Modelling Anchor Text Retrieval in Book Search based on Back-of-Book Index

Hengzhi Wu
Queen Mary, University of London, UK
hzwoo@dcs.qmul.ac.uk

Gabriella Kazai
Microsoft Research Cambridge, UK
gabkaz@microsoft.com

Thomas Roelleke
Queen Mary, University of London, UK
thor@dcs.qmul.ac.uk

ABSTRACT

This paper proposes a probabilistic logic abstraction for modelling *tf*-boosting approaches to anchor text retrieval, adapted for the task of page-search in books. The underlying idea is to view the back-of-book index (BoBI) as a list of anchors pointing to pages in the book. First, we model the direct application of hypertext-based *tf*-boosting to books and show that this naive method of propagating anchor-text from the BoBI does not deliver the desired *tf*-boosting effect. To address this, we then propose a revised anchor-text retrieval model based on a novel voter approach. In this approach, each page of the book, where a given term occurs, acts as a virtual voter to the pages referenced by the BoBI for that term. The *tf*-boosting effect is achieved by propagating term weights from the voter pages to the pages in the BoBI. We use probabilistic Datalog for the high-level abstract modelling of retrieval strategies, which allows for the evolution and transfer of successful techniques from one domain, such as anchor-text retrieval in Web IR, to a similar domain, such as book search.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Algorithms

1. INTRODUCTION

As a result of ongoing digitization and mass-digitization projects around the world, searching over a large collection of digitized books has become a new area of interest in the field of IR. Reflecting this, the INitiative for the Evaluation of XML IR (INEX) has launched a Book Search track[1] in 2007. The track runs two core tasks: 1.) The Book Retrieval task, where the units of retrieval are whole books; and 2.) The Page in Context task, which studies the application of passage and XML element retrieval methods to books. Both tasks have been shown to benefit from the use of structural information extracted from digitized books [11, 15, 14, 20]. In this paper, we focus on the Page in Context task, and view the challenge of locating relevant pages inside books as a task of finding the best entry points into the book content.

We build on the observation that books are typically highly structured and, in particular, we leverage the potential presented by a book's back-of-book index (BoBI) as a searching and browsing tool. Unlike the table of contents, a BoBI provides specific information on relevant sections of text by pointing to key concepts discussed in the book [1, 8]. Its significance is that it distinguishes important topics and concepts, as identified by the author, from other keywords that may appear in the book.

In contrast to the traditional full-text index used in IR, which reflects the global distribution of terms (among pages), but provides limited evidence for selecting best entry points (e.g., best pages), an author generated BoBI may hold the key for identifying the best entry points. Although some term statistics may also be derived from a BoBI (e.g., number of linked pages), the full-text index is still likely to provide a good source of information for estimating relevance. Thus, in this paper, we propose a strategy for combining the traditional full-text index with the BoBI by integrating anchor text retrieval in the context of the page finding task.

The underlying idea is to view the BoBI as a set of anchors pointing to respective parts of the text in the book. Given this view, the question of how to best utilise this source of information for the identification and retrieval of relevant pages inside a book arises.

Term propagation mechanisms in the form of anchor text or hypertext retrieval have been successfully applied in Web IR [2], and have also been found to be beneficial for enterprise [9], and XML retrieval [19]. The underlying principle of term propagation (also referred to as context augmentation) is the use of terms from linked source documents in the representation of a destination document. A link may be a structural link between components in a document hierarchy, or a hyperlink connecting two web pages, for example. A consequence of term propagation is that the destination document receives a term frequency boost, which can then lead to a higher retrieval score being assigned to the document by a search system.

In this paper, we investigate the modelling of term propagation strategies adapted to the task of book search, exploiting the BoBI as a list of anchor texts linking to pages in the book. We employ probabilistic Datalog (PD) [4], an expressive high level language, for modelling information retrieval taking into account the specifics of book search.

[1] http://www.inex.otago.ac.nz/tracks/books/books.asp

SIGIR 2008 Workshop on Focus Retrieval
July 24, 2008, Singapore

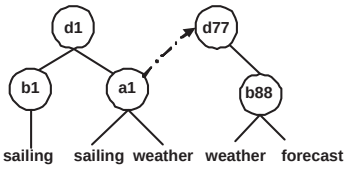

Figure 1: Structured documents with hyperlink

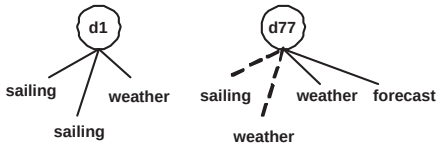

Figure 2: Augmentation through hierarchy and hyper-link

The two main contributions of this paper are: 1.) The modelling of *tf*-boosting strategies in PD, and 2.) The proposal of the voter model for *tf*-boosting.

The paper is structured as follows. Section 2 introduces anchor text retrieval and modelling in PD. Section 3 discusses why the classic anchor text retrieval strategy fails in book search, and introduces a voter model for the task. Section 4 formally defines anchor text retrieval in book search, modelling the strategies in PD. Finally, section 5 summarises the paper and outlines future work.

2. BACKGROUND

In this section, we present two examples of anchor text retrieval applied to structured and hyperlinked documents, and review the basic principles of probabilistic modelling in PD.

2.1 Context Augmentation

Term propagation in anchor text retrieval is an application of context augmentation [13, 16], where the occurrence of a term in the anchor text (of the source document) is propagated to the destination document via a structural or hyperlink that connects the two documents. As a result, the content of the destination document is augmented and the term frequencies (*tf*) of terms in the destination document are increased.

We present two context augmentation mechanisms, depending on the type of linkage between documents. We illustrate these using the small sample collection of two documents, d_1 and d_{77}, shown in Figure 1. Each document in the collection is a tree with nodes and edges. The nodes of a tree may be composite elements, which contain other nested nodes, and leaf nodes, which contain the text of the document. Nodes are labelled based on the following convention: "d" represent the root node of a document, "b" is short for body, and "a" indicates an anchor. A solid line between nodes represents a structural link and a dash-dot arrow from an anchor element implies a hyperlink.

2.1.1 Hierarchical augmentation

Hierarchical augmentation is typically applied to structured documents (e.g., in XML IR), where the occurrence of a term in a child-node is propagated to the representation of its parent node. In this case, the term frequency is propagated upwards along the structure of the document tree. One reason to do this is to support the retrieval of composite nodes, which may provide a better match than their child nodes alone (e.g., when one query term is contained in one child node and another query term in another child node) [5].

For example, in Figure 1, we obtain the augmented representation of d_1 by propagating the *tf*'s of the terms that occur in the child nodes: b_1 and a_1. We use the notation $n_L(t, c)$ to denote the *tf* of a term t in a context c, where n_L stands for number of locations. The term frequencies in the child nodes are thus $n_L(sailing, b_1) = 1$ and $n_L(sailing, a_1) = 1$, and $n_L(weather, a_1) = 1$. After augmentation, we obtain the following *tf*'s in d_1: $n_L(sailing, d_1) = 2$ and $n_L(weather, d_1) = 1$. Augmentation, thus, allows for d_1 to act as a combined representation of its child nodes. This is illustrated in the left side of Figure 2.

One typical question in hierarchical augmentation concerns whether to propagate raw term frequencies or apply some kind of normalization or weighting [17]. In this paper, we model both raw and weighted term frequency propagations.

2.1.2 Anchor-text-based augmentation

Applying augmentation in a hyperlinked environment allows to combine evidence from external documents for the representation and retrieval of a hyperlinked document.

Tf-boosting is one such strategy, which directly propagates and combines term frequencies into the representation of the destination document. Anchor text retrieval using *tf*-boosting have in fact been shown to improve retrieval performance in Web IR [2]. Furthermore, *tf*-boosting was found to be more effective than other link-based strategies in Web IR [10].

We illustrate *tf*-boosting using the sample collection of Figure 1. Here, the anchor element a_1 has a hyperlink pointing to document d_{77}. Assuming that the representation of d_{77} already contains the terms "weather" and "forecast" as a result of hierarchical augmentation from b_{88}, we start with the following *tf*'s in d_{77}: $n_L(weather, d_{77}) = 1$ and $n_L(forecast, d_{77}) = 1$. The terms "sailing" and "weather" in a_1 are terms of the anchor text, which is often interpreted as a description of the destination document. By propagating the terms of the anchor text to the destination document, we can obtain a *tf*-boosted representation of d_{77}, where the *tf* of "weather" is boosted from 1 to 2. In addition, the term "sailing" which did not occur in d_{77} originally, is now also represented. After propagation, the term frequencies for d_{77} are: $n_L(sailing, d_{77}) = 1$, $n_L(weather, d_{77}) = 2$, and $n_L(forecast, d_{77}) = 1$. This is illustrated in the right hand side of Figure 2.

2.2 Modelling in Probabilistic Datalog

We model retrieval strategies in probabilistic Datalog (PD) [4]. PD is a combination of deterministic Datalog (a query language used in deductive databases) and probability theory. It is a highly expressive and flexible platform for modelling and prototyping different retrieval strategies.

To introduce PD, we implement the example term propagation strategies discussed in Section 2.1. First, we create the following relational structures to store the necessary information about the collection (see Figure 4): The `term(Term,Context)` relation is used for storing term occurrences along with their corresponding location information, e.g., the term "sailing" occurs in the anchor element a_1. The location information is stored as

```
pdRule := {assumption} goal ':-' body
assumption :=
    'DISJOINT' | 'INDEPENDENT' | 'SUBSUMED' |
    'SUM' | 'MAX'
goal ::= NAME {assumption} '(' varList ')'
body ::= subgoalList
varList := var {',' varList}
var := VARIABLE
subgoalList := subgoal {'&' subgoalList}
subgoal := NAME '(' argList ')' {'|' evidenceKey}
argList := arg {',' argList}
arg := VARIABLE | constant
evidenceKey := '(' varList ')'
```

Figure 3: Extract of PD Syntax

term	
Term	Context
sailing	d1/b1
sailing	d1/a1
weather	d1/a1
weather	d77/b88
forecast	d77/b88

part_of	
Child	Parent
d1/a1	d1
d1/b1	d1
d77/b88	d77

link	
Context	URL
d1/a1	www.bbc.co.uk

site	
Context	URL
d77	www.bbc.co.uk

Figure 4: Relations of link-based retrieval

a path, expressed in XPath [2], however, for readability we use a simplified syntax throughout this paper, e.g., "d1/a1". The hierarchical structure of documents in the collection is represented in the `part_of(Child,Parent)` relation, e.g., "d1/a1" is a part of "d1". The hyperlink structure is modelled through two relations: `link(Context,URL)` and `site(Context,URL)`. The `link` relation associates the XPath of an anchor node with the URL of the destination, while the `site` relation maps the URL to the destination document.

The syntax of our PD is given in Figure 3. In this syntax, everything between a pair of curly brackets, i.e. '{' and '}', is optional; the assumption 'SUM' is an alias of 'DISJOINT' hence the two are interchangeable.

A PD rule consists of a goal, a body and arguments. A rule is evaluated such that the goal is true if and only if the body is true. For example, the following rule demonstrates a common term matching strategy in IR, stating that if T occurs in a query and T is a term in a document D, then D is retrieved.

```
retrieve(D) :- query(T) & term(T, D).
```

The weights (probabilities) associated with the results of a rule are obtained through probability computations, which involve probability estimation and probability aggregation. Probability estimation assigns initial probabilities to the rows of a relation; the probabilities may be term-based, i.e. *tf*, or document-based (or element-based in XML), i.e. *df*. Probability aggregation, on the other hand, combines probabilities of one or multiple relations, e.g., by summation or multiplication.

[2]http://www.w3.org/TR/xpath

term(T,S) & part_of(S,D)			
Term	Context	Child	Parent
sailing	d1/b1	d1/b1	d1
sailing	d1/a1	d1/a1	d1
weather	d1/a1	d1/a1	d1
weather	d77/b88	d77/b88	d77
forecast	d77/b88	d77/b88	d77

(a) intermediate result of rule body

augTerm	
Term	Parent
sailing	d1
sailing	d1
weather	d1
weather	d77
forecast	d77

(b) hierarchical augmentation

augTerm_d		
$1/N_L(d)$	Term	Parent
1/3	sailing	d1
1/3	sailing	d1
1/3	weather	d1
1/2	weather	d77
1/2	forecast	d77

(c) probability estimation

w_augTerm		
$P(t\|d)$	Term	Parent
2/3	sailing	d1
1/3	weather	d1
1/2	weather	d77
1/2	forecast	d77

(d) probability aggregation

Figure 5: Intermediate results of hierarchical augmentation

For instance, to estimate the within-document *tf*'s of a term T given that it occurs in document D, we have the following rule:

```
w_term(T, D) :- term(T, D) | (D).
```

The rule contains a special form of our probabilistic Datalog variant: The "| (D)" is the so-called evidence key, so that the tuples in the goal "w_term" (stands for weighted term) have the probabilistic semantics $P(T|D)$. We will return to this in more detail in sections 4.2 and 4.3. Further information about PD can be found in [4, 7, 18].

2.2.1 *Hierarchical augmentation in PD*

Let us now present a model of augmentation in PD. First, we give the PD rules for the hierarchical augmentation described in Section 2.1.1:

```
augTerm(T, D) :- term(T, S) & part_of(S, D).
augTerm_d(T, D) :- augTerm(T, D) | (D).
w_augTerm SUM(T, D) :- augTerm_d(T, D).
```

The first rule yields the augmented term frequency for the term T in document D by evaluating the predicates that T occurs in a context S, and that the context S is part of a document D. The second rule estimates the *tf* for `augTerm`. Finally, the third rule aggregates the probabilities and yields the final results.

Figure 5 illustrates the steps for processing this hierarchical augmentation rule. Figures 5(a) and 5(b) show the intermediate and final results of the first rule; Figure 5(c) shows the result of the probability estimation, i.e. the second rule; and Figure 5(d) shows the results of the probability aggregation, i.e. the third rule.

2.2.2 *Anchor-text-based augmentation in PD*

The PD rules for anchor text retrieval with *tf*-boosting presented in section 2.1.2 are given as follows:

```
augTerm(T, D) :- term(T, S) & part_of(S, D).
augTerm(T, D) :- term(T, A) & link(A, U) & site(D, U).
augTerm_d(T, D) :- augTerm(T, D) | (D).
w_augTerm SUM(T, D) :- augTerm_d(T, D).
```

term(T,A) & link(A,U) & site(D,U)					
Term	Context	Context	URL	Context	URL
sailing	d1/a1	d1/a1	www.bbc.co.uk	d77	www.bbc.co.uk
weather	d1/a1	d1/a1	www.bbc.co.uk	d77	www.bbc.co.uk

(a) intermediate result of rule body

augTerm	
Term	Parent
sailing	d77
weather	d77

(b) anchor-text-based augmentation

augTerm	
Term	Parent
sailing	d1
sailing	d1
weather	d1
weather	d77
forecast	d77
sailing	d77
weather	d77

(c) Union of hierarchical and anchor-text-based augmentation

augTerm_d		
$1/N_L(d)$	Term	Parent
1/3	sailing	d1
1/3	sailing	d1
1/3	weather	d1
1/4	weather	d77
1/4	forecast	d77
1/4	sailing	d77
1/4	weather	d77

(d) probability estimation

w_augTerm		
$P(t\|d)$	Parent	Term
2/3	sailing	d1
1/3	weather	d1
1/2	weather	d77
1/4	forecast	d77
1/4	sailing	d77

(e) probability aggregation

Figure 6: Intermediate results of tf-boosting

The first rule performs hierarchical augmentation to obtain the original within-document *tf* of terms. The anchor text based *tf*-boosting is performed by the second rule in two steps: 1.) The propagated term frequency for the term T in document D is obtained by evaluating the predicates that the term T appears in a context A, where A has a hyperlink to a URL U, which is associated to a document D; and 2.) The propagated *tf* is combined with the original *tf* of term T in document D. The third and fourth rules estimate and aggregate probabilities, respectively, to yield the final probability scores.

The procedures for processing the *tf*-boosting rules (rules 2-4) are illustrated in Figure 6. Figures 6(a) and 6(b) show the intermediate results for obtaining the propagated term frequencies, i.e. the first step of *tf*-boosting; Figure 6(c) shows the combination of the boosted and the original *tf*'s, i.e. the second step of *tf*-boosting. Figures 6(d) and 6(e) show the results of the probability estimation and aggregation processes.

3. BACK-OF-BOOK INDEX (BOBI)

As mentioned earlier, the rich structure of books is viewed as potentially useful for improving the retrieval effectiveness of book search approaches. Generally, the table of contents (TOC) and the BoBI are regarded as the two main sources of information to aid readers in locating relevant content in books [12]. While the benefits of a hyperlinked TOC's seem obvious, the use of the BoBI in book search tasks is less trivial. In this section, we investigate and discuss the applicability of BoBI for page-level search in books.

3.1 TF-Boosting and the BoBI: Discussion

A BoBI is, in essence, a cross-reference lockup mechanism aimed to support readers of printed books to locate information quickly using keywords as their entry into the content of the book. A BoBI is usually created by the author (or editor)[3] of the book, and as such, it reflects a somewhat personalised view on what the important terms/concepts of a book are. On the other hand, it represents a relevance association in the form of $\langle term, page \rangle$ pairs, similar to an inverted index used in IR. Furthermore, it can also be likened to

[3]Although automatic index generation tools may also be used.

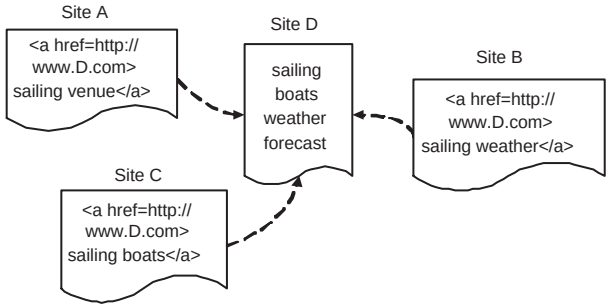

Figure 7: Anchor text topology in Web IR

hyperlink structures of the Web, whereby the anchor text of a BoBI term points to a content page.

Inspired by the successful application of anchor text retrieval in Web IR, one may attempt to incorporate the same mechanisms into book search by applying them to the BoBI of books. However, as we demonstrate next, such a direct application of the technology is not suitable for books.

Let us illustrate this with an example, contrasting the link structure of the Web and that of a book. Figure 7 shows three web pages A, B and C, each of which links to a fourth D page. Anchor text retrieval here is based on the principle of exploiting the in-link structure of page D's Web graph and propagating terms from the source pages to D. The more in-links to a page, the more terms are propagated and the stronger the effect of *tf*-boosting.

In a book, the BoBI contains out-links to pages of the book, where index terms occur (see Figure 8). In contrast to the Web scenario, however, the referred pages are likely to receive only a small number of in-links from the BoBI (one in-link per referred page is the most likely case). This means that the boost on referred pages is likely to have little effect. To address this, we propose in the next section a modification to the classic anchor text retrieval strategy and adopt this new voter model approach for the task of page retrieval in books.

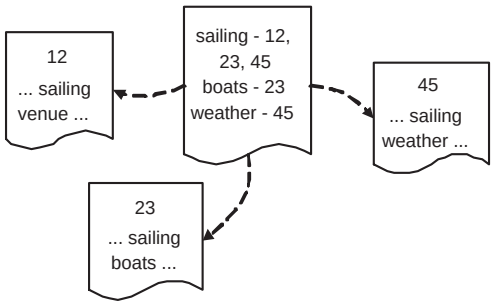

Figure 8: Anchor text topology in book search

3.2 TF-Boosting and the BoBI: A Voter Model

The occurrence of a term in the BoBI of a book implies that it is considered (by the author) of more importance than the rest of the terms on the referenced page that do not appear in the BoBI. Intuitively, the pages 'cited' in the BoBI could be considered as "best entry points" for looking up information in the book. This is further supported by the observation that topics (and alas associated keywords) spanning a number of pages are often only listed in the BoBI once, directing the reader to the first page in a sequence of pages on the topic. To incorporate this characteristic of the BoBI within a retrieval strategy, we employ *tf*-boosting as a method for combining traditional retrieval strategies based on term weights with adapted anchor text retrieval based on the BoBI.

Assuming that the BoBI terms are exact copies of the terms that occur in the referenced pages [4], the BoBI can be considered a subset of a full-text index (FTI) generated at the page-level (i.e., using pages as the document units). Given the intersection of FTI and BoBI, the terms common to both sets could be viewed as providing an association between the pages in the FTI and the pages in the BoBI, each shared term acting as a kind of anchor text link. Based on this view, we can consider each page associated with a given term in the FTI as a virtual voter to the pages referenced by the BoBI: by propagating term weights to the pages in the BoBI, each voter thereby votes for the "best entry points" in the book.

A range of possible options exist for deriving a suitable voting weight, i.e. the propagated term weight. One option is to redistribute the terms in the FTI over the page frequency of terms in the BoBI, i.e. $n_{P,bobi}(t)$ (see section 4.3 for formal definition).

Based on this idea of a voter model, we propose the following *tf*-boosting approach:

Distribute the page frequency of a term t in the FTI across the occurrences of the term in the BoBI. With regards to voting this means that each entry of the BoBI receives votes from entries of the FTI.

We detail our model based around the above voting-based *tf*-boosting idea in the next section.

4. MODELLING TF-BOOSTING FOR PAGE SEARCH IN BOOKS

[4]In history books, for example, expanded phrases, such as those of historical events, may be used in the BoBI instead of the actual terms appearing in the text.

This section defines anchor-text-like retrieval strategies that lead to *tf*-boosting for pages of a book. First, we detail the schema of our book collection. We then present a naive anchor-text propagation approach, and a refined approach expressed as a voter model.

4.1 Schema

A PD engine uses database-like tables for indexing, where a schema is a list of tables and corresponding attributes in the header of tables. We designed our schema based on a collection of books provided by the INEX 2007 Book Search track [5]. A book in this corpus is stored in DjVu XML [6], where the basic structure can be summarised as follows:

```
<DjVuXML>
 <BODY>
  <OBJECT data="file.." ...>
  <PARAM name="PAGE" value=".."/>
  [...]
   <REGION>
    <PARAGRAPH>
     <LINE>
      <WORD coords="..."/>
      <WORD coords="..."/>
      [...]
     </LINE>
     <LINE> [...] </LINE>
     [...]
    </PARAGRAPH>
   </REGION>
   [...]
  </OBJECT>
  [...]
 </BODY>
</DjVuXML>
```

The original source was subsequently converted to an XML syntax, referred to as OCRML, which also contains additional structure mark-ups. Example snippets from the converted XML structure are given below:

```
<page id="54" pageNumber="23">
  <section id="123" label="SEC_BODY">
    hundreds of sailing boats are beached [...]
  </section>
</page>

<page id="224" pageNumber="123">
  <section id="54321" label="SEC_INDEX">
    sailing 12, 23, 45
  </section>
</page>
```

The first segment shows terms in the body of the book's content. This is identified by the label `SEC_BODY`. The second segment shows terms in the BoBI, which is labelled by `SEC_INDEX`. Each element has an unique identifier, reflecting its location in the document tree. The actual page numbers (as printed inside the book) are given by the "pageNumber" attribute.

Based on the OCRML structure and following the conventions introduced in section 3, we define the following schema for our indexes. The `fti_raw` table stores our (non-aggregated) full-text index, which is populated with terms from the body of the book (i.e., content labelled with `SEC_BODY`), while the table

[5]http://inex.is.informatik.uni-duisburg.de/2007/bookSearch.html
[6]http://djvu.sourceforge.net/doc/man/djvuxml.html

fti_raw	
Term	PageID
sailing	/book1/page23
sailing	/book1/page23
sailing	/book1/page29
...	...
weather	/book1/page19
weather	/book1/page20
...	...

(a) terms in book pages

bobi	
Term	PN
sailing	13
sailing	20
weather	3
weather	6
weather	100
boats	66

(b) terms in a BoBI

page_map	
PageID	PN
/book1/page23	7
/book1/page29	13
/book1/page31	15
...	...

(c) mapping from page IDs to page numbers

bobiPageFreq		
Probability $P(t\|p)$	Term t	PN p
1/2	sailing	13
1/2	sailing	20
1/3	weather	3
1/3	weather	6
1/3	weather	100
1	boats	66

(d) estimate the page frequencies of terms in the BoBI

fti_all	
$tf_{\text{fti}}(t)$	Term t
8	sailing
15	weather
6	boats

(e) aggregate the total terms weights in the FTI

fti (full-text index)			
Tuple weight $tf_{\text{fti}}(t,p)$	Term t	PageID p	
2	sailing	/book1/page23	(7)
1	sailing	/book1/page29	(13)
2	sailing	/book1/page31	(15)
3	sailing	/book1/page36	(20)
1	weather	/book1/page19	(3)
5	weather	/book1/page20	(4)
1	weather	/book1/page22	(6)
3	weather	/book1/page24	(8)
2	weather	/book1/page26	(10)
3	weather	/book1/page116	(100)
1	boats	/book1/page82	(66)
5	boats	/book1/page83	(67)

(f) tuples with original term weights

nbfti (naive boosted fti)		
Tuple weight $tf_{\text{nbfti}}(t,p)$	Term t	PageID p
$2+0=2$	sailing	$\cdots$ (7)
$1+1=2$	sailing	$\cdots$ (13)
$2+0=2$	sailing	$\cdots$ (15)
$3+1=4$	sailing	$\cdots$ (20)
$1+1=2$	weather	$\cdots$ (3)
$5+0=5$	weather	$\cdots$ (4)
$1+1=2$	weather	$\cdots$ (6)
$3+0=3$	weather	$\cdots$ (8)
$2+0=2$	weather	$\cdots$ (10)
$3+1=4$	weather	$\cdots$ (100)
$1+1=2$	boats	$\cdots$ (66)
$5+0=5$	boats	$\cdots$ (67)

(g) propagates term weights to the pages in the BoBI with naive model

vbfti (voter boosted fti)		
Tuple weight $tf_{\text{vbfti}}(t,p)$	Term t	PageID p
$2+0=2$	sailing	$\cdots$ (7)
$1+8/2=5$	sailing	$\cdots$ (13)
$2+0=2$	sailing	$\cdots$ (15)
$3+8/2=7$	sailing	$\cdots$ (20)
$1+15/3=6$	weather	$\cdots$ (3)
$5+0=5$	weather	$\cdots$ (4)
$1+15/3=6$	weather	$\cdots$ (6)
$3+0=3$	weather	$\cdots$ (8)
$2+0=2$	weather	$\cdots$ (10)
$3+15/3=8$	weather	$\cdots$ (100)
$1+6/1=7$	boats	$\cdots$ (66)
$5+0=5$	boats	$\cdots$ (67)

(h) propagates term weights to the pages in the BoBI with voter model

Figure 9: *Tf*-boosting for book search with a naive model and a voter model

called `bobi` is our back-of-book index, built from content labelled with `SEC_INDEX`. A `page_map` table is used for mapping the 'printed' page numbers of a book to the page IDs.

Page mapping is a practical requirement of processing books in their digitized form, while applications that need to handle such complexity will benefit from the expressiveness and flexibility provided by PD.

The three tables for indexing books are shown in Figures 9(a), 9(b), and 9(c).

With respect to the notation used in Figure 9 in general: The tuple weight corresponds to the number of tuples in which a term-page pair occurs. For example, $tf_{\text{fti}}(t,p) := n_{L,\text{fti\_raw}}(t,p)$, where (t,p) is a tuple, and $n_{L,\text{fti\_raw}}(t,p)$ is the number of tuples in table `fti_raw` (Figures 9(a)) in which (t,p) occurs.

In the next sections, we discuss the modelling of anchor-text *tf*-boosting. First, a naive model using the classic Web IR strategy is defined; and then a tailored voter model for book search is introduced.

4.2 Naive Model

A naive model performs direct term augmentation thus achieving *tf*-boosting to the pages in the BoBI. The idea comes from the Web IR anchor text *tf*-boosting, where terms in anchor texts contribute term frequencies to destinations. In a naive model, the terms in the BoBI are viewed as anchor texts, while their associating pages are viewed as destinations. For instance, in the example OCRML segment labelled by `SEC_INDEX` in section 4.1, the term "sailing" has three referenced pages "12", "23", and "45". Thereby, there are three virtual anchors, and "sailing" is an anchor-text being boosted in the destination pages.

A mathematical definition of the naive model is given by the following formula:

DEFINITION 1. *Boosted* tf*: naive model:*

Let $\text{tf}_{fti}(t,p) := n_L(t,p)$ *be the bare within-page term frequency of term* t *in page* p*, i.e. the number of locations in page* p *at which term* t *occurs.*

Let bobi be the set of anchors, and let $\text{tf}_{bobi}(t,a_p) := n_L(t,a_p)$ *be the within-anchor term frequency, where anchor* a_p *points to page* p*. Then, the naive boosted within-page term frequency is the sum of the bare within-page term frequency and the term frequencies of the anchors pointing to the page.*

$$\text{tf}_{nbfti}(t,p) := \text{tf}_{fti}(t,p) + \sum_{a_p} \text{tf}_{bobi}(t,a_p) \tag{1}$$

In probabilistic Datalog, the following rules model the naive boosting strategy [7]:

```
%1. term frequency in fti
fti SUM(T, X) :- fti_raw(T, X).

%2-3. maps page numbers and page IDs
```

[7]Lines start with '%' are for comment purpose.

```
fti_dist DISTINCT(T, X) :- fti_raw(T, X).
fti_map(T, X, P) :- fti_dist(T, X) & page_map(X, P).

%4-5. augmentation / tf-boosting
nbfti(T, X) :- fti(T, X).
nbfti(T, X) :- bobi(T, P) & fti_map(T, X, P).

%6. estimates probability scores
w_augTerm SUM(T, X) :- nbfti(T, X) | (X).
```

The arguments T, X and P correspond to "Term", "PageID" and "Page Number" (PN) in the schema, respectively. The first rule adds up the term frequencies to obtain the book-wide term frequency. Strictly speaking, this is not a probabilistic operation, since the tuple weights in `fti` (see Figure 9(f)) are total counts rather than probabilities [8]. The `fti` has two columns, i.e. "Term" and "PageID", while for better readability, we also show the corresponding "PN" values (between parentheses) after the page ID, and we will refer to "PN" in our latter discussions. The second and third rules map the page numbers to corresponding page IDs. The fourth and fifth rules perform naive *tf*-boosting. The augmented result is shown in the Figure 9(g). The last rule estimates the probability scores.

The issue here is that the naive modelling of *tf*-boosting does not deliver a reasonable boosting effect; this is because there is at most *ONE* anchor per term. Therefore, the next section describes the idea to relate the indexed (in the BoBI) and non-indexed occurrences of a term, and to propagate the non-indexed occurrences to the indexed occurrences, thus, leading to a boost of the term frequencies in the indexed pages.

4.3 Voter Model

Based on the discussion in section 3.2, a voter is a page occurring in the FTI. Voters vote for the destination pages that occur in the BoBI, and that share the same keywords. In other words, each voter holds anchors to destination pages (there could be more than one destinations, depending on the number of pages in BoBI). The local term frequency of a voter corresponds to the votes it can potentially assign to destinations (candidates). Each destination obtains a portion of the total amount of votes. There are various ways to allocate votes to destinations, and we illustrate here the unbiased voting where the votes are evenly distributed over destinations. Refined methods for vote allocation are to be studied in future work. For example, the vote distribution could consider page properties such as length, location in book, exhaustiveness and/or specificity of the page.

The mathematical definition of the unbiased voter model is given as:

DEFINITION 2. *Boosted* tf*: voter model:*

Let $\text{tf}_{fti}(t) := n_{L,fti\_raw}(t)$ *be the book-wide term (location) frequency. This frequency is equal to the sum over the within-page term frequencies:* $n_{L,fti\_raw}(t) = \sum_p n_{L,fti\_raw}(t,p)$*. In a less formal way, the total (book-wide) term frequency is*

$$\text{tf}_{fti}(t) = \sum_p \text{tf}_{fti}(t,p)$$

where $\text{tf}_{fti}(t,p) := n_{L,fti\_raw}(t,p)$ *is the within-page term frequency.*

The book-wide term frequency of a term t *is distributed over the BoBI entries. Let* $n_{P,bobi}(t)$ *denote the number of pages the respective term entry points to in the BoBI. Note that* $n_{P,bobi}(t) = n_{L,bobi}(t)$*, i.e. the number of pages is equal to the number of locations/tuples, if the BoBI is distinct, i.e. there are no multiple term-page entries of the same term to the same page.*

Then, the boosted term frequency is defined as follows:

$$\text{tf}_{vbfti}(t,p) := \text{tf}_{fti}(t,p) + \frac{\text{tf}_{fti}(t)}{\text{tf}_{bobi}(t)} \tag{2}$$

*The respective term frequencies (*tf*) correspond to total counts of term occurrences in the raw FTI, as the next equation illustrates.*

$$\text{tf}_{fti}(t,p) = n_{L,fti\_raw}(t,p) + \frac{n_{L,fti\_raw}(t)}{n_{L,bobi}(t)} \tag{3}$$

The probabilistic Datalog rules for modelling the voter model are as follows:

```
%1. term's page frequency in bobi
bobiPageFreq(T, P) :- bobi(T, P) | (T).

%2. term frequency in fti
fti SUM(T, X) :- fti_raw(T, X).

%3-4. maps page numbers and page IDs
fti_dist DISTINCT(T, X) :- fti_raw(T, X).
fti_map(T, X, P) :- fti_dist(T, X) & page_map(X, P).

%5-6. obtains statistics for augmentation
bobiIDFreq(T, X) :-
    bobiPageFreq(T, P) & fti_map(T, X, P).
fti_all SUM(T) :- fti(T, X).

%7-8. augmentation / tf-boosting
vbfti(T, X) :- fti(T, X).
vbfti(T, X) :- bobiIDFreq(T, X) & fti_all(T).

%9. estimates probability scores
w_augTerm(T, X) :- vbfti(T, X) | (X).
```

The first rule yields the term's page frequency in the `bobi` (see Figure 9(d)). The tuples in the goal "bobiPageFreq" are conditional probabilities of the form $P(T, P|T)$, i.e. the tuple probabilities are conditioned by the term. The "| (T)" attached to the sub-goal is the evidence key, and this describes the conditional probability. The second, third and fourth rules are similar to the naive model. The fifth rule transfers the page frequency to page IDs, and the sixth rule aggregates the total book-wide *tf* (see Figure 9(e)); these two rules prepare the statistics for the latter *tf*-boosting. The seventh and eighth rules conducts the proposed boosting, the result of `vbfti` is shown in Figure 9(h). Finally, the last rule estimates the probability scores.

Considering the boosting effect of the voter model, we take a closer look at the boosted FTI, the `vbfti` in Figure 9(h). For example, the two occurrences of the term "sailing" which appear in the BoBI (i.e., on page 13 and page 20) receive an equal share of the total *tf* of "sailing" in the FTI, boosting the *tf* by $8/2$ to a total score of 5 in the case of page 13, and 7 for page 20.

[8]The description of a full probabilistic model is part of future research. For the time being, we propose the frequency-based model, since the probabilistic model implies an early aggregation of frequency counts, and this does not deliver the *tf*-boosting effect as applied in hypertext retrieval.

A beneficial effect of the voting-based *tf*-boosting strategy is that since it promotes the pages referenced in the BoBI, it correctly leads to a more user-oriented retrieval paradigm. For example, assuming that the BoBI lists only the first occurrence of a term (at the start of a topic), the system will rank the page referenced in the BoBI ahead of other pages containing the term, and alas directing the user to the page where they should start reading. For instance, page 3 is the first page about "weather", while the adjacent page 4 continues the discussion. Without the voter model or with the naive model, page 4 gets ranked higher, but with the voter model score of page 3 is boosted above page 4's score. Based on this assumption, a best entry point strategy which aims to direct user to those points in the text where they should start reading [3] can be feasibly implemented. In effect, using the voter model, the best entry point selection method of the author, as implemented within the index term selection strategy they applied can be directly reflected in the retrieval and its benefits passed directly onto the user.

5. SUMMARY

This paper proposes a *tf*-boosting model for book/page search. The initial starting point was to apply anchor-text retrieval to the back-of-book index, this idea being based on the fact that a BoBI is a list of anchors pointing to pages. However, the starting point turned out to be "naive" in the sense that the propagation of anchor-text from the BoBI does not deliver the desired *tf*-boosting effect, and, in turn, this observation motivated this paper.

For achieving a *tf*-boosting effect from the BoBI, we needed to revise the hypertext-based modelling. This revision leads to a voter model proposed for *tf*-boosting in book search: the overall idea is to distribute the book-wide term frequency of a term to the pages that are indexed in the BoBI; this corresponds to a voting model in the sense that a term votes for the pages to which its term frequency shall be added.

One of the main contributions of this paper is that the modelling of the voter model demonstrates the benefit of high-level abstract modelling of retrieval strategies. Traditional anchor-text retrieval is modelled in probabilistic Datalog rules; these rules are not directly applicable for book search, and need to be evolved. This evolution demonstrates that through the high-level abstraction of search strategies, successful techniques of one domain (here, anchor-text retrieval in hypertext) can be transferred to another domain (tf-boosting for book/page search).

Future work will include the investigation of different voter models. For example, deciding how many votes an entry page should receive based on proximity (term to page, or page to page). In addition, the evaluation of retrieval effectiveness will be examined with real and large scale collections.

Acknowledgements

We would like to thank Bodin Dresevic and Dejan Lukacevic at the Microsoft Development Centre in Serbia for the use of their tool to mark-up books in OCRML.

6. REFERENCES

[1] N. Abdullah and F. Gibb. Using a task-based approach in evaluating the usability of bobis in an e-book environment. In *ECIR*, pages 246–257, 2008.

[2] N. Craswell, D. Hawking, and S. E. Robertson. Effective site finding using link anchor information. In *SIGIR*, pages 250–257, 2001.

[3] K. Finesilver and J. Reid. User behaviour in the context of structured documents. In *ECIR*, pages 104–119, 2003.

[4] N. Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110, 2000.

[5] N. Fuhr and K. Großjohann. XIRQL: An XML query language based on information retrieval concepts. *ACM Trans. Inf. Syst.*, 22(2):313–356, 2004.

[6] N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors. *Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX), Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2008.

[7] N. Fuhr and T. Roelleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems (TOIS)*, 14(1):32–66, 1997.

[8] D. J. Harper, I. Koychev, and S. Yixing. Query-based document skimming: A user-centred evaluation of relevance profiling. In *ECIR*, pages 377–392, 2003.

[9] D. Hawking. Challenges in enterprise search. In *ADC*, pages 15–24, 2004.

[10] D. Hawking, E. M. Voorhees, N. Craswell, and P. Bailey. Overview of the trec-8 web track. In *TREC*, 1999.

[11] G. Kazai and A. Doucet. Overview of the INEX 2007 book search track (BookSearch'07). In Fuhr et al. [6].

[12] G. Kazai and A. Trotman. Users' perspectives on the usefulness of structure for XML information retrieval. In *ICTIR*, 2007.

[13] M. Lalmas and T. Roelleke. Four-valued knowledge augmentation for structured document retrieval. In *Proceedings of the 13th International Symposium on Methodologies for Intelligent Systems (ISMIS), Lyon, France*, June 2002.

[14] R. R. Larson. Logistic regression and EVIs for XML books and the heterogeneous track. In Fuhr et al. [6].

[15] W. Magdy and K. Darwish. CMIC at INEX 2007: Book search track. In Fuhr et al. [6].

[16] T. Roelleke. *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects*. Shaker Verlag, Aachen, 1999. Doktorarbeit (PhD thesis).

[17] T. Roelleke, M. Lalmas, G. Kazai, I. Ruthven, and S. Quicker. The accessibility dimension for structured document retrieval. In *Proceedings of the BCS-IRSG European Conference on Information Retrieval (ECIR), Glasgow*, March 2002.

[18] T. Roelleke, H. Wu, J. Wang, and H. Azzam. Modelling retrieval models in a probabilistic relational algebra with a new operator: the relational Bayes. *VLDB J.*, 17(1):5–37, 2008.

[19] F. Weigel, K. U. Schulz, and H. Meuss. Exploiting native XML indexing techniques for XML retrieval in relational database systems. In *WIDM*, pages 23–30, 2005.

[20] H. Wu, G. Kazai, and M. Taylor. Book search experiments: Investigating IR methods for the indexing and retrieval of books. In *ECIR*, pages 234–245, 2008.