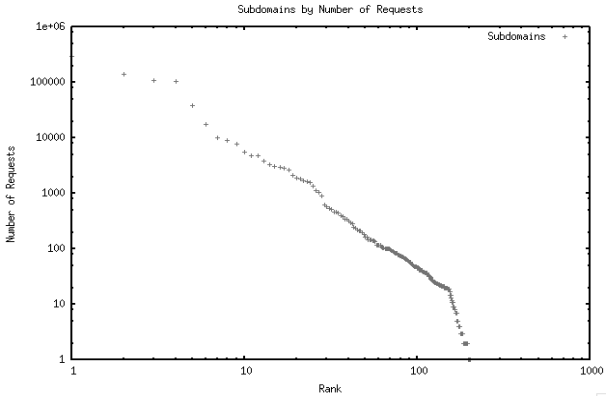


Figure 2: Frequency of use of all versions.

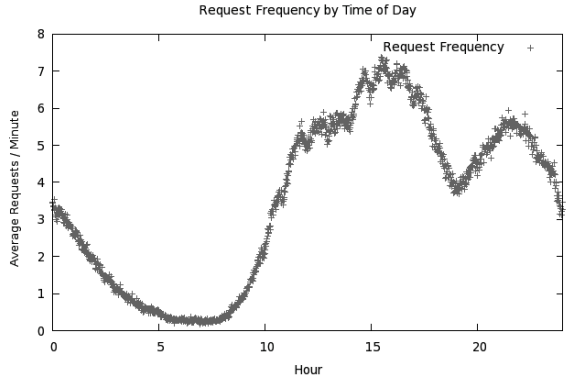


Figure 3: Access to the Wikipedia by time of day.

The Wikipedia exists in many different languages and forms. Each of these versions has its own subdomain of *wikipedia.org*. In the log there are 202 references to different variants (including spelling errors). The most common is the English Wikipedia while the least common (occurring only once) is species.wikipedia.org, the Wikipedia Free Species Directory.

Figure 1 graphs the frequency of use of those variants of the Wikipedia seen in the log more than 500 times. The graph shows that English (subdomain *en*) is the primary language used at Otago, with European and Asian languages also popular. The Māori Wikipedia (subdomain *mi*) was the 17th most popular version, accessed 2,953 times.

All of the subdomains shown in Figure 1 are identified by the ISO 639 codes for their languages, except *www* (an entry point to Wikipedia, having links to the most popular language versions), *simple* (the Simple English Wikipedia, in which articles are written at a level suitable for non-native English speakers or children), *meta.wikimedia.org* (a wiki containing information useful to editors of the various Wikimedia projects), and *nostalgia* (a static copy of a 2001 version of Wikipedia).

Figure 2 shows the request frequency of all subdomains of *wikipedia.org* and *wikimedia.org*. It shows that the subdomains do not completely follow a power-law distribution.

Timestamps in a search engine log are relative to the search engine location. It is therefore not possible to know the user-time at which each query was given. In a proxy log of the type used in this study, however, the user time is the same as the time recorded at the proxy.

Figure 3 shows the mean number of requests per minute at each hour of the day. At midnight there is moderate access steadily falling to low at 5am where access picks up and stabilizes at about 11am. A local peak is seen at 3pm with a dip at dinner-time, picking up at about 7pm and falling again at about 10pm. Student use of the Wikipedia is round-the-clock.

This finding is in line with results seen by others. Zhang & Moffat [14] found that there was no hour of the day at which the MSN search engine was completely unused from within the US. The US, however, is a somewhat larger geographical area then the University of Otago (and has a larger population).

Publicly available search engine logs tend to cover a very short period of time. The MSN log is one month in length, the Excite logs are one day, and the Alta Vista log is about six-weeks. From such short logs it is not possible to make any observations about seasonal user behaviour, analyses have been restricted to daily patterns.

Zhang & Moffat [14] present a day-by-day analysis of the MSN log, which covers May 2006. They show a clear drop in use over weekends and a pattern of peaking early in the week and dropping towards the end.

Shown in Figure 4 is the total number of Wikipedia requests per day seen in the proxy log. Use is clearly seasonal varying from fewer than 1,000 accesses per day in December to over 14,000 accesses per day in June and October. Unsurprisingly the peak is around the university's exam period.

It is reasonable to conclude from this seasonal access pattern that the Wikipedia forms an important part of the student study regime at the University of Otago. If this is the case then it is also reasonable to expect many of the most frequently requested pages to be related to academic study.

The 20 most frequently requested Wikipedia articles are shown in Table 2. The homepage (*Main Page*) is the most viewed Wikipedia page, being requested with more than 23 times the frequency as the next most popular page. This is as expected as many users will enter the Wikipedia via the homepage rather than typing an article's URL manually.

Column 3 shows a manual classification of the given pages into the categories *Work-Related* (*W*),

Informational (*I*) and *Entertainment* (*E*). Of the top 20, half (10) can be considered work-related while the other half are entertainment (2) and informational (8). Most of the work-related pages are medical, reflecting the importance of the medical sciences to the University. This provides further evidence that the Wikipedia is, indeed, being used by students as an aid to their study during the exam period.

It should be noted that the classification is ad-hoc, and was arbitrarily chosen by the two authors. In particular, all medical pages in the table are classified as work-related on the assumption that these pages are mostly requested by the university's large number of medical students, rather than by people seeking medical advice. The classification of some pages is clearly ambiguous; the *Treaty of Waitangi* page could be considered informational due to the treaty's relevance to the location of the university (New Zealand), or work-related due to its potential relevance to History students.

Plotted in Figure 5 is the number of times each of the 340,477 requested articles was retrieved (ordered by frequency). There are a small number of pages requested a very large number of times. (Those articles appear to be informational pages about the Wikipedia, Wikis, New Zealand, the University of Otago, and death!) This distribution of request frequencies suggests that more useful results could come from clustering pages by subject area. We hypothesise that this would show other subject areas being looked up with comparable frequency to the medical sciences, but that those requests would be distributed among a greater number of pages, leading to their absence in Table 2.

It is not only reassuring that the Wikipedia is used for study purposes within the university, but also reassuring that it is not primarily used for smut. Spink et al. [13] provide a list of the 75 most frequently seen search terms in the Excite query log, the top 10 of which are: *and*, *of*, *sex*, *free*, *the*, *nude*, *pictures*, *in*, *university*, *pics*. It appears as though the Wikipedia is being used honourably by students.

3.2. Session Statistics

Identifying a user's session in a search engine query log has proven to be problematic because it is not clear what the user is doing between one log entry and the next. The same problem exists when looking at a proxy log such as the one used in this study, because only the user actions that result in an HTTP request are recorded.

The proxy log used in this study distinguishes users, and identifies the requested page, dates, time, etc., but not the referrer. Therefore, although it is known what was done, by whom, and when, it is not certain what a user was doing before making a particular request. Identifying a user's session under these circumstances is problematic because without the referrer it is difficult to identify the start (or end) of a session.

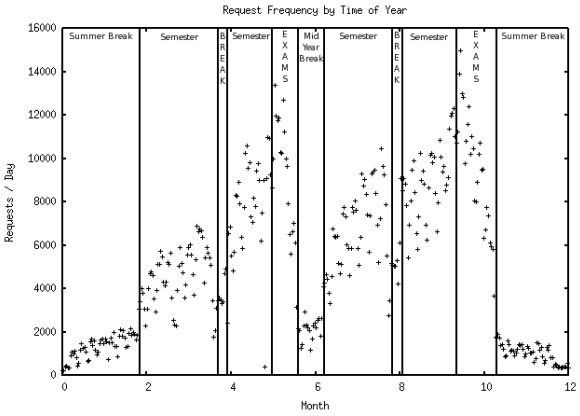


Figure 4: Access to the Wikipedia by date. Semester-times, breaks and examination periods are indicated.

Table 2: Top 20 most retrieved pages, classified as Work-Related (W), Informational (I) or Entertainment (E)

Page	Requests	Class
Main Page	75583	I
Wiki	3256	I
New Zealand	1686	I
Deaths in 2008	1315	I
University of Otago	861	I
Dunedin	859	I
Standard deviation	857	W
Wikipedia	806	I
Dopamine	669	W
Blood pressure	561	W
The Dark Knight (film)	557	E
Aldosterone	556	W
Glycolysis	546	W
Tyrosinase	541	W
Gossip Girl (TV series)	541	E
Treaty of Waitangi	516	I
Tuberculosis	514	W
Meningitis	512	W
Multiple sclerosis	511	W
HIV	510	W

He & Göker [2] define a web search session as a set of consecutive requests by a user with no longer than some time limit from one request to the next. They conclude that for web search log analysis the optimal time is between 10 and 15 minutes. There was, however, very little difference observed between the sessions produced using a time limit of 15 minutes and those produced using a time limit of 60 minutes.

It is reasonable to assume that a user navigating the Wikipedia will spend longer reading documents than a user searching the web spends reading a results list. For this reason, and for this study, a session is defined as a set of consecutive requests by the same user with a gap of no more than 60 minutes between adjacent requests. Further investigation is needed to determine whether or not this is a suitable time limit for proxy logs.

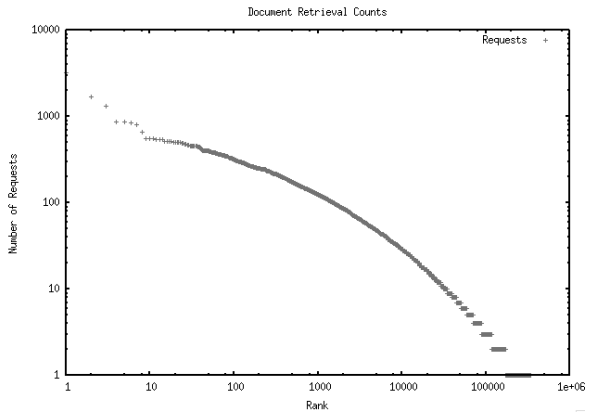


Figure 5: Number of times each document is retrieved ordered from most to least frequent.

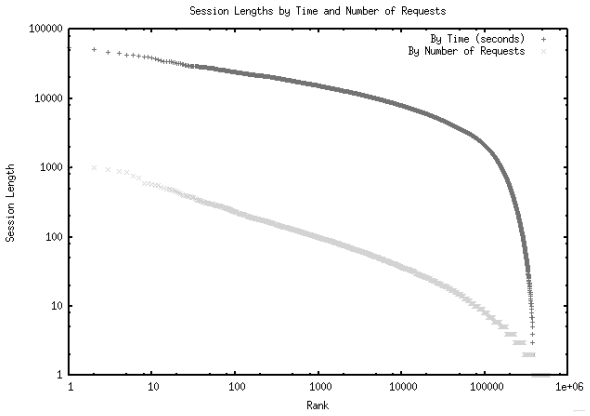


Figure 6: Session lengths ordered from longest to shortest. Sesson times in seconds and in number of clicks are both shown.

Table 3: Top 20 non-wikipedia session origins

Count	Source
2030	http://rds.yahoo.com/
1527	http://nz.wrs.yahoo.com/
1433	http://content.answers.com/
427	http://hk.wrs.yahoo.com/
203	http://s.scribd.com/
203	http://wrs.search.yahoo.co.jp/
154	http://au.wrs.yahoo.com/
149	http://mycroft.mozdev.org/
130	http://tw.wrs.yahoo.com/
129	http://sp.ask.com/
110	http://uk.wrs.yahoo.com/
82	http://www.scribd.com/
81	http://digg.com/
76	http://static.getfansub.com/
76	http://www.microsoft.com/
68	http://www.nationmaster.com/
60	http://www.apple.com/
57	http://wrs.yahoo.com/
52	http://i.ixnp.com/
52	http://pixel.quantserve.com/

Session length can be measured in several ways including the number of requests and the total time between the first and last request. In the case of a single-request session, however, the session time must be considered to be zero because it is impossible to tell how long the user spent looking at the single page that was requested.

In Figure 6 the sessions from the proxy log are shown ranked from the longest to the shortest. In total there were 577,973 sessions. When measured by time, the longest had 26 requests over 86,441 seconds (1 day and 41 seconds), and the median had 2 requests over 93 seconds. It is reasonable to conclude that the longest session is not human generated (one click an hour for a day) and so there are, in all likelihood, robots running at the university that are downloading data from the Wikipedia each hour.

When measured by number of requests, the longest had 2,340 requests over 8,550 seconds (a mean of one click every 3.76 seconds for 2 hours 22 minutes and 30 seconds), and the median had 3 requests over 93 seconds. Again it is reasonable to conclude that the longest session is not a human, but a robot.

In some cases users chose to search the Wikipedia using a search engine. In these cases they might have either added the word *Wikipedia* to their query or site-restricted their search to a *wikipedia.org* site.

Table 3 shows the top 20 non-Wikipedia site origins appearing in the log. It is important to recall that the analysed log only includes requests that contain the substring *Wikipedia* – and so this table does not truly reflect the number of sessions originating outside the Wikipedia. It is surprising that Google does not appear, but this is possibly because of Google's use of asynchronous requests for result lists on supporting browsers.

Coupling this result with the number of requests for the Wikipedia homepage leads to the conclusion that the students tend to go directly to the Wikipedia and then search, rather than using an Internet search engines to find information in the Wikipedia.

3.3. Link Statistics

The primary motivation for this investigation is the understanding of how users navigate the Wikipedia so that this knowledge may be used to improve the performance of link recommender systems.

For the purpose of this investigation a user is deemed to have clicked a link in order to retrieve an article if, within a session, there was a page requested earlier in that session that contains a link to the retrieved page.

An alternative would be to consider only the user's most recently requested page as a potential link source, which would reduce the number of false positives. This was rejected because of anecdotal evidence that users surfing the Wikipedia have multiple pages open at once, meaning that the user's click sequence may resemble part of a breadth-first traversal of the link graph.

For brevity, the term *click* will hereafter be used without qualification to refer to a request that is believed to have been caused by a click on a particular link. It is important to note that this information may

not be accurate, and a proxy log with referrers should ideally be used in future research.

Presented in Table 4 are the top 20 most clicked links. Of particular note is the link from the homepage to Deaths in 2008. This can be directly attributed to the link "Recent Deaths" at the bottom of the "in the news" section of the homepage. Of the top 20 links, 13 are clearly work-related while 5 are entertainment and 2 are informational.

Shown in Figure 7 is the distribution of link clicks ordered from most popular to least popular. By inspection it can be seen to roughly follow a power-law distribution. Most links are clicked only once but some links are very popular.

Figure 8 shows the distribution of clicked links on a per document basis. It can be seen that of the links in a document, very few were clicked even though there are many links in the documents. This cannot be explained by the presence of "boilerplate" links such as the *What links here* link because these links are not included in the collection from which the relevant data was extracted.

Table 4: Source and target articles of the 20 most clicked links.

Source	Target	Clicks	Class
Main Page	Deaths in 2008	3092	I
NAD	Nicotinamide adenine dinucleotide	282	W
Nicotinamide adenine dinucleotide	FAD	239	W
Tyrosinase	Melanin	233	W
Lactate	Lactic acid	219	W
ADH	Vasopressin	206	W
Tyrosine	Dopamine	202	W
Heroes	Heroes (TV series)	186	E
Main Page	Wikipedia	181	I
Melanin	Melanocyte	179	W
South Park	List of South Park episodes	176	E
Gossip Girl	Gossip Girl (TV series)	174	E
Adjuvant	Immunologic adjuvant	162	W
Thiamine pyrophosphate	Pyruvate dehydrogenase	161	W
Heroes (TV series)	List of Heroes episodes	151	E
House (TV series)	List of House episodes	141	E
Systole	Systole (medicine)	133	W
Vitamin E	Tocopherol	133	W
Melanin	Melanoma	124	W
Diaphragm	Thoracic diaphragm	124	W

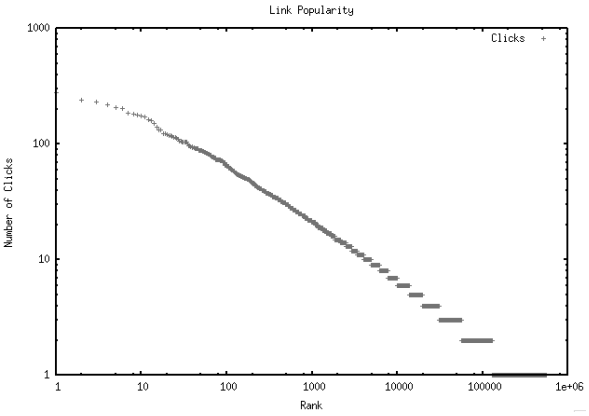


Figure 7: Frequency of use of clicked links

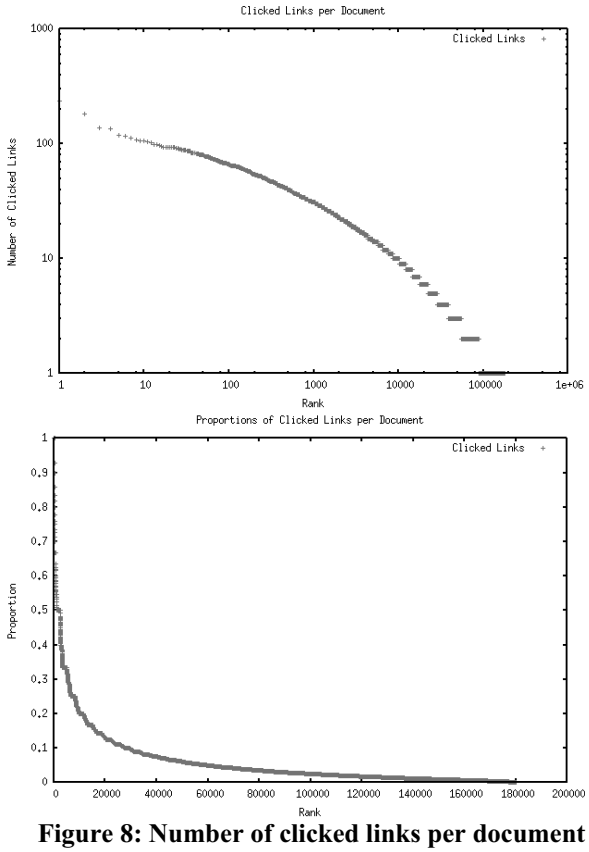


Figure 8: Number of clicked links per document by absolute count (above) and relative to the number of links in the document (bottom). In most documents only one link was clicked despite there being many links that might have been chosen.

Huang et al. [4] present the metric used in the INEX Link-the-Wiki track. It is a mean average precision (MAP) based metric which assumes that all relevant links are equally relevant. This assumption may not be valid; the users may show bias for certain links. In future work we will examine these potential biases by determining the prior probability of the click frequency distributions seen in each document. Given the already observed bias from the homepage to the recent deaths page it is reasonable to believe that some links are more popular than others. If this is the case then

the appropriateness of the INEX Link-the-Wiki metrics should be examined.

6.4% of those links that are clicked are from the *See Also* section of the document whereas remaining 93.6% are from the running text. 6.4% is also the proportion of links in those documents that are *See Also* links. This suggests that there is no user preference to these links over the running text links. This is surprising because the *See Also* links are at the bottom of the page, although Fitts's Law may apply.

INEX offers two tasks in the Link-the-Wiki track: file-to-file linking, and anchor-to-BEP (best entry point) linking. In the former the task is to identify articles related to a new article to be added to the Wikipedia. This is equivalent to the task of adding *See Also* links to an article. In the latter task the link discovery system must identify anchor-texts in the running text of the new article and targets within the Wikipedia.

The discovery that running-text links appear to be as important as the *See Also* links suggests that the two INEX tasks are also equally important.

Potamias et al. [12] propose an algorithm for approximating the shortest path between two nodes in a large graph. Several hubs are chosen based on an estimate of their centrality in the graph, and a single-source shortest path calculation is performed from each hub to all nodes in the graph. The shortest path estimate for a pair of nodes is calculated by determining the length of the path between the nodes through each hub in turn, and taking the shortest of those paths.

The actual path taken in each session was computed and the lengths of the paths are shown in Figure 6. The shortest path they could have taken (from the start to the end of their session) can be estimated using the algorithm of Potamias et al. The difference is the *slack* in the session. That is, assuming the user has one information need per session and upon fulfilling it they stop using the Wikipedia, the number of wasted clicks (and consequently the amount of wasted time) can be estimated.

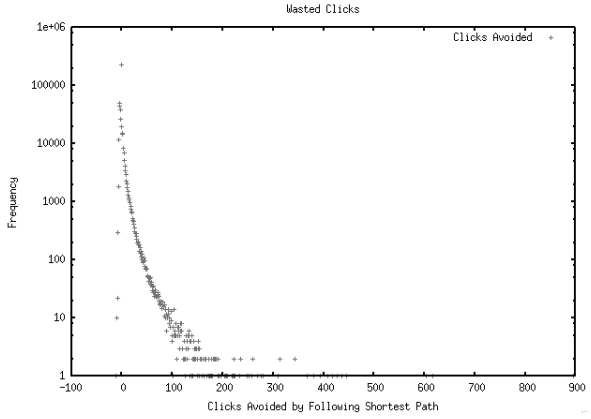


Figure 9: Number of clicks that could be saved if the user navigated the Wikipedia using the shortest path from the start of their session to the end of their session.

Figure 9 shows the difference between the actual length and the estimated shortest path for each session. Positive numbers indicate that clicks would be saved if the user had chosen the shortest path; negative numbers are due to users arriving at their destination by methods other than clicking links.

The shortest path estimation algorithm was used because of the number of sessions and the magnitude of the link-graph. It should be noted that the result is always pessimistic. It computes a number that is no smaller than the shortest path. Despite this, 83,761 (14%) user sessions would be reduced in length if the user had followed the shortest path. In 192,375 (33%) sessions the user found a path shorter than the estimated shortest path (perhaps by searching). 231,317 sessions are optimal. For the remaining 70,520 sessions no path could be found (the link graph is not strongly connected).

Assuming users are doing their utmost to find the information they seek, it is pertinent to ask why they waste so many clicks in their information seeking. Further investigation is needed; however it could be due to information overload. Given the extensive interlinking between Wikipedia articles, it may simply be too difficult to spot which links to click. If this is the case then a reduction in the size of the link graph (that is, the removal of links) may result in a better user experience. This result is in line with the manual assessment experiments of Huang et al. [5], which suggest that many of the links in the Wikipedia are not relevant. Further, since 33% of the sessions are shorter than the shortest path, it is reasonable to conclude that users' current response to viewing over-linked documents is to resort to searching.

The mean number of clicks that could be avoided if a user followed the shortest path is 0.018 clicks per session.

However, it is also possible that many of the wasted clicks seen are a result of users browsing Wikipedia for trivia, merely because they find it interesting. (For example, clicking links that go from the name of a day, month or year to a list of events that happened in that time period.) It is therefore important not to take the link-graph reduction goal to its logical conclusion by removing all trivial links, as this would diminish users' enjoyment of Wikipedia, which might in turn cause the non-trivial information content in Wikipedia to stagnate. Therefore, it is important to balance the removal of links that hinder navigation with the retention of links that, while not strictly relevant, are sometimes used and do not hinder navigation.

It is pertinent to ask whether the first document the user viewed *should have been* linked to the last document they viewed. Computing this is equivalent to solving the link discovery problem, but an estimate might be made using one of the previously published link discovery algorithms. The Itakura & Clarke [6] algorithm as implemented by Jenkinson et al. [7] is fast and might make a good candidate algorithm, as

might Geva's title matching algorithm [1]. Computing the optimal link graph for the Wikipedia is left for future work.

4. Discussion and Conclusions

The University of Otago student proxy server logged all accesses to the Internet for the 2008 calendar year. From this log all accesses to the Wikipedia were extracted and analysed. In total 16,665,418 requests were made by 17,635 users.

The analysis suggests that students use the Wikipedia primarily as an encyclopaedia for study-related purposes. They typically use it for a very short period of time (a few minutes) and search from the Wikipedia rather than via an Internet search engine. They prefer to use it close to exams, and they use it at all times of the day and night.

The analysis of the link statistics suggests that there is some bias in the users' click pattern, as very few of the available links are clicked, but further work is needed to determine the nature of this bias. Users appear to click on a very small proportion of the links in a document, but there is no bias towards *See Also* or running-text links. If indeed there is bias, then it may be appropriate to re-examine the metrics used to measure the performance of link discovery systems.

On the assumption that a user is trying to fulfil one information need in each session, the amount of slack in a user session was computed. In 14% of sessions the user did not choose the shortest path from the start of their session to the end. In 33% of cases the user found a path shorter than the shortest path which suggests that the link-graph of the Wikipedia is not helping those users and they are resorting to methods other than browsing in order to find their information.

This study was conducted with the goal of improving link discovery systems such as those seen in the INEX Link-the-Wiki track. The results suggest that by removing non-useful links from the Wikipedia (simplifying the graph) the user will find it easier to browse in order to fulfil their information need, but it is important not to take this too extreme, and to remove *harmless* links merely because they are not relevant, as this would decrease the utility of the Wikipedia.

Further work might be conducted on the proxy log. Previous studies have suggested that 4-digit year links are not considered relevant by INEX assessors. The nature of the links the user clicked remains unknown, as does the nature of relevant links in the INEX assessments.

The INEX Link-the-Wiki track has two tasks. In the file-to-file task a set of randomly selected documents are chosen from the Wikipedia. The links between those documents and the Wikipedia are removed and the system must predict the links that were present. As a consequence of the Wikipedia log entries having been extracted from the full proxy log, there now exists a complete year-long log of which articles were chosen and which links were clicked. This log might be used as the source of articles for the INEX track. If the articles were chosen from those accessed in the log then performance could be measured relative to those links that were clicked.

The log might also be used in the Link-the-Wiki anchor-to-BEP task in which the link discovery system must choose anchors and target document / best entry point pairs. Although best entry points are not typically linked to in the Wikipedia, the anchor text and target document pairs can be deduced from the Proxy log using the method outlined above.

Much of this study was devoted to understanding how university students use the Wikipedia. It is heartening to see the use is generally related to their study, but disheartening to see that use is driven by the examination schedule.

5. References

[1] Geva, S., *GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia*. INEX 2007 pp. 404-416.

[2] Göker, A. and D. He, *Analysing Web Search Logs to Determine Session Boundaries for User-Oriented Learning*, In *Adaptive Hypermedia and Adaptive Web-Based Systems*. 2000. pp. 319-322.

[3] Huang, D.W., et al., *Overview of INEX 2007 Link the Wiki Trac.* INEX 2007 pp. 373-387.

[4] Huang, W.C., S. Geva, and A. Trotman, *Overview of INEX 2008 Link the Wiki Track*, INEX. 2008p. 314-325.

[5] Huang, W.C., A. Trotman, and S. Geva, *The Importance of Manual Assessment in Link Discovery*, SIGIR 2009

[6] Itakura, K.Y. and C.L. Clarke, *University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks*, INEX 2007. pp. 417-425.

[7] Jenkinson, D., K.-C. Leung, and A. Trotman, *Wikisearching and Wikilinking*, in *pre-proceedings of INEX 2008*. 2008.

[8] Kamps, J., M. Koolen, and A. Trotman, *Comparative Analysis of Clicks and Judgments for IR Evaluation*,.WSCD 2009.

[9] Metzger, M.J., A.J. Flanagin, and L. Zwarun, *College student web use, perceptions of information credibility, and verification behavior.* Comput. Educ., 2003. **41**(3):271-290.

[10] Mihalcea, R. and A. Csomai, *Wikify!: linking documents to encyclopedic knowledge*. CIKM 2007. pp. 233-242.

[11] Milne, D. and I.H. Witten, *Learning to link with wikipedia*, CIMK 2008 pp. 509-518.

[12] Potamias, M., et al., *Fast shortest path distance estimation in large networks*, CIKM 2009.

[13] Spink, A., et al., *Searching the Web: The public and their queries.* JASIST 2001. **53**(2):226-234.

[14] Zhang, Y. and A. Moffat. *Some Observations on User Search Behavior*. ADCS 2006. pp. 1-8

Feature Selection and Weighting Methods in Sentiment Analysis

Tim O'Keefe
School of Information Technologies
University of Sydney
NSW 2006, Australia
toke9145@uni.sydney.edu.au

Irena Koprinska
School of Information Technologies
University of Sydney
NSW 2006, Australia
irena@it.usyd.edu.au

Abstract *Sentiment analysis is the task of identifying whether the opinion expressed in a document is positive or negative about a given topic. Unfortunately, many of the potential applications of sentiment analysis are currently infeasible due to the huge number of features found in standard corpora. In this paper we systematically evaluate a range of feature selectors and feature weights with both Naïve Bayes and Support Vector Machine classifiers. This includes the introduction of two new feature selection methods and three new feature weighting methods. Our results show that it is possible to maintain a state-of-the art classification accuracy of 87.15% while using less than 36% of the features.*

Keywords Information Retrieval, Natural Language Techniques and Documents

1 Introduction

The opinions of other people have always been important to us, and in particular we are often concerned with the prevailing sentiment of those opinions. Often governments want to know how voters feel about a policy, corporations want to know how customers feel about a product and movie goers want to know if others would recommend a movie. The idea behind sentiment analysis is to provide this information by building a system that can classify documents as positive or negative, according to the overall sentiment expressed within those documents.

Early approaches to sentiment analysis tended to focus on classifying documents according to the out-of-context sentiment of individual features [14]. While these approaches did not require domain-specific training data, their accuracy was quite poor. Subsequent research focused on supervised learning techniques that are common in text categorisation tasks [9], such as Support Vector Machine (SVM) and Naïve Bayes (NB) classifiers. Though these techniques are far more accurate than the earlier text-based approaches, they are a lot more computationally expensive to run due to the large number of features.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

In fact, in the Pang et al. [9] movie review data set that has become the *de facto* standard there are just under 51,000 unique words and symbols. Very few of these features actually provide useful information to the classifier, so feature selection can be used to reduce the number of features. Despite the fact that its use is commonplace, there has been little research into the effects of different methods of feature selection in sentiment analysis. In this paper we address this gap by comparing three feature selection methods at a number of selection thresholds, using six feature weighting methods. The feature selection methods include Categorical Proportional Difference (PD), a recently proposed method that was successfully used for topic-based text categorisation, and two methods based on sentiment values from SentiWordNet (SWN) [2] that we introduce: SWNSS and SWNPD. The feature weighting methods include Feature Frequency (FF), Feature Presence (FP), TFIDF, and three other methods based on words grouped by their SWN values that we introduce: SWN-SG, SWN-PG and SWN-PS. All tests were conducted using both SVM and NB.

Our results show that PD and SWNSS were able to maintain or improve accuracy when used with suitable weightings while SWNPD tended to reduce accuracy, though not in all cases. SVM with PD as a feature selector achieved our highest accuracy of 87.15% which is comparable with the state-of-the art, but uses a vastly reduced set of features.

2 Background

While there was some early work in word-level sentiment analysis [3] and a semi-automatic approach to document-level sentiment analysis [13], the real genesis of document-level sentiment analysis was the work of Turney [14]. The basic idea behind Turney's approach was to average the sentiment of the adjectives within each document and then classify the document depending on whether the average was positive or negative. To find the sentiment of adjectives, Turney used the AltaVista search engine to determine how often individual adjectives co-occured with the words "excellent" and "poor." Words that co-occured more often with "excellent" were deemed positive and words co-occuring more often with "poor" were deemed negative.

Authors	Data split	Classifier	Cross Validation	Feature Selection	Baseline Accuracy (%)	Best Accuracy (%)
Pang et al. [9]	700+ 700-	NB, ME, SVM	3-fold	No	N/A	82.9
Pang & Lee [8]	1000+ 1000-	NB, SVM	10-fold	Yes	87.15	87.2
Mullen & Collier [7]	700+ 700-	Hybrid SVM (Turney values, Osgood values, lemma models)	10-fold	No	83.5	86
König & Brill [6]	1000+ 1000-	Pattern-based, SVM, Hybrid	5-fold	No	87.5	91
Abbasi et al. [1]	1000+ 1000-	Genetic Algorithms (GA), Information Gain (IG), IG + GA	10-fold	Yes	87.95	91.7
Prabowo & Thelwall [10]	1000+ 1000-	Hybrid (rule + closeness measure + SVM)	10-fold	No	87.3	87.3

Table 1: Results reported in the literature on various versions of the Pang et al. [9] movie review data set.

The first use of supervised learning in sentiment analysis was by Pang et al. [9]. Their aim was to determine whether sentiment analysis could be treated as a special case of topic-based categorisation with two topics: *positive* and *negative*. To achieve this they tested Naïve Bayes (NB), Maximum Entropy (ME), and Support Vector Machine (SVM) classifiers, all of which have performed well in topic-based categorisation. For features, they used the words and symbols of the documents as either a unigram or a bigram bag-of-features, with unigrams generally performing better. They tested Feature Frequency (FF) and Feature Presence (FP) and found that by using a SVM with unigram FP they could achieve an accuracy of 82.9% in a 3-fold cross validation test. Table 1 lists some of the best results that have been reported in the literature.

2.1 Feature Selection

Most researchers employ basic feature selection in their work in order to improve computational performance, with a few using more complicated approaches [5, 8, 1]. To date there have only been two papers that have entirely focused on using feature selection to improve sentiment analysis. The first was by Pang & Lee [8], who used a SVM trained on subjective and objective text to remove objective sentences from the corpus. In their initial results they found that document sentiment classification accuracy actually declined. They then conducted some "non-obvious feature engineering" by making it more likely that sentences adjacent to removed sentences would be removed as well, which slightly improved accuracy over their baseline.

The other work that used sophisticated feature selection was by Abbasi et al. [1]. They found that using either information gain (IG) or genetic algorithms (GA) resulted in an improvement in accuracy. They also combined the two in a new algorithm called the Entropy Weighted Genetic Algorithm (EWGA), which achieved the highest level of accuracy in sentiment analysis to date of 91.7%. The drawback of this new method is that while it can efficiently classify items, it is very computationally expensive to conduct the initial feature selection, since both GA and IG are expensive to run.

2.2 SentiWordNet

SentiWordNet (SWN) is an extension of WordNet that was developed by Esuli & Sebastiani [2], which is intended to augment the information in WordNet with information about the sentiment of the words in WordNet. Our research uses the information provided by sentiment in some detail, so we will describe it here. Each synset in SWN has a positive sentiment score, a negative sentiment score and an objectivity score. When these three scores are summed they equal one, so they give an indication of the relative strength of the positivity, negativity and objectivity of each synset. Esuli & Sebastiani [2] obtained these values by using several semi-supervised ternary classifiers, all of which were capable of determining whether a word was positive, negative, or objective. If all the classifiers agreed on a classification then the maximum value was assigned for the associated score, otherwise the values for the positive, negative and objective scores were proportional to the number of classifiers that assigned the word to each class.

The drawback in using SWN is that it requires word sense disambiguation to find the correct sense of a word and its associated scores. Whilst there has been significant research into this problem, we decided that it was out of scope to use any sophisticated word sense disambiguation for this project, so we simply took the highest positive and negative values that we could find for each word. This is based on the assumption that in a subjective document it is reasonably likely that the most subjective sense of a word is being used. Preliminary testing confirmed that using the most subjective senses

tended to outperform the senses that are known to be most frequent.

3 Data & Evaluation

We use two different supervised learning approaches to sentiment analysis: Support Vector Machines (SVM) and Naïve Bayes (NB). SVM and NB classifiers were originally used in sentiment analysis by Pang et al. [9], who found that SVM classifiers generally outperformed NB. In order to be as comparable to Pang & Lee as possible we use the SVM implementation developed by Joachims [4], called SVM_{LIGHT}. For Naïve Bayes we use the implementation available in Weka [15].

The data set we use is the set of 1000 positive and 1000 negative movie reviews from IMDb[1] that was introduced in Pang et al. [9]. For all of our experiments we conduct 10-fold cross validation, and we use paired t-tests at a confidence level of 0.05 to establish significance.

4 Feature Weighting Methods

4.1 Unigram Features

In the domain of sentiment analysis, and more generally text categorisation, it is common to use the words and symbols within the corpus as features in the feature vectors. Though there are other ways of representing the words and symbols, we will be using unigrams, where each unique word or symbol is counted as one feature. Pang et al. [9] found that unigrams fairly comprehensively out-performed bigrams and combinations of unigrams and bigrams. The different feature weights for the unigrams are discussed below.

4.1.1 Feature Frequency (FF)

The simplest way to represent a document with a vector is the feature frequency method that was originally used in sentiment analysis by Pang et al. [9]. The method uses the term frequency, i.e. the frequency that each unigram occurs within a document, as the feature values for that document. So if the word "excellent" appeared in a document ten times, the associated feature would have a value of ten.

4.1.2 Feature Presence (FP)

Pang et al. [9] were also the first to use feature presence in sentiment analysis. Feature presence is very similar to feature frequency, except that rather than using the frequency of a unigram as its value, we would merely use a one, to indicate that the unigram exists in the document. Multiple occurrences of the same unigram are ignored, so we get a vector of binary values, with ones for each unique unigram that occurs in the document, and zeros for all unigrams that appear in the corpus but not in the document.

[1] http://www.imdb.com

4.1.3 Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a common metric used in text categorisation tasks [11], but its use in sentiment analysis has been less widespread, and surprisingly it does not appear to have been used as a unigram feature weight. TF-IDF is composed of two scores, term frequency and inverse document frequency. Term frequency is found by simply counting the number of times that a given term has occured in a given document, and inverse document frequency is found by dividing the total number of documents by the number of documents that a given word appears in. When these values are multiplied together we get a score that is highest for words that appear frequently in a few documents, and low for terms that appear frequently in every document, allowing us to find terms that are important in a document.

4.2 SentiWordNet Word Groups

While unigram features have emerged as the most accurate approach to sentiment analysis, there has still been significant work in using other types of features [14, 7, 10]. While most of this previous research has shown that grouping or summing words based on their out-of-context sentiment has not performed well on its own [14, 9], some researchers have used these sorts of features to augment unigrams [7]. We add to this research by using SWN to put the words found in each document into groups, which we can then use as features for classifiers.

4.2.1 SWN Word Score Groups (SWN-SG)

One of the interesting features of SWN is that there are only a limited number of values that the positive and negative word scores can take on, due to the way those scores are calculated. We can take advantage of this fact to group words with the same positive or negative score, so that rather than having features that correspond to words, we have features that correspond to groups of words. The value of a feature would then be the number of words in the document that have the same positive or negative SWN score. So for example if the sentence "The acting was excellent, the special effects were amazing, and the script was terrific" appeared in a document we might find that "excellent," "amazing," and "terrific" all had the same positive score. When we turn that sentence into a feature vector one of the features would correspond to that positive score and would have a value of three, since there are three words with that score.

4.2.2 SWN Word Polarity Groups (SWN-PG)

Since SWN gives words both a positive and negative score, we can find whether a word is more positive than negative and vice versa. This allows us to define two features, positive and negative, which correspond to the counts of positive and negative words respectively. So

words that are more positive than negative add one to the positive feature and words that are more negative add one to the negative feature. The end result is a feature vector with two features, the first being the number of positive words and the second being the number of negative words in the document.

4.2.3 SWN Word Polarity Sums (SWN-PS)

The final feature type that we introduce is similar to the word polarity groups, except that we actually sum the positive and negative scores, rather than just tallying the number of words with those scores. So when we convert a document into a feature vector there are two features. The first one is the sum of the SWN positive scores of all words that have a higher positive than negative score. The second feature is the sum of the SWN negative scores of all words that have a higher negative score than positive score. Any words that have no positive and no negative score, or where the positive and negative scores are equal, are ignored. The scores are adjusted for document length, so different length documents can be more accurately compared.

5 Feature Selection

When we set out to classify a document we generally start off with a very large number of words that need to be considered, even though very few of the words in the corpus are actually expressing sentiment. These extra features have two clear drawbacks that we would like to eliminate. The first is that they make document classification slower, since there are far more words than there really needs to be. The second is that they can actually reduce accuracy, since the classifier must consider these words when classifying a document.

Clearly there is an advantage in using fewer features, so in order to remove some of the unnecessary features, we use *feature selection*. As the name suggests, feature selection is a process where we run through the corpus before the classifier has been trained and remove any features that seem unnecessary. This allows the classifier to fit a model to the problem set more quickly since there is less information to consider, and thus allows it to classify items faster. In this section we describe several different methods of feature selection.

5.1 Categorical Proportional Difference (PD)

Categorical Proportional Difference (PD), introduced by Simeon & Hilderman [12], is a metric which tells us how close to being equal two numbers are. We can use this to find unigrams that occur mostly in one class of documents or the other, by using the positive document frequency and negative document frequency of a unigram as the two numbers. In other words if a unigram occurs predominantly in positive documents or predominantly in negative documents then the PD of the unigram will be close to one, whereas if it occurs in about as many positive documents as negative documents then its PD will be close to zero. While Simeon & Hilderman use a more general equation for multi-class problems, we use a simplified equation for our two-class problem, which is as follows:

$$\frac{|PositiveDF - NegativeDF|}{PositiveDF + NegativeDF}$$

A high score from this equation indicates that the unigram is telling us a lot, and a low score indicates that the unigram is telling us very little. For example if the word "actor" appears in exactly as many positive documents as negative documents then finding the word "actor" in a new document will tell us nothing about it and as such its PD score will be zero. Conversely, if the word "excellent" appears in only positive documents then finding the word "excellent" in a new document would give us a good clue that the document is positive, and as such it would have a PD score of one. So to use PD as a feature selector we simply need to remove any features where the result of the equation is less than or equal to some threshold value.

5.2 SWN Subjectivity Scores (SWNSS)

The SWN feature selector is actually able to distinguish objective and subjective terms, which is useful since only subjective terms should carry sentiment. To do this we use the SWN *subjectivity score*, which is found by adding the positive and negative SWN scores of a unigram together. This is the opposite of the *objectivity score* that is defined by Esuli & Sebastiani [2], but its use is equivalent. To use it as a feature selector we simply remove any unigrams whose subjective score is less than a certain threshold. When this feature selector is used, unigrams that are not found in SWN, such as names and misspellings, are removed from the corpus as well (although arguably the names of certain actors could give strong clues about the quality of a movie).

5.3 SWN Proportional Difference (SWNPD)

While the SWN subjectivity feature selector can find words that have some *a priori* sentiment attached, it cannot tell us whether that sentiment is consistent or meaningful. It is entirely possible that a word may have a SWN subjectivity score of one, indicating that it is very subjective, but its positive and negative scores may be 0.5 each. This may make the word uninformative as a feature so there could be value in removing it. To do this we define SWN Proportional Difference, which uses the SWN positive and negative scores in the PD equation, as follows.

$$\frac{|SWNPos - SWNNeg|}{SWNPos + SWNNeg}$$

Similarly to PD, SWNPD will be high for words that are mostly positive or negative, and low for words that are

a mix of both. By using this score we hope to remove subjective words that have an ambiguous polarity from the corpus.

6 Results and Discussion

Table 3 shows in bold the best results achieved for each classifier with each feature selection method. The best accuracy result was 87.15%, which was achieved using PD feature selection with a threshold of 0.125 (which uses 18,149 features or 36% of the total) and FP as a feature weighting method. For comparison, Table 1 shows other results reported in the literature. All approaches used the same dataset which was created by Pang et al. [9] and is the *de facto* standard for sentiment analysis. Note that the evaluation methodology and the number of instances varies between the approaches which makes it difficult to compare the results. Having said that, our best accuracy is 4.55% lower than the best reported result of 91.7% by Abbasi et al.[1].

Our approach offers several key advantages though. Firstly, Abbasi et al's EWGA method is quite computationally expensive. Our best result, though less accurate, is much more computationally efficient, and can make both classification and training faster. Our method is also much simpler and easier to implement. Furthermore we start from a baseline that is 2% lower than Abbasi et al, which reduces the significance of the accuracy difference. The next best accuracy of 91% was achieved by König & Brill [6], who used pattern matching techniques. Their method is also very computationally expensive and has the additional drawback of requiring human intervention. Other approaches in the literature tend to have an accuracy that is similar to ours [7, 5, 9, 8], though without using feature selection.

6.1 Comparison of Classifiers

Figure 1 shows the best accuracy for the two classifiers with all the different feature weighting methods and feature selection methods. For the unigram based feature weights, our results confirm the findings of Pang et al.[9], which is that SVM classifiers are significanly more accurate than NB classifiers. However, for the word group based feature weights the results are less clear. In 8 of the 12 best results for the word group based feature weights, there was less than 0.5% difference between the NB and SVM classifiers, though in the remaining four cases the SVM clearly performed better. This finding shows that while SVM classifiers are substantially more accurate than NB classifiers for unigram based feature weights, they may not necessarily be the best approach for other types of features.

6.2 Comparison of Feature Selectors

Table 3 compares the results between the three feature selectors and the baseline where no feature selection was used for both SVM and NB. The results show that

	PD	SWNSS	SWNPD
0		14,617 (28.71%)	
0.125	18,149 (35.64%)	8,250 (16.2%)	7,433 (14.6%)
0.25	14,860 (29.18%)	7,094 (13.93%)	6,870 (13.49%)
0.375	10,342 (20.31%)	6,061 (11.9%)	5,943 (11.67%)
0.5	9,180 (18.03%)	4,919 (9.66%)	5,750 (11.29%)
0.625	6,716 (13.19%)	3,607 (7.08%)	4,868 (9.56%)
0.75	6,034 (11.85%)	2,302 (4.52%)	4,485 (8.81%)
0.875	5,767 (11.33%)	1,326 (2.6%)	4,431 (8.7%)
1	5,758 (11.31%)	739 (1.45%)	4,431 (8.7%)

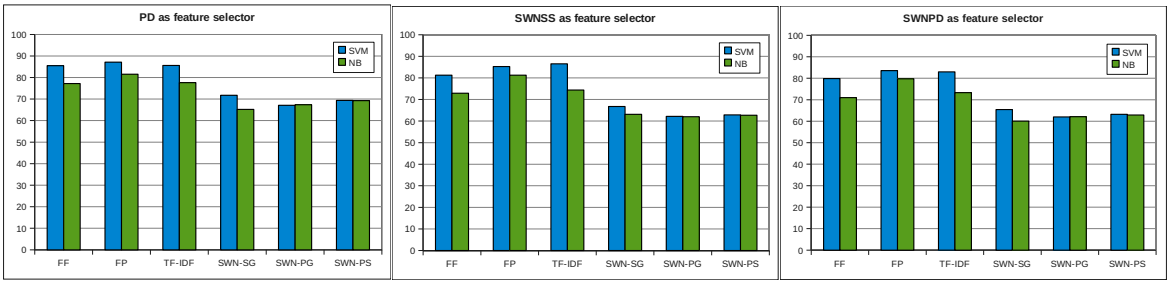
Table 2: Number of selected features by each feature selector for the various selection thresholds.

PD and SWNSS were successful in maintaining classification accuracy when used with appropriate thresholds, and SWNPD was able to maintain accuracy in all cases except for three. PD in particular was able to statistically significantly improve accuracy for nine out of 12 combinations of classifiers and feature weights, while SWNSS and SWNPD were able to improve accuracy in three and one cases respectively. Table 2 shows the number of features selected by each feature selection method at each threshold.

From the results in Table 3 one might conclude that PD was the best feature selection method. However, Figures 2a and 2b provide more information. They show that at low thresholds PD is quite successful at improving accuracy for all of the feature weights, but at higher thresholds accuracy drops sharply. Conversely, both SWNSS and SWNPD have relatively flat lines, indicating that they are more able to find the most effective features at any threshold.

6.3 Comparison of Feature Weights

Figure 2 a), c) and e) show the results for SVM for the three feature selection methods respectively, while Figure 2 b), d) and f) show the same for NB. The x-axis corresponds to the feature selection threshold; as the threshold increases, the number of selected features decreases. The starting point marked with a 'B' corresponds to the baseline where no feature selection is used. In general we found FP was the most accurate feature weighting method, which is in agreement with the results of Pang et al. [9]. Interestingly, the accuracy of FF increased steeply when feature selection was applied. We speculate that this was due to the presence of stop-words, so we conducted a further test of FF with SVM and all words appearing in 1000 or



(a) (b) (c)

Figure 1: Accuracy results (%) for SVM and NB when used with different feature selectors with different thresholds and the six feature weighting methods.

	None	PD	SWNSS	SWNPD
FF	72.5	85.5 ↑ t=0.25	81.3 ↑ t=0.375	79.85 ↑ t=0.125
FP	**85.95**	**87.15** t=0.125	85.3 t=0	**83.55** ↓ t=0.125
TF-IDF	85.9	85.6 t=0.125	**86.55** t=0	82.95 ↓ t=0.125
SWN-SG	65.5	71.75 ↑ t=0.25	66.75 t=0.5	65.45 t=0.125
SWN-PG	62.2	67.1 ↑ t=0.5	62.2 t=0.125	62 t=0.5
SWN-PS	62.85	69.35 ↑ t=0.25	62.85 t=0.375	63.2 t=0.125

(a) SVM Results

	None	PD	SWNSS	SWNPD
FF	68.65	77.2 ↑ t=0.5	72.9 ↑ t=0.875	71 t=0.125
FP	**80.65**	**81.5** t=0.25	**81.3** t=0.125	**79.75** t=0.125
TF-IDF	75.3	77.6 ↑ t=0.25	74.4 t=0.25	73.3 ↓ t=0.125
SWN-SG	59.9	65.2 ↑ t=0.375	63.1 ↑ t=1	60.05 t=0.125
SWN-PG	62	67.35 ↑ t=0.5	62 t=0.125	62.1 t=0.25
SWN-PS	62.7	69.25 ↑ t=0.25	62.7 t=0	62.9 t=0.125

(b) NB Results

Table 3: Comparison between the three feature selection methods and no feature selection for SVM and NB with all six feature weightings. The best accuracy (%) for each feature selector is shown in bold with statistically significant gains over the baseline marked with an up arrow (↑) and statistically significant losses marked with a down arrow (↓).

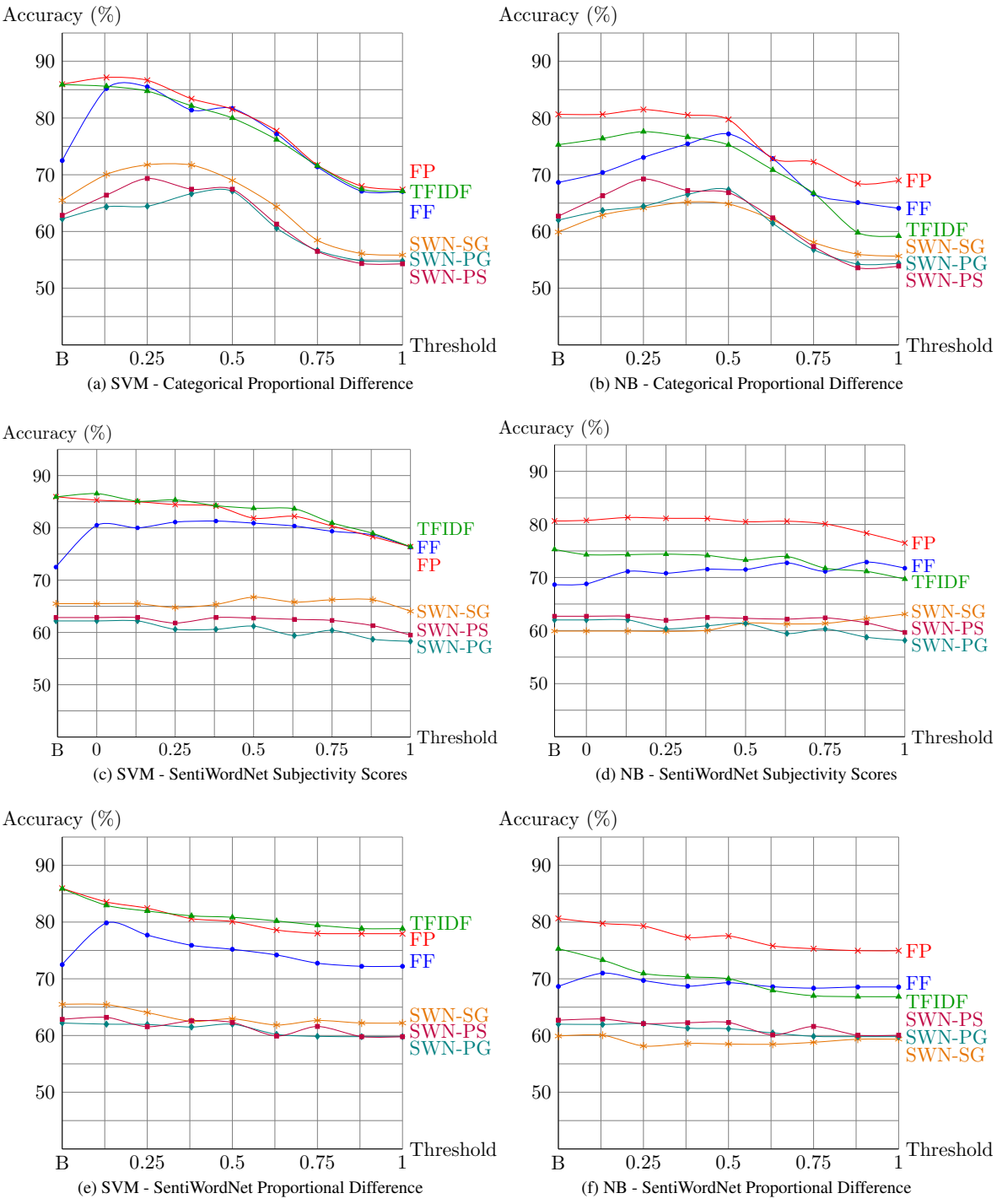
more documents removed. This achieved an accuracy of 83.95%, which indicates that the case for ignoring FF is not as clear cut as the results of Pang et al. [9] suggest.

Unigram based methods consistently outperformed the SWN word group methods for both SVM and NB with all combinations of feature weights and selectors. This finding is in agreement with the findings by Pang et al. [9] and Turney [14], who both noted that summing any out-of-context sentiment scores of individual words does not seem to capture the subtleties that exist in subjective writing. The features produced by SWN-SG, SWN-PG, and SWN-PS illustrate this point quite effectively since they all have approximately equal scores for positive and negative words regardless of the sentiment of the document. This is shown in Figure 3, where we would expect the positive bars to be higher for positive documents and the negative bars to be higher for negative documents. Instead the bars are approximately equal, indicating that there are about as many positive and negative words in positive documents as there are in negative documents.

7 Conclusions

In this paper we empirically and systematically evaluate the performance of a number of feature selection and feature weighting methods for sentiment analysis. In particular, we introduce two new feature selection methods - SWNSS and SWNPD - and compare them, at a number of selection thresholds, with PD, a recently proposed method, shown to be very successful for topic-based classification. We also introduce three feature weighting methods - SWN-SG, SWN-PG and SWN-PS - and compare their performance with the standard and popular FF, FP and TF-IDF methods. The experiments are conducted using two classifiers, SVM and NB, on the movie review data set that has become the *de facto* standard dataset for sentiment analysis.

We achieved an accuracy of 87.15% using PD as a feature selector, FP as a weighting mechanism and SVM as a classifier. This is a promising result as it is comparable with previous state-of-the-art results but is much less computationally expensive. All the feature selectors we tested were able to improve the



(a) SVM - Categorical Proportional Difference

(b) NB - Categorical Proportional Difference

(c) SVM - SentiWordNet Subjectivity Scores

(d) NB - SentiWordNet Subjectivity Scores

(e) SVM - SentiWordNet Proportional Difference

(f) NB - SentiWordNet Proportional Difference

Figure 2: Accuracy results (%) for SVM and NB when used with different feature selectors with different thresholds and the six feature weighting methods.

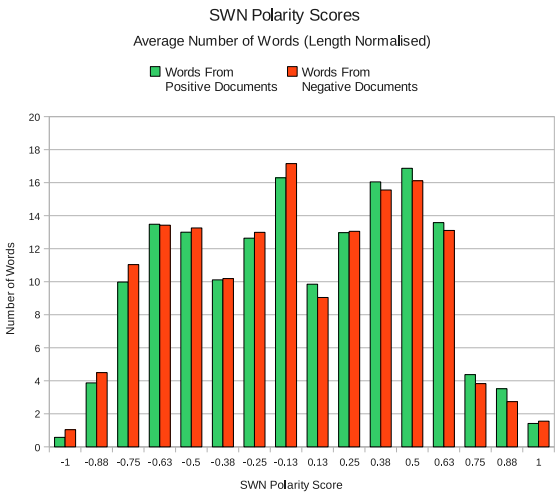


Figure 3: Average number of words with each SWN positive and negative score, from each class of documents.

performance over the baseline without feature selection when used with appropriate weighting methods. Overall, PD was the most successful at improving accuracy, although SWNSS was able to achieve the smallest feature sets whilst maintaining accuracy. The unigram based feature weights - FP, FF and TF-IDF - outperformed SWN-SG, SWN-PG and SWN-PS. Overall, FP was the most successful feature weighting method for both SVM and NB.

Future work will include evaluating more feature selection methods, particularly some of the common ones from text categorisation, such as information gain and χ^2. It would also be valuable to combine some of the feature selectors to see if better feature sets can be produced. Lastly, there would be significant value in repeating these tests on another data set.

References

[1] A Abbasi, HC Chen and A Salem. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions On Information Systems*, Volume 26, Number 3, 2008.

[2] A. Esuli and F. Sebastiani. SentiWordNet: a publicly available lexical resource for opinion mining. In *Proc. of LREC 2006 - 5th Conf. on Language Resources and Evaluation*, Volume 6, 2006.

[3] V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proc. of the eighth conf. on European chapter of the ACL*, pages 174–181, 1997.

[4] T. Joachims. *Making large-scale support vector machine learning practical, Advances in kernel methods: support vector learning*. MIT Press, Cambridge, MA, 1999.

[5] A. Kennedy and D. Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, Volume 22, Number 2, pages 110–125, May 2006.

[6] A. C König and E. Brill. Reducing the human overhead in text categorization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 598–603, 2006.

[7] T. Mullen and N. Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing EMNLP*, pages 412–418, 2004.

[8] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of the ACL*, pages 271–278. ACL, 2004.

[9] B. Pang, L. Lee and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP '02: Proc. of the ACL-02 conf. on Empirical methods in natural language processing*, pages 79–86. ACL, 2002.

[10] R. Prabowo and M. Thelwall. Sentiment analysis: A combined approach. *Journal of Informetrics*, 2009.

[11] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, Volume 34, Number 1, pages 1–47, 2002.

[12] M. Simeon and R. Hilderman. Categorical proportional difference: A feature selection method for text categorization. In *AusDM*, pages 201–208, 2008.

[13] R. M. Tong. An operational system for detecting and tracking opinions in on-line discussions. In *Working Notes of the ACM SIGIR 2001 Workshop on Operational Text Classification*, pages 1–6, 2001.

[14] P. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *ACL '02: Proc. of the 40th Annual Meeting on ACL*, pages 417–424. ACL, 2002.

[15] I. H Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. San Francisco, Morgan Kaufman Publishers, 2005.

The Use of Topic Representative Words in Text Categorization

Su Nam Kim[♠] *and Timothy Baldwin*[♡]

[♠] [♡] Computer Science and Software Engineering
[♡] NICTA VRL
University of Melbourne
Victoria, 3056, Australia
{snkim,tim}@csse.unimelb.edu.au

Min-Yen Kan[♣]

[♣] Computer Science
National University of Singapore
Singapore, 117417, Singapore
kanmy@comp.nus.edu.sg

Abstract *We present a novel way to identify the representative words that are able to capture the topic of documents for use in text categorization. Our intuition is that not all word n-grams equally represent the topic of a document, and thus using all of them can potentially dilute the feature space. Hence, our aim is to investigate methods for identifying good indexing words, and empirically evaluate their impact on text categorization. To this end, we experiment with five different word sub-spaces: title words, first sentence words, keyphrases, domain-specific words, and named entities. We also test TF·IDF-based unsupervised methods for extracting keyphrases and domain-specific words , and empirically verify their feasibility for text categorization. We demonstrate that using representative words outperforms a simple 1-gram model.*

Natural Language Techniques and Documents, Text Categorization

1 Background and Motivation

Automatic text categorization is the task of classifying documents into a set of predefined categories. It is one of the more heavily researched areas in natural language processing (NLP) due to its immediate applicability in applications such as text filtering [1], word sense disambiguation [11] and automated authorship attribution and genre classification [8].

The conventional approach to text categorization utilizes supervised machine learners such as support vector machines (SVMs) and Maximum Entropy (ME) models, and represents each document as a bag of word n-grams [40, 14, 10]. Empirically, SVMs have been shown to be superior to other machine learning techniques such as Naive Bayes (NB), Rocchio and decision trees over a range of tasks [40, 10].

While the predominance of research in text categorization is on machine learning models, there has also been significant research on feature extraction [4, 2, 24, 21] and feature weighting/selection [18, 41, 7]. While the majority of research has used simple n-grams to represent documents [4], this has been expanded in various ways, including word clusters [2], complex nominals [24], words from automatically extracted sentences [21], and title words/keyphrases(or keywords) [13]. Similarly, while most research has used simple term weighting (*TF* and/or *TF·IDF* variants), some have used attributes such as mutual information [18], chi-square [41], and gain ratio [7] to weight and/or select features.

Our interest is in the impact of different term types on text categorization. Our intuition is that not all word n-grams equally represent the topic of a document, and thus using all of them can potentially dilute the feature space. Hence, our aim is to investigate methods for identifying good indexing words, and empirically evaluate their impact on text categorization. To find representative topic words, we tested five different word groups: *title words, first sentence words, domain-specific words, named entities*, and *keyphrases*. *Title words* and *first sentence words* are based on the notion of *document zoning*. *Domain-specific words* and *named entities*, on the other hand, are typified as occurring with markedly-high occurrence in documents of particular domains. Finally, keyphrases are representative words, as identified by dedicated methods such as [12] and [36]. We also test combining the different term types with conventional terms n-grams.

A secondary area of interest in this research is exploration of the utility of unsupervised term extraction methods. As a result, we are particularly interested in the utility of unsupervised keyphrase and domain-specific word extraction methods on text categorization.

2 Zone-based Term Extraction

Our first term extraction method is based on document zoning, i.e. the extraction of terms based on the document structure. A common approach in keyphrase extraction and topic detection is to use titles as a representation of the document topic. For example, [26] showed that sentences in particular article sections, such as the introduction and conclusion, contain more keyphrases in scientific articles.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

In our work, we drew on methods such as [21] in extracting important sentences from documents based on the simple heuristic that the title and first sentence often contains key facts about the news story. From these observations, we select the *title words* and *first sentence words* as candidate terms. In each case, we extract out the component 1-grams, to minimize reliance on parsing or manual processing. We also filter terms by their combined occurrence in the document set, selecting only those terms which occur with frequency ≥ 1, 2 or 3. The final number of title words is 8,622, 3,878, and 2,357, for cutoffs of 1, 2 and 3, respectively, and the corresponding number of first sentence words is 11,565, 5,819, and 3,905, respectively. These numbers are based on the evaluation data described in Section 6.

3 Keyphrases

Keyphrases are simplex (i.e. 1-gram) nouns or noun phrases that represent the key ideas of the document. Keyphrases can serve as a condensed summary of the document and also as high-quality index terms. In the past, the majority of keyphrase studies have used three types of statistics to extract keyphrases: (1) document co-occurrence, i.e. *TF·IDF*-style statistics relating keyphrases to their relative co-occurrence across documents [12, 26]; (2) keyphrase co-occurrence, i.e. the extent to which keyphrases occur together in the same documents [37]; and (3) term co-occurrence, i.e. local contiguity of terms in keyphrases [28].

We quickly summarize related work first. `KEA` [12] is a very simple and popular keyphrase extraction and indexing tool. It uses two main features: *TF·IDF* to capture document co-occurrence, and *distance* to signify the relative locality of keyphrase occurrences within documents. These features have been broadly used in keyphrase extraction, e.g. by [37] in addition to keyphrase co-occurrence. [26] extended the basic `KEA` approach by applying linguistic features such as document zones. `GenEx` [36] uses more syntactic features, such as document positions and stemming. [3] uses head noun-based heuristics. [35] use modelling based on information loss between preceding and proceeding document extents. `Textract` [28] ranks keyphrase candidates by their degree of domain-specificity and term cohesion in a text analysis system. [38] uses information from clustered documents for keyphrase extraction over single documents.

3.1 Unsupervised Keyphrase Extraction

As keyphrases are known to be representative of document topics, it is also natural to use them as terms for document categorization. Hulth and Megayesi [13] used a supervised keyphrase extraction method, seeded with 500 abstracts annotated with keyphrases. To avoid documents without keyphrases, they controlled the number of keyphrases to between 3 and 12.

While supervised techniques work well, they require manually-built annotated corpora, which has implications both in terms of resource creation and domain adaptability. We are interested in minimizing such efforts, and thus committed to using unsupervised or minimally-supervised methods. To the best of our knowledge, very few unsupervised keyphrase extraction methods exist. Therefore, we used the features used in `KEA` to build our own unsupervised keyphrase extractor. That is, we use *TF·IDF* and *first position*, i.e. the inverse of the offset from the start of the document, such that documents which occur earlier in the document are preferred as keyphrase candidates. First, we calculate the score for each candidate as shown in (1), combining *TF·IDF* and first position.

Word set	T1(.02)	T2(.04)	T3(.06)
original	7,889	5,733	4,497
1+NP	25,343	15,257	10,679

Table 1: Number of collected keyphrases

$$Score = \textit{TF·IDF} + (1 - \frac{\textit{first position of } W_i}{\textit{\# of total terms}}) \quad (1)$$

We then extract the top-N candidates as keyphrases. In other keyphrase extraction research, N has typically been set to 15, but in our case, we decided to experiment with different thresholds. This is because the documents used in text categorization testbeds are short, and thus result in comparatively few keyphrase candidates. We selected thresholds by examining the score drop. Specifically, we set the threshold to the point at which the number of domain-specific terms gained at the current similarity value is no more than a fixed proportion (e.g. 2%) of keyphrases previously selected. Due to this use of threshold, our keyphrase extractor did not assign any keyphrases for a few documents.

Keyphrases can be either simplex nouns or NPs. [13] found that breakdown-keyphrases (i.e. all unigrams contained within a keyphrase) performed better for text categorization. Hence, we also convert keyphrases into their component unigrams. However, we observed that whole keyphrases are often better descriptors of the document topic (e.g. *import goods* vs. *goods*). Thus, we tested another set, called 1+NP, which combines 1-grams with the original keyphrases.

Table 1 shows the number of collected keyphrases for the entire document collection (see Section 6) at different threshold settings, for both the original keyphrases and 1+NP. Figure 1 additionally shows the proportion of documents containing different numbers of keyphrases for the three thresholds.

To assess the quality of our unsupervised keyphrase extractor, we sampled 100 documents from the training data and had two human annotators manually assign keyphrases to 50 documents each. The total number of manually-assigned keyphrases in the 100 sample documents was $1,486$. Performance is shown in Table 2.

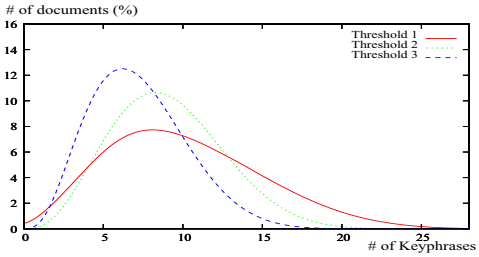


Figure 1: Proportion of documents assigned differing numbers of keyphrases

	Precision	Recall	Fscore
T1(.02)	9.76%	23.85%	13.85%
T2(.04)	15.32%	15.62%	15.47%
T3(.06)	21.02%	10.86%	14.32%

Table 2: Performance of keyphrase extraction

4 Domain-Specific Terms

Automatic domain-specific term extraction is a classification process where the terms are categorized using a set of predefined domains with supervised machine learning models. It has been studied for application in areas such as keyphrase extraction [12, 38] and word sense disambiguation [19].

Much of the work has been carried out using supervised machine learning techniques in the context of term categorization and/or text mining. [9] focused on simplex terms using corpus comparison, and verified the collected data using automatic and manual validation. [31] projected the categorized terms onto a predefined set of semantic domains exploiting web knowledge, and used the context to map the terms onto domains. [29] proposed an unsupervised method for extracting domain-specific terms, and used them to check word and keyword error rates.

In this paper, we test two unsupervised domain-specific word extraction approaches, drawing on work in the context of keyphrase extraction [16]. The first one (**D1**) is based on simple *TF·IDF*. The second method (**D2**) was proposed by [29], and is based on the difference in *TF* for a given domain relative to other domains, based on:

$$\textbf{D2} = domain\_specificity(w) = \frac{\frac{c_d(w)}{N_d}}{\frac{c_g(w)}{N_g}} \qquad (2)$$

where $c_d(w)$ and $c_g(w)$ denote the number of occurrences of term w in the domain text and general document collection, respectively. N_d and N_g are the numbers of terms in the domain corpus and in the general corpus, respectively. If term w does not occur in the general corpus, then $c_g(w)$ is set to 1; otherwise it is set to the highest count in the general corpus.

We use the same thresholding method for the two methods as described in Section 3.1.

Method	Term set	T1(.02)	T2(.04)	T3(.06)
D1	original	2,918	1,573	1,157
	1+NP	3,969	1,918	1,344
D2	original	3,692	2,759	2,368
	1+NP	7,169	5,021	4,215

Table 3: Number of collected domain-terms words

	Overlap	D1	D2
T1	1,612	55.24%	43.67%
T2	593	37.70%	21.49%
T3	404	34.92%	17.06%

Table 4: Overlap between domain-specific words collected by D1 and D2

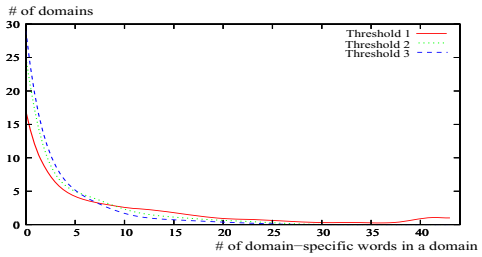


Figure 2: Number of domains containing differing numbers of domain-specific terms for D1

Table 3 details the number of terms and 1+NP extracted by D1 and D2 over the document collection described in Section 6, over three different threshold values. We also calculated the overlap in terms extracted by the two methods, and report the numbers in Table 4. The numbers in the second and third columns show the portion of terms extracted by the D1 and D2, respectively, which overlap with terms extracted by the second method.

The number of domains containing differing numbers of terms is shown in Figures 2 and 3. D1 produced less domain-specific words in total (as shown in Table 3), but the keyphrases are better distributed across the domains.

In separate research, we manually evaluated the terms extracted by the two methods, and found that D1 marginally outperformed D2 [16].

5 Named Entities

Named entity recognition is the task of identifying atomic elements in a document which belong to predefined categories such as location, person, and organization. It has been applied to contexts including Question-Answering (QA) [23] and information retrieval [34]. The standard approach is based on structured classification methods such as hidden Markov models (HMMs) or conditional random fields (CRFs). Recently, research has focused on semi-

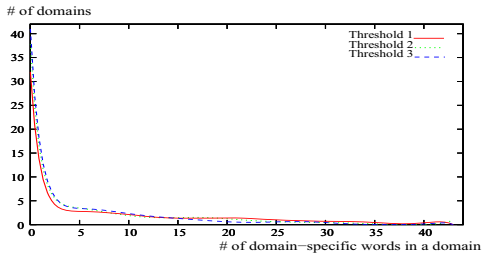


Figure 3: Number of domains containing differing numbers of domain-specific terms for D2

Length	F1($f \geq 1$)	F2($f \geq 2$)	F3($f \geq 3$)
original	11,431	6,538	4,650
1+NP	23,440	9,883	6,234

Table 5: Number of extracted named entities

supervised [27] and/or unsupervised approaches [5] to named entity recognition.

The relevance of named entities (NEs) to this research is that we expect they will be indicative of document domains. For example, *Gulf* and *Kuwait* often occur in the domain of *oil* and not other domains. Thus, we trial named entities as a term type in text categorization.

We experiment exclusively with the named entity recognition software of the University of Illinois at Urbana-Champaign (`UIUC NER`).[1] `UIUC NER` makes extensive use of non-local features and external knowledge resources (i.e. gazetteers extracted from Wikipedia), as well as semi-supervised learning. It identifies four entity types (i.e. person, location, organization and miscellaneous), and is reported to have achieved 90.80 F1-score over the CoNLL-03 NER shared task

Table 5 shows the number of named entities extracted by `UIUC NER` over our document collection (see Section 6). We used three different frequency cutoffs to select the candidate NEs ($f_{NE} \geq 1, 2, 3$), and once again experimented with both the original NEs and the 1+NP method of breaking down the NEs.

6 Text Categorization

We now describe our integrated approach for performing text categorization, incorporating the various extracted term types from the preceding sections.

As our dataset, we use the Reuters newswire corpus, with 21,450 articles from 1987, spanning 135 topics. The number of articles with no category label, one label and multiple labels are 31%, 57% and 12%, respectively. This dataset has been used widely for text categorization research. In particular, we use the *Modified Lewis Split*, comprising 7,771 training and 3,019 test documents across 90 domains.[2]

In preprocessing, we performed part-of-speech (POS) tagging using the *Lingua* POS tagger, and POS-sensitive lemmatization using `morpha` [22].[3] Then we built classifiers using SVM^{light},[4] with *TF·IDF* term weighting in an attempt to generate as competitive as possible a text categorization system.

As our benchmark, we use 1-grams with a frequency cutoff of 1, 2 and 3 (i.e. all terms occurring less than N times are ignored), along with stopping. The best results were achieved for a frequency cutoff of 3, with a micro-averaged F-score of 78.54%.

Table 6 shows the text categorization performance of the various term extraction methods, organized into four groups: (1) individual extraction methods; (2) the combination of all extraction methods; (3) the combination of individual extraction methods with 1-grams; and (4) the combination of all extraction methods with 1-grams. In each case, we report the micro-averaged precision, recall and F-score ($\beta = 1$) for the given method over the test data. All values which surpass the benchmark performance (F3) at a level of statistical significance (based on approximate randomisation, $p < 0.05$) are indicated in **bold**. In Table 6, F1, F2 and F3 refer to the three frequency cutoffs used for title words, first sentence words and named entities ($f \geq 1, 2, 3$), while T1, T2 and T3 refer to the three thresholds used for keyphrases and domain-specific words. We also present the performance over the top-10 topics in Table 7.

7 Text Categorization Results

Looking first at the individual methods (the top section of Table 6), we notice that only keyphrases were able to surpass the performance of the benchmark, closely followed by title and first sentence words, then named entities, and finally domain-specific terms. Almost no difference was observed between using the original terms extracted by each of the methods, and combining the original terms with their unigram components (1+NP). In general, the standalone methods tended to do better in terms of both precision and recall for lower cutoff/threshold values, that is larger numbers of noisier terms tended to boost performance across the board.

When we combine all five term extraction methods (considering D1 and D2 separately), the results exceed those of the benchmark in all cases for the lowest threshold/cutoff values, and in select cases for higher values. None of these gains were found to be statistically significant, and yet the result is encouraging as the best of the combined methods outperforms the best of the standalone methods,

[1] `http://l2r.cs.uiuc.edu/~cogcomp/asoftware.php?skey=FLBJNE`

[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

[3] The only use we made of the POS tags was in lemmatization.

[4] `http://svmlight.joachims.org/svm_multiclass.html`

Word	Length	F1/T1 Prec.	F1/T1 Recall	F1/T1 Fscore	F2/T2 Prec.	F2/T2 Recall	F2/T2 Fscore	F3/T3 Prec.	F3/T3 Recall	F3/T3 Fscore
Benchmark	1	87.15%	70.26%	77.80%	87.48%	70.53%	78.09%	87.98%	70.93%	78.54%
Title (T)	1	87.48%	70.53%	78.09%	87.58%	70.61%	78.18%	87.58%	70.61%	78.18%
First (F)	1	87.58%	70.61%	78.18%	87.48%	70.53%	78.09%	87.35%	70.42%	77.98%
Keyphrase (K)	1	88.01%	70.95%	78.57%	87.45%	70.50%	78.07%	87.68%	70.69%	78.27%
	1+NP	87.78%	70.77%	78.36%	87.65%	70.66%	78.24%	87.65%	70.66%	78.24%
Domain (D1)	1	86.26%	69.54%	77.00%	85.70%	69.08%	76.50%	83.44%	67.27%	74.49%
	1+NP	86.26%	69.54%	77.00%	85.70%	69.08%	76.50%	83.44%	67.27%	74.49%
Domain (D2)	1	84.67%	68.26%	75.58%	82.78%	66.73%	73.90%	81.75%	65.91%	72.98%
	1+NP	84.67%	68.26%	75.58%	82.78%	66.73%	73.90%	81.75%	65.91%	72.98%
NE (N)	1	86.16%	69.46%	76.91%	85.53%	68.95%	76.35%	84.87%	68.42%	75.76%
	1+NP	86.32%	69.59%	77.06%	85.53%	68.95%	76.35%	85.17%	68.66%	76.03%
T+F+K+D1+N	1	87.98%	70.93%	78.54%	87.91%	70.87%	78.48%	87.78%	70.77%	78.36%
	1+NP	88.11%	71.03%	78.66%	87.72%	70.71%	78.30%	87.91%	70.87%	78.48%
T+F+K+D2+N	1	88.05%	70.98%	78.60%	87.95%	70.90%	78.51%	88.01%	70.95%	78.57%
	1+NP	88.15%	71.06%	78.69%	88.08%	71.01%	78.63%	88.25%	71.14%	78.77%
B3+Title	1	87.72%	70.71%	78.30%	87.85%	70.82%	78.42%	87.55%	70.58%	78.15%
B3+First	1	87.78%	70.77%	78.36%	87.62%	70.63%	78.21%	87.82%	70.79%	78.39%
B3+Keyphrase	1	88.18%	71.09%	78.72%	87.85%	70.82%	78.42%	88.05%	70.98%	78.60%
	1+NP	88.31%	71.19%	78.83%	88.38%	71.25%	**78.89%**	88.15%	71.06%	78.69%
B3+D1	1	87.95%	70.90%	78.51%	88.08%	71.01%	78.63%	87.95%	70.90%	78.51%
	1+NP	87.95%	70.90%	78.51%	88.08%	71.01%	78.63%	87.95%	70.90%	78.51%
B3+D2	1	87.45%	70.50%	78.07%	87.32%	70.59%	77.95%	87.68%	70.69%	78.27%
	1+NP	87.45%	70.50%	78.07%	87.32%	70.59%	77.95%	87.68%	70.69%	78.27%
B3+NE	1	87.58%	70.61%	78.18%	87.68%	70.69%	78.27%	87.98%	70.93%	78.54%
	1+NP	87.58%	70.61%	78.18%	87.65%	70.66%	78.24%	87.45%	70.50%	78.07%
B3+T+F+K+D1+N	1	88.28%	71.17%	78.80%	88.31%	71.19%	78.83%	88.25%	71.14%	78.77%
	1+NP	88.44%	71.30%	**78.95%**	88.15%	71.06%	78.69%	88.21%	71.11%	78.75%
B3+T+F+K+D2+N	1	88.31%	71.19%	78.83%	88.28%	71.17%	78.80%	88.48%	71.33%	**78.98%**
	1+NP	88.44%	71.30%	**78.95%**	88.38%	71.25%	**78.89%**	88.48%	71.33%	**78.98%**

Table 6: Performance of text categorization

Benchmark (F3)	Individual	Individual+1-grams	All candidates	All candidates+1-grams
89.55%	89.59%	89.96%	90.02%	90.07%

Table 7: Performance over the top-10 topics

suggesting that there is complementarity between the term extraction methods. Comparing D1 and D2, our simple *TF·IDF*-based unsupervised term extraction method is marginally superior to D2 (the method of [16]).

Next, when we combine the individual methods with the terms from the benchmark method, the results improve uniformly, with the best-performing method (keyphrases with 1+NP terms) surpassing the benchmark method at a level of statistical significance. This indicates that keyphrases, as extracted using our adaptation of KEA, can successfully complement simple 1-grams in text categorization.

Finally, when we combine the benchmark term representation with all of the term extraction methods, we again achieve statistically significant gains almost 50% of the time, once again pointing to the utility of term extraction methods in text categorization applications. Comparing these results with those for the standalone term extraction methods combined with the benchmark system, the full set of five methods is not able to improve significantly beyond the performance of keyphrase extraction with the benchmark system.

Looking to the results over the top-10 topics, we find a similar trend, with keyphrases producing the best standalone performance, and all term extraction methods combined with 1-grams producing the best overall performance.

8 Conclusions

In this work, we evaluated the impact on text categorization of five representative term extraction methods, namely title words, first sentence words, keyphrases, domain-specific words, and named entities. We used the output of the different methods, either individually or in combination, as the source of terms for text categorization, and verified that we were able to achieve statistically significant improvements over a benchmark text categorization method using either keyphrase extraction in combination with the benchmark term representation, or the combination of all term extraction methods, again in combination with the benchmark term representation. On the basis of this, we concluded that keyphrases were the pick of the terms experimented with, but also that there is complementarity between the different term types.

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] G. Amati and D. DAloisi and V. Giannini and F. Ubaldini, A framework for filtering news and managing distributed data, Journal of Universal Computer Science, 1997, 3(8), pp. 1007–1021.

[2] L.D. Barker and A.K. McCalluma, Distributional clustering of words for text categorization, In Proceedings of 21st ACM International Conference on Research and Development in Informatoin Retrieval, 1998, pp.96–103.

[3] K. Barker and N. Cornacchia, Using noun phrase heads to extract document keyphrases, In Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, 2000, pp. 40–52.

[4] W.B. Cavnar and J.M. Trenkle, N-gram-based text categorization, In Proceedings of SDAIR, 1994, pp. 161–175.

[5] M. Collins and Y. Singer, Unsupervised Models for Named Entity Classification, In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999, pp. 100–110.

[6] D. Okanohara and Y. Miyao and Y. Tsuruoka and J. Tsujii, Improving the Scalability of Semi-Markov Conditional Random Fields for Named Entity Recognition, In Proceedings of COLING/ACL, 2006, pp. 465–472.

[7] F. Debole and F. Sebastiani, Supervised term weighting for automated text categorization, In 18th ACM Symposium on Applied Computing, 2003, pp.784–788.

[8] J. Diederich and J. Kindermann and E. Leopold and G. Paass, Authorship attribution with support vector machines, Applied Intelligence, 2003, 19(1/2), pp.109–123.

[9] P. Drouin, Detection of Domain Specific Terminology Using Corpora Comparison, In Proceedings of the 4th LREC, 2004, pp. 79–82.

[10] S. Dumais and J. Platt and D. Heckerman and M. Sahami, Inductive learning algorithms and representations for text categorization, In Proceedings of CIKM, 1998, pp. 148–155.

[11] G. Escudero and L. Marquez and G. Rigau, Boosting applied to word sense disambiguation, In Proceedings of 11th European Conference on Machine Learning, 2000, pp. 129–141.

[12] E. Frank and G.W. Paynter and I. Witten and C. Gutwin and C.G. Nevill-Manning, Domain Specific Keyphrase Extraction, In Proceedings of the 16th IJCAI, 1999, pp. 668–673.

[13] A. Hulth and B. Megayesi, A Study on Automatically Extracted Keywords in Text Categorization, In Proceedings of the 21st COLING/ACL, 2006, pp. 537–544.

[14] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, In Proceedings of ECML, 1998, pp. 137–142.

[15] M. Kida, M. Tonoike, T. Utsuro and S. Sato, Domain Classification of Technical Terms Using the Web, Systems and Computers, 2007, 38(14), pp. 2470–2482.

[16] S. Kim, T. Baldwin and M-Y. Kan, An Unsupervised Approach to Domain-Specific Term Extraction, In Proceedings of the Australasian Language Technology Workshop 2009, to appear.

[17] Y. Ko and J. Park and J. Seo, Improving text categorization using the importance of sentences, Information Processing and Management, 2004, 40(1), pp. 65–79.

[18] D.D. Lewis, An evaluation of phrasal and clustered representations on a text categorization task, In 15th ACM International Conference on Research and Development in Informaton Retrieval, 1992, pp. 37–50.

[19] B. Magnini and C. Strapparava and G. Pezzulo and A. Gliozzo, The role of domain information in word sense disambiguation, Natural Language Engineering, 2002, 8(4), pp. 359–373.

[20] Y. Matsuo and M. Ishizuka, Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information, International Journal on Artificial Intelligence Tools, 2004, 13(1), pp. 157–169.

[21] R. Mihalcea and S. Hassan, Using the essence of texts to improve document classification, In Proceedings of RANLP, 2005.

[22] G. Minnen and J. Carroll and D. Pearce, Applied morphological processing of English, Natural Language Engineering, 2001, 7(3), pp. 207–223.

[23] D. Molla and M. van Zaanen and D. Smith, Named Entity Recognition for Question Answering, In Proceedings of ALTW, 2006, pp. 51–58.

[24] A. Moschitti and R. Basili, Complex linguistic features for text classification, In Proceedings of 26th European Conference on Information Retrieval Research, 2004, pp.181–196.

[25] D. Nadeau and P.D. Turney and S. Matwin, Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity, In cogprints, 2006, pp. 266–277.

[26] T. Nguyen and M.Y. Kan, Key phrase Extraction in Scientific Publications, In Proceeding of International Conference on Asian Digital Libraries, 2007, pp. 317-326.

[27] S. Pakhomov, Semi-Supervised Maximum Entropy Based Approach to Acronym and Abbreviation Normalization in Medical Texts, In Proceedings of 40th ACL, 2002, pp. 160–167.

[28] Y. Park and R.J. Byrd and B. Boguraev, Automatic Glossary Extraction Beyond Terminology Identification, In Proceedings of COLING, 2004, pp. 48–55.

[29] Y. Park and S. Patwardhan and K. Visweswariah and S.C. Gates, An Empirical Analysis of Word Error Rate and Keyword Error Rate, In Proceedings of International Conference on Spoken Language Processing, 2008, pp. 2070–2073.

[30] L. Ratinov and D. Roth, External Knowledge and Non-local Features in Named Entity Recognition, In Proceedings of NAACL, 2009.

[31] L. Rigutini and E. Di Iorio and M. Ernandes and M. Maggini, Automatic term categorization by extracting knowledge from the Web, In Proceedings of 17th ECAI, 2006, pp. 531–535.

[32] G. Salton and A. Wong and C.S. Yang, A vector space model for automatic indexing, Communications of the ACM, 1975, 18(11), pp. 61–620.

[33] F. Sebastiani, Machine learning in automated text categorization, ACM Computering Surveys, 2002, 34(1), pp. 1–47.

[34] S. Sekine and K. Sudo and C. Nobata, Extended Named Entity Hierarchy, In Proceedings of LREC, 2002.

[35] T. Tomokiyo and M. Hurst, A Langauge Model Approach to Keyphrase Extraction, In Proceedings of ACL Workshop on Multiword Expressions, 2003, pp.33–40.

[36] P. Turney, Learning to Extract Keyphrases from Text, In National Research Council, Institute for Information Technology, Technical Report ERB-1057, 1999.

[37] P. Turney, Coherent keyphrase extraction via Web mining, In Proceedings of the 18th IJCAI, 2003, pp. 434–439.

[38] X. Wan and J. Xiao, CollabRank: towards a collaborative approach to single-document keyphrase extraction, In Proceedings of COLING, 2008, pp. 969–976.

[39] I. Witten and G. Paynter and E. Frank and C. Gutwin and G. Nevill-Manning, KEA:Practical Automatic Key phrase Extraction, In Proceedings of the fourth ACM conference on Digital libraries, 1999, pp.254–256.

[40] Y. Yang and X. Liu, A re-examination of text categorization methods, In Proceedings of SIGIR, 1997, pp. 42–49.

[41] Y. Yang and J.O. Pedersen, A comparative study on feature selection in text categorization, In Proceedings of 14th International Conference on Machine Learning, 1997, pp. 412–420.

Word Segmentation for Chinese Wikipedia Using N-Gram Mutual Information

Ling-Xiang Tang[1], *Shlomo Geva*[1], *Yue Xu*[1] *and Andrew Trotman*[2]

[1]School of Information Technology
Faculty of Science and Technology
Queensland University of Technology
Queensland 4000 Australia

{l4.tang, s.geva, yue.xu}@qut.edu.au

[2]Department of Computer Science
Universityof Otago
Dunedin 9054 New Zealand

andrew@cs.otago.ac.nz

Abstract *In this paper, we propose an unsupervised segmentation approach, named "n-gram mutual information", or NGMI, which is used to segment Chinese documents into n-character words or phrases, using language statistics drawn from the Chinese Wikipedia corpus. The approach alleviates the tremendous effort that is required in preparing and maintaining the manually segmented Chinese text for training purposes, and manually maintaining ever expanding lexicons. Previously, mutual information was used to achieve automated segmentation into 2-character words. The NGMI approach extends the approach to handle longer n-character words. Experiments with heterogeneous documents from the Chinese Wikipedia collection show good results.*

Keywords Chinese word segmentation, mutual information, n-gram mutual information, boundary confidence

1 Introduction

Modern Chinese has two forms of writings: simplified and traditional. For instance, the word China is written as 中国 in simplified Chinese, but as 中國 in traditional Chinese. Furthermore, a few variants of Chinese language exist in different locales including: Mainland China, Taiwan, Hong Kong, Macau, Singapore and Malaysia. For instance, a laser printer is called 激光打印机 in mainland China, but 鐳射打印機 in Hongkong, and 雷射印表機 in Taiwan.

In digital representations of Chinese text different encoding schemes have been adopted to represent the characters. However, most encoding schemes are incompatible with each other. To avoid the conflict of different encoding standards and to cater for people's linguistic preferences, Unicode is often used in collaborative work, for example in Wikipedia articles. With Unicode, Chinese articles can be composed by people from all the above Chinese-speaking areas in a collaborative way without encoding difficulties. As a result, these different forms of Chinese writings and variants may coexist within same pages. Besides this, Wikipedia also has a Chinese collection in Classical Chinese only, and versions for a few Chinese dialects. For example, 贛語(Gan) Wikipedia, 粵語(Cantonese) Wikipedia and others. Moreover, in this Internet age more and more new Chinese terms are coined at a faster than ever rate. Correspondingly, new Chinese Wikipedia pages will be created for the explanations of such terms. It is difficult to keep the dictionary up to date due to the rate of creation and extent of new terms. All these issues could lead to serious segmentation problems in Wikipedia text processing while attempting to recognise meaningful words in a Chinese article, as text will be broken down into single character words when the actual n-gram word can not be recognised. In order to extract n-gram words from a Wikipedia page, the following problems must be overcome:

- Mix of Chinese writing forms: simplified and traditional
- Mix of Chinese variants
- Mix of Classical Chinese and Modern Chinese
- Out of vocabulary words

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

There may be two options to tackle the these new issues in Chinese segmentation: (1) use existing methods and solutions; or (2) attempt new technique. In general, on the basis of the required human effort, the Chinese word segmentation approaches can be classified in two categories:

- Supervised methods, e.g. training-based, or rules-based methods, which require specific language knowledge. Normally, a pre-segmented corpus is employed to train the segmentation models e.g. PPM [13], or word lexicons need to be prepared for dictionary-based methods e.g. CRF [10].
- Unsupervised methods, which are less complicated, and commonly need only simple statistical data derived from known text to perform the segmentation. For instance, statistical methods using different mutual information formulas to extract two-character words rely on the bi-gram statistics from a corpus [2] [12].

The drawbacks of supervised methods are obvious. The effort of preparing the manually segmented corpus and parameters tuning is extensive. Also, the selected corpus mainly from modern Chinese text source may only cover a small portion of Wikipedia Chinese text. Plus, out-of-vocabulary(OOV) words are problematic for dictionary based methods. Different writing and different variant can lead to different combinations of characters representing the same word. Furthermore, according to the 2nd International Chinese Word Segmentation Bakeoff result summary [1], the rankings of participants results in different corpora not being very consistent which may indicate that the supervised methods used in their segmentation system are form (simplified or traditional) sensitive. To make use of these existing systems, the segmentation could be done the by converting all Chinese text into one unified form, simplified Chinese, for example. However, the resulting performance may be cast in doubt because the Chinese form conversion could not change the way the variant is used radically. For example, 鐳(Radium) character in 鐳射(laser, or 激光 simplified Chinese equivalent), will remain the same after such conversion, and 鐳射 would still not be recognised correctly as a word for the simplified-Chinese oriented segmentation system. At the time of writing, no performance of segmentation targeted on Chinese Wikipedia corpus using the-state-of-the-art systems are reported.

To avoid the effort of preparing and maintaining segmented text and lexicons for different corpora and potential issues when applying existing methods on Chinese Wikipedia articles, a simple unsupervised statistical method called n-gram mutual information(NGMI), which relies on the statistical data from text mining on Chinese Wikipedia corpus, is proposed in this paper. We extend the use of character-based mutual information to be segment-based in order to realize n-gram Chinese word segmentation. To achieve this goal, we introduce a new concept named boundary confidence(BC) which is used to determine the *boundary* between segments. The n-gram words are thus separated by the boundaries. The estimation of boundary confidence is based on the mutual information of adjoining segments. Since n-gram mutual information looks for boundaries in text but not for words directly, it overcomes the limitation of traditional usage of mutual information in the recognition of bi-gram words only.

2 Previous Studies

Pure statistical methods for word segmentation are less well studied in the Chinese segmentation research. They are the approaches that make use of statistical information extracted from text to identify words. The text itself is the only ``training'' corpus used by the segmentation models.

Generally, the statistical methods used in Chinese segmentation can be classified into the following groups: Information Theory (e.g. entropy and mutual information), Accessory Variety, t-score and Others. The accuracy of the segmentation is commonly evaluated using the simple recall and precision measures:

$$R = \frac{c}{N},\ and\ P = \frac{c}{n}$$

R is the recall rate of the segmentation
P is the precision rate of the segmentation
c is the number of correctly identified segmented words
N is the number of unique correct words in the test data
n is the number of segmented words in the test data

In a recent study, an accessory variety(AV) method has been proposed by Feng et al. [4] to segment words in a unsupervised manner. Accessory variety measures the probability of a character sequence being a word. A word is separated from the input text by judging the independence of a candidate word from the rest by using accessor variety criterion in considering the number of distinct preceding and trailing characters. An AV value of a candidate word is the minimal number of distinct preceding or trailing characters. The higher the number, the more independent the word is.

Information theory can help group character sequences into words. Lua [7] [8] and Gan [8] used the entropy measure in their word segmentation algorithm. A character sequence is a possible word if its overall entropy is lower than the total entropy of individual characters. Using this entropy theory for word judgment differently, Tung and Lee [14] considered the relationship of a candidate word with all possible preceding and trailing single-characters appearing in the corpus. The entropy values are calculated for those characters given that they occur in either the left hand side or the right hand side of this candidate word. If entropy values on either side are high, the candidate word could be an actual word. Mutual information and its derived algorithms are mainly used in finding bi-gram words.

Generally, mutual information is used to measure the strength of association for two adjoining characters. The stronger association, the more likely it is that they form a word. The formula used for calculating the association score for adjacent two characters is:

$$A(xy) = MI(x, y) = log_2(\frac{\frac{freq(xy)}{N}}{\frac{freq(x)}{N}\frac{freq(y)}{N}}) \simeq log_2(\frac{p(xy)}{p(x)p(y)}) \quad (1)$$

Here, $A(xy)$ is the association score of bi-gram characters xy; $freq(x)$ is the frequency of character x occurring in the given corpus; $freq(xy)$ is the frequency of two characters sequence (x followed by y) occurring in the corpus; *N* is the size, in characters, of the given corpus; p(x) is an estimate of the probability of character x occurring in corpus, calculated as $freq(x)/N$.

Based on Sproat & Shih's work, Dai et al. [2] further developed an improved mutual information(IMI) formula to segment bi-gram words using regression analysis:

$$Improved\,MI(xy) = 0.39 * log_2(p(xy)) - 0.28 * log_2(p(x)) - 0.23 * log_2(p(y)) - 0.32 \quad (2)$$

Their experiment results indicate using this formula has similar precision with that of original mutual information formula. They also developed another formula called contextual information(CI) formula which considers the frequency of the character preceding and the character following the bi-gram as well. Given a character sequence - $vxyz$, the association strength of bi-gram xy is calculated from:

$$CI(xy) = 0.35 * log_2(p(xy)) + 0.37 * log_2(p(v)) + 0.32 * log_2(p(z)) - 0.36 * log_2(p_{docwt}(vx)) - 0.29 * log_2(p_{docwt}(yz)) + 5.91 \quad (3)$$

Where p_{docwt} is the weighted probability for given the character or bi-gram in corpus by considering frequency of document where that character or bi-gram appears. The contextual information formula has been proven better in term of precision. There is a 7% improvement in average comparing with IMI formula.

3 N-Gram Mutual Information

To overcome the limitations of the mutual information approaches including its extensions IMI and CI in recognising words with two characters only, we propose a new simple unsupervised method - n-gram mutual information (NGMI) to segment n-gram words. Phrase mutual information is developed based on mutual information of segments by expanding it with contextual information. The idea is to search words by looking for the word boundaries inside a given sentence by combining contextual information, rather than looking for words. This was tried by Sun et al. [9]before. The two adjacent characters are ``bounded" or ``separated" through a series of judgment rules based on values of mutual information and difference of t-score. But for NGMI there are no rules involved, and mutual information of **segments** not just adjacent characters is considered. The boundary | of a sub-string $(L|R)$, consisting of a left substring L, and a right substring R, is determined based on the boundary confidence(BC). BC measures the association level of the left and right substrings. The Boundary Confidence of any adjoining segments is defined as:

$$BC(L|R) = MI(L, R) = sgn * (A(LR))^2 \quad (4)$$

Where,

$$sgn = \begin{cases} -1, & if\ A(LR) < 0 \\ 1, & if\ A(LR) >= 0 \end{cases}$$

Here, *A* is the association score of segment S_i and segment S_{i+1}. The lower the mutual information score of *L* and *R*, the more confident we are about the boundary. Generally speaking, characters that occur together frequently have a high mutual information value, indicating a strong association between them; it is then unlikely that there will be a boundary between them. When the boundaries are determined, the characters between the boundaries are considered as candidate words.

For any input string, we have

$$s = c_1c_2c_3 \cdots c_ic_{i+1} \cdots c_n \quad (5)$$

Here, s is the a input string - containing n Chinese characters. There may be a boundary between any pair of adjoining characters c_ic_{i+1}. Given a sequence of n characters we can derive a complete list of all possible segmentations. So for each possible segmentation S, we have

$$S = [c_1c_2 \cdots c_i] \,|\, [c_{i+1}c_{i+2} \cdots c_{i+k}] \,|\, \cdots \,|\, [c_{n-m}c_{n-m+1} \cdots c_n] = S_1S_2 \cdots S_x \quad (6)$$

$[c_1c_2c_3 \cdots c_l]$, or S_i,is a single segment from the entire sequences, a candidate word. Whether a certain segmentation has the correct words selected needs to be decided by a model that can make the best choice based on the ranking scores for all possible segmentations. These scores are calculated by accumulating all boundary confidence values. The n-gram mutual information formula is then defined as:

$$\begin{aligned} NGMI(S) &= [BC(S_1|S_2),\, BC(S_2|S_3),\, \cdots,\, BC(S_{n-1}|S_n)] \\ &= \sum_{i=1}^{n} BC(S_i|S_{i+1}) \\ &= \sum_{i=1}^{n} MI(S_i,\, S_{i+1}) \end{aligned} \quad (7)$$

In previous work by Sproat & Shih [12] and Dai et al. [2], the mutual information was only used to deal with two characters at a time. The N-Gram Mutual Information overcomes this limitation; it is using the mutual information in a manner which is different on two important counts. Firstly, by detecting boundaries the length of the words between adjacent boundaries are of variable lengths, and secondly, by looking at the segmentation of multiple words at once rather than one word at a time. In this paper, boundary confidence is calculated in a few varieties:

MI_{pair}, MI_{sum}, MI_{min}, MI_{max} and MI_{mean}.

Providing that the length of sub-string S_i is *n*, and the length of sub-string S_{i+1} is *m*, we have,

$$\begin{aligned} MI_{pair}(S_i, S_{i+1}) &= MI(C_{rightmost}(S_i), C_{leftmost}(S_{i+1})) \\ &= MI(C_n(S_i), C_1(S_{i+1})) \end{aligned} \quad (8)$$

Here, $C_n(S_i)$ or $C_{rightmost}(S_i)$ is the right most character of S_i , $C_1(S_{i+1})$ or $C_{leftmost}(S_{i+1})$ is the left most character of S_{i+1}. If considering only at most two characters each side of the boundary, we have

$$\begin{aligned} MI_{sum}(S_i, S_{i+1}) &= MI(C_n(S_i), C_1(S_{i+1})C_2(S_{i+1})) \\ &+ MI(C_n(S_i), C_1(S_{i+1})) \\ &+ MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})) \\ &+ MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})C_2(S_{i+1})) \end{aligned} \quad (9)$$

$$\begin{aligned} MI_{min}(S_i, S_{i+1}) &= min(MI(C_n(S_i), C_1(S_{i+1})C_2(S_{i+1})), \\ &MI(C_n(S_i), C_1(S_{i+1})), \\ &MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})), \\ &MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})C_2(S_{i+1}))) \end{aligned} \quad (10)$$

$$\begin{aligned} MI_{max}(S_i, S_{i+1}) &= max(MI(C_n(S_i), C_1(S_{i+1})C_2(S_{i+1})), \\ &MI(C_n(S_i), C_1(S_{i+1})), \\ &MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})), \\ &MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})C_2(S_{i+1}))) \end{aligned} \quad (11)$$

$$\begin{aligned} MI_{mean}(S_i, S_{i+1}) &= (MI(C_n(S_i), C_1(S_{i+1})C_2(S_{i+1})) \\ &+ MI(C_n(S_i), C_1(S_{i+1})) \\ &+ MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})) \\ &+ MI(C_{n-1}(S_i)C_n(S_i), C_1(S_{i+1})C_2(S_{i+1})))/k \end{aligned} \quad (12)$$

Here, $C_{n-1}(S_i)$ is the second character, if it exists, counting backward starting from boundary and the right most character of the left hand side sub-string S_i, $C_2(S_{i+1})$ is the second character, if it exists, counting forward starting from boundary and the left most character of the right hand side sub-string S_{i+1},

$$k = \begin{cases} 2, & length(S_i \, or \, S_{i+1}) <= 2 \, and \, at \, begining \, or \, end \, of \, S \\ 4, & length(S_i \, and \, S_{i+1}) > 2 \end{cases}$$

So

$$NGMI_{pair}(S) = \sum_{i=1}^{n-1} MI_{pair}(S_i, S_{i+1}) \quad (13)$$

$$NGMI_{sum}(S) = \sum_{i=1}^{n-1} MI_{sum}(S_i, S_{i+1}) \quad (14)$$

$$NGMI_{min}(S) = \sum_{i=1}^{n-1} MI_{min}(S_i, S_{i+1}) \quad (15)$$

$$NGMI_{max}(S) = \sum_{i=1}^{n-1} MI_{max}(S_i, S_{i+1}) \quad (16)$$

$$NGMI_{mean}(S) = \sum_{i=1}^{n-1} MI_{mean}(S_i, S_{i+1}) \quad (17)$$

Given overall scores of NGMI(*S*) for all possible segmentations, the lower the score of a segmentation, the more likely for it to have the right splits. For any particular split, if the boundary confidence MI values are negative, we are pretty confident that we are not splitting words in the middle. A detailed example may help explain this, given a short segmentation

$$S_1(ab|cdef) = [MI(ab, cdef)]$$

$$S_2(ab|c|def) = [MI(ab, c), MI(c, def)]$$

and calculate $NGMI_{min}(S_1)$ and $NGMI_{min}(S_2)$,

$$\begin{aligned} NGMI_{min}(S_1) = min(&MI(b, c), MI(ab, c), \\ &MI(b, cd), MI(ab, cd)) \end{aligned}$$

$$\begin{aligned} NGMI_{min}(S_2) = min(&MI(b, c), MI(ab, c), \\ &MI(b, cd), MI(ab, cd)) \\ &+ min(MI(c, d), MI(bc, d), \\ &MI(c, de), MI(bc, de)) \end{aligned}$$

4 Test Data

4.1 In-house Test Data

The following articles were chosen from the Chinese version of the Wikipedia: 本草纲目 (Bencao Gangmu), 马可・波罗 (Marco Polo), 張仲景 (Zhang Zhongjing),贫民百万富翁 (Slumdog Millionaire), 网络评论员 (50 Cent Party), and 风水 (Feng shui). All text from the above might be a mix of classical Chinese, simplified and traditional Chinese, and Chinese language variants. These pages were arbitrarily chosen simply as test pages.

4.2 Bake-off 2005 Test Data

In the Second International Chinese Word Segmentation Bakeoff test set, there are four groups of data (each having training, testing and gold-standard) provided by Academia Sinica, City University of Hong Kong, Peking University and Microsoft Research respectively [11]. The gold-standard data is segmented text following the word specifications defined by the each corpus creator. Each data set contains one form of Chinese writing either simplified or traditional.

	in-house	AS	CU	PKU	MSR
L.G.	81.27%	76.50%	79.58%	78.60%	73.14%
H.G.	18.73%	23.50%	20.42%	21.40%	26.86%

Table 1: Percentage of lower-gram and higher-gram words in test data

4.3 N-Gram Words Statistics For Test Data

Definitions:
Lower-gram words : 1-gram and 2-gram words
Higher-gram words : N-gram words, N > 3.

Table 1 shows around 20% of n-gram words in most test data sets are higher-gram words, and the rest of them (~80%) are lower-gram words, except that the standard gold data from Microsoft Research has lowest percentage of lower-gram words, only 73.14%. These statistical data indicates that simply searching for bi-gram words could not satisfy the need for n-gram word segmentation.

5 Experimental Design

5.1 String Pattern Frequency Table

The statistical information for the Chinese language is obtained through text mining of the Chinese Wikipedia XML corpus [3]. There are 56,662 documents, 27,360,399 Chinese characters, and 11,464 unique Chinese characters in total. For any character sequence with length less than 12, their corresponding frequencies are recorded in a string pattern frequency table. Since the size of such a complete table is very large, only those string patterns appearing in the corpus more than 216 times are kept. 216 is arbitrarily chosen to ensure that the frequency table can fit into program memory.

5.2 Stop Words

We recognise that an extra character like a preposition or a postposition cannot be separated from the actual word because of the strong statistical association. From the string pattern frequency table, the top 20 single-character words with the highest frequency (over 100,000 times) were selected as ``stop words".

5.3 Segmentation Runs

The Chinese segmentation experiments were performed using different segmentation methods and test data. Their run names and descriptions are listed in Table 2:

6 Segmentation Algorithms

6.1 MI Algorithm

The algorithm used in MI run to segment bi-gram words is that of Sproat and Shih [12]. As they did, the bi-gram frequency table keeps those words with a frequency greater than 4, and the threshold is set to 2.5. Given an input string of characters, the association strengths of each pair of adjoining characters are looked up. The pair with highest value is picked, then the second highest. If there are pairs with the same values, the right most pair is chosen. The bi-gram word with the highest value amongst the rest is repeatedly chosen until no pair's score is higher than the threshold. The remaining characters are then considered as one-character words.

6.2 IMI Algorithm

In Dai et al. IMI method, two sets of different algorithms, Comparative Forward Match(CFM) and Forward Match(FM) were implemented to perform the segmentation [2]. The segmentation process for both algorithms starts from the beginning of the sentence, and continues until the end. CFM is slightly more precise than FM in all their experiments. In our IMI run, we use the CFM algorithm to segment bi-gram words. The bi-gram frequency table only keeps words with frequency higher than 4, and the threshold is set to -2.5, which is the parameter used by Dai et al. having a segmentation result with the highest recall and lowest precision rate in all their IMI experiments. Given the sentence ABCDE, for example, the steps of segmenting it with CFM algorithm are:

First only the bi-gram AB is considered. If the association score of it is lower than the threshold, A is a single character word. Then BC is next to be considered. However, if the score of AB is higher than the threshold, then both bi-grams AB and BC are considered. If BC also has a score above the threshold but AB's is higher, AB is then chosen as a word. On other hand, if BC has the higher value, A is then marked as a 1-character word and CD also needs to be considered to decide whether BC is a bi-gram word. This process repeats until all words are segmented.

6.3 NGMI Algorithm

Given a Chinese character sequence: $s = c_1c_2c_3 \cdots c_ic_{i+1} \cdots c_n$, the word segmentation process using the NGMI has following steps:

1. The first *x* characters (in the experiments, *x* was set to 11) : $c_1c_2c_3 \cdots c_x$ are retrieved from the unsegmented text *s* for segmenting.
2. Build a list of all possible segmentations - S_{list} of the *x* characters. The upper bound of S_{list}equals $2^x - 1$. For 11 characters, there will be 1023 permutations, but segmentations will be removed from the list if they contain any

Run Name	Description
ICTCLAS	With ICTCLAS Chinese word segmentation system online demonstration version [6], from Chinese Academy of Sciences, using the in-house test data. It was developed based on multi-layer hidden Markov model [5]
MI	With original mutual information formula [12]using the in-house test data
IMI	With the improved mutual information formula proposed by Dai et al. [2]using the in-house test data
NGMI_PAIR	With $NGMI_{pair}$ formula using the in-house test data
NGMI_SUM	With $NGMI_{sum}$ formula using the in-house test data
NGMI_MIN	With $NGMI_{min}$ formula using the in-house test data
NGMI_MAX	With $NGMI_{max}$ formula using the in-house test data
NGMI_MEAN	With $NGMI_{mean}$ formula using the in-house test data
NGMI_MIN_SW	With $NGMI_{min}$ formula combining stop-words judgment using the in-house test data. The segmentation process repeats on already segmented words with length more than two characters, the words will be further split if the conditions as stipulated are met, (see ``step 6" in the segmentation algorithm)
NGMI_MIN_SW_AS	Same with NGMI_MIN_SW run but using the Academia Sinica test data
NGMI_MIN_SW_CU	Same with NGMI_MIN_SW run but using the City University of Hong Kong test data
NGMI_MIN_SW_PU	Same with NGMI_MIN_SW run but using the Peking University test data
NGMI_MIN_SW_MSR	Same with NGMI_MIN_SW run but using the Microsoft Research test data

Table 2: The list of all segmentation runs

substring that is not in frequency table.

$$S_{list} = \begin{cases} c_1|c_2c_3\cdots c_x \\ c_1|c_2|c_3\cdots c_x \\ c_1|c_2c_3|\cdots c_x \\ \cdots \end{cases}$$

3. For each boundary in the candidate segmentation, apply boundary confidence calculation, and sum the BC scores for each segmentation.

4. Sort S_{list} based on the segmentation scores in ascending order. It means that the lower score, the more likely it is to have the correct boundaries.

5. Choose the first segmentation (having highest rank) as the best segmentation:

$$S_{best} = c_1c_2c_3\cdots c_x = W_1W_2\cdots W_y$$

6. This step will only be executed if this is a stop words elimination run (NGMI_MIN_SW or NGMI_MIN_SW_AS, etc.). For any word-$W_i(c_1c_2\cdots c_k)$ in the best segmentation S_{best} with length more than two characters, it will be further broken down as in previous four segmentation steps(step 2 to step 5) into:

$$W_i = w_{i1}w_{i2}\cdots w_{iz}$$

This further segmentation will be accepted only if it meets the following conditions: both the first segment (w_{i1}) and the last segment (w_{iz}) of W_i are not one-character word; or if either w_{i1} or $w_{iz}$is a one-character word and it is in stop words list. For example, if W_2contains a stop word w_{21} at the beginning, then the best segmentation S_{best} from step 5 now become:

$$S_{best} = W_1[w_{22}w_{23}\ldots w_{2z}]\cdots W_y$$

7. Accept all the segments of S_{best} as words except for putting the last word W_y back into the unsegmented text. The last segmented word is returned to the unsegmented text since the split of these *x* characters was arbitrary and the best segmentation S_{best}may have split a long word in the middle.

8. Start the segmentation loop and repeat the segmentation process from step 1-7, until all the remaining characters are consumed.

The current version of NGMI algorithm isn't optimised yet. The segmentation performance is approximately 6000 words per second.

7 Evaluation and Analysis

In this section we compare the performance of the different segmentation runs on both the in-house test data and the bake-off test data. The comparison of the precisions in the in-house test data is used as major performance measurement for identifying the best NGMI variant. Also, the performance of segmentation runs on the all data is measured by overall recall rate.

7.1 Runs On the In-house Test Data

The precision values and their corresponding numbers of correctly identified words in each run against in-house test data are

given in Table 3. The recall figures of all runs on the in-house test data are given in Table 4.

Table 3 shows the mutual information runs are inherently limited by selecting only bi-gram words; and the NGMI runs are able to extract words with up to seven characters, even though the NGMI runs achieve only around 50% precision rate overall. The number of correctly identified words are similar for all runs on the in-house test data. The mutual information runs identify a high number of bi-gram words accurately, and the NGMI_MIN_SW run produces similar results but with more higher-gram words correctly identified. The results of the NGMI_MIN_SW run demonstrate an increase in the overall precision rate, reaching 62.64% from 53.52%, but the numbers of correctly identified higher-gram words drop. Some of the correct n-gram words are split and lost due to the further segmentation.

Despite the loss of correctly identified n-gram words, the NGMI_MIN_SW run still has the highest recall rate of all runs. The recall rates of mutual information runs, 69.17% and 68.61% respectively, come second and third. Other NGMI runs have slightly over 60% recall rate. The recall rate of ICTCLAS(56.58%) is low considering its relatively high precision. And considering the number of the single character words identified by the ICTCLAS run on the in-house test data is significantly higher than those in other runs but with a low precision, this suggests that the ICTCLAS online word segmentation system is accurate at recognising one form of written Chinese(either simplified or traditional), but it fails in the other. In mixed form documents the use of ICTCLAS could be problematic.

Overall, the supervised methods normally restrict themselves to choose words from the lexicon only, so their segmentation results have relatively a small number of found words. This explains why ICTCLAS has a high precision but a low recall. In contrast, as there isn't a finite correct words set for NGMI runs, the number identified words could be huge. And that leads to the decrement in the precision because of the larger denominator.

n-gram	ICTCLAS		MI		IMI		NGMI_PAIR		NGMI_SUM		NGMI_MIN		NGMI_MAX		NGMI_MEAN		NGMI_MIN_SW	
1	36.35%	434	42.05%	410	42.87%	406	49.39%	365	48.95%	350	48.79%	362	48.65%	343	50.00%	342	49.40%	368
2	87.20%	1042	79.02%	1318	71.06%	1429	71.64%	1329	63.35%	1279	69.74%	1307	61.82%	1279	64.39%	1318	72.76%	1520
3	83.62%	97		0		0	18.49%	113	18.65%	124	19.51%	104	18.50%	123	19.16%	123	29.91%	96
4	75.68%	28		0		0	16.29%	29	13.51%	30	14.21%	27	13.30%	29	13.36%	29	33.33%	8
5	100.00%	3		0		0	9.09%	2	6.25%	2	4.76%	1	6.06%	2	6.06%	2	0.00%	0
6		0		0		0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0		0
7		0		0		0	0.00%	0	0.00%	0	50.00%	1	0.00%	0	0.00%	0		0
Overall	63.03%	1604	65.38%	1728	62.04%	1835	53.88%	1838	48.72%	1785	53.52%	1802	48.00%	1776	49.93%	1814	62.64%	1992

Table 3: The precision and corresponding number of correctly identified words in the in-house test data. Overall precision = # of correctly identified words / # of all detected words.

7.2 Runs On the Bake off Test Data

It has to be noted that all the segmentation runs on the bake-off test data are created directly using the string frequency table obtained from the Chinese Wikipedia corpus without any knowledge of the bake-off training data. The training text is in fact completely independent of the test corpus. The recall figures for all bake-off runs are given in Table 5.

Table 5 shows that the recall rates of all bake-off run are around 70%, which indicates the corpus independent ability of NGMI in segmenting n-gram words. Of course this can be attributed to the fact that the Chinese Wikipedia corpus is a mixed language corpus and hence it covers the language of the bake-off text. Table 4 and table 5 also show that the recall rate of NGMI method using $NGMI_{min}$ formula with stop words elimination is the highest, and consistent (all around

ICTCLAS	MI	IMI	NGMI_PAIR	NGMI_SUM	NGMI_MIN	NGMI_MAX	NGMI_MEAN	NGMI_MIN_SW
56.58%	60.95%	64.73%	64.83%	62.96%	63.56%	62.65%	63.99%	70.26%

Table 4: Recall rate of segmentation runs using in-house test data

NGMI_MIN_SW_AS	NGMI_MIN_SW_CU	NGMI_MIN_SW_PU	NGMI_MIN_SW_MSR
68.96%	72.85%	72.26%	69.86%

Table 5: Recall of segmentation runs on the bake-off test data

70%) through all the runs.

8 Conclusions

In this paper, we have presented a simple unsupervised method NGMI using purely the Chinese text statistics drawn from the Wikipedia corpus to segment n-gram words. It is based on mutual information theory, but overcomes the limitation of the original mutual information based methods in recognising only bi-gram words by introducing the judgment of boundary-confidence of the adjacent segments.

To examine the feasibility of segmentation with n-gram mutual information and to find the best n-gram mutual information formula, a set of segmentation runs including a run using a state-of-the-art word segmentation system (ICTCLAS), two runs using different mutual information formulas (MI and IMI) and five runs using different n-gram mutual information variants ($NGMI_{pair}$, $NGMI_{sum}$, $NGMI_{min}$, $NGMI_{max}$, and $NGMI_{mean}$) were produced for performance comparison. Our experiments show $NGMI_{min}$ method performed best among all variants. The precision, number of correctly identified words, and overall recall rate of NGMI segmentation runs show encouraging results in segmenting n-gram words for Chinese Wikipedia articles.

As NGMI is a simple unsupervised method without needing much knowledge of the language, it will certainly benefit the text processing, when segmentation is required and situations are new to the-state-of-the-art systems, by providing the baseline n-gram word segmentation.

References

[1] Second international chinese word segmentation bakeoff - result summary. `http://www.sighan.org/bakeoff2005/data/results.php.htm`.

[2] Dai, Y., Loh, T. E., and Khoo, C. S. G. A new statistical formula for Chinese text segmentation incorporating contextual information. 82--89.

[3] Denoyer, L., and Gallinari, P. The Wikipedia XML Corpus. Tech. rep.

[4] Haodi Feng, Kang Chen, C. K. X. D. *Unsupervised Segmentation of Chinese Corpus Using Accessor*. Springer Berlin / Heidelberg, 2005, pp. 694--703.

[5] Institute of Computing Technology, Chinese Academy of Sciences. Chinese lexical analysis system ICTCLAS. `http://www.ict.ac.cn/jszy/jsxk_zlxk/mfxk/200706/t20070628_2121143.html`.

[6] Institute of Computing Technology, Chinese Academy of Sciences. ICTCLAS(Institute of Computing Technology, Chinese Lexical Analysis System). `http://ictclas.org`.

[7] Lua, K. From character to word - An application of information theory. *Computer Processing of Chinese & Oriental Languages 4*, 4 (1990), 304--312.

[8] Lua, K., and Gan, G. An application of information theory in Chinese word segmentation. *Computer Processing of Chinese & Oriental Languages 8*, 1 (1994), 115--124.

[9] Maosong, S., Dayang, S., and Tsou, B. K. Chinese word segmentation without using lexicon and hand-crafted training data. In *Proceedings of the 17th international conference on Computational linguistics* (Morristown, NJ, USA, 1998), Association for Computational Linguistics, pp. 1265--1271.

[10] Peng, F., Feng, F., and McCallum, A. Chinese segmentation and new word detection using conditional random fields. 562.

[11] SIGHAN. Second International Chinese Word Segmentation Bakeoff Data. `http://www.sighan.org/bakeoff2005/`.

[12] Sproat, R., and Shih, C. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese & Oriental Languages 4*, 4 (1990), 336--351.

[13] Teahan, W. J., McNab, R., Wen, Y., and Witten, I. H. A compression-based algorithm for Chinese word segmentation. *Comput. Linguist. 26*, 3 (2000), 375--393.

[14] Tung, C.-H., and Lee, H.-J. Identification of unknown words from a corpus. *Computer Processing of Chinese & Oriental Languages 8*, 1 (1994), 115--124.

An Automatic Question Generation Tool for Supporting Sourcing and Integration in Students' Essays

Ming Liu
School of Elec. & Inf. Engineering
University of Sydney
NSW
liuming@ee.usyd.edu.au

Rafael A. Calvo
School of Elec. & Inf. Engineering
University of Sydney
NSW
rafa@ee.usyd.edu.au

Abstract *This paper presents a domain independent Automatic Question Generation (AQG) tool that generates questions which can be used as a form of support for students to revise their essay. The focus here is on generating questions based on semantic and syntactic information acquired from citations. The semantic information includes the author's name, the citation type (describing the aim of the cited study, its results or an opinion), the author's expressed sentiment, and the syntactic information of the citation. Pedagogically, the question templates are designed using Bloom's learning taxonomy where the questions reach the Analysis Level. We used 40 undergraduate students essays for our experiment and the Name Entity Recognition component is trained on 20 essays. The result of our experiment shows that the question coverage is 96% and accuracy of generated questions can reach 78%. This AQG tool will be integrated into our peer review system to scaffold feedback from peers.*

Keywords Question Generation, Electronic Feedback System for Sourcing and Integration in Students'Essay

1 Introduction

Progress made in question answering systems has motivated a recent growth in automatic question generation systems. Two types of question generation tasks are normally considered. The first is text-to-question, where a document is provided to an AQG system that generates a question for which the answer is contained in the text. The second type is as a component of an Intelligent Tutoring System where a dialogue between the student and the ITS, and a set of propositions, is used as the input to the AQG component. In this case the question is aimed at helping the student elicit an answer containing the propositions.

The former AQG systems can support reading comprehension tasks, automatically suggesting questions that tutors can use in their teaching. Similar systems can be used to generate questions in the medical or security domain, where a system suggest questions to a practitioner based on a the case file. The second type of AQG systems is useful in a growing number of tutoring systems that have natural dialogue capabilities (e.g. Autotutor discussed later).

In this study we are concerned with building an AQG component for a third type of pedagogical applications: supporting students in their academic writing. In this context the common way of addressing the AQG problem is substantially changed:

- The driver for the technology is pedagogical so the questions should be framed in a pedagogical theoretical framework.
- The domain may be very general and a corpora for background knowledge might not be available.
- The questions must be generated from a single document, instead of a whole corpora
- The target audience of the questions is the same author of the document. The author should know the answers, so the goal here is to trigger reflection or get the student to expand on a topic.

Most different genres of academic writing contain citations of third party work on which the student is expected to comment (as in a literature review) or which is being used as evidence in an argument. When writing an essay or literature review, students are expected to learn and reason from multiple documents which require the skill of *sourcing* (i.e., citing sources as evidence to support their arguments) and *Information Integration* (i.e., presenting the evidences in a cohesive and persuasive way).

The development of student's sourcing and integration skills can be supported by using trigger questions such as *Does the essay provide evidence for the claims it makes?* or *Does the conclusion follow from the argument?* But such questions are too general and not likely to provide strong support in the process of writing on a specific topic. More specific questions need to be asked.

Most of the current AQG systems rely on shallow semantic parsing with entity recognizers. For example, Name Entity Recognizer,Verbnet [14] and Framenet [1]

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

can only 'understand' the semantic role of the entities such as agent, time, location and object in a sentence and generate factual questions. To generate deep questions related to a student's essay, AQG systems depend on some type of domain knowledge. AutoTutor [8] can generate deep questions, using domain specific knowledge in Computer Literacy or Physics.

This paper describes a new AQG system that includes a name entity recognizer for citation extraction, a pattern-matching based classifier for citation type classification and a sentiment analysis component for detecting the author's opinion polarity. These pieces of information are used to generate template-based questions during student's academic writing activities and targeting specific levels of Bloom's learning objectives taxonomy. Section 2 provides a brief review of the extensive literature focusing on approaches and systems that support learning experiences with sourcing and integration as learning goals. Section 3 describes the system's architecture while Section 4 its evaluation, including coverage and correctness. Section 5 concludes.

2 Related Work

Natural Language Processing techniques have been used to develop a number of tutoring and feedback systems. Section 2.1 reviews some of the projects developing writing support tools, and Section 2.2 systems that generate questions automatically.

2.1 Electronic Feedback System for Sourcing and Integration

Numerous projects have used computational approaches to assessing and providing automatic feedback on writing, most of the focus being on the assessment [15]. Despite a variety of initiatives to improve the quality of automatic feedback the efficacy of the systems remains to be proven and more research is needed. Meanwhile providing timely and appropriate feedback at key stages of the writing process remains a manual task, and a serious challenge for university lecturers.

Some of the early systems include Writers Workshop a system developed by Bell Laboratories, and Editor [16] both focused on grammar and style. Studies on the impact of Editor [2] concluded that the pedagogical benefits of grammar and style checking are limited. It could also be argued that these systems only aimed at supporting writing to communicate and did not address the issue of supporting writing to learn, important in today's curriculum design.

SaK, a writing tutoring system developed at the University of Memphis [18] is based on the notion of voices that speak to the writer during the process of composition. SaK uses avatars to give the impression of giving each voice a face and a personality [18]. Each avatar provides feedback on a different aspect of the composition, saying what is good or bad about the text but without correcting it. SaK uses Latent Semantic Analysis (LSA) to calculate the average distance between consecutive sentences and provide feedback on the overall coherence of the text. LSA is a technique used to measure the semantic similarity between texts and has been described thoroughly elsewhere [11]. SaK can also analyze the purpose of a sentence, identifying clusters of topics amongst the students so when the topic of a new composition is not identified the student can be asked for an explanation or reformulation.

Sourcer's Apprentice Intelligent Feedback (SAIF) [3] is an automated feedback tool for writing essays which can be used to detect plagiarism, uncited quotation, lack of citations and limited content integration problems. Once a problem is detected, SAIF can give helpful feedback to the student as shown in Table 1.

Problem	Feedback prompts student to:
1a. Unsourced copied material (plagiarism)	Reword plagiarism and model proper format.
1b. Unsourced copied material (quotation)	Explicitly credit source and model proper format.
2. Explicit citations	Explicitly make a minimum of 3 citations.
3. Distinct sources mentioned	Cite at least 2 different sources.
4. Excessive quoting	Paraphrase more instead of relying on quotations too heavily.
5. Integration from multiple sources	Include a more complete coverage of the documents in set.

Table 1: Types of Problems SAIF addresses and the intended goal of feedback

SAIF also uses Latent Semantic Analysis (LSA) techniques for plagiarism detection, computing the similarity between each essay sentence and the source sentences in LSA semantic space. For finding the explicit citations, SAIF uses a Regular Expression Pattern Matching technique to detect the explicit citations by recognizing phrases containing the author's name (e.g. According to, As stated in, State). Evaluations showed [3] that SAIF provides helpful feedback for students to use more explicit citations in their essays. However, this tool only addressed some basic problems for sourcing and integration. Moreover, it required a large number of source documents to build the LSA semantic space and a large number of pattern matching rules had to be predefined.

Glosser is an automated feedback system for student's writing [17]. It uses textual data mining and computational linguistics algorithms to quantify features of the text, and produce feedback for the student. This feedback is in the form of generic trigger questions (adapted to each course) and document features that relate to each set of questions. For example, by analyzing the words contained in each paragraph, it can measure how close two adjoining paragraphs are. If the paragraphs are too far this can be a sign of what is called lexical cohesiveness and Glosser flags a small warning sign. Glosser (1.0) provides feedback on four aspects of the writing: structure, coherence, topics, and concept visualization.

Glosser does not address sourcing directly, but four trigger questions (and the text features above) are provided:

1. Are the ideas used in the essay relevant to the question?
2. Are the ideas developed correctly?
3. Does this essay simply present the academic references as facts, or does it analyse their importance and critically discuss their usefulness?
4. Does this essay simply present ideas or facts, or does it analyse their importance?

The AQG algorithms described here are designed to be integrated into Glosser and provide support for sourcing an integration of citation sentences. The students upload a composition and Glosser provides the different forms of feedback. Other approaches for including the automatically generated questions include embedding them within an email, or using them as part of a peer-review process.

2.2 Question Generation

One of the first automatic question generation systems proposed for supporting learning activities was AUTOQUEST [19]. In this case, as in most of the current research questions are generated from external sources that the student *reads* (as opposed to writes).

The approach used here is similar to that of Kunichika et. al. [10] who proposed an AQG approach based on both the syntactic and semantic information extracted from the original text based on DCG (Definite Clause Grammar). Their educational context was the assessment of grammar and reading comprehension around a story. The extracted syntactic features include subject, predicate verb, object, voice, tense and sub clause. The semantic information contains three semantic categories: noun, verb and preposition, used to determine the interrogative pronoun for the generated question. For example, in the noun category, several noun entities can be recognized including the Person, Time, Location, Organization, Country, City, Furniture. In the verb category, the bodily actions, emotional verbs, thought verbs and transfer verbs can be identified. It also builds the semantic links among the time, location and other semantic categories when an event occurs. Because this technique extracts substantial syntactic and time / space semantic information from sentences, the generated questions can be more sophisticated and provide better support. The empirical result shows that 80% questions were considered by experts as appropriate for novices learning English and 93% of the questions were semantically correct.

AutoTutor, developed by the Graesser et al [8] at the University of Memphis, is an ITS that improves student's knowledge in computer literacy and Newtonian physics through an animated agent asking a series of deep reasoning questions that follow Graesser-Person taxonomy [7]. In each of these themes a set of topics have been identified. Each topic contains a focal question, a set of good answer aspects, a set of hints, prompts or elaborations which used to elicit each good answer aspect, a set of anticipated bad answers and so on. The system initiates a session by asking a focal question about a topic and the student are expected to write an answer containing 5-10 sentences. The system can generate hints or prompts for the student to elicit the correct and complete answer. The authors showed that AutoTutor's questioning approach had a positive impact on learning with an effect size on a pretest post-test study of approximately 0.8 standard deviation units in the areas of computer literacy and Newtonian physics. However, the system is domain dependent and requires a large number of human resources to predefine the content of each topic.

3 System Design and Architecture

The AQG tool described here is designed to generate questions from a student's essay and a set of templates designed by the instructor. The system was evaluated using a corpus of student essays discussed in Section 4. Sentences from that corpus are used here as examples on how the questions are generated. The corpus contains essays on the topic "English as a Global language".

In this section we provide an overview of the system's architecture shown in Figure 1 and describe each step in a pipeline process. The input to the system is an essay and the output is the generated questions.

Table 2 shows an example of questions generated by the AQG tool and their mapping to cognitive levels in Bloom's Taxonomy. In this example, the questions are generated from the raw sentence written by a student as part of an essay.

The question generation process follows 3 steps shown in Figure 1:

Step 1. Pre-processing. This includes citation extraction, filtering 'noisy' segments, splitting complex sentences and sentence transformation if it uses a noun or passive voice to refer to resources. There are two major components to perform these tasks: 1 *Sentence Extractor*, performs citation sentence extraction using the combination of trained Stanford Name Entity Recognizer [5], and a Pronoun Resolver, which is implemented by finding the nearest Name Entity appeared before the pronoun, and 2 *Filter* performs the rest of tasks which involved to clean up "noisy" segment, split complex sentences, transform other types of citation form to reporting verb type by using Tregex Pattern Match Techniques[12].

Examples of students' compositions include:

1. **According to Crystal,** *more people in the world speak Chinese than any other language.*

Level	Description	Example
Recognition	Ability to identify the specific content.	1.Who is Graddol? 2.What does Graddol point to in his study?(Sourcing)
Recall	Ability to retrieve the specific content from memory.	The same to Recognition
Comprehension	Ability to understand the learning material in terms of generation inferences, interpretation information, explanation and summarization information.	Why would Graddol point to the social and economic inequality that the dominance of English could lead to? (What evidence does Graddol provide to prove that?) (Sourcing)
Application	Ability to apply the knowledge from the learning material to a problem or situation.	How did you present Graddol opinion as evidence to confirm the thesis in your essay?(Integration)
Analysis	Ability to disassemble the elements and find the relationship between elements.	1. Is Crystal against Graddol's opinion? 2. Since you say Crystal's opinion is against Graddol, can you find the contradictive evidence provided by Crystal? (Integration)

Table 2: An example of questions generated from the sentence *"Graddol on the other hand points to the social and economic inequality that the dominance of English could lead to"*.

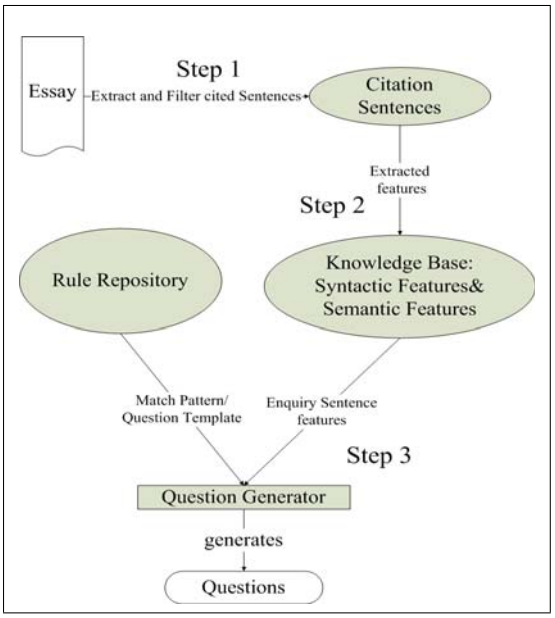


Figure 1: System Architecture

2. Although *Crystal and Graddol use many statistical evidence to discuss the spread of English as a Global language and the resulting consequences of this,***Wallraff actually challenges the notion that English has the global status most people believe it to have.**

3. *Wallraff's* **opinion** *is that there is a rate of growth of other languages in the USA which is higher than the rate of growth of English.*

In sentence 1 the noisy segment is shown in Bold. Sentence 2 is a complex sentence divided into two simple sentences shown in Bold and Italics respectively. Sentence 3 uses the noun 'opinion' to refer to the reference and the system will convert it into a reporting verb type (explained later). The new reported verb type version for Sentence 3 is:

Wallraff states that there is a rate of growth of other languages in the USA which is higher than the rate of growth of English.

To achieve these, the input sentence is parsed into a Phrase Structure tree, and then the Tregex Expressions are used to detect the syntactic patterns, and finally we use Tsurgeon to perform required opersions.

Tregex, developed by Stanford NLP group, is a powerful pattern matching technique which can match an individual word, regular expression, a POS tag or group of POS tags such as a Noun Phrase. Once the matched node is found by the Tregex, the Tsurgeon tool can perform delete, add, remove the node from the syntactic tree as shown in Figure 2 .

According to a study by Hyland [9], there are mainly three grammatical ways to refer to sources, which use Reporting Verb, Noun and Passive construction. Here, we call this as three grammatical patterns for citation. In our implementation, the citation sentence which is either Noun or Passive construction patterns would be transformed into reporting verb pattern because it would be easier to transform the citation sentence with reporting verb pattern into questions in later stage. Therefore, the Tregex Expression are defined to detect the three grammatical patterns and extract right Subject, Predicate Verb, Predicate, Auxiliary Verb for processing in later stage. The code segment in Figure 2 is used to split the complex sentence 2.

```
find_adv=TregexPattern.compile
          ("ADVP =rb >>,(NP >(S > ROOT)) | > S");
find_clause=TregexPattern.compile
          ("SBAR=sbar<(IN<Although|though)<S");
find_comma=TregexPattern.compile
          ("/,/=comma \$ (NP >( S > ROOT ))");
Tsurgeon.parseOperation("delete rb");
Tsurgeon.parseOperation("delete sbar");
Tsurgeon.parseOperation("delete comma");
```

Figure 2: An example of code segment using TregexPattern and Tsurgeon for splitting a complex sentence

Step 2. Syntactic and Semantic features. The purpose of this step is to extract the Syntactic feature and Semantic feature, such as the citation type (Study Result, Author's Opinion, Aim of Study) and the Author's Opinion Polarity. AQG then inserts the Semantic features as facts into a prolog knowledge base to be used in Step 3. There are two components to perform these tasks: *a Sentence Feature Extractor* which performs Syntactic Feature and Semantic feature extraction, and *a Sentiment Classifier* which detects the Author's Opinion Polarity.

Sentence Feature Extractor uses Tregex Expression on the Syntactic Tree for pattern match to extract syntactic features: Subject, Predicate Verb, Link Verb, Modal Verb and Predicate which are essential elements for question generation. In addition to Syntactic Features Extraction, *Sentence Feature Extractor* also uses predefined Reporting verb to define the Citation Type by matching the predicate verb in a sentence. In our database, Reporting Verb have been classified into three categories which correspond to different citation types.

Sentiment Classifier is used to detect the Author's opinion polarity about a topic. For the sentiment analysis AQG defines three elements: Opinion Holder, Topic and Opinion Polarity. At the moment, AQG only handles one Author appearing in a sentence and the opinion holder is the Author mentioned in the citation sentence. The topic is detected by choosing the most frequent noun or noun phrases among citation sentences expressed as a Sentence-Term matrix containing rows corresponding to the citation sentences and columns corresponding to the terms appeared in the sentence. Because AQG doesn't consider the number of times a word appears, a Binary Weighting schema is used. The topic is chosen by finding the term with maximum value and the Equation 1 is defined below, where $a_{ij} = 1$ if the term j appears in the citation sentence i, n is the number of citation sentences in an essay and the m is the number of terms appearing in these sentences.

$$\max_{\forall j \in m}\{\sum_{i}^{n} a_{ij}\} \quad (1)$$

For example, two citation sentences are extracted from an essay.

1. *"The increasing use of English is also negative in respect to the advantage gained by its native-speakers, not to mention the "threat to the identity of nations" through the inevitable increase of use of minority languages (Crystal, 1992)."*

2. *"Graddol on the other hand points to the social and economic inequality that the dominance of English could lead to."*

The word *'English'* has been chosen as Topic because it has the largest value 2 according to Equation 1. After the Opinion Holder and a Topic are detected, AQG detects the Opinion Polarity about the topic. The Opinion Polarity is decided by the Sentiment Region containing sentiment words in a sentence. The size of Sentiment Region is very important and AQG defines it as the set of nearest sentiment words around the topic in a sentence, and use the SENTIWORDNET [13] to determine the sentiment of a word. The SENTIWORDNET, a publicly available lexical resource for opinion mining, is an extension of WORDNET2 [4] and has defined three categories for a word sentiment with some magnitude: positive, negative and neutral.

Sentence	Opinion Holder	Topic	Polarity	Sentiment words list
S1	Crystal	English	Negative	(negative=-1.0), gain=0.5, increase=0.5
S2	Graddol	English	Negative	Inequality=-1.0

Table 3: an example of Author's Sentiment Classification

Table 3 shows the result of Sentiment Classification from the two citation sentences in the above example. *Crystal* is the Opinion Holder for Sentence 1, the *English* is chose as the Topic and the Opinion Polarity is *Negative* because AQG calculates the sum of the two nearest sentiment words: *Negative=-1.0* and *increase=0.5* which is negative. It is similar to sentence 2. Once finishing the sentiment analysis AQG will insert the extracted facts including Opinion Holder,Topic and Opinion Polarity into our prolog knowledge base showed in Figure 3 which will be used to infer if the Author's opinion is against/support each other.

```
#Facts
author(crystal).
author(graddol).
against(graddol,english).
against(crystal,english).
opinion(english).
#Inference rules
support(person1,noun1).
against(person2,noun2).
ally(X,Y):-support(X,Z),support(Y,Z),opinion(Z),X\=Y.
ally(X,Y):-against(X,Z),against(Y,Z),opinion(Z),X\=Y.
enemy(X,Y):-support(X,Z),against(Y,Z),opinion(Z),X\=Y.
enemy(X,Y):-against(X,Z),support(Y,Z),opinion(Z),X\=Y.
```

Figure 3: An example of Author's Opinion Polarity in Prolog knowledge base

Step 3. Generation This is the final step to generate template-based questions where the *Question Generator* uses the extracted syntactic features and the knowledge base, and then matches the predefined patterns in our Rule Repository, and finally generates template-based questions. In our current implementation, we have defined 5 rules and each rule defines the pattern for matching and 5 question templates. Each citation sentence would be applied by only one of the five rules. If a citation sentence matches both reporting verb and sentiment words, we would consider the rule for reporting verb because sentiment words have higher error rate to determine the citation type. In the future, we will use Machine Learning techniques to train a citation type classifier which will use the weight of selected features (reporting verb, sentiment words, numbers and etc) rather than current fixed pattern matching technique. Table 4 shows that the five rules are defined in our Rule Repository.

The Pattern Matching is based on the Reporting Verb and Word Sentiment in the citation sentence. In

Rules	Pattern	Citation Type	The Purpose of Generated Question
Rule 1	Reporting Verb	Opinion	Ask the student to provide evidence which support the Opinion (Sourcing), to provide other Author's contradictive opinion or result about the topic(Integration) if applicable
Rule 2	Reporting Verb	Aim of Study	Ask the student to identify the motivation for this Author's study and the outcome of the study (Sourcing).
Rule 3	Reporting Verb	Result	Ask the student to identify if the Author's Result is objective and what opinion does the result support (Sourcing)
Rule 4	Sentiment Word	Opinion	The same to Rule 1
Rule 5	Sentiment Word	Result	The same to Rule 3

Table 4: The Rule Definition for Patterns and Templates

our database, the reporting verb has been classified under one of three citation types and matches the predicate verb extracted from Step 2. If they are not matched, the sentiment words is used to detect the citation type. In our Rule repository, the question templates are designed according to the citation type. For example,

Graddol on the other hand points to the social and economic inequality that the dominance of English could lead to.

The predicate verb is *point to* and it matches a reporting verb under Opinion Type in our repository, then we apply Rule 1 shown in Table 4 to generate the template-based questions. Table 5 gives an example of question templates defined in Rule 1 and Table 2 shows an example of generated template questions defined in Rule 1. As you noticed, the following questions are generated by using prolog inference engine described in Step 2.

1. Is Crystal against Graddols opinion? 2. Since you say Crystals opinion is against Graddol, can you find the contradictive evidence provided by Crystal? (Integration)

If the sentence does not contain any reporting verb but some sentiment words, then it is also considered as the Author's Opinion. For example,

The increasing use of English is also negative in respect to the advantage gained by its native-speakers, not to mention the "threat to the identity of nations" through the inevitable increase of use of minority languages (Crystal, 1992).

As the word *Negative* has been detected as a sentiment word, the sentence is consider as expressing Author's Opinion, and then AQG applies Rule 4 to generate questions. Rule 5 is similar to Rule 4 for pattern matching except the sentence does not contain the sentiment words and the citation are expressed as Study Result.

Pattern	The predicate verb matches reporting verb for expressing Authors opinion purpose.
Template	• Who is [Author Name]? • What does [Author Name] [predicate verb Lemma]? • In the [Author Name]s study, do you agree that [Author Name] [Predicate]? Have you evaluated [Author Name]s opinion? • Why would [Author Name] [Predicate]? (What evidence does [Author Name] provide to prove that?) • How did you present [Author]'opinion as evidence to confirm the thesis in your essay? • Is [other Author Name] against [Author Name]'s opinion? Since you say [Other Author Name]s opinion is against [Author Name], can you find the contradictive evidence provided by [Other Author Name]?

Table 5: A Example of Question Template in Rule 1

4 Evaluation

This section describes a preliminary evaluation of the technique focused on two aspects : 1) The Question Coverage. 2) The Semantic Correctness of generated questions. In the last section we comment on planned evaluations that will study the learning impact of such a system, and self (the writer's view) and 3rd person reports on the quality features of the questions generated.

The evaluation was performed using 40 essays written by students at the University of Sydney. Students gave informed consent as approved by the Human Ethics Committee of the University of Sydney.

4.1 Question Coverage

The citation sentence extraction approach is based on the Author Name Recognition. The Expected Number of Questions depends on the total number of citation sentences. Table 6 shows that AQG can reach 96% coverage. This dataset contains 127 citation sentences(127*2=254 questions) and 123 citation sentences (123*2=246 questions) are extracted by AQG. We only evaluate 2 generated questions per citation sentence because some template-based questions only require Author Name, a relatively easy task, the evaluation does not include these questions.In other words, two questions are evaluated per citation sentence. For example, in Rule 1 question 3 and 4 are evaluated which is shown in Table 5. The problem for missing these citation sentence extraction is that some Author Names are not identified by the Name Entity Recognizer which cause these citation sentences can not be detected by AQG.

Expected Number of Questions	Number of Generated Questions	The Question Coverage
254	246	96%

Table 6: Question Coverage

4.2 The Correctness of Generated Questions

123 citation sentences were extracted from the 40 essays. Of these, 5 citation sentences had serious grammatical errors which caused the sentence Parser to fail. Therefore only the 118 remaining sentences were considered for evaluation. Because we only evaluate two questions per rule, the total number of evaluated questions is 236.

Table 7 shows that the semantic correctness of question reach to 78%. One of the main problems is that the rules used are too rough to handle multiple Authors appeared in a sentence. For example, the sentence

"*Wallraff suggests that the number of Spanish speakers in the USA has grown by 50% in the 1980-1990 census, thus refuting Crystal and Graddol's arguements for English being a global language.*"

Another major problem is the misclassification for the citation type: Opinion and Result. For example, the sentence

"*Many Chinese-speakers (four out of five of about 2.4 millions) in America prefer to speak Chinese at home rather than English (Wallraff, 1999).*"

In this case, although it contains *prefer* as a sentiment word with a positive term, the citation sentence should be considered as Study Result.

Rules	Number of Generated Questions	Number of Semantic Correct Questions
Rule 1	82	72
Rule 2	12	12
Rule 3	40	36
Rule 4	64	34
Rule 5	38	30
Total	236	184

Table 7: Question Generation Result

5 Conclusion and Discussion

Sourcing and Integration are important quality features in writing, and are part of the skills that college students must learn to master. The importance of asking questions has been shown to be an important part of teaching and learning experiences, so we designed an implemented a tool for automatically generating questions from an essay.

This domain independent AQG tool supports student's essay writing in the areas of sourcing and integration. Although we have not yet been able to assess the impact on student learning, the system was evaluated using real student essays.

Both the question coverage and the semantic correctness of generated questions were evaluated. Although the performance of Name Entity Recognizer would be different under different domain, the focus of current work is on interesting question generation. The pattern matching algorithm is based on Hyland's citation study that describes the most common ways of citing third party work. The algorithm captures the major forms of citation and as shown to have excellent accuracy.

Reasoning techniques were implemented in Prolog to detect when two authors are against each other and the generated question can reach to Analysis Level in Bloom's Taxonomy.

The tool can not only detect how many citations the writer has used in their essay but also generate specific or content related questions. Compared to current question generation systems, our tool can generate pedagogically deep questions in a somewhat domain independent form (it still requires templates that may required adaptation). It also presents novel results for using the authors' sentiment to generate questions.

Some limitations of this early work are obvious as the need to handle multiple authors in a sentence and to improve the classification of the citation type.

In future work, we will integrate the AQG tool into Glosser and to our peer review system so it provides extra information to support students' engagement with the writing (or peer-reviewing) process. for example, in a peer-reviewing scenario, the peer could not only evaluate the essay but also the author's answers to these automatically generated questions and provide better feedback. We will also improve the technique by adding ways for extracting multiple authors' arguments in a sentence and use other machine learning techniques to improve the Citation Type classification accuracy.

Acknowledgements

The authors would like to thank Jorge Villalon and Setphen O'Rourke for the development of TML and Glosser. Ming is partially supported by a N.I. Price scholarship. This project was partially supported by an Australian Research Council Discovery Project DP0986873.

References

[1] C. F. Baker, C. J. Fillmore and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[2] T. J. Beals. Between Teachers and Computers: Does Text-Checking Software Rea lly Improve Student Writing? *English Journal*, pages 67–72, 1998.

[3] M. A. Britt, P. Wiemer-Hastings, A. A. Larson and C. A. Perfetti. Using intelligent feedback to improve sourcing and integration in students' essays. *Int. J. Artif. Intell. Ed.*, Volume 14, Number 3,4, pages 359–374, 2004.

[4] C. Fellbaum and G. Miller. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[5] J. R. Finkel, T. Grenager and M. Christopher. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[6] X. Gong, Y. H.and Liu. Generic text summarization using relevance measure and latent semantic analysis. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–25, New York, NY, USA, 2001. ACM.

[7] A. C. Graesser and N. K. Person. Question asking during tutoring. *American Educational Research Journal*, Volume 31, pages 104–137, 1994.

[8] A. C. Graesser, K. VanLehn, C. P. Rosé, P. W. Jordan and D. Harter. Intelligent tutoring systems with conversational dialogue. *AI Mag.*, Volume 22, Number 4, pages 39–51, 2001.

[9] K. Hyland. Academic attribution: citation and the construction of disciplinary knowledge. *Applied Linguistics*, Volume 20, pages 341–367, 1994.

[10] H. Kunichika, T. Katayama, T. Hirashima and A. Takeuchi. Automated question generation methods for intelligent english learning systems and its evaluation. pages 1117–1124. Proc. of ICCE01, 2001.

[11] T. K. Landauer, D. S. McNamara, S. Dennis and W. Kintsch. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum, 2007.

[12] R. Levy and A. Galen. Tregex and tsurgeon: tools for querying and manipulating tree data structures. *In Proceedings of the Fifth International Conference on Language Resources and Evaluation.*, 2006.

[13] S. Osinski and D. Weiss. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. In *In IIPWM04*, pages 369–377, 2004.

[14] K. K. Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylyania, 2005.

[15] M. D. Shermis and J. Burstein. *Automated essay scoring: A cross-disciplinary perspective*, Volume 16. The MIT Press, 2003.

[16] E. C. Thiesmeyer and J. E. Thiesmeyer. *Editor:A System for Checking Usage, Mechanics, Vocabulary, and Structure*. New York: Modern Language Association, 1990.

[17] J. Villalon, P. Kearney, R. A. Calvo and P. Reimann. Glosser: Enhanced feedback for student writing tasks. In *Proc. Eighth IEEE International Conference on Advanced Learning Technologies ICALT '08*, pages 454–458, July 1–5, 2008.

[18] P. Wiemer-Hastings and A. C. Graesser. Select-a-Kibitzer: A computer tool that gives meaningful feedback on student compositions. *Interactive Learning Environments*, Volume 8, Number 2, pages 149–169, 2000.

[19] J. H. Wolfe. Automatic question generation from text - an aid to independent study. *SIGCUE Outlook*, Volume 10, Number SI, pages 104–112, 1976.

You Are What You Post: User-level Features in Threaded Discourse

Marco Lui and Timothy Baldwin
University of Melbourne
VIC 3010 Australia
saffsd@gmail.com, tb@ldwin.net

Abstract *We develop methods for describing users based on their posts to an online discussion forum. These methods build on existing techniques to describe other aspects of online discussions communities, but the application of these techniques to describing users is novel. We demonstrate the utility of our proposed methods by showing that they are superior to existing methods over a post-level classification task over a published real-world dataset.*

Keywords Document Management, Information Retrieval, Web Documents

1 Introduction

People like to talk. In particular, people like to talk to other people that share their interests, resulting in everything from hobby groups to clubs to professional associations. The internet gives people the ability to talk to each other on an unprecedented scale, and this has fostered the growth of publicly-accessible communities around a gamut of topics, from technology (Slashdot[1]) to knitting (Ravelry[2]), to social interaction for its own sake (Facebook[3]).

The most natural form of communication is through dialogue, and in the internet age this manifests itself via modalities such as forums and mailing lists. What these systems have in common is that they are a textual representation of a *threaded discourse*. The Internet is full of publicly-accessible communities which engage in innumerable discourses, generating massive quantities of data in the process. This data is rich in information, and with the help of computers we are able to archive it, index it, query it and retrieve it. In theory, this would allow people to take a question to an online community, search its archives for the same or similar questions, follow up on the contents of prior discussion and find an answer. However, anyone with any experience in searching for an answer to a technical question online would agree that the situation is seldom that simple.

[1] http://www.slashdot.org
[2] http://www.ravelry.com
[3] http://www.facebook.com

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

One problem with current approaches to accessing threaded discourse data is that they do not take into account the structure of the discourse itself. The bag-of-words (BoW) model standardly used in text classification and information retrieval (IR) discards all contextual information. However, even in IR it has long been known that much more information than simple term occurrence is available. In the modern era of web search, for example, extensive use is made of link structure, anchor text, document zones, and a plethora of other document (and query, click stream and user) features [15].

The natural question to ask at this point is, "What additional structure can we extract from threaded discourse?" Previous work has been done in extracting useful information from various implicit relationships between chunks of data in threaded discourse; we describe this in more detail in Section 2. However, one dimension that has not yet been explored is how we can use information about the identity of the participants to extract useful information from the structure of the discourse. In this paper we will examine how we can extract such *user-level* features, and how we can use them to improve performance over established tasks.

We use the term *threaded discourse* to describe online data that represents a record of messages exchanged between a group of participants. The two most common examples of this are forums and mailing lists. In this paper, the data that we examine in Section 5 originates from a site which bridges both. Indeed, the techniques we describe should generalize to any data which can be mapped into a similar structure.

There are several dimensions to the structure of threaded discourse that can be useful. For this paper, we will focus on the relationships between participants, which we refer to as the *user-level* structure. However, most instances of threaded discourse do not encode relationships between users explicitly. Therefore, we must infer the user-level relationships from relationships in other dimensions of the data. In particular, we focus on the following levels of threaded discourse structure:

Post-level: The individual unit contributions submitted by participants

Thread-level: Groupings of posts into a discussion on a particular topic

Our contribution in this paper is to develop methods for describing users based on their posts to an online discussion forum. We demonstrate the utility of our proposed methods by showing that they are superior to existing methods over a post-level classification task over a dataset from Nabble.[4]

The research presented in this paper forms a component of a larger research agenda on the utility of user-level characteristics in a variety of user forum tasks [**?**].

2 Related Work

This section provides a first-gloss overview of related work on thread- and post-level text classification, and feature-based approaches to capturing user characteristics. We return to present the aspects of this work that are most relevant to our research in greater detail, as detailed below.

Wanas et al. [16] detail a set of post-level features extracted based on a more structured approach. They evaluate their feature set over a classification task involving post and rating data derived from Slashdot. Their task involves classifying discrete posts into one of three quality levels (High, Medium or Low) where the gold-standard is provided by annotations from the community itself. We implement part of their feature set for experiments conducted in this paper; more detail is provided in Section 4.

Agrawal et al. [1] describe a technique for partitioning the users in an online community based on their opinion on a given topic. They find that basic text classification techniques are unable to do better than the majority-class baseline for this particular task. They then describe a technique based on modeling the community as a *reply-to* network, with users as individual nodes, and edges indicating that a user has replied to a post by another user. They find that using this representation, they are able to do much better than the baseline. Fortuna et al. [5] build on the work done by [1], defining additional classes of networks that represent some of the relationships present in an online community. We describe these networks in detail in Section 4, and adapt them to generate user-level features.

Weiner et al. [17, 18] propose a set of heuristic post-level features to predict the perceived quality of posts using a supervised machine learning approach. The data they evaluate over is extracted from Nabble, and they use the ratings provided by users as the gold-standard for a correct classification. They conclude that post-level classification using their feature set provides a substantial improvement over the majority-class baseline. We describe the dataset in greater detail Section 5.1, and use it as the basis of our evaluation.

In work on thread classification, Baldwin et al. [2] attempted to classify forum threads scraped from Linux-related newsgroups according to three qualities:

Task Orientation: Is the thread about a specific problem?

Completeness: Is the problem described in adequate detail?

Solvedness: Has a solution been provided?

They manually annotated a set of 250 threads for these qualities, and extracted a set of features to describe each thread based on the aggregation of features from posts in different sections of the thread. We apply a similar idea, but instead of aggregating over sections of the thread, we aggregate posts from a given user. The results from [2] were inconclusive, but we have found that their feature set can be effective when aggregated by user. Full details of the feature set are presented in Section 5.3.

3 Applications

While our experiments in this paper focus exclusively on a post-level classification task, this research has potential impact in a much wider range of settings, as outlined in this section.

3.1 Information Access

A key application of this paper is to support improved information access over internet forums, building on the work of Baldwin et al. [2]. The underlying intuition here is that not all contributions on a forum are equal in their usefulness, and that we often find that certain users are consistently outstanding in their contribution. Note that this is not the same as the user being an expert — other qualities come into play, such as how clear their explanations are, as well as how much effort they put into responding. Indeed, a relatively inexperienced user may post a detailed description of how he or she tackled a particular problem, which could be extremely valuable to a similar inexperienced user tacking a similar problem.

3.2 User Profiling

In some situations, we may wish to identify users with particular characteristics. For example, Kim et al. [9] use Speech Act Analysis to classify student contributions according to Speech Act categories, thereby identifying roles that participants play, using this information to identify when participants require assistance. This approach can be enhanced with user-level features.

3.3 User Karma

Karma is formalization of the notion of how influential a user is in an online community. It is the subject of much discussion in web communities as it is critical to the self-organizing structure of some communities, such as Reddit.[5] It is even more influential in other

[4] `http://www.nabble.com`

[5] `http://www.reddit.com`

Feature name	*Description*	*Type*
distribution	Mention of the name of a Linux distribution	Boolean
beginner	Mention of terms suggesting the posted is inexperienced	Boolean
emoticons	Presence of "smiley faces"	Boolean
version numbers	Presence of version numbers	Boolean
URLs	Presence of hyperlinks	Boolean
words	Number of words in post	Integer
sentence	Number of sentences in post	Integer
question sentence	Number of questions in post (sentences ending in '?')	Integer
exclaim sentence	Number of exclamations in post (sentences ending in '!')	Integer
period sentence	Number of sentences ending in a period	Integer
other sentence	Number of sentences not falling into the above three categories	Integer

Table 1: The ILIAD feature set

Feature name	*Description*	*Type*
onThreadTopic	Post's relevance to the topic of a thread	Float
overlapPrevious	Post's largest overlap to a previous post	Float
overlapDistance	How far away the previous overlapping post is	Integer
timeliness	Ratio of time interval from previous post to average inter-post interval in thread	Float
lengthiness	Ratio of length of post to average length of post in thread	Float
formatEmoticons	How often emoticons are used with respect to number of sentences	Float
formatCapitals	How often capitals are used with respect to number of sentences	Float
weblinks	How often weblinks are used with respect to number of sentences	Float

Table 2: The WANAS feature set

communities, where it is used to give incremental moderation powers to users (e.g. Stack Overflow[6]). There is no body of formal research associated with it, and sometimes the exact mechanism is a closely-guarded secret. User-level features are relevant to this because they can be used to more fully describe a user, which in turn can be used to compute a karma score that takes into account more aspects of the user's participation.

3.4 Automatic Grading

Lui et al. [12] use a text classification approach to perform content analysis. This task involves automatically grading participation by students in an online learning community. They make use of a fairly simplistic model of the content. It may be possible to improve their approach by extracting more detailed structural information from participants' contributions.

4 User-Level Features

In Section 2, we outlined existing methods for extracting features to describe posts and threads. In this section, we present methods for extracting features for describing *users*.

4.1 Aggregate

The first type of user-level feature we consider are features derived from aggregation over features describing individual posts. We implement two post-level feature sets. The first, which is henceforth referred to as ILIAD, is derived from [2] and is described in Table 1. The second, which is henceforth referred to as WANAS, is derived from [16], and is described in Table 2.

From each of ILIAD and WANAS we derive a user-level feature set by finding the mean of each feature value over all of the user's posts. These feature sets are referred to as $\text{ILIAD}_{\text{AGG}}$ and $\text{WANAS}_{\text{AGG}}$, respectively.

4.2 Network-Based

Fortuna et al. [5] present a method of describing forum data using Social Network Analysis. The network is a graph representation of relationships within the forum, reminiscent of algorithms such as PageRank [4]. In the case of PageRank, each node represents a webpage and each edge represents a hyperlink. In [5], the authors define 3 *author networks*, where each node represents an author, and 2 *thread networks*, in which each node represents a thread. The meaning of an edge varies for each network, and each edge may be directed or undirected according to the network.

The authors then use each of these networks to extract features on a per-post basis. We briefly summarize the method here; more detail is provided in [5].

For Author Networks, each post is assigned a feature vector v of length N, where N is the total number of nodes, or equivalently, the total number of authors in the network. v has at least one feature set to 1, which corresponds to the author of the post. Authors directly

[6] `http://www.stackoverflow.com`

connected to the post author in the network receive a feature value of 1, and authors that are second-level neighbours of the post author are set to a feature value of 0.5. All other values in v are set to 0. Since each post has a unique author, this network can be used to describe authors without modification.

For Thread Networks, the method for computing a feature vector is similar to that for Author Networks. The key difference is that in this instance, the vector v is of length T, where T is the total number of threads in the forum. Therefore, each value v_T in the vector describes a relationship to a particular thread. In [5], the authors are interested in the relationship between posts, so they assign to each post the feature vector of the thread it belongs to. However, in our case we do not wish to describe a post directly; instead, we are interested in describing the author. To do this, we consider every thread that the author has posted in. For each of these threads, we set the feature corresponding to the thread to 1. We then set all the immediate neighbours of the threads to 1 as well, and the second-level neighbours thereafter to 0.5.

In our work, we consider two Author Networks and one Thread Network:

POSTAFTER *(Author Network)*

POSTAFTER is modeled on the *reply-to* network described in [5]. Our data does not contain exact information about the reply structure in a thread, so we approximate this information by the temporal relationship between posts. Effectively, we have made the assumption that within a thread, each post replies to the post immediately preceding it in terms of the time-of-posting. We expect that this will generally be the case, but in the context of the original work by [1] on partitioning users by opinion, it is possible that, given three posts A, B and C, B and C both reply in objection to A, therefore defining a different network from ours. Nonetheless, our results will show that our approximation is admissible in that it can be used to augment a BOW feature set to exceed a benchmark result; we will present evidence of this in Section 5.3.

POSTAFTER is parameterized with two values: $dist$ and $count$. Being an *Author Network*, the nodes represent authors. Two authors $A1$ and $A2$ have a directed edge from $A1$ to $A2$ if and only if $A1$ submits a post to a thread that is within $dist$ posts of a post in the same thread by $A2$ on at least $count$ occasions. For our experiments, we used $dist = 1$ and $count = 3$.

THREADPARTICIPATION *Author Network*

THREADPARTICIPATION is implemented as described in [5]. In this network, nodes are again authors, and each undirected edge indicates that two authors have posted in the same thread on at least k occasions. In the original work, the authors set $k = 5$, but in out case, we use $k = 2$ as the network is too sparse for higher settings of k.

COMMONAUTHORS *Thread Network*

COMMONAUTHORS is implemented as described in [5]. In this network, nodes are threads, and each undirected edge indicates that two threads have at least m authors in common. We followed the original research in setting $m = 3$.

5 Evaluation

We evaluate the effectiveness of the features described in Section 4 by utilizing them for a classification task. In this paper, we focus exclusively on a post-level classification task, which allows us to assess the usefulness of user-level features in describing post-level data.

5.1 Dataset

The data set we are using is based on that from Weimer and Gurevych [17]. The data consists of 16562 posts across 2956 different threads. Separately, there are 4508 annotations spanning 4291 distinct posts, rating the quality of the post. Each annotation consists of an ordinal rating from 1 to 5 stars, with more stars indicating better quality. We filtered the annotated posts by removing all posts with an empty body. We also removed all posts that had an average rating of exactly 3.0. This eliminated posts that were rated 3 once, as well as posts that received contradictory ratings, such as a post rated 1 by one user and 5 by another, leaving 4094 rated posts. We divided posts into two groups, corresponding to posts with an average rating > 3.0, which we consider GOOD, and posts with an average rating ≤ 3.0, which we consider BAD. In the 4094 rated posts, there were 2060 GOOD posts and 2034 BAD posts. Our approach to filtering the data is generally consistent with that in [17]. Differences in our use of the dataset are discussed in Section 6.

5.2 Methodology

For each post, we extracted the feature sets described in Section 4, as summarized in Table 3. For user-level feature sets, we use the features corresponding to the post's author to describe the post. We evaluate various combinations of these feature sets by carrying out 10-fold cross-validation [10], as follows:

1. Divide the data randomly into 10 *partitions*
2. For each partition, train a classifier on the other 9 partitions
3. Use the trained classifier to predict the categories of the instances in the selected partition
4. Pool together the predictions from the 10 iterations and evaluate

The partitioning is performed once and re-used for each pairing of learner and feature set. We repeat this procedure using a number of different learners. The learners used, along with their parameter settings, are as follows:

Label	*Type*
BOW	Post
ILIAD	Post
WANAS	Post
$\text{ILIAD}_{\text{AGG}}$	User
$\text{WANAS}_{\text{AGG}}$	User
POSTAFTER	Author Network
THREADPARTICIPATION	Author Network
COMMONAUTHORS	Thread Network

Table 3: Feature sets used in classification

`SVM`: Support vector machines [8] as implemented in `bsvm` [6], using the package default values which correspond to an RBF kernel.

`SkewAM`: Nearest-prototype skew divergence, as implemented in `hydrat` [13]. This is a Rocchio-style approach [7], where a centroid is computed for each class by finding the arithmetic mean of all the instances of the class. Classification is then carried out by assigning the class of the single nearest neighbour. The distance metric we use is skew divergence [11], with a mixing parameter $\alpha = 0.99$.

`Maxent`: Maximum entropy modeling [3] as implemented in the `Maximum Entropy Toolkit` [19]. We use L-BFGS for parameter estimation [14], with 10 iterations of the training algorithm.

For each cross-validated result, we report the overall classification accuracy (Acc), which is the proportion of correct predictions made by the classifier; a larger number is, naturally, better. When comparing a result to a benchmark value, we also provide the p-value for a two-tailed paired t-test. We can conduct a paired t-test because for each result, the partitions used have been kept constant and thus the performance over them is directly comparable. The null hypothesis is always that the difference in the mean accuracy over all 10 partitions is identical for both results being compared. Therefore, a low p-value indicates that it is highly improbable that the two combinations of feature sets being considered have led to the same results. To facilitate discussion of statistical significance, we will consider a p-value < 0.05 to be statistically significant. This corresponds to the 5% significance level that is commonly reported. In tables, p-values that are statistically significant at the 5% significance level are shown in **bold**.

Our experiments were performed using `hydrat` [13], an open-source framework for comparing classification systems. `hydrat` provides facilities for managing and combining feature sets, setting up cross-validation tasks and automatically computing corresponding results.

5.3 Results

The baseline for this task is a majority-class (`ZeroR`) result of 0.489. Although this is a binary task, the majority-class result is less than 0.5 because the majority class varies across partitions. In 8 of the 10 partitions, it was the overall majority class (GOOD), whereas in 2 of the 10 partitions, it was the majority class in the training data but overall minority class (BAD).

Learner	*Accuracy*
`SVM`	0.780
`SkewAM`	0.812
`Maxent`	0.820
`ZeroR`	0.489

Table 4: Accuracy for each learner when utilizing only the BOW feature set

Feature Set	*Acc*	p
BOW	0.780	—
ILIAD	0.723	2.1×10^{-6}
WANAS	0.751	7.3×10^{-4}
$\text{ILIAD}_{\text{AGG}}$	**0.831**	2.4×10^{-6}
$\text{WANAS}_{\text{AGG}}$	**0.829**	2.7×10^{-4}
POSTAFTER	0.636	5.1×10^{-13}
THREADPARTICIPATION	0.670	1.1×10^{-10}
COMMONAUTHORS	0.671	4.2×10^{-11}

Table 5: Accuracy for each feature set over `SVM` (results higher than the baseline are highlighted in **bold**; p is the probability that the result differs from the benchmark only due to chance)

We establish benchmark results for this task using only the BOW feature set. The overall accuracy for each learner is summarized in Table 4. Immediately, it is apparent that the benchmark result is significantly better than the baseline. The best result over only the BOW feature set is attained by `Maxent`, with an accuracy of 0.820.

Next, we consider each learner over each individual feature set. For `Maxent` and `SkewAM`, this always leads to results that are below the BOW benchmark. For `SVM`, however, the aggregate features $\text{ILIAD}_{\text{AGG}}$ and $\text{WANAS}_{\text{AGG}}$ do better than the BOW benchmark, attaining an accuracy of 0.831 and 0.829, respectively. These are different from the BOW result with $p = 2.4\times10^{-6}$ and $p = 2.7\times10^{-4}$ respectively. Both results are statistically significant. We report results for each feature set in Table 5.

We then investigate the use of the various feature sets to augment BOW, as presented in Table 6. A fairly consistent picture emerges from this: the ILIAD and $\text{ILIAD}_{\text{AGG}}$ feature sets cause performance to drop when combined with the BOW feature set, whereas all other feature sets cause performance to rise with respect to BOW.

We also experimented with *feature ablation*, by examining the result of removing one feature set at a time from the full set of features. The results for this are reported in Table 7. Surprisingly, removing a particular

Learner	*Feature Sets Present*		*Acc*	*p*
SVM	BoW		0.780	—
	BoW	ILIAD	0.746	**0.001**
	BoW	WANAS	**0.790**	0.202
	BoW	$\text{ILIAD}_{\text{AGG}}$	0.768	0.136
	BoW	$\text{WANAS}_{\text{AGG}}$	**0.797**	**0.041**
	BoW	POSTAFTER	**0.780**	0.978
	BoW	THREADPARTICIPATION	**0.790**	0.243
	BoW	COMMONAUTHORS	**0.786**	0.492
SkewAM	BoW		0.812	—
	BoW	ILIAD	0.799	**0.041**
	BoW	WANAS	**0.827**	**0.005**
	BoW	$\text{ILIAD}_{\text{AGG}}$	0.805	0.236
	BoW	$\text{WANAS}_{\text{AGG}}$	**0.830**	**0.001**
	BoW	POSTAFTER	**0.825**	**0.019**
	BoW	THREADPARTICIPATION	**0.827**	**0.008**
	BoW	COMMONAUTHORS	**0.829**	**0.005**
Maxent	BoW		0.820	—
	BoW	ILIAD	0.624	**0.000**
	BoW	WANAS	**0.843**	**0.025**
	BoW	$\text{ILIAD}_{\text{AGG}}$	0.564	**0.000**
	BoW	$\text{WANAS}_{\text{AGG}}$	**0.849**	**0.002**
	BoW	POSTAFTER	**0.834**	0.127
	BoW	THREADPARTICIPATION	**0.836**	0.088
	BoW	COMMONAUTHORS	**0.840**	**0.043**

Table 6: Accuracy for each learner when combining each feature set with BoW (results better than the BoW benchmark for each learner are highlighted in **bold**; p is the probability that a result differs from the benchmark only due to chance, and p-values significant at the 5% level are highlighted in **bold**)

feature set can result in a statistically significant performance increase for both SVM and SkewAM. For SVM, the feature set in question is BoW, whereas for SkewAM, removing THREADPARTICIPATION or COMMONAUTHORS leads to a statistically significant increase in results. Maxent is the only learner where there is no significant increase resulting from removing a single feature set.

Finally, we proceed to test other combinations of feature sets. We exhaustively tested all possible combinations of two and three feature sets, as well as all feature sets, all feature sets minus one, and all feature sets minus ILIAD and $\text{ILIAD}_{\text{AGG}}$. The best combination that we found was using BoW, WANAS and COMMONAUTHORS, with Maxent as the learner. This produced an accuracy of 0.854. However, the top 10 combinations of features and learners all produced very similar results, so we cannot conclude that this is the undisputed best combination overall. We also found that the best combination of feature sets for SVM was different from that for Maxent, but was still extremely close to the best result. The top 10 combinations that we found over the classifiers considered are reported in Table 8.

6 Discussion

As noted in Section 5.1, our dataset is based on data originally used in [17]. Our task is most similar to the ALL task of [17], in that we do not divide the data on the basis of the Nabble sub-forum it originates from. We have also filtered the data slightly differently. The

Learner	*Feature Set*	*Acc*	*p*
SVM	ALL	0.775	—
	−BoW	**0.796**	**0.005**
	−ILIAD	0.775	1.000
	−WANAS	0.775	0.949
	$-\text{ILIAD}_{\text{AGG}}$	0.770	0.508
	$-\text{WANAS}_{\text{AGG}}$	**0.776**	0.897
	−POSTAFTER	0.775	0.949
	−THREADPARTICIPATION	**0.778**	0.731
	−COMMONAUTHORS	**0.777**	0.834
SkewAM	ALL	0.776	—
	−BoW	0.689	**0.000**
	−ILIAD	**0.778**	0.750
	−WANAS	0.769	0.248
	$-\text{ILIAD}_{\text{AGG}}$	**0.788**	0.099
	$-\text{WANAS}_{\text{AGG}}$	0.764	**0.024**
	−POSTAFTER	**0.785**	0.140
	−THREADPARTICIPATION	**0.811**	**0.000**
	−COMMONAUTHORS	**0.812**	**0.000**
Maxent	ALL	0.741	—
	−BoW	0.687	**0.003**
	−ILIAD	0.648	**0.000**
	−WANAS	0.730	0.503
	$-\text{ILIAD}_{\text{AGG}}$	0.697	0.082
	$-\text{WANAS}_{\text{AGG}}$	0.714	0.129
	−POSTAFTER	0.741	0.975
	−THREADPARTICIPATION	0.737	0.768
	−COMMONAUTHORS	0.738	0.825

Table 7: Accuracy for feature ablation over the full feature set for each learner (results better than the BoW benchmark for each learner are highlighted in **bold**; p is the probability that a result differs from the benchmark only due to chance, and p-values significant at the 5% level are highlighted in **bold**)

original authors made use of 3418 posts, whereas we use 4094 posts. The bulk of the difference is due to the original authors eliminating posts which they determined to be non-English. We did not do this because some of our methods do not make use of any language-specific information, so we were still able utilize the non-English data. According to [17], there are 668 non-English posts.

The remaining difference results from the original authors opting to eliminate any posts with 'contradictory ratings', in that the post received ratings both > 3 and ≤ 3, whereas we only eliminated posts where the average rating was $= 3.0$. In practice, the difference is negligible as it only accounts for 8 out of 4291 posts.

The original authors report a maximum accuracy of 0.775 over their ALL task. Their values are not directly comparable to ours because the two tasks are not identical, as we have described above. However, they are very similar, so our best accuracy of 0.854 suggests that our technique would yield an improvement if applied directly to the original task.

We found that, even in isolation, user-level features can outperform a benchmark based on the conventional IR bag-of-words approach, to a high level of statistical significance. This is important because it justifies the use of user-level features for post-level classification tasks. Furthermore, we showed that most of the user level feature sets can be added to the basic bag-of-words model to improve its performance, and that

Learner	*Feature Sets*								*Acc*	*p*
	BOW	ILIAD	WANAS	$\text{ILIAD}_{\text{AGG}}$	$\text{WANAS}_{\text{AGG}}$	POSTAFTER	THREADPARTICIPATION	COMMONAUTHORS		
Maxent	✓		✓					✓	0.854	**0.002**
Maxent	✓		✓		✓	✓	✓	✓	0.850	**0.002**
Maxent	✓				✓	✓			0.850	**0.002**
Maxent	✓				✓				0.849	**0.002**
SVM			✓	✓	✓				0.848	**0.006**
SVM				✓	✓				0.847	**0.004**
Maxent	✓				✓		✓		0.847	**0.006**
Maxent	✓				✓			✓	0.845	**0.012**
Maxent	✓		✓						0.843	**0.025**
Maxent	✓					✓		✓	0.842	**0.027**

Table 8: Top 10 Results over different combinations of learner and feature sets (p is the probability that the result differs from the Maxent BOW benchmark only due to chance; p-values significant at the 5% level are highlighted in **bold**)

this behaviour is consistent across a range of different learners.

Table 8 suggests that the ILIAD feature set is generally ineffective, which may explain the poor results reported in [2]. However, the SVM learner is able to make effective use of the user-level aggregates of ILIAD, $\text{ILIAD}_{\text{AGG}}$, whereas the Maxent learner is not. This is reflected in both the single-featureset experiment reported in Table 5, as well as the overall results in Table 8. The reason for this is not immediately obvious, and further investigation may yield insight into how to reconcile the two.

Another obvious difference between the results from the Maxent and SVM learners is that Maxent performs best in the presence of the BOW features, whereas SVM performs better without the BOW features. This trend is clearly visible in Table 7, where for SVM, removing the BOW features leads to a statistically significant increase in the results, whereas for Maxent and SkewAM, it causes a significant drop. This trend is also visible in Table 8, where we see that the top results for Maxent include BOW, whereas the top results for SVM exclude it. Again the reason for this is not immediately clear. What is clear is that each learner is effective over different sets of features, so there may be scope for further work in terms of applying meta-classification techniques such as stacking in order to further improve results.

It is important to consider the implications of using user-level features for performing a classification task over the 'quality' of a post. The fact that user-level features in isolation can perform better than the baseline is a strong case for the argument that users are consistently good or consistently bad, indicating that the quality of a user's previous posts is a good predictor for the quality of future posts. However, we also expect that the quality of an individual post can vary; it therefore makes sense that the best results we have obtained use a mixture of features, some reflecting purely the content of the post, and some reflecting the overall posting trends of the user.

7 Further Work

Previous studies have either used only a single classification method [5, 16, 17, 18], or have not found significant differences between the relative performance of learners with respect to a given feature set [2]. However, we have seen that over the data being examined in this paper, learners respond better to particular feature sets. We intend to investigate this further by applying the technique to a wider variety of tasks over a greater number of datasets.

We have also adopted a relatively simplistic approach to aggregating post-level features at a user level, simply computing the arithmetic mean of the feature values. Further work would involve taking more information into account, for example the variance and skew in each post-level feature when examined at a user-aggregate level. Another dimension to be taken into account is that a user's knowledge and attitude evolve over time, so we may need to introduce some kind of temporal weighting to the post-level features we aggregate to produce the user-level profile.

Finally, it is also important to note that the gold-standard labels are provided by anonymous internet users, and that each post often has only a single annotation. It is therefore difficult to establish exactly how well the annotation reflects the opinion of the entire community with respect to the post annotated. Further work in this respect would involve establishing datasets where there are a number of annotations for each post, so as to be able to judge inter-annotator agreement and have a feeling for the upper bound in terms of possible classifier performance on the task.

8 Conclusion

In this paper, we have shown that user-level features can improve performance over classification tasks involving posts. We started by defining threaded discourse as an umbrella term for online discussions, and deriving several sets of features for describing users based on techniques for describing other aspects of the threaded discourse.

We evaluated our features over a dataset that has been used in previous research, defining a task similar to that previously investigated. We established a majority-class baseline for the task, as well as a benchmark result based on a conventional bag-of-words model for each post. We investigated our feature sets in isolation, as well as their interactions, across three different off-the-shelf learners. We found that in general, user-level features performed significantly better than simple BoW features on the given task, and that different learners seemed to prefer a different combination of feature sets.

We succeeded in our primary goal of showing that user-level features are effective in classifying posts according to quality, and we expect that the use of these features will generalize well to tasks over other aspects of threaded discourse, for example in profiling users or in classifying threads.

References

[1] Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant and Yirong Xu. Mining newsgroups using networks arising from social behavior. In *Proceedings of the Twelfth International World Wide Web Conference (WWW'03)*, pages 529–535, Budapest, Hungary, 2003.

[2] Timothy Baldwin, David Martinez and Richard Baron Penman. Automatic thread classification for linux user forum information access. In *Proceedings of the Twelfth Australasian Document Computing Symposium (ADCS 2007)*, pages 72–9, Melbourne, Australia, 2007.

[3] Adam L. Berger, Stephen A. Della Pietra and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, Volume 22, Number 1, pages 39–71, 1996.

[4] Sergei Brin and Larry Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, Volume 30, Number 1-7, pages 107–117, 1998.

[5] Blaz Fortuna, Eduarda Mendes Rodrigues and Natasa Milic-Frayling. Improving the classification of newsgroup messages through social network analysis. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM '07)*, pages 877–880, Lisboa, Portugal, 2007.

[6] Chih-Wei Hsu and Chih-Jen Lin. BSVM-2.06. `http://www.csie.ntu.edu.tw/cjlin/bsvm/`, 2006. Retrieved on 15/09/2009.

[7] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94)*, pages 282–291, Dublin, Ireland, 1994.

[8] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, 1998.

[9] Jihie Kim, Grace Chern, Donghui Feng, Erin Shaw and Eduard Hovy. Mining and assessing discussions on the web through speech act analysis. In *Proceedings of the ISWC'06 Workshop on Web Content Mining with Human Language Technologies*, Athens, USA, 2006.

[10] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1137–1145, Montréal, Canada, 1995.

[11] Lillian Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, College Park, USA, 1999.

[12] Andrew Kwok-Fai Lui, Siu Cheung Li and Sheung On Choy. An evaluation of automatic text categorization in online discussion analysis. In *Proceedings of the Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*, pages 205–209, Niigata, Japan, 2007.

[13] Marco Lui and Timothy Baldwin. hydrat. `http://hydrat.googlecode.com`, 2009. Retrieved on 15/09/2009.

[14] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, Taipei, Taiwan, 2002.

[15] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[16] Nayer Wanas, Motaz El-Saban, Heba Ashour and Waleed Ammar. Automatic scoring of online discussion posts. In *Proceedings of the 2nd ACM Workshop on Information Credibility on the web (WICOW '08)*, Napa Valley, USA, 2008.

[17] Markus Weimer and Iryna Gurevych. Predicting the perceived quality of web forum posts. In *Proceedings of the 2007 Conference on Recent Advances in Natural Language Processing (RANLP-07)*, Borovets, Bulgaria, 2007.

[18] Markus Weimer, Iryna Gurevych and Max Mühlhäuser. Automatically assessing the post quality in online discussions on software. In *Proceedings of the 45th Annual Meeting of the ACL: Interactive Poster and Demonstration Sessions*, pages 125–128, Prague, Czech Republic, 2007.

[19] Le Zhang. Maximum entropy toolkit. `http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html`, 2004. Retrieved on 15/09/2009.

Investigating the use of Association Rules in Improving Recommender Systems

Gavin Shaw

School of Information Technology
Queensland University of Technology
Brisbane QLD Australia

g4.shaw@student.qut.edu.au

Yue Xu

School of Information Technology
Queensland University of Technology
Brisbane QLD Australia

yue.xu@qut.edu.au

Shlomo Geva

School of Information Technology
Queensland University of Technology
Brisbane QLD Australia

s.geva@qut.edu.au

Abstract *Recommender systems are widely used online to help users find other products, items etc that they may be interested in based on what is known about that user in their profile.*

Often however user profiles may be short on information and thus when there is not sufficient knowledge on a user it is difficult for a recommender system to make quality recommendations. This problem is often referred to as the cold-start problem.

Here we investigate whether association rules can be used as a source of information to expand a user profile and thus avoid this problem, leading to improved recommendations to users. Our pilot study shows that indeed it is possible to use association rules to improve the performance of a recommender system. This we believe can lead to further work in utilising appropriate association rules to lessen the impact of the cold-start problem.

Keywords Information Retrieval, Personalised Documents, Recommender Systems, Association Rules.

1 Introduction

Recommender systems are designed to understand a users interests, learn from them and recommend items (whether they be products, books, movies etc) that will be of interest to the user. This requires them to personalise their recommendations. Recommendation systems usually work most effectively when user profiles are extensive and/or the applicable dataset has a high information density. When the dataset is sparse or user profiles are short, then recommender systems struggle to provide quality recommendations. This is often known as the cold-start problem.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

Our proposal here focuses on one aspect of the cold-start problem; short user profiles. When a user (can be a new user) has very few ratings in their profile, recommender systems may fail to provide recommendations that interest the user. Both collaborative based [5] and content based [2] can suffer, due to collaborative systems failing to find similar users and content systems having problems due to lack of being able to obtain the content interests of the user.

We propose expanding a user profile (eg. so it contains more ratings) through the use of association rules derived from the dataset. By doing so we expand profiles based on patterns and associations of items, topics, categories etc (which should give relevant consequents) and thus give more information to a recommender system. This would reduce the effect of the cold-start problem and result in better quality recommendations earlier on.

2 Related Work

Much work has been done in the area of recommender systems. Work focusing on solving the cold-start problem includes collaborative & content hybrids [1] [5], ontology based systems [3] and taxonomy driven recommender systems [6] [7]. However, all of these proposals have drawbacks. The hybrid systems can lack novelty, resulting in recommendations that are excessively content centric [7]. The onotolgy based system requires a well defined ontology to be created, something that can be difficult and would limit the system to what is defined/covered within the ontology. Taxonomy based systems work better, but still have low performance. Also the HTR system proposed in [6] performs only marginally better than the TPR system proposed in [7], although it is more time efficient. The taxonomy based approach in [7] does have the advantage of being able to be applied to many domains.

Work has also focused on the other cold-start problem, when a new item is introduced and recommendations are required, but no one has yet rated that item [5]. For this cold-start problem collaborative systems can not help, but content based systems can [5]. We focus on the cold-start problem of new users, rather than new items.

3 Proposed Approach - Using Association Rules to Expand User Profiles

Here we outline our proposed approach and investigation into solving the cold-start problem in recommender systems.

3.1 Background

In its simplest form, a recommender system takes what it knows about a particular user (perhaps a profile) and attempts to predict what that given user would be interested in and recommend those items to the user. Information can take the form of what a user has rated, what other users with similar tastes have rated, the content of the rating (if any) and the content of the item(s). In the case of TPR [7] it uses a given user's rating history of items and the user's implicit taxonomic preferences to determine new items that will interest the user. Both a user's history and implicit taxonomic preferences can be easily obtained from the dataset without any extra effort on the users behalf.

The drawback to this is when a user has a small number of ratings (what we refer to as a 'short profile') it becomes difficult to effectively determine the users implicit taxonomic preferences. Thus recommendation quality suffers.

3.2 Expanding User Profiles

sually at the heart of every recommender system are the user profiles. It is from this that recommenders base their work. In order to improve the quality of recommendations being made to users with short profiles, we propose to use association rules to expand the number of entries in their profiles.

We have a set of users $U = \{u_1, u_2, ..., u_n\}$ and a set of items $I = \{i_1, i_2, ..., i_m\}$. Every user $u \in U$ has a set of rated items $R(u) \subseteq I$ whereby if an item *i* is in the set *R* then user *u* has rated it. We also have a taxonomy *T* containing topics (or categories) *t* in a multi-level structure, where each topic has one parent or supertopic, but may have many children or subtopics. Thus the taxonomy can be visualised as a tree. Each path from the root to a leaf is called a descriptor which is an ordered list consisting of the topics on the path. For any item, it may have more than one descriptor. Let $D = \{d_1, d_2, ..., d_o\}$ be the set of all descriptors, for an item $i \in I$, its descriptors can be represented as $\{d_1(i), d_2(i), ...\}$ which is a subset of *D*. All of this information can be used to expand existing short profiles.

Firstly, using the set of users *U* and the taxonomy *T* we can build a transactional dataset where each user *u* is a 'transaction' and all the topics *t* in the taxonomy make up the datasets attributes. Then we populate the dataset using the set of users *U*, the set of rated items *R* and the set of taxonomic descriptors *D*. This is done by determining the items *i* rated by the user *u* and their positions within the taxonomy. Each item will correspond to one or more paths through the taxonomy from the root to a leaf. That is, the item may have one or more descriptors. For a user u_x and $a \in R(U_x)$, using the descriptors $\{d_1(a), d_2(a), ...\}$, we place a positive value '1' in the user's transaction at each topic involved in these descriptors. All other topics in the transaction will be marked with a negative value '0'. From this we can construct a transactional dataset that shows users' interests in topics, not items.

Second, we then mine the transactional dataset for frequent patterns and derive association rules from these patterns. The frequent patterns and rules will not come from just one taxonomy level, but rather mutiple levels and will also include cross-level patterns and rules. This will give us association rules between topics that interest users. These rules allow us to discover topics that frequently appear together as part of a user's interest. This rule set will then be used to expand user profiles to solve the cold-start problem.

Next, we create the user profiles that will be needed by the recommender system. For our investigation we use the TPR system first proposed in [7]. In order to achieve this we will use the set of users *U*, the set of rated items *R*, the taxonomy *T* and the set of descriptors *D* to create a set of user profiles $P = \{p_1, p_2, ..., p_n\}$. Here for each user $u \in U$ we determine the leaf topics that correspond to each item $i \in R(u_x)$ through the use of the descriptor(s) *d(i)*. These leaf topics are then added to the profile $p(u_x)$ so that $p(u_x)$ contains a list of leaf topics for which user u_x has rated at least one item *i* in each leaf topic *t*. The set of user profiles *P* is known as the user taxonomy profiles and is used by the TPR approach to perform recommendations. This set of profiles *P* will serve as our baseline.

Finally, we expand the user profiles. For this we take the set of user profile *P* and the association rule set we derived in the second step. For each user profile $p(u_x)$ we extract all of the topics *t* within and generate a list of all the combinations possible from the group of topics. Each combination represents a possible antecedent of an association rule. We take each combination and search the set of association rules for any rules that have that exact set of topics as its antecedent. If such a rule exists we can then take its consequent and the topics within and add them to the profile $p(u_x)$. Thus this generates a set of expanded user profiles which we show in our experiments have the potential to improve recommendations over profiles that have not been expanded.

3.3 Imposing Restrictions on User Profile Expansion

We have outlined our proposal for using association rules to expand user profiles in order to improve recommender system quality. However, it is possible that we may want to place limitations on the expansion of user profiles.

1. Restrict the expansion to short profiles. The idea behind this proposal is to expand users who have very few ratings and thus suffer from the cold-start problem. Users with many ratings do not have this problem. Thus a restriction should be imposed on how many topics can be in the user profile *p* before there is too many to warrent expansion. This limit would be dependent on several factors.

2. Restrict the number of rules used when expanding a user profile. It is entirely possible that when deriving the association rules from the transactional dataset that a large list may be generated. It is also possible that from this, when expanding a user profile that a large number of rules and their consequents will be considered for inclusion in the expanded user profile. This may lead to poorer performance as many more topics are added and more items from a wider selection become recommended. Therefore it may be benefical to limit the number of association rules that are used in expansion to those that have the highest support, confidence or other appropriate interestingness measure.

4 Experiments and Evaluation

Here we outline the pilot experiment we undertook to study the value of our proposal to use association rules in expanding user profiles to improve recommendation quality.

4.1 Evaluation Metrics

In order to evaluate the performance of the baseline set of profiles and the expanded set of profiles we follow the same approach detailed in [6]. The past ratings of each user $u \in U$ is divided into a training component and a test component. For the experiments, the recommender system will recommend a list of *n* items for user u_i based on the training set. The recommendation list will be evaluated against the test set. For our experiments we use exactly the same training and test sets as used in [6].

In our work we use precision, recall and F1-measure to determine the overall performance of the recommender system. This allows us to compare the standard approach against our proposal of using association rules for user profile expansion.

4.2 Dataset

For this investigation we use the BookCrossing dataset (obtained from http://www.informatik.uni-freiburg.de/ cziegler/BX/) which contains users, books and the ratings given to those books by the user. The taxonomy tree and descriptors are orginally sourced from Amazon.com and are exactly the same as those used in [6]. From this we build a transactional dataset that contains 92,005 users (transactions) and 12,147 topics from the taxonomy. The dataset is populated using the descriptors that belong to 270,868 unique books. This dataset is then mined to derive the association rules from it.

From the BookCrossing dataset we also build the base set of user profiles *P*. This set of profiles contains 85,415 distinct users with a total of 10,662 leaf topics contained in the taxonomy. As already mentioned the ratings for each user are divided into a training set and a test set. The set of user profiles *P* is based on the training set. The average number of leaf topics in a user profile is 27.08 and the highest number of leaf topics in a given user profile is 3,173 leaf topics. This set of user profiles will serve as the baseline in our experiments and is also the set that will be expanded using the derived association rules.

4.3 Experiment Results

To validate our proposal we conducted a series of experiments to see whether using association rules to expand user profiles improves recommendation quality. From the transactional dataset we set the minimum confidence threshold to 50% and are able to derive 37,827 association rules using the MinMax rule mining algorithm [4]. We then go through the user profiles in the training set and for any profile $p \in P(train)$ that has 5 or less topics listed we attempt to expand using the association rules. This yields a total of 15,912 user profiles which we consider to be short profiles. After attempting profile expansion we then make up to 10 recommendations for these 15,912 users and measure the overall performance of the recommender system. We compare our proposed approach (involving the expanded profiles) against the baseline of the same 15,912 user profiles with no expansion. All experiments use the TPR recommender system first presented in [7].

As shown in Table 1 the baseline set of user profiles (which there is no profile expansion) scores only 0.00619, 0.0571 and 0.0112 for precision, recall and F1-measure respectively. When using expanded profiles we manage to achieve up to 0.00815, 0.0754 and 0.0147 for precision, recall and F1-measure. This is an improvement of approximately 31.5% over the baseline. This level of improvement was achieved when we used the top 5 rules (ranked by their confidence score) to expand user profiles. Table 1 also shows that the performance of our proposed approach improves as more rules are used in expanding a user's profile. However, the improvement is between using the top 2

Table 1: Experimental results for TPR using the short user profiles.

Approach	Precision	%	Recall	%	F1-Measure	%
Baseline (No Rules)	0.00619		0.0571		0.0112	
Expanded (1 Top Rule)	0.00649	4.77%	0.0595	4.28%	0.0117	4.72%
Expanded (2 Top Rules)	0.00714	15.21%	0.0655	14.66%	0.0128	15.16%
Expanded (3 Top Rules)	0.00732	18.15%	0.0672	17.77%	0.0132	18.12%
Expanded (4 Top Rules)	0.00792	27.79%	0.0729	27.75%	0.0143	27.79%
Expanded (5 Top Rules)	0.00815	31.54%	0.0749	31.22%	0.0147	31.51%

or top 3 rules is small. A similar situation also occurs between the top 4 and top 5 rules.

This experiment shows that the overall performance of the recommender system can be improved through the use of our proposed approach, with a 30+% improvement being achieved, which we believe supports our proposal. The efficiency of the recommender is not negatively impacted, as while our expanded profiles take longer to make recommendations for, the time taken is inline with that needed to process a profile with a similar number of topics without profile expansion.

We also conducted a second smaller experiment. We took the 15,912 user profiles that had been deemed to be short and while attempting to expand them, we determined which profiles were actually expanded. From the 15,912 short profiles we were able to expand 11,273 profiles. We then evaluated the recommender system on just these profiles. As Table 2 shows the use of the top 5 association rules to expand these user profiles resulted in an improvement in the performance of the recommender system of 39.7% over the baseline. This experiment shows that for users whose profile can be expanded, noticable improvements in recommendation performance are achievable. Thus it appears that association rules can make a difference in recommender system performance.

Table 2: Experimental results for TPR using only the short user profiles that were sucessfully expanded.

Approach	F1-Measure	%
Baseline (No Rules)	0.0113	
Expanded (5 Top Rules)	0.0158	39.74%

5 Conclusion and Future Work

In this paper we proposed the idea of using association rules to expand user profiles in order to improve recommendations. We outline an approach whereby the rules can be discovered and used, increasing the number of topics in a user profile that only has a few existing ratings. Our experiment shows that the proposed approach can improve the performance of a recommender system under the cold-start problem. This approach allows a user profile to obtain more topic information without extra input from a user and allows a new user to get better recommendations faster.

Further work includes discovering if there is a better measure to rank the rules so that the rules selected are the best for expanding the user's profile. Also more investigation into how many of the top rules to use needs to be undertaken. This would help determine if it is possible to use too many rules during profile expansion such that recommender performance is degraded. Finally, with the issue of redundant rules, this application could be used to help confirm that rules removed as redundant do not cause information loss. This could be done by comparing the performance of a rule set containing redundant rules against one that does not.

Acknowledgements Computational resources and services used in this work were provided by the HPC and Research Support Unit, Queensland University of Technology, Brisbane, Australia.

References

[1] R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modelling and User-Adapted Interaction*, Volume 12, pages 331–370, 2002.

[2] P. Melville, R. J. Mooney and R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. In *18th National Conference on Artificial Intelligence (AAAI'02)*, pages 187–192, Edmonton, Canada, July 2002.

[3] S. E. Middleton, H. Alani, N. R. Shadbolt and D. C. De Roure. Exploiting Synergy Between Ontologies and Recommender Systems. In *The Semantic Web Workshop, World Wide Web Conference (WWW'02)*, pages 41–50, Hawaii, USA, May 2002.

[4] N. Pasquier, R. Taouil, Y. Bastide and G. Stumme. Generating a Condensed Representation for Association Rules. *Journal of Intelligent Information Systems*, Volume 24, pages 29–60, 2005.

[5] A. I. Schein, A. Popescul, L. H. Ungar and M. Pennock. Methods and Metrics for Cold-Start Recommendations. In *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'02)*, pages 253–260, Tampere, Finland, August 2002.

[6] L.-T. Weng, Y. Xu, Y. Li and R. Nayak. Exploiting Item Taxonomy for Solving Cold-start Problem in Recommendation Making. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI'08)*, pages 113–120, Dayton, Ohio, USA, November 2008.

[7] C.-N. Ziegler, G. Lausen and L. Schmidt-Thieme. Taxonomy-driven Computation of Product Recommendations. In *International Conference on Information and Knowledge Management (CIKM'04)*, pages 406–415, Washington D.C., USA, November 2004.

The Methodology of Manual Assessment in the Evaluation of Link Discovery

Wei Che (Darren) Huang
Faculty of Science and Technology
Queensland University of Technology
Brisbane, Australia
w2.huang@student.qut.edu.au

Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand
andrew@cs.otago.ac.nz

Shlomo Geva
Faculty of Science and Technology
Queensland University of Technology
Brisbane, Australia
s.geva@qut.edu.au

Abstract - *The link graph extracted from the Wikipedia has often been used as the ground truth for measuring the performance of automated link discovery systems. Extensive manual assessments experiments at INEX 2008 recently showed that this is unsound and that manual assessment is essential. This paper describes the methodology for link discovery evaluation which was developed for use in the INEX 2009 Link-the-Wiki track. In this approach both manual and automatic assessment sets are generated and runs are evaluated using both. The approach offers a more reliable evaluation of link discovery methods than just automatic assessment. A new evaluation measure for focused link discovery is also introduced.*

Keywords

Wikipedia, Link Quality, Manual Assessment, Evaluation.

1. Introduction

The Wikipedia free encyclopedia is the most popular collaborative information repository on the web. It continues to enjoy increasing popularity amongst web users as well as amongst a diverse set of knowledge content editors [1]. Wikipedia documents are densely linked in the traditional way, from text anchors in one document to a target document. Although external links to other web pages outside the Wikipedia also exist, the link structure within the Wikipedia is quite different from that of the Web. The use of hyperlinks on the Web tends to vary, ranging from elaboration to referential to navigational. Text anchors do not necessarily denote the concept of the target, and even if they do they often take the user to a different but related web site.

The Wikipedia link structure is typically built by matching text anchors to semantically related entries. Most links within the Wikipedia have a strong semantic relationship between the anchor context and the target context. The purpose of a Wikipedia link is almost invariably to provide more detailed information about something. The majority of the links are conceptual rather than navigational.

Proceedings of the 14th Australian Document Computing Symposium, Sydney, Australia, 4 December 2009.

In a growing collection, such as the Wikipedia, the maintenance of the link graph can quickly become more time-consuming and complicated than adding content. Newly created documents should be linked to from text anchors in existing pages. Links to deleted documents must be erased. There is also general maintenance of the link graph for documents that change or are extended.

Several [2, 3, 4, 5] automated *link discovery* algorithms have been proposed as methods to alleviate the link maintenance problem. The INEX *Link-the-Wiki* Track [6] takes the traditional link discovery problem a step further with *focused link discovery*. The aim is to identify text anchors in a source document and a best entry point (*BEP*) in a target document. In HTML a BEP is a named anchor and an anchor-to-BEP link is specified using *#name* on the end of the target document's URL.

Focused systems are potentially more useful to the user because of the reduced need to navigate (especially in a long document or on a mobile device). Anchor-to-BEP link discovery is also a harder (and more interesting) problem than *anchor to document* discovery because of the focused relationship between the anchor context and the target document BEP context. The current method of link discovery is based on the page name matching or similarity. A broad range of technologies, e.g. natural language processing, data mining, machine learning, information retrieval, information extraction and link discovery, are encouraged to integrate to resolve the issue of linking anchor to best entry points.

After two years of INEX experiments it appeared as though the problem of the file-to-file link discovery was solved. Two fundamentally different approaches (anchor link analysis [3] and page name analysis [2]) could identify high quality links when evaluated against links already in the Wikipedia. Near perfect precision scores at high recall levels were seen.

However, after extensive manual assessment of INEX runs it became clear that the use of the existing link graph lead to biased and optimistic evaluation [7]. It appears as though the near perfect scores are achieved because a substantial proportion of the links in the Wikipedia are in fact generated automatically using similar methods to those used by the link dis-

covery systems being evaluated. Manual assessment appears to be essential for robust evaluation of link quality.

There are other reasons for manually assessing link discovery systems at INEX:

1. There appear to be many links in the Wikipedia that are not useful. Some links are inserted automatically and may not be considered relevant by users of the Wikipedia (for instance, links to *year* documents).
2. The Wikipedia is largely linked from an anchor to a whole document; best entry points are rarely seen. It is, therefore, not possible to use the existing Wikipedia link graph to evaluate *anchor to BEP* link discovery systems.
3. In the Wikipedia it is quite reasonable to expect some anchors to target multiple destinations. There could, for example, be a variety of thematic links, multilingual links, or links which extend the anchor's context with varying degrees of complexity (simple vs. full Wikipedia). The existing link graph does not support the evaluation of systems which support multiple links per anchor discovery.

The need for a robust and standardized manual assessment and evaluation methodology is the motivation for this paper. We hope that this methodology will be adopted for link discovery experiments beyond INEX 2009.

2. Wikipedia

There are more than 200 different language versions of the Wikipedia (September 2009). They are freely available as a database and are particularly well suited to IR experiments.

Between 2006 and 2008 INEX used a dump of the English Wikipedia consisting of 659,388 documents. For 2009 INEX has used a fresh dump consisting of 2,666,192 documents. The documents were converted from the original Wiki-markup to XML.

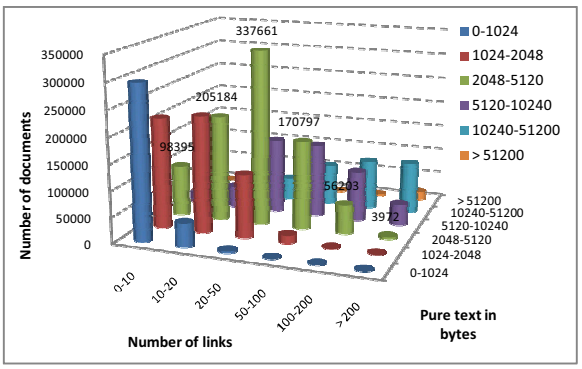


Figure 1: Relationship between document length and number of link in the INEX 2009 Wikipedia collection.

Presented in Figure 1 is the relationship between document size and the number of outgoing links. There are no very short documents with a high link count and there are very few long documents with few links. Most documents link to between 10 and 100 different pages within the collection.

There are 24,168 homonym disambiguation pages. These pages are not suitable for link discovery experiments as they are (essentially) content-free.

3. Related Work

The 2008 INEX Wikipedia collection [8] was converted from Wiki-markup into XML for XML-IR experiments. The 2009 INEX collection was also converted into XML but for use in a broader set of experiments. One point of difference between the two collections is the semantic annotations present in the 2009 collection (see Schenkel et al. [9]).

The Wikipedia has already been used as an IR corpus in several evaluation initiatives. Since INEX 2006 it has been used for the evaluation of ad hoc XML retrieval and for XML Document Mining. At CLEF 2006, it was used for question answering [10].

A link suggestion tool, *Can We Link It*, was developed by Jenkins [11]. It extracts a number of anchors which have not been discovered in an article and that might be linked to other Wikipedia documents**.** Using this tool the user can add new anchors and corresponding links back from a Wikipedia article. Mihalcea & Csomai present the *Wikify* [4] system. It integrates automatic keyword extraction and word sense disambiguation to identify the important concepts in a document and links these to corresponding documents in the Wikipedia.

Link discovery systems are typically evaluated against the Wikipedia itself. Pages are selected as IR topics, the algorithms are run over the topics, and the result compared to the links that are already in the document. Mihalcea & Csomai [4] used the Turing test to further validate their results. Milne & Witten [5] used the Mechanical Turk to solicit links for the AQUAINT collection. INEX considers link discovery to be a recommender task and so the results list is ranked; set based evaluation is inappropriate.

Two evaluation frameworks, DIRECT [12] at CLEF and EPAN [13] at NTCIR, provide a GUI and modules for evaluation. INEX assesses all topics, and also uses a GUI evaluation tool for ad hoc retrieval.

4. Experimental Methodology

A subset of the Wikipedia collection is chosen as a topic set. All anchor links to and from the topics, from and to the collection, are removed (orphaning the documents). Specifically, a random set of (6600 in 2008, 5000 in 2009) documents was chosen as the topics for file-to-file linking; track participants no-

minated topics (50 in 2008, 33 in 2009) for anchor-to-BEP linking. The goal is to identify both outgoing and incoming links from and to those topics.

INEX offers two linking tasks: file-to-file and anchor-to-BEP. The former is a low-cost entry-level task for new participants (and as a sanity check for the latter task). The task is to identify up to 250 documents that the topic should link to; no anchor or BEP need be identified.

In the anchor-to-BEP task the system can identify up to 50 outgoing anchor texts per topic. For each anchor at most 5 target document/BEP pairs are allowed. For incoming link discovery, a set of at most 250 anchors (in the collection) targeting BEPs in the topic are to be identified. Both incoming and outgoing links are from anchor to BEP.

A text anchor is identified by its position (offset and length) within the document. A BEP is identified by its position. Positions are specified as character offsets (excluding markup) from the document start.

Participants were invited to submit runs. In total 30 runs were submitted in 2008. It was prior to the 2009 submission deadline at the time of writing.

5. Manual Assessment

5.1 Methodology

In 2008 two sets of assessments were generated, one from the Wikipedia and the other from the runs.

The Wikipedia ground-truth assessment set consisted of just those links already in the Wikipedia. It is an automatically generated set of links from anchors to documents.

Submission runs were *pooled*. The pooling process combines overlapping anchor texts to form a pool-anchor which is presented to the assessor. A pool-anchor might contain a number of anchors as well as a set of target BEP links. All the links already in the Wikipedia topic were added to the pool. The pool was then manually assessed.

For the purpose of evaluation it is assumed that all non-assessed links are non-relevant. However, as the pool was exhaustively assessed, there is a reusability issue and does not affect submitted runs. We note that the same convention is used in other forums and tracks (such as TREC).

A validation tool was provided and distributed to assist developers of focused link discovery systems. It allowed participants to view their submissions in an interface similar to that used by the assessors. The tool helped participants debug their submissions (the calculation of BEP can be non-trivial), as well as perform sanity checks on their algorithms.

5.2 Assessment

Built on experience using the INEX ad hoc assessment tool, a GUI-based relevance assessment tool (i.e. *GPXrai*) was custom designed and built for the manual assessment of link discovery pools (see Figure 2). The interface is comprised of a split screen.

The topic pane is located on the left hand side. The right hand pane is used to show the target document. Two distinct assessment modes are provided, one for outgoing links, the other for incoming links.

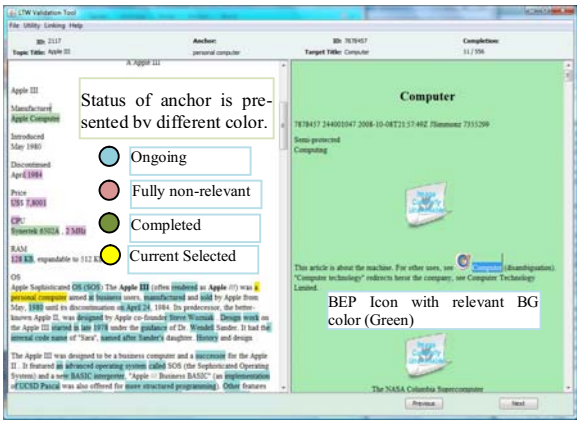


Figure 2: INEX 2009 Link-the-Wiki Assessment Tool

Outgoing links are initially assessed. The topic document is displayed with highlighted pool anchors. In the first instance the assessor goes through all the anchors and rejects (a mouse right-click) those which are obviously irrelevant. The pool can contain many such anchors, for instance year or other coincidental links. Each link for each remaining anchor is then displayed, in turn, with the right pane showing the target document. The assessor then either rejects (a mouse right-click), or accepts the target as relevant (by double-clicking to indicate the BEP, then mouse left-click).

Incoming links are assessed in a similar manner, but the locations of the anchor and BEP are swapped. Now the anchors are from other documents and the BEPs are inside the topic document. The assessor is required to accept or reject each prospective link.

In INEX 2008 the pools contained between 405 and 1722 links. Assessment logs suggest that between 4 and 6 hours were required to assess a topic. On average, only 7.4% of a pool was judged relevant.

6. Evaluation

A portable (Java) evaluation tool, *LtwEval*, was developed for evaluation purposes. It is GUI based and provides numerous evaluation metrics including: precision, recall, MAP, and precision@R. Different runs can be evaluated and compared to each other. Precision/recall graphs can be generated for sets of runs (see Figure 3). Anchor-to-BEP runs can be eva-

luated as either file-to-file or anchor-to-BEP. The tool was distributed to participants.

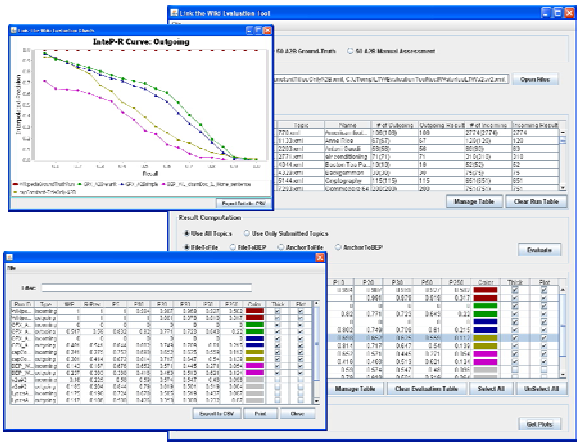


Figure 3: INEX 2008 Link-the-Wiki Evaluation Tool

The "best" metric to use for focused link discovery evaluation is not obvious. As with all metrics, it is important to first define the use-case of the application. The assumption at INEX is that link-discovery is a recommendation tasks. The system produces a ranked list of anchors and for each a set of recommended target/BEP pairs. The user navigates a limited number of anchors and selects only a few to embed in the new document.

A link discovery system might identify a very large number of possible links. The Wikipedia has a page for each letter in the Latin alphabet and so each letter of each word might be linked. It also contains potentially overlapping links, for example there is a page for *world*, a page for *war*, and a page for *world war*. The user is expecting the system to identify relevant anchors and links, and to place these at the top of the results list. The list should also be comprehensive because it is not clear that the document author can know a priori which links will be relevant to a reader of the document. That is, link discovery is a recall oriented task

The Mean Average Precision based metrics are very good at taking rank into account and are recall oriented. They are also very well understood. A good metric for link discovery should, consequently, be based on MAP. The difficulty is computing the relevance of a single result in the results list.

For the anchor *The Theory of Relativity*, an equally good anchor might be *Relativity*. For evaluation purposes it is assumed that if the target is relevant and the anchor overlaps a relevant anchor then the anchor is relevant; $f_{anchor}(i) = 1$. This is subtly different from the *world war* problem above, different in so far as the target must also be relevant. Of course, this definition of relevant anchor aids in reusability.

The assessor might have assessed any number of documents as relevant to the given anchor. If the target of the anchor is in the list of relevant document then it is considered relevant; $f_{doc}(i) = 1$. In the INEX ad hoc track the BEP is considered to be subjective. If the search engine can put the relevant passage on the user's screen then it is considered a "hit". The contribution of the links' BEP is a function of distance from the assessor's BEP:

$$f_{bep}(j) = \begin{cases} \dfrac{n - 0.9 \times d(x,b)}{n} & if\ 0 \le d(x,b) \le n \\ 0.1 & if\ d(x,b) > n \end{cases}$$

Where $d(x,b)$ is the distance between submission BEP and result BEP in character. Therefore, the score of $f_{bep}(j)$ varies between 0.1 (i.e. d is greater than n) and 1 (i.e. the submission and result BEPs are exactly matched). The score of 0.1 is reserved for the right target document with an indicated BEP not in range of *n*. *n* typically is set up as 1000 (characters). The score of a result in the results is then:

$$P = \left[(f_{anchor}(i)) \times \frac{\left(\sum_{j=1}^{m} \left(f_{doc}^{i}(j) \times f_{bep}^{i}(j)\right)\right)}{m_i} \right]$$

Where m is the number of returned links for the anchor and m_i is the number of relevant links for the anchor in the assessments. As the result list is restricted to 5 targets per anchor m_i is capped at 5 for evaluation. A perfect run can thus score a MAP of 1.

7. Conclusion and Outlook

Although it has appeared as though link discovery is a solved problem, manual assessment of participants runs at INEX 2008 showed that, in fact, it is not. The INEX result raises new questions about methodologies for link discovery evaluation, and in particular focused link discovery systems.

In this contribution we propose and describe a new comprehensive methodology. This methodology is based on manual assessment of link relevance. A new metric is proposed to measure the performance of a run. Our methodology is being used for the INEX 2009 Link-the-Wiki track.

Our further work will focus on evaluation quality and on the efficiency of the manual assessment. This will be done using assessor surveys and interviews.

We remain fascinated by the appalling performance of the Wikipedia itself when evaluated against the manual assessments. It is our expectation that, once the methodology is stable, link discovery systems will outperform human created hypertext links.

References

[1] Alexa, The Web Information Company *http://www.alexa.com/topsites.*

[2] Geva, S., *GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia*, INEX 2007, pp. 404-416.

[3] Jenkinson, D., K.-C. Leung, A. Trotman, *Wikisearching and Wikilinking*, INEX 2008, pp. 374-388.

[4] Mihalcea, R., A. Csomai, *Wikify!: linking documents to encyclopedic knowledge*, CIKM 2007, pp. 233-242.
[5] Milne, D., I.H. Witten, *Learning to link with wikipedia*, CIKM 2008, pp. 509-518.
[6] INEX (2009) http://www.inex.otago.ac.nz/tracks/wiki-link/wiki-link.asp.
[7] W.C. Huang, Trotman, A., Geva, S (2009), *The Importance of Manual Assessment in Link Discovery*, SIGIR 2009, pp. 698-699.
[8] Denoyer, L., Gallinari, P. (2006) The Wikipedia XML Corpus, *ACM SIGIR Forum*, 40(1):64-69.
[9] Schenkel, R., Suchanek, F. M., Kasneci, G. (2007) YAWN: *A Semantically Annotated Wikipedia XML Corpus*, BTW 2007, pp. 277-291.
[10] WiQA: Question answering using Wikipedia (2006) http://ilps.science.uva.nl/WiQA/index. html
[11] Jenkins, N., *Can We Link It*, http://en.wikipedia.org/wiki/User:Nickj/Can_We_ Link_It.
[12] Mitanura, T., Nyberg, E. Shima, H., Kato, T., Mori, T., Lin, C.Y., Song, R., Lin, C. J., Sakai, T., Ji, D., Kando, N., *Overview of the NTCIR-7 ACLIA Taska: Advanced Cross-Language Information Access* NTCIR-7, pp. 11-25.
[13] Di Nunzio, G. M. and Ferro, N., *DIRECT: A System for Evaluating Information Access Components of Digital Libraries*, ECDL 2005, pp. 483-484.

Web Indexing on a Diet: Template Removal with the Sandwich Algorithm

Tom Rowlands, *Paul Thomas*, and *Stephen Wan*
CSIRO ICT Centre
tom.rowlands@csiro.au, *paul.thomas@csiro.au*, *stephen.wan@csiro.au*

Abstract *Web pages contain both unique text, which we should include in indexes, and template text such as navigation strips and copyright notices which we may want to discard. While algorithms exist for removing template text, most rely on first completing a crawl and then parsing each page. We present a cheap and efficient algorithm which does not parse HTML and which requires only a single pass of the document. We have used two web corpora to investigate the performance of a retrieval system using our algorithm and have found similar eectiveness with an index 9-54% smaller. Further experiments using a marked-up corpus have shown 97% of desired lines are returned.*

Keywords Web documents, information retrieval

1 Introduction

Retrieving information from within a web document is made more difficult by the presence of template text. Such templates include, for example, the header and footer information that sandwiches the real content of the document. These are typically inserted automatically by HTML authoring tools and scripts that dynamically generate HTML pages, in order to provide a website with a consistent look-and-feel. Ideally, an information retrieval system would be able to discard such template material when it doesn't contribute to the topic of a page.

In this paper, we treat template detection and removal as a longest common subsequence (LCS) problem, giving an efficient solution. Our experiments with the WT10g corpus and an enterprise data set demonstrate gains in efficiency with low complexity.

2 Related work

Related work has been characterised as using either a *local* or a *global* approach [3]. A local approach examines a page in isolation to find the template material. In contrast, a global approach determines shared templates by examining two or more documents from a collection.

Most approaches handle templates with a two-pass algorithm: the first pass identifies the template and the second extracts it. Approaches to identifying the template have included structural comparisons, often using the document object model of the HTML document. Tree comparison methods have been used to examine similarities in HTML tag elements [8]. Similarly, Wang et al. [9] look for tables specifying layout. Visual blocks have been segmented using classification approaches [7].

In contrast to examining document structure, other approaches simply examine page text and are thus cheaper to run. Word-level features such as term frequency and word position statistics have been exploited to induce templates [2]. A similar approach using text fragment frequencies is explored by Gibson et al. [3]. Our work is similarly non-structural but does not require any statistical modelling.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

3 The sandwich algorithm

We investigate template detection and removal from the viewpoint of improving the efficiency of a web search engine. As such, we start with the constraint that the solution must be able to operate as documents are crawled.

The algorithm is derived from the intuition that, given the prevalence of HTML authoring tools and website content management systems, documents in the same directory will likely share the same template. The template lines are detected by comparing the target file—line by line—with a sibling document in the directory, referred to here as a *peer*. The longest common subsequence (LCS) of lines is a non-contiguous set of lines in common to a document and its peer. Our approach assumes all such lines are from a template and can be discarded. The remaining lines are considered indexable material and kept. If there are no other pages in the directory, and therefore no candidate peers, no template removal is attempted.

Our approach is global but reduces to a single pass. That is, identification of the template is performed per document, and template material is removed at the same time. As a result, this approach can be implemented in a crawler before material is stored. If the crawl is breadth-first, in most cases an appropriate peer will simply be the last page crawled.

Different algorithms produce the LCS in $O(n^2)$ to $O(n \log n)$ time [5, 6], where n is the number of lines in each document. No HTML parsing is required; the algorithm is entirely independent of

the markup language.[1] The algorithm can remove template text from "split" content, where template material is injected in between portions of useful text. Implementation is straightforward and simpler than competing approaches, which makes template removal an option where engineering resources are limited.

4 Experiments

Our early experiments consider two measures. First, we examine the effectiveness and efficiency of a retrieval system which employs the sandwich algorithm. Second, a corpus with templates explicitly marked allows us to investigate our algorithm's accuracy. (These only provide a quick check of the algorithm's performance; in this first work we have not conducted an in-depth comparison with competing, more complex, techniques.)

To investigate the performance of a retrieval system which incorporates the sandwich algorithm, we used PADRE [4]—which implements a variant of BM25—and two corpora. The WT10g corpus, used by the TREC web track [1], includes about 1.7 million web pages from a variety of hosts. Peers were found for 92% of pages. We used three sets of associated queries ("topics"). Topics 451–500 (from TREC 2000) and 501–550 (from TREC 2001) are reverse engineered from search engine query logs. Topics EP1–145, also from TREC 2001, concentrate on finding home pages. Since by removing navigation blocks we will remove a number of links to each site's entry page, performance on this latter set of queries seems likely to degrade.

The second corpus is in the media domain, and was collected from a large, national media organisation's website. It comprises about 760,000 documents for which peers were found for 98%. 88 queries were used from a sample of the organisation's query log, with judgements by an author who was familiar with the organisation. A subset of this corpus has templates explicitly marked.

In these experiments, which used a pre-existing crawl, a page's peer was based on its name n: it was that page in the same directory whose name was closest to n. "Closest" was defined with respect to edit distance.

The first question we ask is: *how much more efficient can an index be if templates are removed?* To our knowledge, template removal approaches have not been examined by this measure. Table 1 summarises the size of each corpus with and without processing; and the number of postings in an index of each.

Since a lot of templates are formatting or scripting instructions, which will not be indexed anyway, the savings in postings are less than the savings in corpus size—however even the smallest saving, 9% of postings for WT10g, seems worthwhile, and the figures for the media corpus represent a substantial savings. The figures for WT10g are smaller than the 40–50% suggested by Gibson et al., but the WT10g crawl is older than the one used there and the use of templates has been growing since [3]. By insisting on exact string matches we are also conservative in identifying possible templates.

[1] If the input is known to be, e.g., HTML or SGML then a tokeniser could be run first and the LCS computed over streams of tokens. We have not yet pursued this idea.

	As-is	LCS removed
WT10g	10.7 GB	9.0 (−17%)
	1.4×10^9 postings	1.2×10^9 (−9%)
media	12.3 GB	4.0 (−67%)
	1.1×10^9 postings	0.5×10^9 (−54%)

Table 1: Corpus and index sizes for two corpora, before and after processing.

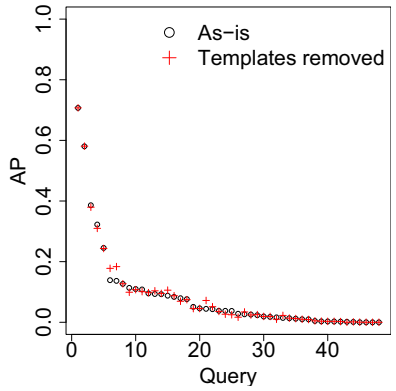


Figure 1: AP scores for queries 451–500, processed with and without templates in the index.

Although a substantial fraction of the index has been removed, retrieval performance is unaffected. Figure 1 illustrates the AP scores for each of topics 451–500: on most queries there is no discernable change and overall there is no significant difference (Wilcoxon $p > 0.99$). Topics 501–550 and EP1–145, and the media set, are similar ($p > 0.2$, $p > 0.5$, and $p > 0.4$ respectively).

A further question is: *how accurate are we in removing templates?* We compared our output, line by line, with a subset of the media corpus explicitly marked by the organisation. Blank lines and contentless HTML (e.g. a sole <p> on a line) were not considered in the comparison. The precision and recall of lines classified as non-template material (and hence kept) is 57% and 97% respectively, with an F1 score of 0.59. The algorithm is correctly keeping the great majority of interesting text, although our conservative approach means we are also keeping a portion of templates.

5 Conclusions and Future Work

Templates represent a substantial, though generally uninformative, portion of text on the web. Removing templates leads to a reduction in index size, without a drop in query performance. Line-based LCS

comparison provides a cheap method for template detection and removal, allowing for easy integration within a web crawler. In future work, we intend to use the sandwich algorithm with question answering systems and automatic text summarisers, both of which can benefit greatly with the accurate removal of irrelevant template material.

Acknowledgements

We thank the anonymous reviewers for their feedback and for their useful ideas.

References

[1] Peter Bailey, Nick Craswell and David Hawking. Engineering a multi-purpose test collection for web retrieval experiments. *Info Proc & Management*, Volume 39, Number 6, pages 853–871, November 2003.

[2] Laing Chen, Shaozhi Ye and Xing Li. Template detection for large scale search engines. In *Proc. ACM Symposium on Applied Computing*, pages 1094–1098, 2006.

[3] David Gibson, Kunal Punera and Andrew Tomkins. The volume and evolution of web page templates. In *Proc. WWW*, pages 830–839, 2005.

[4] David Hawking, Peter Bailey and Nick Craswell. Efficient and flexible search using text and metadata. Technical Report 2000/83, CSIRO Mathematical and Information Sciences, 2000. `http://es.csiro.au/pubs/hawking_tr00b.pdf`.

[5] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, Volume 24, Number 4, pages 664–675, 1977.

[6] James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest common subsequences. *Comm. ACM*, Volume 20, Number 5, pages 350–353, 1977.

[7] Ruihua Song, Haifeng Liu, Ji-Rong Wen and Wei-Ying Ma. Learning block importance models for web pages. In *Proc. WWW*, pages 203–211, 2004.

[8] Karane Vieira, Altigran S da Silva, Nick Pinto, Edleno S de Moura, João M B Cavalcanti and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proc. CIKM*, pages 258–267, 2006.

[9] Yu Wang, Bingxing Fang, Xueqi Cheng, Li Guo and Hongbo Xu. Incremental web page template detection. In *Proc. WWW*, pages 1247–1248, 2008.

Analyzing Web Multimedia Query Reformulation Behavior

Liang-Chun Jack Tseng
Faculty of Science and Technology
Queensland University of Technology
Brisbane, QLD 4001, Australia
ntjack.au@hotmail.com

Dian Tjondronegoro
Faculty of Science and Technology
Queensland University of Technology
Brisbane, QLD 4001, Australia
dian@qut.edu.au

Amanda Spink
Faculty of Science and Technology
Queensland University of Technology
Brisbane, QLD 4001, Australia
ah.spink@qut.edu.au

Abstract *Current multimedia Web search engines still use keywords as the primary means to search. Due to the richness in multimedia contents, general users constantly experience some difficulties in formulating textual queries that are representative enough for their needs. As a result, query reformulation becomes part of an inevitable process in most multimedia searches. Previous Web query formulation studies did not investigate the modification sequences and thus can only report limited findings on the reformulation behavior. In this study, we propose an automatic approach to examine multimedia query reformulation using large-scale transaction logs. The key findings show that search term replacement is the most dominant type of modifications in visual searches but less important in audio searches. Image search users prefer the specified search strategy more than video and audio users. There is also a clear tendency to replace terms with synonyms or associated terms in visual queries. The analysis of the search strategies in different types of multimedia searching provides some insights into user's searching behavior, which can contribute to the design of future query formulation assistance for keyword-based Web multimedia retrieval systems.*

Keywords Web log analysis, multimedia search, query reformulation, search strategy

1. Introduction

The prevalence of multimedia information on the Web has changed user's information need from textual to multi-modal (i.e. audio, image, and video) searching. Multimedia search is more complex compared to general Web searches as evidenced by the longer session times and query lengths [11, 18]. Web multimedia search users also perform many query modifications, and have more difficulties in finding the appropriate terms to represent their needs. In addition, image search has the longest session length (i.e. more queries per session) [19] and more terms per query than video and audio searches [18]. Therefore, it is important to investigate user's multimedia query formulation behavior in order to better understand the characteristics and obstacles in different types of multimedia searches.

Existing studies have attempted to understand user's information searching behavior and the search trends from Web log analysis [7, 15, 19]. These studies have shown that users submit relatively short search queries, typically around three terms per query [15]. Most users do not review many results, typically only the first result page [9, 11]. Such little contextual information and brief interaction between the user and the search engine limited the understanding of user's searching behavior, especially when the analysis is based on individual transaction records [13, 16]. Thus, it is necessary to investigate multiple queries in order to provide more contextual information for Web log analysis.

The current study aims to discover multimedia query reformulation behavior and search strategies by applying novel log analysis procedures. To our knowledge, this is the first study to automatically analyze contextual information beyond two consecutive transaction logs. This approach also allows us to compare the search strategy characteristics among different types of multimedia searches and provide insights for future system development.

2. Related studies

2.1 Limitations of current Web log analysis

Web logs can be effectively used to understand general users' online searching behavior on a large scale [6, 8, 9, 11, 15] and are generally more objective and non-intrusive than other data collection methods [6]. Such unique characteristics make Web log data representative of user's unaltered behavior and thus regarded as the most convenient way to study real users [6]. However, the findings from individual transaction log are usually limited to the descriptive data without explanatory information about user's

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

searching behavior [16]. Recent studies have begun extracting contextual information from consecutive query modifications [5, 13, 14]. However, most studies are limited in the amount of queries that can be investigated because of the need for manual reviewing processes [12, 16, 20]. Other studies using large-scale data only examine searching behavior based on two consecutive query modifications. Thus, the full potential of contextual Web log analysis has yet to be discovered.

The analyses of contextual search information mainly focus on identifying new search sessions based on query modifications between consecutive queries [5, 10, 13]. He, Goker, and Harper [5] compared the effectiveness of using the time interval between two clicks and query modification patterns in detecting new search sessions. While the combination of these two methods produced the best results, query modification patterns accounted for the majority of the improvements. Ozmutlu and Cavdur [13] applied this method to Excite search engine logs. Their findings supported the usefulness of query modification patterns. Query modification pattern and time interval also have a significant effect on judging topic shifts [14]. In the comparison with several Support Vector Machine methods, the use of query modification pattern achieved at least 95% precision and recall in topic continuation, and 35% or more in topic shift cases, far better than its SVM counterparts. Similarly, Lau and Horvitz [12] used query modification pattern and time intervals between two consecutive queries to successfully predict user's upcoming search behavior based on a Bayesian probability model. Query modification has also been used for studying the uptake and effectiveness of terminology feedback provided by retrieval systems [1].

2.2 Web query reformulation behavior and search strategies

The term "modification" and "reformulation" have been used interchangeably in many Web log analysis studies without explicit clarification of the differences [10, 16]. In this study, we use "reformulation" to refer to user's overall behavior of formulating different versions of related queries in a session, whereas "modification" represents each change to the query. Thus query modifications can be classified into certain patterns and the overall query reformulation behavior implies user's search strategies.

Bruza and Dennis [4] investigated user's query reformulation behavior by manually classifying more than one thousand queries into one of the eleven types of query modifications. With the exception of the repeating queries, term substitution was found to be the most dominant type of query modifications, followed by term addition and deletion. A similar finding was also reported from the study on a meta-search engine Dogpile.com [10]. Jansen, Spink, and Narayan [10] investigated query reformulation behavior among large-scale Web log data. Despite the large proportion of formulating new search queries, query reformulation (which is equivalent to substitution in [4]) accounted for more than 15% of all eight types of modifications, with specialization (i.e. addition) occurring more than twice of generalization (i.e. deletion). They also concluded that major search content transitions were between Web and image collections.

Currently, only limited query modification studies have investigated more than two consecutive queries to infer user's search strategies [16] or tactics [2]. Rieh and Xie [16] manually investigated 313 sessions of five modifications or more to classify the overall query reformulation approach into one of the eight distinct strategies, including: *generalized*, *specified*, *dynamic*, *parallel*, *block-building*, *multi-tasking*, *recurrent*, and *format* (details in Section 4.6). Although they did not report the frequency for each strategy, they concluded that the first four (i.e. generalized, specified, dynamic, and parallel) are the most popular strategies. A similar categorization of search strategies can also be found in [2].

3. Research questions

Our focus is user's Web multimedia searching behavior which can be revealed by consecutive query modifications. We investigate the entire session of user's query modifications to infer the searching behavior. The three main questions that we attempt to answer are:

1. What are the frequent modifications in Web multimedia queries and do they differ among the multimedia searches?

2. What can the sequence of query modifications tell us about user's query reformulation behavior?

3. What search strategies can be inferred from query modification sequences? How can they contribute to the improvement of keyword-based Web multimedia retrieval systems?

4. Methodology

4.1 Dogpile log aggregation and query modification records

Dogpile is one of the leading online meta-search engines, which incorporates the indices of top search results from Google, Yahoo!, MSN Live, and Ask.com. For this study, a total of 1,228,310 records taken on May 15th, 2006 have been used in our analysis. The original Dogpile transaction log contains five fields that we use for our analysis:

IP: the IP address of the computer submitting the query.

Cookie: the unique identifier which Dogpile system sends to a particular computer with a pre-defined valid period.

Time: the time of the day when user submits the query.

Query: the original search text submitted to the system.

Vertical: the search type option which user selected on Dogpile's search page. In this study, we separate the logs with "images", "video", and "audio" option selected and replicate the analyses for comparison.

4.2 Browsing record aggregation

The Dogpile transaction logs are sorted based on different user identification (i.e. a unique combination of Internet Protocol (IP) and cookie) in a chronological order. Consecutive transaction logs with identical queries represent browsing records and are aggregated with the foremost record. If the last record has the same time stamp as the first record in current browsing aggregation, the duration will be logged as zero length (e.g. the last aggregation in Table 1 and Table 2). Table 1 and 2 illustrate the original log and aggregated record respectively.

IP	Cookie	Time	Query	Vertical
64.105.73.70	2187RDPA47YLJOB	6:05:33 PM	pod of dolphins	Images
64.105.73.70	2187RDPA47YLJOB	6:06:03 PM	group of dolphins	Images
64.105.73.70	2187RDPA47YLJOB	6:06:18 PM	group of dolphins	Images
64.105.73.70	2187RDPA47YLJOB	6:06:18 PM	group of dolphins	Images
64.105.73.70	2187RDPA47YLJOB	6:08:56 PM	dolphins	Images
64.105.73.70	2187RDPA47YLJOB	6:08:56 PM	dolphins	Images
64.105.73.70	2187RDPA47YLJOB	6:08:56 PM	dolphins	Images

Table 1. Original Dogpile search logs with browsing records

Session No.	Current query	Modified terms	Modification pattern	Duration
1	pod of dolphins		I	0:00:30
1	group of dolphins	pod,group	R	0:02:53
1	bottlenose dolphins	group of,bottlenose	R	0:00:00

Table 2. Query modification table with aggregated modification records

4.3 Modification pattern classification

Each aggregated transaction record is classified into a query modification pattern based on the content of the current query and the previous query. We use four modification patterns for our classification. The definitions for each modification pattern are:

Initial query (I): current query has no terms in common with the previous query

Addition modification (A): current query contains all search terms from the previous query, as well as some new terms

Deletion modification (D): current query omits some terms from the previous query

Replacement modification (R): deletion and addition of terms happen simultaneously to form the current query

Thus an initial query represents a new search topic since no search terms are carried over from previous query. Some studies also classify replacement modification as "reformulation" [5, 13, 14]. However, as users can freely reformulate the query by changing the order of search terms without affecting the search results, we use the term "replacement" to clearly indicate such modification. Details of the classification algorithm can be found in [5]. We built a program to automatically classify queries by their modification patterns and aggregate consecutive browsing records in Table 2.

4.4 Session aggregation

A search session is a series of related queries submitted by same user. In addition to being defined by a unique combination of IP and cookies, a query with no terms in common with its preceding query is regarded as the beginning of a new session, thus classified as the "initial query". By calculating the number of sessions with same IP and cookie combination, we are able to identify the average search topics submitted by a user.

4.5 Modification sequence

Once the query modification records have been generated, consecutive modifications within each session can be classified into several predetermined modification sequences. We used our program to identify the occurrence of thirty-six types of modification sequences, incorporating two or three predetermined modifications. Sessions with less than two modifications are discarded as they provide little information about user's behavior. The two–modification-sequences comprise one initial query (I), followed by two query modifications which can be either of the replacement, addition, or deletion modification. Thus, nine patterns (3*3) can be formulated for the two-modification sequences. Similarly, twenty-seven patterns of three-modification-sequences (3*3*3) can be formulated. The main purpose of this analysis is to discover the frequent patterns of modification sequences that users follow, thus revealing user's preference for consecutive query modifications and providing in-depth information for search strategy analysis.

4.6 Search strategies based on modification sequence analysis

When typical modification sequences emerge from our analysis, we calculate the changes in the number

of query terms within each sequence. Such changes can determine if users adopt some particular search strategies. We construct our strategy classification based on the higher level categorization in [16]. The list of modification strategies used in this study, as well as the detail descriptions of our analysis assumptions are as follows:

Generalized reformulation

A user may begin with several search terms and subsequently drop some of the terms to include more results. This generalized reformulation is often manifested by consecutive term deletion changes [16]. It can also be characterized by replacing the query with fewer terms. Modification sequences in which subsequent queries always have fewer or equal terms to the precedent queries belongs to this category.

Specified reformulation

When a user persistently specifies a query by adding more terms or changing to more specific phrases, we classify this approach as specified reformulation. In our analysis, modification sequences in which a subsequent query always has more or equal terms to its preceding query belongs this category.

Dynamic reformulation

When a user inconsistently switches between generalized and specified reformulation, we characterize such approach as dynamic reformulation. Such modification pattern manifests the unplanned nature of user's search process. Users who adopt this search strategy generally have the most unconsolidated search problems, and require more interaction with the retrieval system. Modification sequences in which subsequent queries can have either fewer or more terms than precedent queries exhibit dynamic search strategy.

Constant reformulation

Constant search occurs when a user modifies terms of the same concept level which shares some common characteristics, for example when substituting with related objects (e.g. from PC to Mac) or synonyms. This strategy is characterized by having a constant number of query terms across the entire modification sequence, regardless of the existence of replacement modifications. The same query specificity suggests a one-to-one relationship between the original and new terms. We used the term "constant reformulation" to reflect this unique characteristic.

5. Results

5.1 Query modification

From Table 3, image searches are the dominant type of multimedia search in our dataset with more than 50% of sessions and users attributed to image searches. Audio is the second popular type of multimedia search whereas video is the least popular. As Table 4 shows, initial queries are the majority of query modification across all multimedia searches. Replacement modification is more than twice of the addition modification in visual searches (i.e. image and video searches) but much less in audio searches. Deletion is the least type of modification in all searches.

Comparing the distribution of the four modifications in multimedia searches, audio search users are more likely to formulate new search topics as they have larger proportion of initial queries and more topics per user than image and video searches. The number of topics submitted by both image and audio users varies a lot (SD=21.60 and 22.39 respectively) while video users shows a much uniformity pattern (SD=8.33). For the number of modifications, image and video users have the same amount of modifications (1.71 modifications on average) while audio users show slightly fewer modifications per session (1.63 on average). Overall, image and video search users are very similar in terms of query modifications.

	Image	%	Video	%	Audio	%	Total
Log records	597,760	48.7	231,941	18.9	398,609	32.5	1,228,310
Sessions	183,825	52.9	52,405	15.1	110,945	32.0	347,175
Users	60,701	52.1	21,677	18.6	34,088	29.3	116,466

Table 3. Statistics of image, video, and audio search logs in Dogpile dataset

	Image	%	Video	%	Audio	%
Initial	183,825	58.6	52,405	58.5	110,945	61.2
Replacemen	82,292	26.2	22,225	24.8	33,645	18.6
Addition	30,716	9.8	8,817	9.8	22,553	12.4
Deletion	16,757	5.3	6,124	6.8	14,176	7.8
Total	313,590	100.0	89,571	100.0	181,319	100.0
Topics per user						
Average	3.03		2.42		3.25	
SD	21.60		8.33		22.39	
Modifications per session						
Average	1.71		1.71		1.63	
SD	1.68		1.68		1.42	

Table 4. Statistics of query modification records

5.2 Modification sequence

Two-modification-sequence analysis

The frequencies of each modification sequence pattern (in percentages) are presented in Table 5 and 6. Table 5 signifies the popularity of replacement modification in all types of multimedia searches, as evidenced by the dominance of *I-R-R* and *I-A-R* sequences. On the contrary, the unlikelihood of consecutive deletion modification is manifested by the low occurrence of *I-D-D* sequences (i.e. less or equal to 1% in all multimedia searches).

Figure 1 shows that both image and video searches have prominently more *I-R-R* sequences than audio searches. The audio searches have much more *I-A-D* sequences than the other two types of searches, thus

making it more evenly distributed in the top three modification sequence patterns. All multimedia searches show a similar distribution beyond the top three patterns.

	Image	Video	Audio
I-R-R	41.2%	39.2%	26.8%
I-A-R	24.9%	22.2%	23.6%
I-A-D	10.9%	13.4%	21.9%
I-R-A	5.6%	5.3%	6.2%
I-R-D	5.4%	6.8%	6.9%
I-D-A	5.1%	6.2%	7.1%
I-A-A	3.9%	3.3%	4.3%
I-D-A	2.3%	2.6%	2.1%
I-D-D	0.6%	1.0%	1.0%

Table 5. Comparison of the frequencies in two-modification-sequence patterns

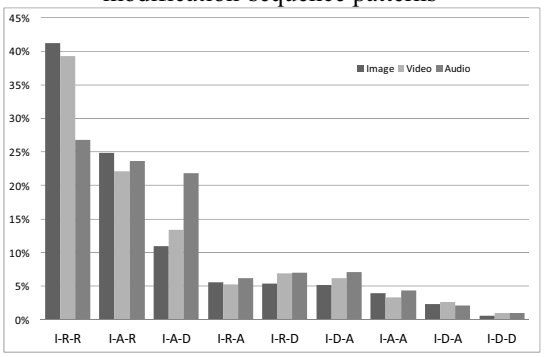


Figure 1. The distribution of the two-modification-sequence patterns among Image, Video, and Audio searches

Three-modification-sequence analysis

The dominance of replacement and addition modification continued in the analysis of three–modification-sequences. As shown in the high frequencies of both *I-R-R-R* and *I-A-R-R* sequences in Table 6, about 50% of all three modification sequences in image and video searches are associated with replacement and addition modifications. Similarly to the distribution in two-modification-sequence analysis, both image and video searches have much higher proportion of consecutive replacement modifications (i.e. *I-R-R-R* sequences) than audio searches. The top three modification sequence patterns distribute more evenly in audio searches with a slightly more *I-A-D-A* sequences than the other two types of searches. For modification sequence patterns beyond the top five, all multimedia searches demonstrate similar distribution, thus provide little information for characterizing different types of multimedia searches. Figure 2 shows that *I-R-R-R* sequences are more prominent in both image and video searches as the distribution decreased more in the top three modification sequence patterns than audio searches. The top five patterns account for over half of the three-modification–sequence in all multimedia searches and only one pattern contains the deletion modification. When we further differentiate *I-R-R* sequence into *I-R-R-R*, *I-R-R-A*, and *I-R-R-D* sequences, the prevalence of replacement over addition and addition over deletion continued (*I-R-R-D* not shown in Table 5). Hence, user's preference for replacing terms and the unlikelihood of deletion modification in the early stage of query modification can be confirmed.

	Image	Video	Audio
I-R-R-R	35.3%	32.4%	21.6%
I-A-R-R	20.1%	17.3%	16.7%
I-A-D-A	5.3%	6.3%	11.1%
I-R-A-R	4.5%	4.1%	3.6%
I-R-R-A	3.9%	3.5%	2.6%
Total	69.1%	63.7%	55.6%

Table 6. Comparison of the top 5 frequencies in three-modification-sequence patterns

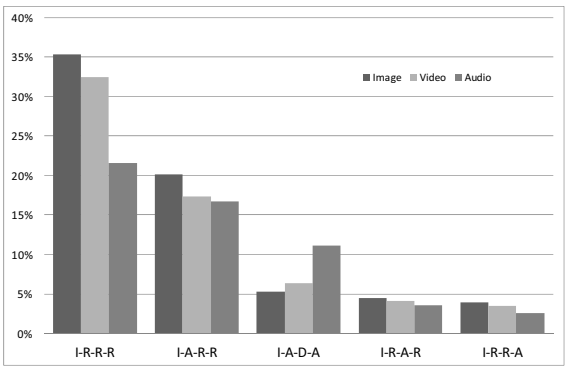


Figure 2. The distribution of the top 5 three-modification-sequence patterns among Image, Video, and Audio searches

5.3 Search strategies based on modification sequence

We investigated search strategies based on the consecutive replacement sequences (i.e. the *I-R-R* and *I-R-R-R* sequences) because of their prominence in the modification sequence analysis. As Table 7 shows, about 40% of all *I-R-R* sequences exhibit a dynamic search strategy. From Table 8, the proportion of dynamic search increases to more than 50% in *I-R-R-R* sequences. While this large proportion of dynamic search can be anticipated, constant search which accounts for nearly one-third of all consecutive replacement sequences is more revealing. Because the query length is held at constant within each session in constant searches, it appears to be a one-to-one relationship between the replaced term pairs. A reasonable explanation is the interchange of synonyms or associated terms of the same construct (e.g. "PC" to "Mac", "UK" to "USA", or "girls" to "boys"). Both Table 7 and 8 show more specified searches than generalized searches, but the difference is only

noticeable in image searches. This indicates that image users are more prone to adopt specified strategy (i.e. gradually adding more search terms as the searching progresses) than other types of multimedia users. In other words, image users progressively consolidate or learn more information about their problems through the interaction with the Web search engine. The percentage for the search strategy analysis from *I-R-R* and *I-R-R-R* sequences are presented in Table 7 and Table 8 respectively.

	Image	Video	Audio
Dynamic	40.1%	40.6%	42.7%
Constant	34.8%	34.3%	28.5%
Specified	15.2%	12.8%	15.2%
Generalized	9.9%	12.3%	13.5%

Table 7. The percentage of each search strategy from *I-R-R* modification sequences

	Image	Video	Audio
Dynamic	52.9%	52.9%	58.3%
Constant	27.2%	25.7%	22.1%
Specified	11.9%	11.3%	10.1%
Generalized	8.0%	10.1%	9.4%

Table 8. The percentage of each search strategy from *I-R-R-R* modification sequences

As shown in Figure 3 and 4, both strategy analyses from *I-R-R* and *I-R-R-R* sequences suggested the highest constant search strategy in image searches. Thus image searches require most synonym or related term replacement modification than other types of multimedia searches, and such characteristic should benefit image searches more from term suggestion functionalities when refining the search queries. While video searches have slightly less proportions of constant searches than in image searches, they shared very similar distribution across the four types of search strategies. On the other hand, audio search users are more prone to adopt a dynamic search strategy.

In order to verify the replaced terms in constant search sequences, we implemented a Brill tagger[1] [3] to identify the part-of-speech of the replaced term pairs (i.e. the terms from the original query paired with the terms from the replacement query). Among the randomly selected 1465 constant *I-R-R-R* modification sequences, a total of 3003 replaced term pairs have been successfully tagged using the Brill tagger. More than 70% of these term pairs (2125 in total) have same part-of-speech, reassuring our explanation of interchanging between synonyms or associated terms of same construct in these constant search sequences.

[1] Details on the tagger implementation can be found in [17].

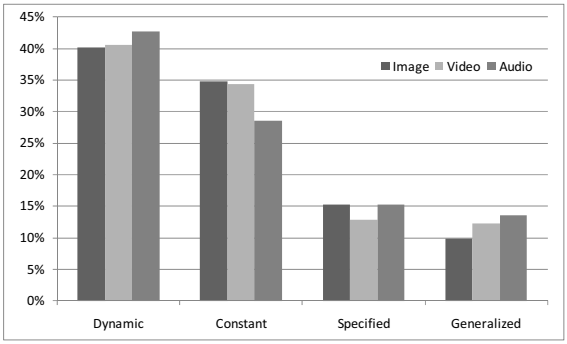


Figure 3. Comparison of the search strategies from *I-R-R* modification sequences

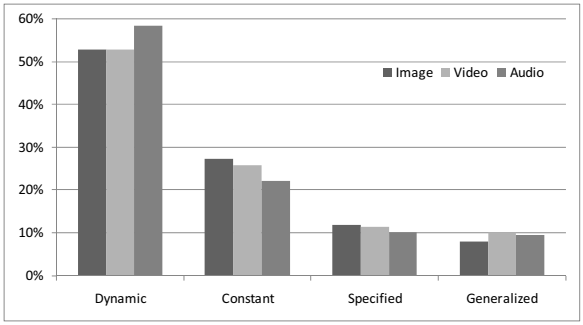


Figure 4. Comparison of the search strategies from *I-R-R-R* modification sequences

6. Discussion and future work

The statistics of query modification revealed that all multimedia search users shift their search topics more than refining their queries. Such phenomenon is most evident in audio searches as initial queries are more than triple of the replacement queries. The replacement queries are more than twice the addition queries in both image and video searches, whereas audio searches have notably more addition queries. Deletion queries are the least type of modifications in all multimedia searches, especially in image searches. Overall, when users do modify their queries, they tend to replace their search terms rather than adding or removing them. Such modification tendency is more prominent for visual searches (i.e. image and video searches). Although the number of topics searched by one user varies a lot, users searched around two to three different topics on average. In terms of in-session modification analysis, the majority of users only perform little modifications to their queries and visual search users modify their queries slightly more than audio users.

The analysis of modification sequence pattern suggests the tendency to replace and add search terms when modifying visual queries. The distribution of modification sequences shows a tendency toward the consecutive replacement modification sequences (i.e. the *I-R-R* and *I-R-R-R* sequences) in visual searches. This tendency also distinguishes visual searches from audio search and suggests the need for interchanging related search terms. In other words, visual search

users are more willing to interact with the system than audio search users.

In terms of search strategies, the changes in number of terms within *I-R-R* and *I-R-R-R* sequences reveal that about 40%-50% of users engage in dynamic searches. This typically reflects the unplanned nature of Web multimedia searching, which manifests the need for initiating several guessing runs to consolidate user's problem, or to find the appropriate search terms. Nevertheless, about one third of users adopt constant search strategy in which they replace search terms with an equal number of terms, suggesting the high likelihood of interchanging with synonyms or related terms. This constant search strategy also differentiates visual searches from audio searches. While visual searches always have higher proportion of constant searches, image users adopt most constant search strategy among all multimedia searches. Hence it can be assumed that image users should benefit most from knowledge or ontology based query expansion or term suggestion assistance. The reason of less constant search strategy in audio searches may be that audio searchers tend to use the song title or singer's names in their queries [19], resulting the replacement of these proper nouns other than interchanging similar terms in visual searches.

When the change of terms shows a unidirectional pattern, all multimedia searchers are more prone to adopt the specified approach. This finding is consistent with prior study's conclusion on user's primary concern of retrieval precisions [9]. In particular, image search users show a stronger preference for adopting this approach than other types of multimedia users. Typical scenario would be that image search users need to see widely before they know exactly what they are searching for or how their target images should look like. This characteristic implies the importance of a browsing tool that helps users compare different results and thus consolidate their problems quicker. A hierarchy arrangement of the results or term suggestions should also be useful.

Compared with general Web search studies, the current study findings are consistent with Jansen and Spink's [8] conclusion on the complexity of user's Web search behavior as one-query session increased over the years and users modify their queries less and less. This is to say that general Web users share the same characteristics with our user pool. Hence the effectiveness of the interaction between the user and the system is substantial to the improvement of query modification process. Future work should include a user study to understand the reasons behind each modification, as well as the corresponding search strategies. A semantic level analysis of replaced terms would also help discover the aspects of multimedia content that users modify most, such as the visual descriptors or the semantic meanings of the retrieved objects.

The prevalence of consecutive replacement modifications implies the need for an effective relevance feedback mechanism that would help users refine the importance of their query terms, perhaps with advanced search term suggestions based on the replaced terms (e.g. automatically displays synonyms or associated terms when user deletes a term). In terms of search strategies, the current study confirms the preference for the specified approach among image searchers. An interactive retrieval system that can gradually obtain more information about user's image problem would be helpful in guiding the user to explore the entire collection, and hence improve the query reformulation effectiveness.

7. Limitations

Due to the aim of using automatic approach to discover user's query modification behavior, this study only perform the part-of-speech analysis of replaced terms in constant search sequences. This limits our understanding of the types of terms being modified during the reformulation process. However, user's overall search strategy can still be inferred from our analysis. Although we have successfully discovered some unique characteristics among different types of multimedia searches, these findings are yet to be compared with general Web searches to address the differences in terms of query modification behavior and search strategies.

8. Conclusion

The current study investigated users' multimedia searching behavior based on their query modification methods. Our analysis showed that around 60% of query modifications are to formulate new search topics. Image and audio users searched more topics on average than video users. Our approach to analyze Web multimedia query modifications went beyond two consecutive queries. The analysis of session modifications revealed that visual search users (i.e. both image and video users) modify their queries slightly more than audio users. Visual search users also tend to replace search terms with other related terms rather than merely narrowing or broadening their searches. Generally speaking, visual searches showed similar modification patterns with much more consecutive replacement modifications than in audio searches. In terms of search strategies, the relatively high proportion of constant search strategy in visual searches indicates the importance of term suggestion assistance that helps user find the synonyms or related terms more easily. Our search strategy analysis also showed the tendency of adopting a specified approach in image searches, which suggests a need for query formulation assistance to help users gradually specify of their problems.

We present an automatic analysis procedure in this study, thus maximizing the ability to apply the same

analysis to different data sets, as well as allowing comparisons with general Web user's searching behavior. By adopting the analysis procedure, it is possible to extract more information about user's query modification behavior, especially the search strategies based on the statistical evidence. Future multimedia retrieval systems can utilize these different search characteristics to improve query formulation process and search efficiency.

9. References

[1] P. Anick. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 88-95, 2003.

[2] M. J. Bates. Information search tactics. *Journal of the American Society for Information Science,* Volume 30, pages 205-214, 1979.

[3] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the workshop on Speech and Natural Language*, pages 112-116, 1992.

[4] P. D. Bruza and S. Dennis. Query Reformulation on the Internet: Empirical Data and the Hyperindex Search Engine. In *Proceedings of the RIAO 97 Conference,* pages 488-499, 1997.

[5] D. He, A. Goker, and D. J. Harper. Combining evidence for automatic Web session identification. *Information Processing & Management,* Volume 38, pages 727-742, 2002.

[6] B. J. Jansen. Search log analysis: What it is, what's been done, how to do it. *Library & Information Science Research,* Volume 28, pages 407-432, 2006.

[7] B. J. Jansen, A. Goodrum, and A. Spink. Searching for multimedia: analysis of audio, video and image Web queries. *World Wide Web,* Volume 3, pages 249-254, 2000.

[8] B. J. Jansen and A. Spink. An analysis of Web searching by European AlltheWeb. com users. *Information Processing and Management,* Volume 41, pages 361-381, 2005.

[9] B. J. Jansen and A. Spink. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing & Management,* Volume 42, pages 248-263, 2006.

[10] B. J. Jansen, A. Spink, and B. Narayan. Query Modifications Patterns During Web Searching. In *Fourth International Conference on Information Technology (ITNG'07)*, pages 439-444, 2007.

[11] B. J. Jansen, A. Spink, and J. Pedersen. An analysis of multimedia searching on AltaVista. In *5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 186-192, 2003.

[12] T. Lau and E. Horvitz. Patterns of Search: Analyzing and Modeling Web Query Refinement. In *Proceedings of the 7th international conference on user modeling*, pages 119-128, 1999.

[13] H. C. Ozmutlu and F. Cavdur. Application of automatic topic identification on Excite Web search engine data logs. *Information Processing & Management,* Volume 41, pages 1243-1262, 2005.

[14] S. Ozmutlu. Automatic new topic identification using multiple linear regression. *Information Processing & Management,* Volume 42, pages 934-950, 2006.

[15] S. Ozmutlu, A. Spink, and H. C. Ozmutlu. Multimedia web searching trends: 1997-2001. *Information Processing & Management,* Volume 39, pages 611-621, 2003.

[16] S. Y. Rieh and H. Xie. Analysis of multiple query reformulations on the web: The interactive information retrieval context. *Information Processing & Management,* Volume 42, pages 751-768, 2006.

[17] T. Simpson and T. Dao. WordNet-based semantic similarity measurement. *The Code Project.com*, Oct. 1, 2005. [Online]. Available: http://www.codeproject.com/KB/string/semanticsimilaritywordnet.aspx. [Accessed: Jun. 10, 2009].

[18] A. Spink and B. J. Jansen. Searching multimedia federated content web collections. *Online Information Review,* Volume 30, pages 485-495, 2006.

[19] D. Tjondronegoro, A. Spink, and B. J. Jansen. A study and comparison of multimedia Web searching: 1997-2006. *Journal of the American Society for Information Science and Technology,* Volume 60, pages 1756-1768, 2009.

[20] M. Zhang, B. J. Jansen, and A. Spink. Information Searching Tactics of Web Searchers. In *69th Annual Meeting of the American Society for Information Science and Technology,* Austin, USA, 2006.

Term Clustering based on Lengths and Co-occurrences of Terms

Michiko Yasukawa and Hidetoshi Yokoo
Department of Computer Science
Gunma University
1-5-1 Tenjin-cho, Kiryu, Gunma 376-8515, Japan
{*michi, yokoo*}*@cs.gunma-u.ac.jp*

Abstract *Document clustering is useful for addressing vague queries and managing large volumes of documents. However, conventional algorithms for document clustering do not consider the lengths of terms in the cluster labels. Some cluster labels have considerably different lengths. Cluster labels with different lengths result in wasted space on the screen. To counter this problem, we have developed a new method for term clustering. Our method considers both lengths and co-occurrences of terms while clustering them. Therefore, our method can achieve an efficient document search even with limited area on the screen.*

Keywords Information Retrieval, Web Documents

1 Introduction

A single-term query is usually ambiguous, and it results in a large number of documents. Search result clustering is very effective in managing such a large number of searched documents[1][2][3]. We have developed a model for classifying a set of searched documents into clusters of related terms[4]. The developed system was found to be useful for PC users but not for the users of mobile terminals. This is because the number of terms in each cluster label varies. Further, the number of letters in each term varies. For example, the number of letters in *cafe* is less than half the number of letters in *restaurant*. The situation worsens when we use a proportional font to represent the cluster labels. In a proportional font, the space required to represent the letter "w" is larger than that required for "i," thereby resulting in wasted space on the screen (Figure 1 (a)). In order to make optimal use of the limited space on mobile terminals, we propose a new clustering method. Our proposed method generates a set of related-term clusters that fit in a rectangular region (Figure 1 (b)). The related-term clusters are based on the co-occurrence of related terms and are supposed to be intuitively better understood by users than randomized related terms. This is because co-occurrent terms in documents are supposed to be terms associated with each other. According to Meyer and Schvaneveldt[5], pairs of associated terms such as (BREAD-BUTTER) and (NURSE-DOCTOR) are more promptly recognized by users than pairs of unassociated terms such as (BREAD-DOCTOR) and (NURSE-BUTTER). Hence, the proposed clusters are considered to be effective in the selection of preferable terms on the display screen by users.

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

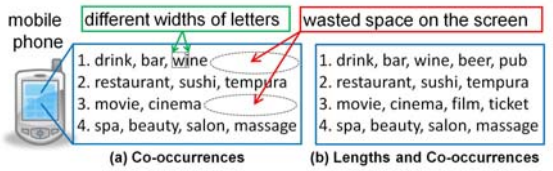


Figure 1: Comparison between clustering methods

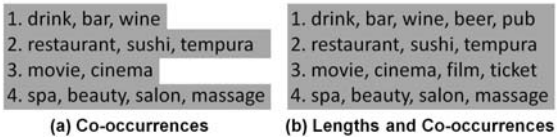


Figure 2: Comparison between occupied areas

Further, users do not have to input each letter in the terms when using related-terms clusters. Users may simply select preferable terms on the screen. Moreover, users have an option of selecting a number on the screen for accessibility; for example, they can push button "2" to indicate a set of terms "restaurant, sushi, tempura" at once. As shown in Figure 2, clustering (b) can be more informative than clustering (a) because the results of the former occupy a larger area on the screen.

2 Proposed Method

Our proposed method is described as follows. We assume that mobile users will enter a short query (typically just one term such as a location name) and will seek suggested terms in response to the query; The system should present a well-organized menu of various suggestions in response to the query, and the user will then select one of the suggestions in the menu as an expanded requirement. After this, the system will present a number of web pages related to that expanded requirement. The proposed method is explained in the following paragraphs.

In our proposed clustering method, we first generate a set $L(Q)$ of terms related to the short primary query

Q and determine the relationship between the elements in $L(Q)$. Specifically, we denote the i-th term selected from $L(Q)$ as $t_i(Q)$. For example, for $Q =$ "Shinjuku," $t_i(Q) =$ "restaurant" may be a related term. Next, we define a query consisting of Q and $t_i(Q)$ as $q_i = \langle Q, t_i(Q) \rangle$. From the web pages that are searched by q_i, we extract the adjacent terms of $t_i(Q)$. We call these terms *association terms* of $t_i(Q)$. Let $A_i(Q)$ be the list of association terms of $t_i(Q)$. Note that $A_i(Q)$ may include another related term $t_j(Q)$. This is because the term $t_j(Q) =$ "sushi" may be adjacent to $t_i(Q) =$ "restaurant" in the web pages of $Q =$ "Shinjuku." In order to determine the relationship between the terms $t_i(Q)$ and $t_j(Q)$ with respect to the primary query Q, we define their co-occurrence score, $score_{ij}$, by

$$score_{ij} = (and_{ij}/or_{ij}) * (1 + \log(and_{ij})), \quad (1)$$

where and_{ij} denotes the number of lists of association terms that include both $t_i(Q)$ and $t_j(Q)$ and or_{ij} denotes the number of lists of association terms that include either $t_i(Q)$ or $t_j(Q)$. The equation is defined empirically on the basis of our exploratory experiments. We have observed that in order to consider the co-occurrences of terms, the equation should amplify and_{ij}; however, the amplification must not be excessive.

In the algorithm, we set the minimum and maximum acceptable lengths per line of the display screen to $\ell_{\min}$ and $\ell_{\max}$, respectively.

***Algorithm*—**Rectangular Clustering

(Step 1) Read a list $L(Q)$ of terms related to every query Q. Determine the length of each term in the list $L(Q)$. Here, the length is the actual length of the term on the screen.

(Step 2) For every pair $t_i(Q)$ and $t_j(Q)$ of terms in $L(Q)$, calculate $score_{ij}$ using equation (1).

(Step 3) For every term $t_i(Q)$ in $L(Q)$, select the two highest co-occurrence terms $t_{k_1}(Q)$ and $t_{k_2}(Q)$. Then, merge the selected terms to generate a primitive cluster $c_i = \langle t_i(Q), t_{k_1}(Q), t_{k_2}(Q) \rangle$. Note that terms may overlap in the primitive clusters. Before proceeding to Step 4, calculate the score of c_i as the sum of $score_{ik_1}$ and $score_{ik_2}$.

(Step 4) Remove overlapping terms from clusters. If there are overlapping terms among multiple clusters, retain only those terms that are in the cluster with the highest co-occurrence score. Eliminate all terms that are repeated in other clusters.

(Step 5) Determine the total length of each cluster to alter the cluster. If the total length of a cluster is less than ℓ_{min}, merge the cluster with another cluster. If two clusters c_i and c_j had common terms when they were primitive clusters, they can be merged.

(Step 6) Determine the total length of each cluster to decide whether to select or reject the cluster. If the total length is adequate, select the cluster for a cluster label. If the total length is less than ℓ_{min}, reject the cluster. If the total length is greater than ℓ_{max}, select terms from the cluster as many as possible until the total length is in the range between ℓ_{min} and ℓ_{max}.

(Step 7) Remove the terms used for the cluster labels from the list $L(Q)$. If $L(Q)$ is empty or if no more cluster labels are generated, write out the cluster labels, and end the algorithm. Otherwise, return to Step 3 and continue.

3 Implementation

In order to measure the actual length of a term on the screen, we use Graphviz[1] and IPA font[2]. With this software and font, we can generate the text image of the term. Then, we measure the lengths of terms by using the generated images. In order to calculate $score_{ij}$, we used a tool called GETA[3] for large-scale text retrieval. We used Search API of Yahoo!JAPAN[4] to collect search results of (1) related terms; (2) URLs, titles, and summaries; and (3) web pages. An actual application of the proposed method in a mobile web search system has been demonstrated in [6].

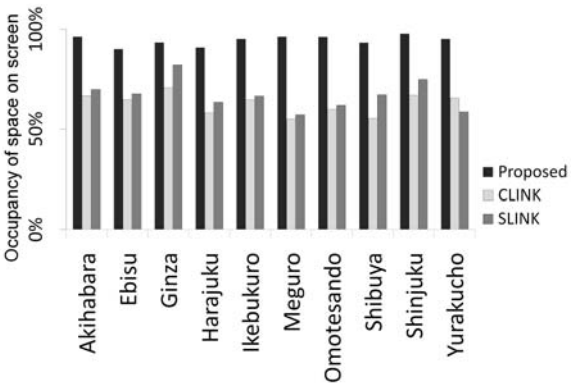


Figure 3: Length of terms on the screen

4 Experiment

We compare the proposed algorithm with two other clustering algorithms— complete-link clustering (CLINK) and single-link clustering (SLINK). These algorithms are widely used conventional algorithms and have been described in detail in [7]. While our algorithm considers both lengths and co-occurrences of terms, these conventional algorithms consider only co-occurrences of terms.

[1] http://www.graphviz.org/
[2] http://ossipedia.ipa.go.jp/ipafont/
[3] http://geta.ex.nii.ac.jp/e/
[4] http://developer.yahoo.co.jp/

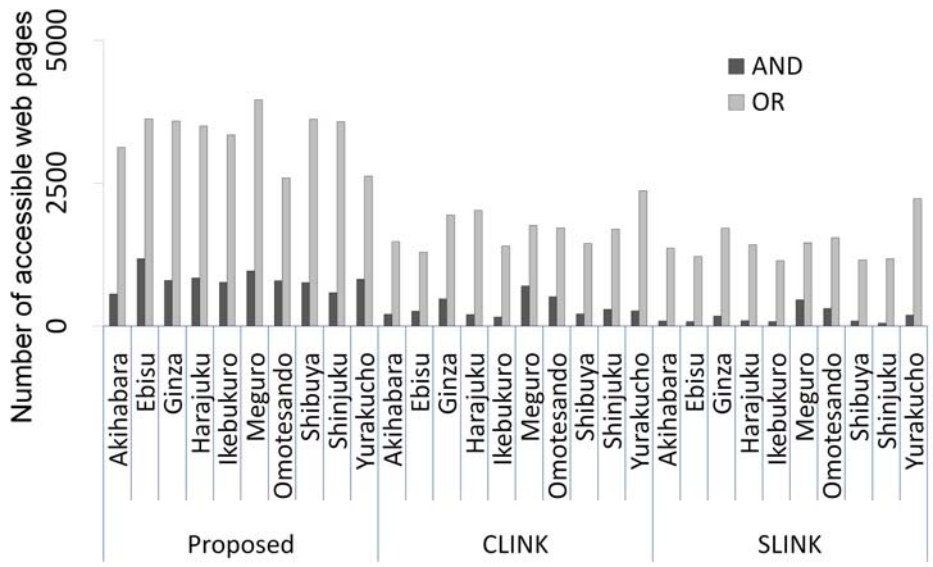


Figure 4: Accessible web pages for different terms on screen

The names of major places in Tokyo were used as queries in the experiment. For each query, 100 related terms and 10,000 web pages were obtained. Term clusters that fit in a rectangular region of 160×160 pixels were generated using the 16-pixel proportional font. In the experiment, the parameters of CLINK and SLINK were adjusted to generate as many clusters as possible with each cluster having two or more terms.

4.1 Area Occupied on Screen

One of the key features of the proposed method is that it takes into consideration the term lengths, thereby optimizing the use of screen space. We investigated the total length ℓ_s of the clusters for each query and then calculated the ratio of the total length ℓ_s of the clusters to the total length ℓ_r of the lines in the rectangular region. In Figure 3, we can observe that the term clusters generated by using the proposed algorithm occupy a larger area on the screen as compared to SLINK and CLINK. Hence, the proposed algorithm is considered to provide more information than others.

4.2 Efficiency of Web Search

Another key feature of the proposed method is its high search efficiency. In Figure 4, "AND" indicates the condition that the web pages include two or more terms in the clusters, e.g., ((restaurant AND sushi) or (sushi AND tempura) or (tempura AND restaurant)). Further, "OR" indicates the condition that the web pages include one or more terms in the clusters, e.g., (restaurant OR sushi OR tempura). The proposed algorithm enables users to obtain desired pages more efficiently than conventional algorithms.

5 Conclusion

We have proposed a new clustering method that enables efficient term clustering in a mobile web search. In the proposed method, a set of primitive clusters are generated on the basis of the co-occurrences of terms. Then, the clusters are altered on the basis of the co-occurrences and lengths of terms. Finally, the clusters are evaluated and adjusted on the basis of the lengths of terms. Term clusters obtained by the proposed method effectively use a small rectangular region on the screen. Hence, the clusters are informative and can aid mobile users to search documents efficiently. In the future, we intend to apply the proposed method to various information retrieval systems.

References

[1] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *SIGIR*, pages 46–54, 1998.

[2] D. Beeferman and A. L. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

[3] S. Osinski. Improving quality of search results clustering with approximate matrix factorisations. In *ECIR*, pages 167–178, 2006.

[4] M. Yasukawa and H. Yokoo. Related terms clustering for enhancing the comprehensibility of web search results. In *DEXA*, pages 359–368, 2007.

[5] D. E. Meyer and R. W. Schvaneveldt. Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, 90:227–234, 1971.

[6] M. Y. Yasukawa and H. Yokoo. Clustering search results for mobile terminals. In *SIGIR*, pages 880–880, 2008.

[7] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. The MIT PRESS, 1999.

WriteProc: A Framework for Exploring Collaborative Writing Processes

Vilaythong Southavilay, Kalina Yacef
School of Information Technologies, The University of Sydney
NSW 2006, Australia
vstoto@it.usyd.edu.au, kalina@it.usyd.edu.au
Rafael A. Calvo
School of Electrical and Information Engineering, The University of Sydney
NSW 2006, Australia
rafa@ee.usyd.edu.au

Abstract *Collaboration and particularly collaborative writing is an increasingly essential skill needed in the workplace and education. Until recently most of the focus of research has been the final product of the writing, rather than the process itself. In this paper, we propose an innovative framework for investigating collaborative writing processes. The WriteProc framework utilizes both process and text mining tools to analyze the process that groups (or individual) writers follow, and how the process correlates to the quality and semantic features of the final product. Furthermore, WriteProc is integrated with existing web 2.0 writing tools, providing full support for writing, reviewing and collaboration. We describe the architecture that integrates tools for analyzing the process and semantics of the writing. We also provide a case study on data collected from a group of undergraduate students writing collaboratively an essay, with peer reviewing and use of an automatic feedback tool.*

Keywords Document workflows, web documents, process mining

1 Introduction

Computer-Supported Collaborative Work (CSCW), particularly Collaborative Writing (CW), has received attention since computers have been used for word processing. Due to the availability of the Internet, people increasingly write collaboratively by sharing their documents in a number of ways. Writing individually and collaboratively are considered essential skills in most industries, academia, and government. This has led to increased research on how to support the production of better documents.

In Education, computer-supported writing has been studied for decades. Goldberg et al. [6] collected a decade of empirical data and in a meta-study found "that when students write on computers, writing becomes a more social process in which students share their works with each other". They also noted that when using computers, students prefer to make revisions while producing, rather than after producing, text. Between initial and final drafts, students also tend to make more revisions when they write with computers. In most cases, students also tend to produce longer passages when writing with computers. In addition, review feedback, especially peer review, has been recognized as one effective way to learn writing [3, 4]. When students write with computers, they engage in the revising of their work throughout the writing process, more frequently share and receive feedback from their peers, and benefit from teacher input earlier in the writing process. Although these studies show that computer-supported writing including automatic feedback tools efficiently assists students in writing and reviewing, understanding the writing process is crucial for developing support technologies for CW.

Over the past two decades, there has been abundant text-mining research for improving the support of quality writing. But work such as automatic scoring of essays [11], visualization [9], and document clustering [1] focus on the final product, not on the writing process itself. Our vision is to investigate how ideas and concepts are developed during the process of writing could be used to improve not only the quality of the documents but more importantly the writing skills of those involved.

Improving the process of writing requires understanding how certain sequence patterns (i.e. the steps a group of writers follow) lead to quality outcomes. We see the sequence pattern as comprised both of time events (as used in other process mining research) and of the semantics of the changes made during that step.

We combine here two techniques: process mining, which focuses on extracting process-related knowledge from event logs recorded by an information system, and semantic analysis, which focuses on extracting knowledge about what the student wrote (or edited). The

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

field of process mining covers many areas, like performance characteristics (e.g. throughput times), process discovery (discovery of the control flow), process conformance (checking if the event log conform specification), and social networks (e.g. cooperation) [2]. Particularly, process mining analysis is necessary to understand group awareness, and writers' participation and coordination. Text mining combines indexing, clustering, latent semantic analysis and other techniques studied by the document computing community.

In this paper, a conceptual framework and tools for supporting collaborative writing (CW) are introduced. Our framework is based on a taxonomy of collaborative writing proposed by Lowry et al. [8] and defines writing activities, strategies, work modes and roles involved in CW. With this taxonomy, the framework incorporates process mining and text mining technologies in order to gain insight of collaborative writing process.

The remainder of the paper is organized as follows. In Section 2, WriteProc, a framework for supporting CW and the analysis of its process and semantics is presented. A case study of process mining for a reviewing tool, Glosser is then presented in Section 3. Finally, Section 4 provides discussion of our case study and future work planned in this area.

2 WriteProc

Let us describe WriteProc, a framework for analyzing individual and collaborative writing process. It consists of three tools: writing, reviewing and analysis tools. The analysis tool utilizes both process and text mining techniques.

Our aim of developing WriteProc is to assist individual or groups of writers during the writing process. Particularly, WriteProc can advise writers with reviewing feedback and visualization of the analyses of writing activities and text changes during the process of writing.

2.1 Overall conceptual description

The framework integrates a front-end writing tool which not only supports collaborative writing activities, but also stores all revisions of documents created, shared and edited by groups of writers. Each revision of particular documents must contain all needed information such as edited text, timestamp of committing change, and identification of the writer. In order to perform analysis of writing process for particular documents, all revisions of the documents are retrieved and traced.

A reviewing tool is also embedded in the framework. It assists writers in revising their own pieces of writing and reviewing others works. After receiving feedback generated automatically by the reviewing tool, writers can edit and change their documents' content accordingly. The tool keeps records of writers' reviewing activities in event logs. The event logs of the tool are then extracted to gain an insight on how writers use the reviewing tool and how review feedback affects changes in reviewed documents.

Process and semantic analysis tools are used in the framework. Based on both the information (such as timestamp and writers' identification) of all revisions and event logs of reviewing activities, a process mining tool is used to discover sequence patterns of writing activities. The process analysis provides a way to extract knowledge about writers' interaction and cooperation. The analysis can identify interactions' patterns that lead to a positive outcome and indicate patterns that may lead to problems. In addition, a text mining technique is performed to analyze text-based changes of all revisions of documents. The text-based analyses can provide semantic meaning of changes in order to gain insight into how writers develop idea and concept during writing process.

2.2 Implementation

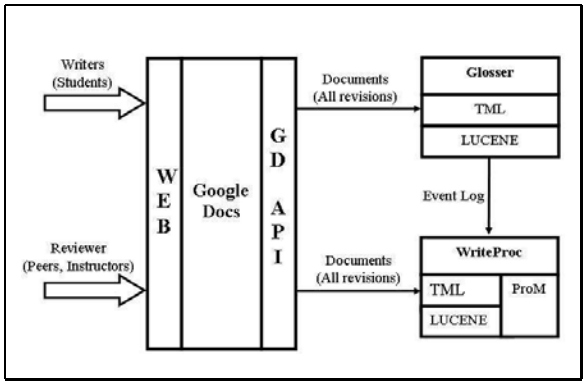


Figure 1: WriteProc: A framework supporting collaborative writing.

Based on the overall concept described above, the framework utilizes process and text mining technologies. It employs open-source utilities of those techniques to conduct analysis of writers' interaction and text in order to assist writers in identifying and realizing their writing process in collaborative manner. Figure 1 shows the framework for supporting collaborative writing (CW).

2.2.1 Writing environment: Google Docs

In order to use a process and semantic analysis tool in real scenarios, the tool must be closely integrated to the writing environment. Tools such as Microsoft Word or OpenOffice do not keep traces of the writing process. Web 2.0 tools such as Google Docs (and the incipient Microsoft Word Live) allow users to write on a web application (or offline and then synchronizing). The service provider keeps the different versions of the document. Therefore, we selected Google Docs in our implementation of WriteProc.

In WriteProc, Google Docs (GD) is used as a front-end writing tool of the CW. It is a web-based utility with most needed functionalities for word processing and it allows users to share their documents with other team

members and to write synchronously. Users can access GD through their web browsers from anywhere and at anytime they want. Each user needs a Gmail account to access the tool that they can obtain from Google free of charge.

At the center of the framework is Google Document Lists Data API (GDAPI) used to integrate GD to our CW system as shown in Figure 1 The API allows WriteProc to retrieve and track all versions of documents created, shared and edited among groups members. In GD, each document created is uniquely assigned a document identification number. The GD also keeps track of all version numbers of each document by incrementing its version numbers each time the document is edited. Every time a writer makes changes and edits a particular document, the identification of the writer, the edited content of the document, timestamp of committing changes and the version number of the edited document can be retrieved and stored at the central relational database of CW system by using the API. This information extraction is executed seamlessly offline and users as writers are not aware of it and are able to perform their writing tasks seamlessly. The API also provides us the ability to build an interface to create and share documents in CW system. This can be very helpful for instructors or supervisors to create and assign documents to groups of writers and reviewers without accessing GD. An appointed owner of a document can edit it, where as an assigned 'viewer' can only review it.

2.2.2 Reviewing tool: Glosser

Glosser is a web-based application providing support for writing in English [16]. It was designed and implemented to support a review feedback model. Figure 2 shows such a model. Glosser assists users to revise their own document and review other documents. It has the analysis and revision tracking system used for reviewing. Writers can use Glosser in order to gain insight into their essays' structure and coherence. To review particular documents, the system consists of several functionalities, as shown in Table 1:

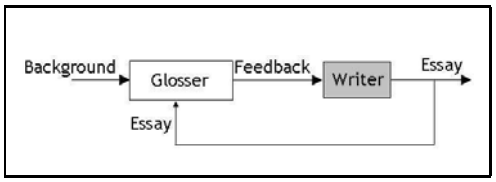


Figure 2: Automated writing feedback.

In the case study described in Section 3, students used a reviewing tool, Glosser [16]. A document created and shared among a group of writers can be reviewed in Glosser, which also accesses each revision using the Google Document Lists Data API. Users can access Google Docs from Glosser or vice versa.

Tool	Description
Home Tool (HOT)	showing basic statistics such as numbers of words and revisions.
Topic Tool (TOT)	checking if content provides evidence to support its topic sentences.
Flow Tool (FLT)	reviewing coherence and checking how paragraphs and sentences follow from previous ones.
Keyword Tool - HTML (KTH)	showing semantic flow.
Keyword Tool - Graph (KTG)	depicting the visualization of semantic flow.
Group Tool (GRT)	showing participation of authors for different versions.

Table 1: Reviewing tools of Glosser

2.2.3 Process and text mining tools

The interesting components of WriteProc are the process and text mining tools. The event log of Glosser is stored at the central relational database. The event log is used as a source to a process mining tool in order to gain an insight on writing activities and writers' interaction. The process mining tool utilized in the WriteProc is ProM [15]. In the next section, ProM will be used to demonstrate a process mining technique for our case study. In addition, an independent measure is developed to analyze the changes in each version of the documents in order to understand the nature of changes and the level of these changes. The analysis uses a text mining technique to find semantics changes among all versions of documents. This technique uses information from all the versions of documents performed by groups of writers. The text information of each version stored in the central database is indexed using Lucene [7] so that text produced by a group of writers can be systematically searched, sorted, filtered, and highlighted. After indexing all versions of documents, the system then analyzes the relationship between them and their terms using Text Mining Library (TML) in order to produce a set of concepts and nature of text changes in all versions of the documents.

3 Case study

As a way of evaluating the architecture and implementation of WriteProc and illustrating how it can be used, we discuss a case study where the tool is used to study writing processes in a software engineering unit conducted during the first semester of 2009 at the University of Sydney. It is important to note that Human Research Ethics Clearance has been completely granted from the university for this study. All students involving in the study signed an informed consent.

There were 58 students in the course, which was E-business Analysis and Design. They were

organized in groups of two and asked to write Project Specification Documents (PSD) for their proposed e-business projects. Each group had to submit one PSD of between 1,500 and 2,000 words (equivalent to 4-5 pages). Students were required to write their PSD on Google Docs and share the documents with the course instructor. They were asked to submit their PSD using Glosser, a reviewing tool mentioned in Section 2.2.2. The submitted PSD was reviewed by other two students who were members of different groups. Students had one week to review each others' documents and submit their feedback. After getting feedback on their documents from their peers, students could revise and improve their writing if necessary before submitting the final version one week later. The submission of the final version of PSD also used Glosser. The total event log file of the system consisted of usage data of Google Docs and Glosser for three weeks. In addition to this log file, the marks of the final submissions of the PSD together with a very good understanding of the quality of each group through the semester was used to correlate behaviour patterns to quality outcomes. In particular, to be able to give insight into how students used the reviewing tool for revising their own documents and reviewing others and to give recommendation to improve the system, we performed a process diagnostics method to give a broad overview of students' interaction and collaboration.

3.1 Log Preparation

Unlike data preprocessing for workflow mining [5], our approach used a data preprocessing method for behavior pattern mining [12]. This method was used with a process mining tool like ProM [15]. Glosser's event log was a typical Web server log which was a text file. The first step of data preprocessing was to filter and clean up the data. The next step of data preprocessing was to define process instances (cases). Our approached used a document as a notion of process instance. We utilized the concept of perspective, proposed by Song et al. [13] to partition event sequences. Our perspective of the event data log was based on documents. Particularly, we wanted to find out how users interact and coordinate for writing and reviewing documents. The final step in data preparation was to transform the log file to a standard format for process mining. Process mining tools such as ProM use MXML (as in *Mining XML*) files as sources [15]. The transformed MXML file was then used as a source for a process mining tool like ProM.

3.2 Log inspection

After preprocessing, the resulting event log consisted of 29 documents with a total of 4,677 events. Each process case represented one document. There were 8 different types of events (Section 3.4 described the process model and event types). The bar chart of Figure 3 shows the number of events for each of the 29

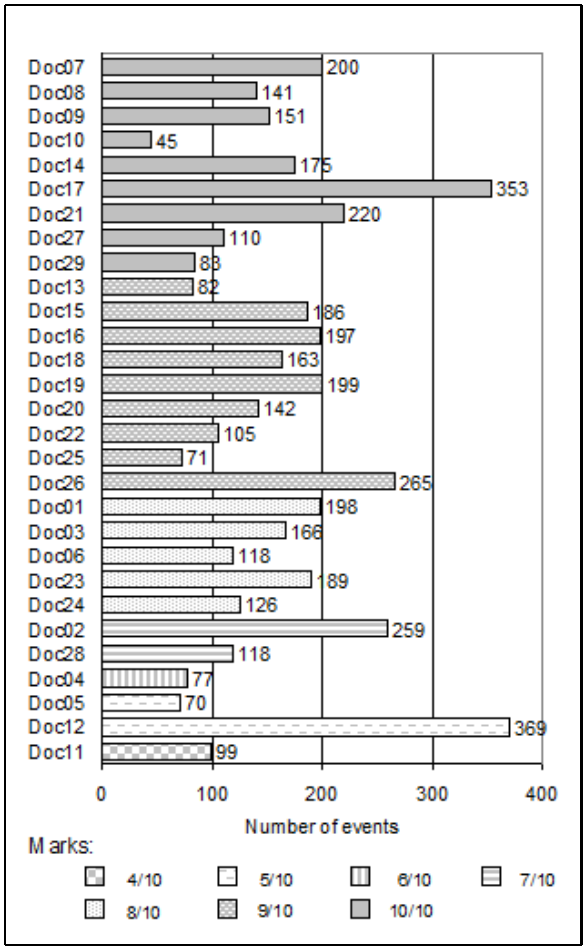


Figure 3: Comparing number of events of 29 documents ranked by their final marks.

documents, represented by the length of the bar (*DocXX* denotes the document of *Group XX*). The documents are ranked based on their final mark ranging from 4/10 to 10/10. For example, *Doc07*, *Doc08*, *Doc09*, *Doc10*, *Doc14*, *Doc17*, *Doc21*, *Doc27* and *Doc29* all obtained the highest mark, i.e. 10/10, while *Doc11* obtained the lowest mark of 4/10. On average there are 161 events per document. The maximum number of events is 369, with *Doc12*. *Doc10* has the smallest number of 45 events associated with it.

Based on the number of events presented in Figure 3, we could not distinguish the better from the weaker groups. Although Group 12 has the maximum number of interaction events, it was ranked in the 6th place. In contrast, the document of Group 10 with the least number of interactions was given the highest mark. In addition, simple statistics drawn from the figure could not clearly provide understanding of students' interaction. Therefore, further analysis was made in order to distinguish group performance and cooperation. We will describe it next.

3.3 Historical snapshot of reviewing activities

The Dotted Chart Analysis utility of ProM [10] was used to analyze students' reviewing activities. The dot-

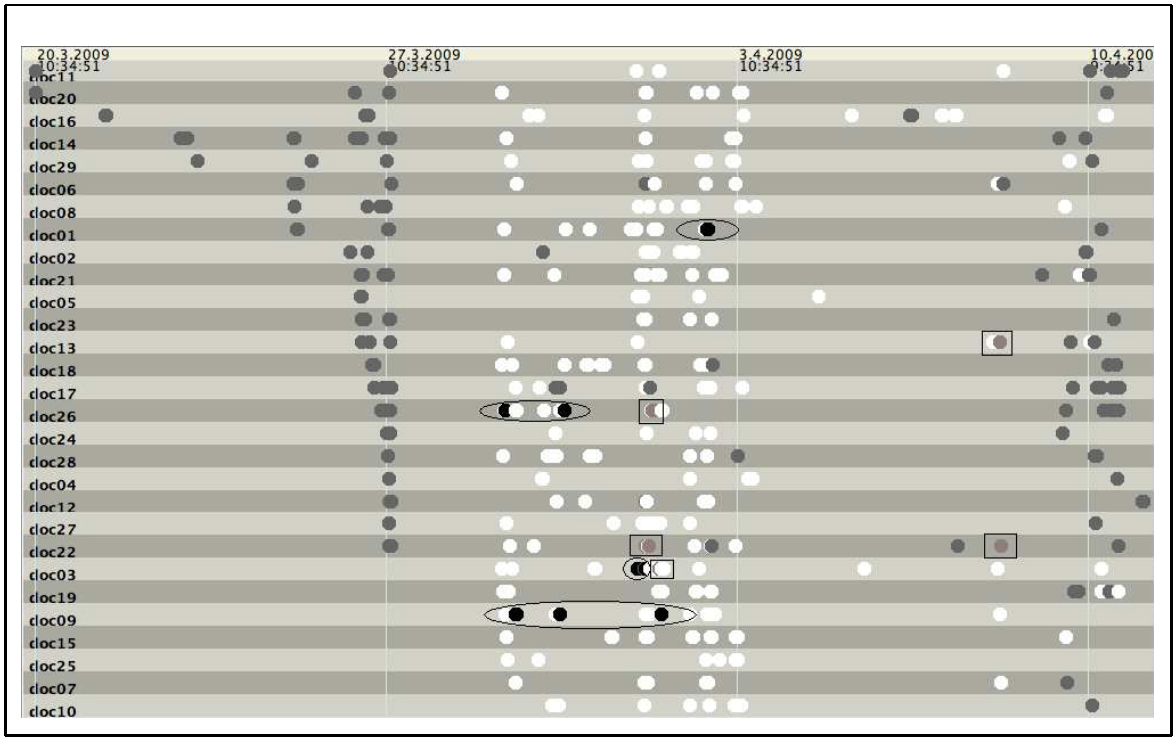


Figure 4: Dotted chart of 29 reviewed documents ordered by their first events' timestamps (from ProM tool [10]). Grey denoted events generated by by authors; white by reviewers, black by reviewers' group member (indicated by ovals) and brown by others (indicated by rectangles).

ted chart is similar to a Gantt chart [14], showing the spread of events over time by plotting a dot for each event in the log. Figure 4 illustrates the output of the dotted chart analysis of students' interaction for reviewing their PSD documents. All instances (one per document) are sorted by start time (the first event ever happening for a particular document during the three-week usage of the system). As shown in the figure, there are three important dates due to the three compulsory submissions: PSD for peer review on 27th March 2009; feedback of peer review on 3rd April 2009, and final PSD on 10th April 2009. In the figure, points represent events occurring at certain time. For particular documents, different color denotes events generated by different roles of users: grey events generated by authors, white events by reviewers assigned for peer review, black events (circled in the figure) by team members of assigned reviewers, and brown events (shown in rectangles) by *non-author* users who were neither assigned reviewers nor assigned reviewers' team members.

We can clearly see from the figure that 22 documents have been revised using Glosser in the first week before the submission for peer review. Obviously, those documents were only used in the system by the authors as indicated by grey events. There were 7 documents starting in the second week. They belonged to groups: 7, 9, 10 (received the same marks of 10/10); 15, 19, 25 (9/10); and 3 (8/10). This means that these documents have never been revised by their authors using Glosser before submitting for peer review. Nevertheless, these seven documents received high marks in the final assessment.

In addition, we observed that all activities of peer review happened in the second week before the submission of feedback. Most of the reviewing activities were performed by the assigned reviewers as indicated by white dotted events. This met the intention of the course of using Glosser for peer review. There are two interesting types of events in Figure 4. Firstly, 4 documents have events originated by students who were not the authors nor the assigned reviewers, as can be seen by black dots of documents of groups: 9 (received a mark of 10/10); 26 (9/10); and 1, 3 (8/10). Those events suggest that students either assisted their team members to review their assigned documents or performed the peer review task together with their group members sitting side-by-side using only one account. We discussed this matter with the course instructor who was also aware of this problem and will try to find a solution to prevent this problem happening in the next semester. Secondly, there are a small number of events where students reviewed others' documents which were not assigned to them nor to their team members for peer review, as indicated by brown dotted events for documents of groups: 3, 13, 22, and 26. These documents received good marks ranging from 8/10 to 9/10. We believe this happened when students shared their own PSD to their friends to assist them using Glosser. We

will perform further investigation to prevent this case from happening next year.

In addition, from Figure 4 we can notice that eight different documents were not revised by their authors using Glosser before the final submission. These documents were 8, 9 (10/10); 15, 16, 19, 25 (9/10); 3 (8/10); and 5 (5/10). Except document of group 5, all documents received the top three highest marks. In fact, three of them (9, 15 and 25) have never been revised by their authors using Glosser at all. This implies that the better groups used feedback received from peer review and the instructor as main source for revising their PSD. They did not spent much time using Glosser for revising their own documents. It is also interesting to note that reviewing activities did not evenly spread out for the three-week period of running the system. In fact, the system has only been used extensively for peer review in the second week as we can see in the figure. There were not many interactions in the third week. However, a number of activities happened a few days before the final submission.

To sum up, the dotted chart tool in ProM allows us to analyze reviewing activities in order to seek information on how each of 29 documents was reviewed by groups of students with different roles. We further investigated patterns of students' interaction for reviewing those documents, as described in the next subsection.

3.4 Process discovery and sequence analysis

From the event log of our case study data, we extracted the process model shown in Figure 5, which represents the process common to all the groups. Groups began with events of opening a particular document (ROD). Then, the reviewing tool was requested (TOR). After that, different reviewing activities were performed and the resulting feedbacks were displayed. The process reiterated until users logged off or closed their browsers. As discussed in Section 2.2.2, the reviewing activities involve these tools: HOT, FLT, KTG, GRT, TOT, and KTH.

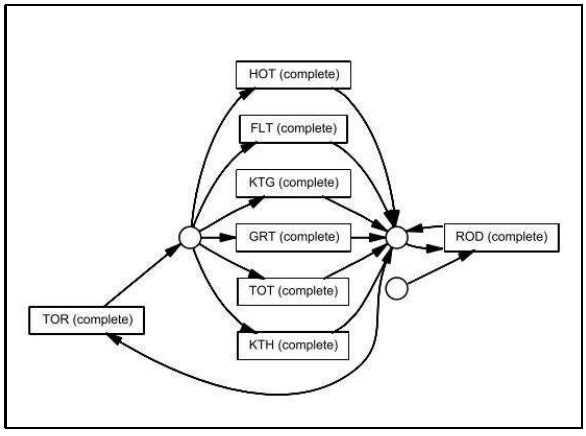


Figure 5: Process model of usage of the reviewing tool, Glosser.

We were naturally interested in finding out more about individual group activity and the path each group was following in this process. ProM provides a Performance Sequence Analysis plug-in to find the most frequent paths in the event log [2]. Figure 6 illustrates the interaction for documents of two groups in the course, with Group 1 (received a mark of 8/10) at the top and Group 29 (10/10) at the bottom. All eight events represented on horizontal axis are according to events discovered by the process model mentioned above. We examined sequence patterns for all documents of 29 groups. We discovered that only one document (*doc10*) was not used with all reviewing tools. In fact, the authors and reviewers of the document only utilized the HOT tool.

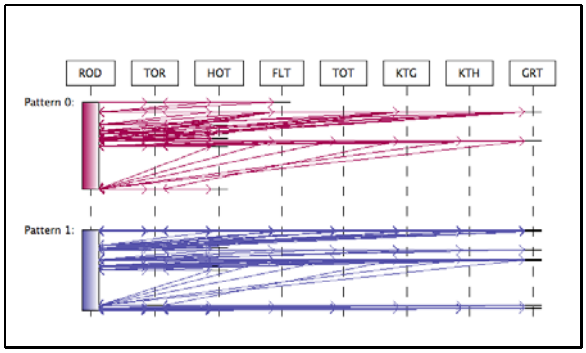


Figure 6: Sequence analysis of documents of Group 1 (above) and Group 29.

We have also used the same plug-in to extract all reviewing interactions for each document. This further investigation gives an insight on how different users reviewed documents using Glosser. For instance, Figure 7 depicts the users' interactions of two documents (*doc01* on the right and *doc29* on the left). Each column represents a user, where G29-1 is user 1 of Group 29 and so on. We analyzed all documents and found that more than half of them were revised by only one author using Glosser. In other words, although students worked in groups, only one member actually performed the reviewing task using the system.

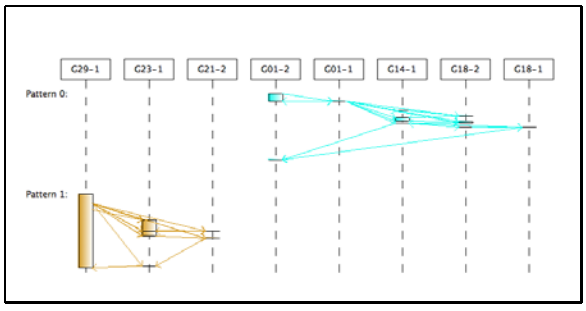


Figure 7: Users' interaction of Group 1 (right) and Group29.

In this section, we illustrated the potential of process mining techniques in understanding how writers react to peer review feedback and to the use of an automatic feedback tool like Glosser. In the log preparation, our

notion of process instance is based on documents because we would like to analyze user interactions on a same document. Dotted chart and sequence analyses were used to gain insights on how students reviewed their documents.

4 Discussion and conclusion

The work described here is a work in progress. While the case study presented here illustrates how our framework, WriteProc can be used, the data we used did not allow us to discover sequence patterns correlated to better outcomes. Although a pattern of users' interaction can be extracted for a particular group, there are different patterns for different groups. Indeed, we could not draw a significant pattern among groups in order to distinguish the better from the weaker groups. However, this gave us direction for the next step of our work. One way to improve our understanding of what writing processes lead to better outcomes so software tools can be used to provide advice during the writing process, is to use text mining techniques. For collaborative writing, we would like to have insights on how each version of documents changes in order to understand the writing process of each document. Although we are able to track all versions of documents that were reviewed in the system, this tracking analysis does not yet give us meaningful insights about the purpose of the text changes between each version. One possibility to systematically capture and interpret writing activities in collaborative writing is to understand changes in text written in each version of the document. Currently, we are working on extracting changes in concepts and ideas during the writing of documents. The text mining algorithms use vector representations of the documents accounting for the temporal nature of the data and the character of writing interaction. The result of the text mining tool will be analyzed and combined with the outcome of process mining (like the one described in the current case study).

Based on the process mining tool illustrated in the case study and text mining techniques as described above, we are developing WriteProc to provide visualization depicting users' interaction and collaboration in order to support writing activities. For example, a user interface can be built to assist a group of writers in identifying a plan for their writing process. This plan is created at the beginning of writing process representing a master plan of all writing activities and tasks. At particular point in time, writers can specify which stage they are on their writing process. During a time of writing, the system monitors if current writing activities are according to the writer's specification. For instance, a leader of a group of writers assigns all writing tasks to his or her members. The group leader specifies that the group is currently drafting its documents. WriteProc will track the group's writing activities and perform semantic analysis of the written texts. If it finds out, for example, that the members are actually outlining the documents (instead of drafting), it will provide information about their writing activities as feedback to the group. In this case, the writers can either adjust and modify their writing process specification, or investigate and change their written content according to the feedback given by the system.

In conclusion, we contribute here the description of WriteProc a framework that combines process and text mining techniques. The architecture of the system is described together with its integration to Google Docs as an environment for users to do the actual writing, and to the Google API that allows the tool to collect the revision information. A case study with a real teaching scenario is described and used to show how the tool can be used to analyze the process component of a collaborative writing task.

Acknowledgements This project has been funded by Australian Research Council DP0986873. The authors would like to thank Jorge Villalon and Stephen O'Rourke for their help with Glosser and TML.

References

[1] N. O. Andrews and E. A. Fox. *Recent Developments in Document Clustering*. Technical Report TR-07-35, Computer Science, Virginia Tech, 2007.

[2] M. Bozkaya, J. Gabriel and J. M. van der Werf. Process diagnostics: A mothod based on process mining. In *International Conference on Information, Process, and Knowledge Management*, February 2009.

[3] P. A. Carlson and F. C. Berry. Using computer-mediated peer review in an engineering design course. *IEEE Transactions on Professional Communication*, Volume 51, Number 3, pages 264–279, Sept. 2008.

[4] K. Cho and C.D. Schunn. Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers & Education*, Volume 48, Number 3, pages 409–426, 2007.

[5] C. A. Ellis, K. Kim and A. J. Rembert. Workflow mining: Definition, techniques, and future directions. *Workflow Handbook*, pages 213–226, 2006.

[6] A. Goldberg, M. Russell and A. Cook. The effect of computers on student writing: A meta-analysis of studies from 1992 to 2002. *Journal of Technology, Learning, and Assessment*, Volume 2, 2003.

[7] E. Hatcher and O. Gospodnetic. *Lucene in Action*. Manning Publications Co., 2004.

[8] P. B. Lowry, A. Curtis and M. R. Lowry. Building a taxonomy and nomenclature of collaborative writing to improve interdisciplinary research and practice. *Journal of Business Communication*, Volume 41, pages 66–99, 2003.

[9] S. O'Rourke and R. A. Calvo. Semantic visualisations for academic writing support. In Vania Dimitrova, Riichiro Mizoguchi, Benedict du Boulay and Art Graesser (editors), *14th Conference on Artificial Intelligence in Education*, pages 173–180. IOS Press, July 2009.

[10] ProM. Version 5.2. `http://prom.win.tue.nl/tools/prom/`, 2009.

[11] M.D. Shermis and J. Burstein. *Automated essay scoring: A cross-disciplinary perspective*, Volume 16. MIT Press, 2003.

[12] J. Song, T. Luo and S. Chen. Behavior pattern mining: Apply process mining technology to common event logs of information systems. In *IEEE International Conference on Networking, Sensing and Control*, Sanya, April 2008.

[13] J. Song, T. Luo, S. Chen and Feng Gao. The data preprocessing of behavior pattern discovering in collaboration environment. In *IEEE/WIC/ACM International Conferrence on Web Intellengence and Intenllent Agent Technology*, pages 521–525, Silicon Valley, USA, November 2007.

[14] M. Song and W. M. P. van der Aalst. Supporting process mining by showing events at a glance. In *7th Annual Workshop on Information Technologies and Systems*, pages 139–145, 2007.

[15] B. F. van Dongen, H.M.W. Verbeek A. K. A. de Medeiros, A. J. M. M. Weijsters and W. M. P. van der Aalst. The ProM framework: A new era in process mining tool support. *Lecture Notes in Computer Science: Application and Theory of Petri Nets*, pages 444–454, 2005.

[16] J. Villalon, P. Kearney, R.A. Calvo and P. Reimann. Glosser: Enhanced feedback for student writing tasks. In *The 8th IEEE International Conference on Advanced Learning Technologies*, Santander, Spain, July 2008.

An Analysis of Lyrics Questions on Yahoo! Answers: Implications for Lyric / Music Retrieval Systems

Sally Jo Cunningham, Simon Laing

Computer Science Department
University of Waikato
Hamilton 3240 New Zealand

{sallyjo, simonl} @cs.waikato.ac.nz

Abstract This paper analyzes 237 questions posted to Yahoo! Answers, a popular community-driven question and answer service. The questions are all natural language and are self-categorized by their poster as being related to music lyrics, and as such they provide a rich context for understanding lyrics-related information behavior outside the constraints imposed by specific lyrics retrieval systems. We categorize the details provided in the queries by the types of music information need and the types of music details provided, and consider the implications of these findings for the design of music/lyric systems and for music retrieval research.

Keywords User studies, multimedia document retrieval, music digital libraries

1 Introduction

Creating a useful and usable music retrieval system is a notoriously difficult task. A music document may consist of a symbolic representation of a work (eg, a score or MIDI encoding), an audio file (eg, MP3), an image (eg, a CD cover), textual metadata (a work's title, artist, composer, etc.), lyrics, a video of a performance—or a combination of any or all of the above [4]. Significant problems have yet to be resolved with document / query representation schemes, retrieval algorithms, and interface support in this challenging research area.

This paper focuses on identifying problems in developing systems for supporting lyrics-based information needs. At first glance it would appear that creating a lyrics-based music digital library would be one of the more straightforward development efforts in music retrieval, given that text-based retrieval is a better understood endeavor than image, video, and audio retrieval. This paper is a preliminary investigation into whether or not existing music retrieval research can address (or is addressing) support for lyrics retrieval systems.

Our approach is based on developing an understanding of what people want to find, and how they describe what they want, when they are trying to satisfy a lyrics information need. To that end, we analyze a set of lyrics related questions posted on Yahoo! Answers, an open Web-based question and answer forum. Once this understanding emerges of what lyrics seeking behavior 'in the wild' (that is, outside the constraints of a retrieval system, and as expressed in natural language) then we can identify remaining problems in supporting lyrics retrieval.

2 Previous work

At present music retrieval research is only lightly informed by an understanding of user needs. For a variety of reasons—including intellectual property law, limited access to a significant and standard music testbed, and lack of access to usage records for emerging commercial music systems—it has been difficult for researchers in music retrieval to develop or exploit data concerning the music information behavior of target users. This situation is particularly problematic in that the common assumptions of 'typical' music behavior made by retrieval researchers and music system developers have been found to differ markedly from actual music behavior in the real world [4].

Query log analysis of music related interactions on Web search engines (eg, [12]) yield extremely coarse-grained information on music behavior; sessions are generally short, queries are generally brief, and the log provides no insight into the searchers' motivations, intended use of retrieved music documents, or satisfaction with the search results. Few usage studies exist of music digital libraries or specific music collections (eg, [5], [8]). These types of investigations are necessarily limited to providing insights into the usability of features implemented in the system studied; log data cannot suggest additional functionality or document types appropriate for the users. For both search engines and digital libraries, the user's information need is obscured by the requirement of complying with the query formats of a specific system.

What is required, then, is a source of authentic music information behavior and needs. Earlier examinations of music behavior are based on information requests harvested from music-related

newsgroups [3], question-answer services [7], and archives of mailing lists [2]. These resources are seeing use to the extent of providing immense quantities of raw data on a scale similar to web logs; however, manual analysis methods limit in practice the size of a harvested dataset to at most a few hundred requests. This type of investigation complements log analysis with a finer-grained understanding of music behavior.

Most technical music retrieval research focuses on integrating lyrics with audio: for example, aligning lyrics to audio signals (eg, [9]); or using lyrics as a basis for thematic or genre clustering and classification of related audio files (eg, [10]). Lyric retrieval has proved to be a special case of text retrieval, inspiring additional research into problems such as identifying and matching multiple (non-identical) lyrics for a single song [6] and supporting search over lyrics that are syllabicated as performance instructions [13].

3 Data gathering and analysis

Yahoo! Answers is an internet based reference site that allows users to both submit and answer questions. Unlike some earlier 'ask an expert systems' (eg, Google Answers), there is no charge to post a question and no financial reward to answer questions. Instead, the system is driven by a 'points' and 'levels' arrangement that rewards posters of correct answers with status within the Yahoo! Answers community.

When posting a question to Yahoo! Answers, the user is required to specify one or more categories for it. We focus in this paper exclusively on Entertainment & Music > Music > Lyrics posts. Yahoo! Answers sees heavy use; as of September 2009, the Lyrics subcategory alone contained over 226,000 questions that had been 'resolved' (that is, had received at least one acceptable response).

We harvested 250 questions posed on a single day in September 2009, from the newly posted ('open') section of the Lyrics category. Twelve were discarded as duplicates and one discarded as off topic, leaving 237 questions for analysis. The average question length was approximately 58 words; the longest question contained 291 words (a request for an explanation of a song's meaning, including the full lyrics), and the shortest a mere 7 ('*What are some of.....? your favourite lyrics?*'). By contrast, audio queries to conventional search engines are far more brief (eg, [12] report an average of 3.1 terms in a 2006 study of the metasearch engine Dogpile).

Grounded theory ([11]) was used to develop categories to elicit characterizations of the desired outcome for the queries (Section 4) and the information features provided by the poster (Section 5). Initial categories were established by bringing together features from previous studies of natural language music–related questions (eg, [1], [3], [7]). These categories were regarded as tentative and were revised based on examination of the Yahoo! Answers Lyrics queries. An iterative coding process was employed, continuing until the two researchers agreed on both the coding categories and the codes assigned to each question.

4 Characterizing the desired outcome

At this point, we examine the types of music information that the posters have specified that they would like to receive as a response to their question—that is, the types of music document or details that they are seeking (Table 1).

Category	No. of queries	% (of 237)
Lyrics	51	21.6%
Metadata	95	40.3%
Identification	36	15.3%
Copy	6	2.5%
Example of type	16	6.8%
Explanation	16	6.8%
Feedback	18	7.6%
Creative Practice	7	3.0%
Other	7	3.0%

Table 1. Desired responses to questions

- *Lyrics*: requests for the complete lyrics to a song, or for specific lines (sometimes in a specific performance of a song)
- *Metadata*: requests for the title of a song and/or its artist / composer ('who it's by').
- *Identification:* questions asking some variation on 'what is this song?' without further specification of the desired result.
- *Copy*: requests to obtain a copy of an audio or video version of a song (by downloading or streaming).
- *Example of type*: requests for a song that fits into a specified category or genre (eg, a 'love song').
- *Explanation*: requests for 'the meaning' of a song and/or portions of the lyrics
- *Feedback*: the question solicits feedback on original song lyrics.
- *Creative Practice*: requests for technical or creative process information to be used in creating new songs.
- *Other*: questions that fall outside the above categories.

A close examination of the questions and their posted answers indicates Metadata and Identification can be collapsed into a single category; the desired result in both is a single song matching the given

criteria, with title and/or artist provided as a response. The Copy category is obviously closely related; a link to a song's audio will allow the poster to verify whether that song is indeed the requested music, and further the site hosting the audio (eg, YouTube) commonly includes music metadata such as title and author. A further breakdown of the 95 Metadata requests indicates that the title is the primary identifier for a song: 91 questions request a title, 25 ask for both title and artist, and 4 request the artist only.

The next largest category is that of requests for full or partial Lyrics for a specific song—the only surprise being that this is not the largest category, given that the poster has explicitly tagged the question as Lyrics focused. Most Lyrics requests appear to assume that there is only one set of lyrics for a song—they ask for '*the words*' or '*the lyrics'*.

For a minority of the Lyrics requests, the lyrics desired are to a specific performance or version of a song and so may not necessarily be the authoritative lyrics (eg, one question presents an audio link and asks, *'Can anyone decipher the lyrics up to the 25th second? plz? i am a nice guy?'*). There may not even exist an authoritative version for some songs, or portions of a song: for example, a 'freestyle' improvisation (*'the song is called "close my eyes" by Matisyahu. I can not find the lyrics for the freestyle he does in the middles of the song'*). Some queries explicitly request non-authoritative versions of the lyrics: for example, *'what's the lyrics of the song paradise by the kpop group "the melody"? the english translation, hangul and romanization please!*' The goal of existing work on identifying multiple sets of lyrics for a single song [6] is to identify the authoritative version and eliminate 'mistakes' in other lyrics; these queries suggest that alternative lyrics should not necessarily be rejected, and that the identification of different versions may be more difficult than previously anticipated (for example, in matching translations to the original).

Example of type questions are not answered by a specific song (a 'known item search'), but instead seek to elicit one or more songs that match a type description. Picking out an answer from a set of potential matches is problematic; the standard default for a music retrieval system is to present textual metadata (eg, song title and artist), which is unlikely to convey the point of similarity between a song and the type description (eg, *'A Happy optomistic, catchy song'*). Providing appropriate support for browsing remains an open problem in music retrieval; coming to a deeper understanding of the song facets that are used to judge a match is required to drive interface development (eg, tempo? lyrics? affect?).

Explanation questions *('What Is This Song About?'*) require a deep understanding of the semantics of the lyrics, and are unlikely to be addressable by automated retrieval systems. Similarly, requests for *Feedback* and critique of original lyrics written by the poster and assistance in the *Creative Practice* of creating audio are well beyond the capacities of existing digital libraries. However, these questions highlight that a great deal of music behavior is embedded in a social context—we listen to music at social gatherings, talk about the latest hits in casual conversation, and play songs on the radio or a CD as we drive. It seems appropriate that a music retrieval system should support music experts, aficionados, and keen novices in discussion and in community-based reference services—that the vision of a music digital library could include people as well documents and software.

5 Characterizing the information features provided

The features or characteristics used to describe the 204 Lyrics, Bibliographic Details, Copy, Identify, Explanation, and Example queries are as follows:

Category	No. of queries	% (of 204)
Lyric fragments	113	47.9%
Storyline	24	10.2%
Video references	18	7.6%
Metadata	95	40.3%
Genre/Style	42	17.8%
Orchestration	30	12.7%
Similarity	11	4.7%
Where heard	50	21.2%
Undesired result	7	3.4%
Other	2	0.8%

Table 1. How the information needs are described

- *Lyric fragments*: the remembered portions of a desired song.
- *Storyline* or message: a description of 'what happens' in a song, or a message conveyed by the song (eg, 'I love her and miss her').
- *Video references*: details about a video including the desired song (most frequently a music video for the song itself), provided either as a link to a video file or as a text description of the action occurring in the video.
- *Metadata*: bibliographic details, further broken down into Title, Artist, Collection Title, Date, Remix, and Tempo.
- *Genre or style*: can be a standard genre such as R&B, or a genre constructed by the poster (eg, 'contemporary, modern').
- *Orchestration:* an indication of the instruments and vocal parts in a recording.
- *Similarity:* another song or an artist that is similar to the desired song(s).

- *Undesired result*: another song, artist, or performance that is not the desired result.
- *Where heard:* the circumstances in which the poster heard a song performance or broadcast.

Finding a song based on the lyrics can be surprisingly difficult. Frustratingly, a person may remember the *Storyline* or gist of the song but not recall any of the lyrics themselves *('the the song talks about hating someone so much they wish they would some how die')*. The lyrics for a song can be difficult to understand as sung, making it difficult to construct a text search based on the known partial lyrics ('*I have NO CLUE what ANY of the lyrics are except two words because I saw someone mouth them while the song was playing behind me… All I know is in the chorus it's "something something git'cha git'cha"'*). A related difficulty is the mondegreen—a misheard lyric that may seem plausible but is incorrect ('*someone in the backmground singing fly high or sky high or something like that*'). It can be difficult to decide how to enter lyrics as search terms; should "*git'cha git'cha*" be entered written? As *Get Ya Get Ya*? *Get You Get You*? *Gitcha Gitcha*? Moreover, some lyrics are not dictionary words ('*Cannot remember any of the lyrics for the life of me besides the chorus lyics which simple go: ooo ooo OOoo ooo ooo, ooo ooo OOoo ooo ooo (repeat)*'). These problems push conventional IR matching techniques such as latent semantic analysis to their limits and beyond.

A word or phrase in the lyrics may be too common to be helpful in constructing a search, but the manner in which it is sung can be distinctive enough to be useful (*"Free-ee-e-e-ee"*). Combining facilities for text and 'sung' audio in a query would neatly solve this problem (eg, [9]).

Posters are sometimes able to point to songs *Similar* to the desired result, or conversely to indicate songs that are known to not be an answer to the question ('*its definitely not Land of 1000 dances'*). Facilities for indicating closeness/distance of results to an exemplar would be useful for these queries and also to represent a song's degree of membership in a *Genre*.

Metadata provided is frequently tentatively presented as likely to contain errors ('*im not sure of the name of it i believe it's called "spirit"*'; '*i think it is by nirvana or rhcp or something like that*')—understandably, since if the person had the correct metadata then they could answer their question themselves. The challenge for a retrieval system is to gracefully identify similar values to those suggested, for query refinement or ranking of results (eg, terms related to spirit, groups whose music is similar to that of Nirvana or the Red Hot Chili Peppers).

Where the poster heard the song might be useful in answering the question ('*What was the song played at the end of GH on 9/22/09?*')—or it might not ('*What's the name of a song I heard at Red Lobster?*'). Again, this type of detail suggests the value of a community-based answering service to work with heavily context dependent questions.

6 Conclusions

This paper analyzes a set of Lyrics-related questions to tease out the types of details presented to describe the information need (Section 5) and the expected responses (Section 4); the findings can inform further music retrieval research and development by suggesting new directions in search facilities, browsing structures and interfaces, and document representation.

References

[1] D. Bainbridge, S.J. Cunningham, and J.S. Downie. How people describe their music information needs: a grounded theory analysis of music queries. In *4th International Conference on Music Information Retrieval (ISMIR)*, Baltimore, Maryland, 2003.

[2] Cunningham, S.J., Bainbridge, D., Falconer, A. More of an art than a science: playlist and mix construction. In *International Conference on Music Information Retrieval (ISMIR '06)*, Vancouver.

[3] J.S. Downie and S.J. Cunningham. Towards a theory of music information retrieval queries: system design implications. In *3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.

[4] J.S. Downie. Music Information Retrieval. *Annual review of information science and technology*, Volume 37, pages 295-340, 2003.

[5] M. Itoh. Subject search for music: quantitative analysis of access point selection. In *1st Annual International Symposium on Music Information Retrieval*, Amherst MA, 2000.

[6] P. Knees, M. Schedl, and W. Gerhard. Multiple lyrics alignment: automatic retrieval of song lyrics. In *6th International Conference on Music Information Retrieval (ISMIR '05)*, pages 564-569, London, UK, 2005.

[7] J.H. Lee, J.S. Downie, and S.J. Cunningham. Challenges in cross-cultural / multilingual music information seeking. In *6th International Conference on Music Information Retrieval (ISMIR '05)*, London, UK, 2005.

[8] J.R. MCPherson and D. Bainbridge. Usage of the MELDEX digital music library. In *2nd Annual International Symposium on Music Information Retrieval*, Bloomington IN, pages 19-20, 2001.

[9] M. Muller, F. Kurth, D. Damm, C. Fremery, and M. Clausen. Lyrics-based audio retrieval and multimodal navigation in music collections. In *European Conference on Digital Libraries 2007*, pages 112-123, Berlin, 2007.

[10] R. Neumayer and A. Rauber. Integration of text and audio features for genre classification in music retrieval. In *29th European Conference on Information Retrieval*, pages 724-727, Rome, Italy, 2007.

[11] A. Strauss and J. Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage, 1990.

[12] D. Tjondronegoro, A. Spink, and B.J. Jansen. Multimedia web searching on a meta-search engine. In *12th Australasian Document Computing Symposium*, pages 80 – 83, Melbourne, Australia, 2007.

[13] B. Wingenroth, M. Patton, T. DiLauro. In *ACM/IEEE Joint Conference on Digital Libraries '02*, pages 308-309, Portland, Oregon, 2002.

Positive, Negative, or Mixed? Mining Blogs for Opinions

Xiuzhen Zhang, Zhixin Zhou, Mingfang Wu
School of CS & IT, RMIT University
GPO Box 2476v, Melbourne 3001, Australia
{*xiuzhen.zhang, zhixin.zhou, mingfang.wu*}*@rmit.edu.au*

Abstract *The rich non-factual information on the blogosphere presents interesting research questions. In this paper, we present a study on analysis of blog posts for their sentiment by using a generic sentiment lexicon. In particular, we applied Support Vector Machine to classify blog posts into three categories of opinions: positive, negative and mixed. We investigated the performance difference between global topic-independent and local topic-dependent opinion classification on a collection of blogs. Our experiment shows that topic-dependent classification performs significantly better than topic-independent classification, and this result indicates high interaction between sentiment words and topic.*

Keywords blog, sentiment analysis, opinion classification, opinion words, information retrieval

1 Introduction

With the wide availability of broadband network facilities, internet has become an indispensable channel for people to communicate. More and more people are publishing their own experience and opinions, as well as seeking other people's opinions. With the explosive amount of information generated daily, it is almost impossible for people to read through all the information even on a narrow topic. This demands for new techniques to help track sentiment trends and search for various opinions, a task that is very different from factual information task as in traditional information retrieval; and sentiment analysis is a key component of such techniques.

Sentiment analysis is the technology to evaluate a text and predicate the text's subjectivity (subjective versus objective) and/or sentiment (positive versus negative). A general approach is to find out those keywords from the text that are of evaluative feature or sentiment orientation to represent the text, and to use a classification method to predicate the text's probability of belonging into pre-defined categories; the classifier is usually trained on a set of labeled texts.

The above approach has shown success in some earlier work where sentiment analysis was used to reviews from a certain domain, such as a movie review or a product review [8], while less success was seen in those studies conducted within the TREC (Text REtrieval Conference) Blog Track on polarity search task [6]. On the other hand, existing studies have also shown that mixed sentiment is especially challenging [5] given its uncertainty in nature.

The TREC Blog Track's polarity task is to identify the polarity of the opinions in the retrieved documents (blogs) in respond to a search topic. A problem with the evaluation of this task is that sentiment analysis is mingled with topic search and rank task, as a result, it is hard to ascertain the effectiveness of a certain sentiment analysis method.

This paper presents a study on analysis of blog posts for their sentiments, or opinions. Specifically a blog post is analyzed and classified into three categories: positive sentiment, negative sentiment or mixed sentiment. We adopt a dictionary-based approach by using a generic sentiment lexicon developed by a linguistic study [12]. We propose to represent blog posts as bags of sentiment words and use the Support Vector Machine (SVM) [3] learning model to classify blog posts. Given that both the sentiment lexicon and the classification model are generic, our research question is: if a *global* classification of blog posts accross topic genres would achieve a similar performance as a *local* classification of blog posts of a certain topic genre.

The remainder of this paper is organised as follows. We review some related work in Section 2 and describe the sentiment lexicon used in this study in Section 3. We present our classification approach in Section 4, and experiment setup and evaluations in 5. We discuss the experiment result and conclude the paper in Section 6.

2 Related Work

Sentiment analysis was initially applied to a corpus of documents that are from the same genre, such as a corpus of movie reviews or a corpus of product review. The task of sentiment analysis is to specify if a document (or a review) expresses a positive or negative opinion. Naturally most studies adopted machine learning classification approaches [1, 8, 11]. Pang et al [8] applied and compared three machine learning methods, naive bayes, maximum entropy and support vector machines,

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

on a corpus of movie reviews with uniform class distribution. Their results showed that the support vector machine model generally performed the best.

While most sentiment analyses classify comments or documents into two categories: positive versus negative, Koppel and Schler [5] argued that there were other comments that might express a mixed or neutral sentiment. Their study showed that by incorporating neutral category can lead to significant improvement in overall classification accuracy, and this is achieved by properly combining pairwise classifiers.

With so many opinionated documents available on the Web, people are actively seeking other people's opinion toward a certain topic. In this case, we need to do more than sentiment analysis: we need first to retrieve a set of documents that are about the topic, then judge if a document indeed contains any opinion at all - so called *subjectivity analysis*, then analyse if a subjective opinion or sentiment is positive, negative, or mixed. Such an opinion polarity finding task was introduced in TREC 2007 conference [6]. A commonly adopted approach by participants is to use baseline search engines to search topic-relevant documents first, and then use polarity-finding heuristics to re-rank documents for polarity. Machine learning models have not been widely used to improve the polarity classification accuracy.

3 A Generic Sentiment Lexicon

Identification of sentiment words is fundamental to sentiment analysis and classification. There are two broad methods to identify sentiment words and build sentiment lexicon. One method is through manual construction in which annotators manually annotate a list of words or phrases [9] or find and annotate sentiment words from a given corpus [8, 12].

Another method is to build a lexicon from a small number of seed words with pre-determined sentimental polarity, and then populate the seed list through learning or other relationships. For example, Hatzivassiloglou and McKeown [2] expanded a seed list by adding those words that are linked to seed words through conjunction such as *and, or, but, either-or*, or *neither-or*; while Kim and Hovey made use of WordNet to populate seed words through synonym and antonym relationships [4].

In our study, we use the sentiment lexicon developed by Wiebe et al. [12]. This lexicon list has 8221 annotated words resulted from manual annotation of a 10,000-sentence corpus of news articles of various topics. The following is an example of such an annotation:

type=strongsubj len=1 word1=admire pos1=verb stemmed1=y priorpolarity=positive

The property *prior polarity* indicates the attitude being expressed by the word *admire* and has three values: *positive*, *negative* and *neutral*. The neutral tag are those subjective expressions that do not have positive or negative polarity. The property *type* indicates the expression intensity and here it has binary values: *strong* or *weak*. As annotation was done within context of a sentence, the grammar function of a word is also annotated, for example, the word *admire* here is a verb. Thus a word may occur twice or more in the list depending on which grammar function a word acts in the original text for annotation, for example, the word "cooperation" is annotated as *adjective* and *none*. This list also includes words with multiple morphemes, for example, cooperate, cooperation, cooperative, and cooperatively.

4 Opinion Classification

This section presents our classification method.

4.1 Support Vector Machine

Support Vector Machine (SVM) has been widely used in text categorisation, and with reported success [3]. In an SVM model, objects are represented as vectors. In learning a model to classify two classes, the basic idea of SVM is to find a hyperplane, represented by a vector, that separates objects of one class from objects of other classes at a maximal margin. When using a linear kernel, SVM learns a linear threshold function. With polynomial and radial basis kernels, SVM can also be used to learn polynomial and radial basis classifiers.

SVM$_{multiclass}$ [1] is an implementation of the multiclass SVM, and is based on Structural SVMs [10]. Unlike regular SVMs, structural SVMs can predict complex objects like trees, sequences, or sets. SVM$_{struct}$ can be used for linear-time training of binary and multiclass SVMs under the linear kernel. Features extracted jointly from inputs and outputs are used to form an optimal separation plane.

4.2 Opinion Word Extraction

To apply a classification model effectively, a key issue is feature selection, i.e. what input will be given to a classification model. The feature selection is application dependent - how do we want to classify a set of documents, and what are prominent features from a set of documents that can separate them from each other. For the sentiment classification task, it is intuitive that we identify those opinion words from a set of documents as classification features.

In this study, we simply treated opinion words as tokens and do not apply natural language processing methods such as Part-Of-Speech tagging to analyse the grammatical function of those words. We applied Porter stem method to the list and group different forms of the same word, and this leaves us 4919 "words".

A closer look at the stemmed opinion words reveals some interesting facts. There are 103 words that are of contradictory polarities. After we removed these words, we had 4816 words with unique sentiment

[1] Avaiable at `http://svmlight.jochims.org/svm_multiclass.html`

polarity. However, there are also some words that have mixed levels of strength. In lieu of this, we created a new level of strength and named it "contextual strength"; there are a total of 194 in this category. The distribution of opinion words in term of polarity and strength is summarised in Table 1.

	Positive	Negative	Neutral	Total
Strong	954	2061	107	3192
Contextual	81	98	14	194
Weak	544	783	163	1490
Total	1579	2942	284	4816

Table 1: Distribution of opinion words

4.3 Opinion Word Vectors

In information retrieval, each document is represented by all word tokens from a collection. However, for the purpose of opinion classification, we represent a document as a vector of opinion word tokens and ignore those words that do not express any sentiment. As in retrieval models, we weight each feature (an opinion word) of the document vector. The $tf \times idf$ weight of an opinion word f in a document d is:

$$w_{fd} = tf_{fd} \times log\frac{|D|}{|D_f|}$$

where tf_{fd} is the frequency of f in d. $|D|/|D_f|$ is inverse document frequency of f — $|D|$ is the number of documents in the collection, and $|D_f|$ is the number of documents containing f. We expect that this model is general enough to be applied to opinion classification.

5 Evaluation

5.1 Topic-independent versus Topic-dependent Classification

Opinion classification is usually applied to a set of documents that are of same genre or about a similar topic such as movie reviews and product reviews. With a huge number of opinionated documents on the Web and the nature of inexact match of a Web search engine, it is unlikely that we can always get a set of documents from the same genre to be classified. As a sentiment lexicon is independent of semantic topic of a document, we then investigate if there exists any difference between classification of documents that are about mixed topics and documents about a topic; we call these two types of document classification topic-independent (or global) classification and topic-dependent (or local) classification respectively.

5.2 Experiment Set-up

The TREC Blog track 2006 collection Blog06 [7] is a sample of the blogosphere crawled from 6 December 2005 to 21 February 2006. The collection is 148GB in total, and comprises three components: XML feeds of 38.6GB, which are the blogs, Permalink documents of 88.8GB, which are the blog posts with associated comments, and HTML homepages of 28.8GB, which are the entries to blogs. The permalink documents are the unit for the opinion finding task and polarity tasks.

Category	TREC-2006	TREC-2007
Negative	3,707 (32.15%)	1,844 (26.34%)
Mixed	3,664 (31.78%)	2,196 (31.37%)
Positive	4,159 (36.07%)	2,960 (42.29%)
Total	11,530	7,000

Table 2: Distribution of document categories in TREC-2006 and TREC-2007

The content of a blog post is defined as the content of the blog post itself and the contents of all comments to the post. A blog post is considered having subjective content if "it contains an explicit expression of opinion or sentiment about the target, showing a personal attitude of the writer" [7]. Fifty topics were selected by NIST from a collection of queries of a commercial search engine for the opinion retrieval task. For a topic, permalink documents are tagged with NIST relevance judgement, with the following categories (or scales) [7]: not judged(-1), not relevant(0), relevant(1), negative(2), mixed(3) and positive(4).

The Blog06 collection was used for both TREC-2006 and TREC-2007 Blog Track. Fifty (different) topics were used for each conference. For each topic, we selected documents with NIST assessor relevance judgement scale of 2 (negative), 3 (mixed - both positive and negative) and 4 (positive) for our study. Table 2 shows the distribution of documents in different categories in TREC-2006 and TREC-2007 respectively.

Zettair search engine [2] was used to index documents with the sentiment lexicon. Each document was converted into a vector of opinion words with the weighting scheme as described in Section 4.3.

5.3 Topic-independent Opinion Classification

To train the topic-independent opinion classification model, we pooled and indexed all documents from 50 topics in TREC-2006. SVM model was then trained on the converted opinion-word vectors with judgement scale $>=2$. Ten-fold cross validation experiment was conducted on all 10,737 documents of 50 topics. It showed an overall accuracy of 52.90$\pm$3%, that is 52.9% of documents correctly classified, with a standard deviation of 3%.

5.4 Topic-dependent Opinion Classification

To examine the interactions between topics and opinion classification accuracy, topics of TREC-2006 that contain at least 10 documents from each opinion category

[2] http:www.seg.rmit.edu.au/zettair/

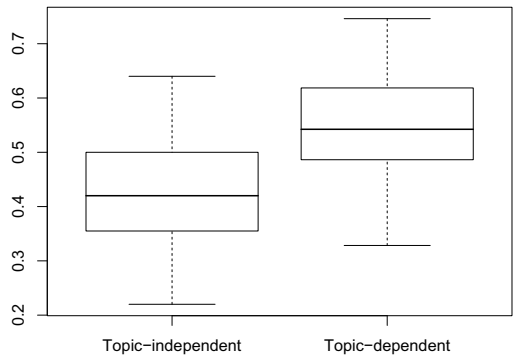


Figure 1: Classification accuracy: Topic-independent vs. topic-independent

(recall that there are 3 categories positive, negative or mixed) were extracted, this resulted in 36 topics and 9,771 documents in total.

To evaluate the accuracy of topic-dependent opinion classification, we individually indexed documents from the same topic, and applied ten-fold cross validation experiment to each topic collection accordingly. On average, the topic-dependent model achieved an accuracy of 63±13%, significantly higher than that achieved by the topic-independent model.

5.5 Blind Test of the Classification Model

The topic-independent classification model trained on documents with TREC-2006 judgments were blind tested on documents with TREC-2007 judgements. 27 topics that contain at least 10 documents in each category were used in our study. The model showed an accuracy of 42% on the whole collection. The drop in performance compared to that of 10-fold croass validation (52.9±3%) may be attributed to the change of topics between the two collections, which in turn suggests that there is strong correlation between topics and opinion words.

On the other hand, in the 10-fold cross validation experiment on the TREC-2007 collection, the topic-dependent model achieved an average accuracy of 55%. We extracted individual topic's accuracy for the topic-independent model, and used a paired Wilcoxon test to compare the difference in classification accuracy between the topic-independent model and the topic-dependent model. The improvement in classification accuracy of the topic-dependent model over that of the topic-independent model is statistically significant ($p < 0.001$). Figure 1 shows the summary of two models. As we can see that the topic-dependent model achieved higher accuracy than the topic-independent model.

6 Conclusion

In this paper we have described our research on opinion classification of blogs. We have investigated the difference of global classification of documents from mixed topics and local classification of documents from the same topic. Our experiment on the TREC Blog collections has shown that the local classification is significantly more accurate than the global classification. This might be because that documents from the same topic tended to have a similar set of sentiment words. Our future research will concentrate on developing topic-specific opinion classification models, especially it is anticipated that the annotation of opinion words tensity can be used to further improve such models.

Acknowledgements The authors would like to thank Steven Garcia for his various help with using Zettair.

References

[1] K. Dave, S. Lawrence and D. M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product review. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528, 2003.

[2] V. Hatzivassiloglou and K.R.McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of 8th Conference on European chapter of the association for computational linguistica*, pages 174–181, 1997.

[3] T. Joachims. Text categorisation with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98*, 1998.

[4] S. M. Kim and E. Hovy. Determining the sentiment of opinions. In *Proc. of the Coling Conference*, 2004.

[5] M. Koppel and J. Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, Volume 22, Number 2, 2006.

[6] C. MacDonald, I. Ounis and I. Soboroff. Overview of TREC-2007 blog track. In *Proceedings of TREC-2007*, Gaithersburg, USA, 2008.

[7] I. Ounis, M. de Rijke, C. MacDonald and I. Soboroff. Overview of trec-2006 blog track. In *Proceedings of TREC-2006*, Gaithersburg, USA, 2007.

[8] B. Pang, L. Lee and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[9] P. Subasic and A. Huettner. Affect analysis of text using fuzzy semantic typing. *IEEE Transactions on Fuzzy systems*, Volume 9, pages 483–496, 2001.

[10] I. Tsochantaridis, T. Hofmann, T. Joachims and Y. Altun. Support vector learning for interdependent and structured output spaces. In *Proc.: of ICML-2004*, 2004.

[11] P. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the ACL*, 2002.

[12] J. Wiebe, T. Wilson and C. Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, Volume 1, Number 2, 2005.