An Artificial Intelligence Approach To Information Retrieval

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

ABSTRACT

Current approaches to information retrieval rely on the creativity of individuals to develop new algorithms. In this investigation the use of genetic algorithms (GA) and genetic programming (GP) to learn IR algorithms is examined.

Document structure weighting is a technique whereby different parts of a document (title, abstract, etc.) contribute unevenly to the overall document weight during ranking. Near optimal weights can be learned with a GA. Doing so shows a statistically significant 5% relative improvement in MAP for vector space inner product and Croft's probabilistic ranking, but no improvement for BM25. Two applications of this approach are suggested: offline learning, and relevance feedback.

In a second set of experiments, a new ranking function was learned using GP. This new function yields a statistically significant 11% relative improvement on unseen queries tested on the training documents. Portability tests to different collections (not used in training) demonstrate the performance of the new function exceeds vector space and probability, and slightly exceeds BM25. Learning weights for this new function is proposed.

The application of genetic learning to stemming and thesaurus construction is discussed. Stemming rules such as those of the Porter algorithm are candidates for GP learning whereas synonym sets are candidates for GA learning.

Categories and Subject Descriptors

H3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval – *retrieval models.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Information Retrieval, Genetic Algorithms, Genetic Programming.

1. INTRODUCTION

There exists a plethora of models for information retrieval. Proposing a new model is a tradition that traces its history to the 1970s. Early examples include the vector space model [22] and the probability model [18]. More recent models include the set-based model [16] and the logic model [12].

Evaluation of information retrieval systems also has a long tradition. The first TREC [7] was held in 1992. In 2002, the INEX [3] workshops for evaluation of XML retrieval started.

The model and the evaluation are inextricably related. A theoretical model is of little value if it performs badly. Conversely, if a method performs well before an underpinning model is developed, it remains important. Such was the case with Okapi BM25 [19], which has been shown to perform well, even though it is a mish-mash of other models.

The best performing method to date is that of the human brain. The evaluation forums take it as read. Each system is evaluated by comparing the computer-generated output to the human-made judgments.

Information retrieval systems are routinely evaluated against standard test sets. In a typical scenario a researcher develops a new technique, implements it, then compares it with existing techniques. The result is positive if a significant improvement in the appropriate metric is shown. In the case of *ad hoc* retrieval of whole documents, the standard metric is mean average precision, while significance is determined using the Wilcoxon's signed rank test [28].

Development of a new technique is comparable to guessing what a human does; yet explicit in every judgment is what a human did. It seems likely, therefore, that the judgments can themselves be used to improve information retrieval techniques.

The Holy Grail of information retrieval is 100% precision and 100% recall, relative to the human judge. This cannot be achieved (multiple human judges often don't agree on judgments [23]), but significant improvements in precision are expected.

2. PROPOSED RESEARCH

Artificial intelligence techniques will be used to improve precision. Standard test sets and judgments from TREC will be used for learning and evaluation. *Ad hoc* retrieval of whole XML documents will be examined, but the techniques are also applicable to relevance feedback. Web retrieval will not be examined

2.1. Improving Precision

Three approaches will be explored: first, the use of document structures to improve precision, second, general purpose ranking methods, and finally, the combination of the two techniques.

2.1.1. Structure Weighted Ranking

TREC documents are distributed in XML. Wall Street Journal (WSJ) documents, for example, include structure markup for title, first paragraph, text, as well as other fields. Term occurrences could be weighted based on which of these structures they occur. This technique is not new [4]. Precision improvements have already been shown [26], and different weight choosing techniques have been tried [10; 17]. However,

there is no systematic method of choosing the weights, which may be why choosing weights was a recurring question at INEX 2003.

The weights can be encoded in an array, with one array element for each document structure weight. The natural choice of learning method for this encoding is the genetic algorithm [8]. Initially a population of individuals is chosen with randomized weights. At each generation the mean average precision of each individual is computed. Individuals for the next generation are then chosen through reproduction, mutation and crossover.

Optimal document structure weighting will not reduce mean average precision when compared to retrieval without structure weighting. If all document structures are weighted equal and 1, the equivalent of un-weighted retrieval occurs. Performance of weighted retrieval (with optimal weights) can therefore always at least equal that of un-weighted retrieval over a large set of queries. Further, GAs are a well-established optimization technique so it should be possible to put a performance upper bound on this technique.

2.1.2. General Purpose Ranking

Any precision improvements gained through structure-weighted retrieval will be gained relative to a baseline. That baseline is the precision of the chosen un-weighted ranking function. Many functions have been published, so many that taxonomies of ranking functions exist [29]. These taxonomies account for as many as 100,000 ranking functions.

Exhaustively testing 100,000 functions is impractical, but a directed search is not. GP [11] is a well-established technique for such a search. By contrast to GA, GP represents individuals as an abstract syntax tree. The ranking function itself is an individual. Mutation and reproduction are similar to that in GA; crossover is the swapping of branches from tree to tree.

The evidences to combine (the atoms in the ranking function) should be those "freely" available in an inverted file (for efficiency reasons). This not only includes term frequencies, but also document-lengths, collection size, and the highest document term frequency, amongst others.

Significant improvements are expected with this technique, however they have not yet been seen. Previous experiments to combine a limited amount of *tf.idf* like evidence on the cystic fibrosis collection [24] failed to show a significant improvement [14]. Meanwhile, experiments to learn more general ranking functions for HTML have failed to better Okapi BM25 without using document structures [2].

Prior results are contrary to expectation. They suggest *tf.idf* and Okapi BM25 are perfect ranking functions for the examined collections, and cannot be bettered. This is unlikely.

Close examination of the prior results shows why they are unexpected. The learned ranking function should be at least a combination of evidence and operators used in the baseline function. Unless this is the case the baseline could not be learned and it is reasonable to assume it will not be bettered.

In other words, if an already existing ranking function f() is the combination of evidence (α) and operators (β), it cannot outperform a GP learned function being the combination of evidence (ε) with operators (ϕ) if α is a subset of ε, and β is a subset of ϕ. This is because the GP could learn f(). Further, the learning can be seeded with f() guaranteeing at least f().

Choice of evidence is important. Previous experiments had negative results because the choice of evidence was inadequate.

2.1.3. The Structure Weighted / Learned Function

It should be possible first to learn an improved general purpose ranking function, then to learn improving structure weights for such a function. Better, the two could be learned in conjunction, this way any interaction of the two techniques would be exploited during learning.

In prior experiments, term frequencies in some HTML tags were used in a GP learning experiment [2]. This resulted in an improvement over un-weighted retrieval with Okapi BM25. An improvement is also expected if this technique is applied to non-web documents.

2.2. Improving Precision and Recall

There already exist many techniques to improve recall; examples include relevance feedback, stemming and thesaurus. These techniques can, instead, be thought of as pure precision enhancers. Every newly identified relevant document takes a prevision score above 0 where, while unidentified, its precision score was 0. This is increasing precision by identifying additional relevant documents. Optimizing precision using conflation is to ask a new question: which terms, when conflated, will increase precision regardless of the recall effect? As an optimization problem this lends itself to genetic learning.

2.2.1. Relevance Feedback

In relevance feedback, a user performs a search, the result is returned for judging, then, with knowledge of the judgments, the search is re-evaluated and the new results retrieved.

The techniques already discussed could be used for relevance feedback. After the initial judging round, the original query and a set of judgments are known – all that is needed to learn a ranking function and structure weights.

Pre-existing techniques for relevance feedback [21] such as query expansion and term weighting could be used in conjunction with this learning technique. GA has already been used to learn term weights [27].

2.2.2. Stemming

During indexing, each word in the document collection is converted algorithmically into a stem. All words with the same stem are then indexed as if they were a single term. The intention is to merge the postings for all words with a common morphology. For example, the postings for "treatment", "treating" and "treat" would be merged into the stem "treat". When a user searches for "treats", that word is stemmed to "treat" and the postings for the stem are retrieved, finding documents not containing the search term.

Many stemming algorithms exist (e.g. [15]). Often they exist as a sequence of re-write rules. Each rewrite rule is a program statement. It should, therefore, be possible to learn to stem using GP.

Investigation into stemming effectiveness has shown stemming is ineffective [5]. This is perhaps because the "stemming quality" of a stemming algorithm is measured with stemming

error rate, but the "IR quality" is measured with mean average precision, two independent measures. If a stemming algorithm is developed with the sole purpose of increasing mean average precision, an improvement on this negative result might be seen.

2.2.3. Thesaurus

Query expansion using a thesaurus is common. There are general-purpose thesauri in the public domain but few document collections are general-purpose. Domain specific thesauri (e.g. UMLS [1]) exist, but are costly to hand produce. Automated thesaurus generation has also been tried [20].

A thesaurus can be represented as a set of bit-strings, one for each thesaurus entry. Each bit-string has a bit for each unique term in the document collection. If, in a given bit-string, two bits are set, those terms are synonyms.

These bit-strings can be learned with a GA. A population is seeded with individuals having random bits set and a traditional GA is used to learn good combinations. Selective pressure is applied to increase precision irrespective of the effect on recall.

The same technique can be used to learn phrasal synonyms (e.g. 'information retrieval' and IR). Individual content baring phrases would first be identified using pre-existing techniques [9], the document collection indexed to include these terms, and the bit-string thesaurus learning applied.

Terms to exclude from a search (antonyms) could also be learned the same way.

3. PORTABILITY OF RESULTS

Documents, queries and judgments are needed for all the proposed experiments. Collecting the judgments is a time consuming and costly exercise so the TREC collections are used. However, unless the results of these learning experiments are portable from one document collection to another, learning on TREC data will produce results that can't be used elsewhere.

Experiments that learn structure weights are tied to structure. For XML, this is not just the DTD, but rather the particular use of the DTD. These results are not portable from collection to collection.

Portability is expected from learned ranking functions, so long as no evidence tying the function to the particular document collection is used. Additionally, the portability can be tested by measuring the performance of any learned function on a diverse set of document collections. Statistical comparison to other portable ranking functions such as BM25 will determine if, or not, these functions are portable.

Equally, stemming and synonym sets learned on one collection are expected to port to other collections.

4. PRELIMINARY RESULTS

Initial investigation into learning document structure weights was conducted with inner product, probabilistic and Okapi BM25 ranking.

The inner product function was a straightforward tf.idf implementation:

$$w_{dq} = \sum_{t \in q} \left(tf_{td} \times vidf_t \right) \cdot \left(tf_{tq} \times vidf_t \right)$$

where

$$vidf_t = \log_2 \frac{N - n_t + 1}{n_t}$$

The probabilistic function was that according to Harman [6]:

$$w_{dq} = \sum_{t \in q} \left(C + pidf_t \right) \times \left(K + \left(1 - K \right) * \frac{tf_{td}}{\max(tf_d)} \right)$$

where

$$pidf_t = \log_2 \frac{N - n_t + 1}{n_t}$$

where $C = 1$, $K = 0.3$.

The BM25 implementation was faithful according to that of Robertson *et al*. [19]

In each case, N was the number of documents, n_t the number of documents in which the term occurs and tf_{td} is the number of occurrences of term t in document d (likewise tf_{tq} in the query).

The training set was the TREC WSJ (1987-1992), with topics 151-200. Topics with fewer than 5 judgments were discarded. The population size was 50 and several experiments were run for 25 generations. Evaluation was against topics 101-150; results are shown in Table 1.

Table 1: Differences in MAP for un-weighted retrieval (MAP) and weighted retrieval (W-MAP) for the evaluation topics. P is computed using Wilcoxon's signed rank test.

Function	MAP	W-MAP	Imp	Imp%	P-Value
BM25	0.2289	0.2281	-0.0008	-0.35%	0.92
Probability	0.1675	0.1787	0.0112	6.69%	0.00
Inner Prod	0.1657	0.1735	0.0078	4.71%	0.00

A 5% improvement on un-weighted retrieval is seen in inner product and probability model (significant at 1%). Okapi BM25 shows no improvement. No examined technique significantly outperformed un-weighted Okapi BM25 [25].

To place an upper bound on the improvement, the training and evaluation queries were pooled and a series of experiments run to optimize structure weights across all queries. The results were similar to those shown in Table 1, this technique yields an improvement of about 5% in inner product and probability, but of less than 1% in BM25.

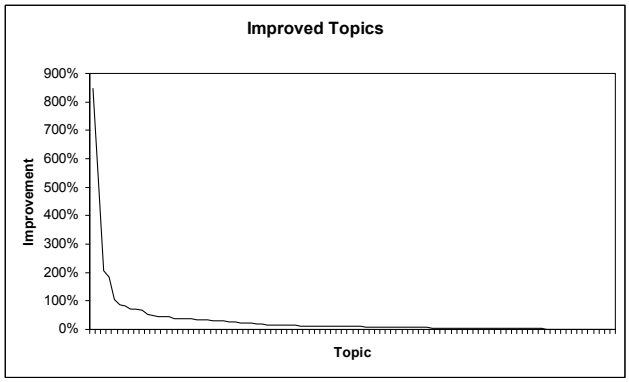

Figure 1: Few topics show large improvements while many topics show small improvements. About 50% of topics show an improvement of over 10%.

Table 3: Comparison of different ranking functions on different document collections.

Function	WSJ	CysFib	CR	FR94	FT	Trec 4	FBIS	LATimes	Trec 5	Trec 4+5
Run 5	0.2849	0.2860	0.1727	0.1820	0.2223	0.1702	0.2566	0.1881	0.1834	0.1625
BM25	0.2553	0.2728	0.1949	0.2079	0.2022	0.1767	0.2082	0.1996	0.1824	0.1658
Probability	0.1890	0.2781	0.1481	0.1311	0.1240	0.0857	0.1423	0.1352	0.1155	0.0855
Inner Prod	0.1497	0.2873	0.0992	0.0863	0.1203	0.0346	0.0665	0.0797	0.0473	0.0286
MAP Imp	0.0297	0.0133	-0.0223	-0.0259	0.0200	-0.0065	0.0484	-0.0115	0.0010	-0.0033
Imp%	11.63%	4.86%	-11.42%	-12.48%	9.91%	-3.70%	23.24%	-5.78%	0.54%	-2.00%
P-Value	0.00	0.04	0.91	0.95	0.19	0.83	0.00	0.50	0.17	0.33

Table 2: Improvements for query-customized weights.

Function	MAP	Each	Imp	Imp%	P-Value
BM25	0.2418	0.2775	0.0357	14.77%	0.00
Probability	0.1780	0.2056	0.0276	15.48%	0.00
Inner Prod	0.1567	0.2033	0.0466	29.70%	0.00

Training on each query in isolation showed improvements for BM25 ranging from 10 times (0.047 to 0.443 for topic 103) through to less than 1%. The mean average precision improvement for inner product was 30%, and was 15% for probability and Okapi BM25. This result suggests this technique might be successful for relevance feedback, further experiments are required. Table 2 shows the improvement in MAP. In Figure 1 the improvements in each query are shown ordered from most to least. In these experiments, the definition of $vidf_t$ was changed to

$$vidf_t = \log_2 \frac{N+1}{n_t}$$

there was no theoretic justification for doing so and the experiments have not been conducted using the earlier definition. It should be noted that the performance of inner product using this later definition is worse than that used earlier.

Experiments to learn a general purpose ranking function were conducted on the same training set (WSJ, Topics 151-200). Several runs of 100 individuals for 100 generations were conducted. Learning was elitist and was seeded with other ranking functions including BM25.

Evaluation was against WSJ (topics 101-150), collections from TREC disks 4 and 5 (topics 301-350), and the cystic fibrosis collection [24]. A diverse set of evaluation collections is necessary because ranking performance is known to vary greatly from collection to collection [29]. The evaluation results are shown in Table 3 where Run 5 is the best learned function to date. From this, each of BM25 and Run 5 out-performed the other an equal number of times, but of the results significant at the 5% level, Run 5 always outperformed BM25. Function Run 5 has proven to be portable from collection to collection.

5. DISCUSSION

This investigation centers around one question: How can artificial intelligence techniques be used to improve information retrieval? Already shown is how GA and GP can be used to improve precision. The advantage of these algorithms over others (e.g., Neural Networks) is in the symbolic results. The ranking function can be examined. The thesaurus results can be printed as a thesaurus. More importantly, the results can be moved from one document collection to another and can be expected to continue to perform well.

Examination of precision and recall is an arbitrary choice, and may not be the best choice. Surely categorization can be improved with AI. How could it be used in a question / answer system? How could these techniques be used to improve the interactive experience of a user? Could GP be used for index compression? Perhaps an intelligent caching mechanism would improve throughput? Clearly AI is important for clustering, but can genetic techniques be used?

Still unanswered AI questions include: do other AI techniques (such as particle swarm optimization [13]) better fit this problem domain? What better encodings exist than those proposed herein? Have these techniques been tried before? What efficiency issues should be examined?

The most important unanswered question is: In what future directions can (and should) this approach be taken?

REFERENCES

[1] Anonymous. (2001). *UMLS knowledge sources* (12 ed.): U.S. Department of Health & Human Services, National Institutes of Health, National Library of Medicine.

[2] Fan, W., Gordon, M. D., Pathak, P., Xi, W., & Fox, E. A. (2004). Ranking function optimization for effective web search by genetic programming: An empirical study. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*.

[3] Fuhr, N., Gövert, N., Kazai, G., & Lalmas, M. (2002). INEX: Initiative for the evaluation of XML retrieval. In *Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval*.

[4] Fuller, M., Mackie, E., Sacks-Davis, R., & Wilkinson, R. (1993). Structured answers for a large structured document collection. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 204-213).

[5] Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science,* 42(1), 7-15.

[6] Harman, D. (1992). Ranking algorithms. In W. B. Frakes & R. Baeza-Yates (Eds.), *Information retrieval: Data structures and algorithms* (pp. 363-392). Englewood Cliffs, New Jersey, USA: Prentice Hall.

[7] Harman, D. (1993). Overview of the first TREC conference. In *Proceedings of the 16th ACM SIGIR Conference on Information Retrieval*, (pp. 36-47).

[8] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

[9] Kim, W., & Wilbur, W. J. (2001). Corpus-based statistical screening for content-bearing terms. *Journal of the American Society for Information Science and Technology,* 52(3), 247-259.

[10] Kim, Y.-H., Kim, S., Eom, J.-H., & Zhang, B.-T. (2000). SCAI experiments on TREC-9. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, (pp. 392-399).

[11] Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press.

[12] Losada, D. E., & Barreiro, A. (2001). A logical model for information retrieval based on propositional logic and belief revision. *Computer Journal,* 44(5), 410-424.

[13] Løvbjerg, M., Rasmussen, T. K., & Krink, T. (2001). Hybrid particle swarm optimizer with breeding and subpopulations. In *Proceedings of the 3rd Genetic and Evolutionary Computation Conference*.

[14] Oren, N. (2002). Reexamining *tf.idf* based information retrieval with genetic programming. In *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology (SAICSIT)*, (pp. 224-234).

[15] Porter, M. (1980). An algorithm for suffix stripping. *Program,* 14(3), 130-137.

[16] Pôssas, B., Ziviani, N., Meira, W., & Ribeiro-Neto, B. (2002). Set-based model: A new approach for information retrieval. In *Proceedings of the 25th ACM SIGIR Conference on Information Retrieval*, (pp. 230-237).

[17] Rapela, J. (2001). Automatically combining ranking heuristics for HTML documents. In *Proceedings of the 3rd International Workshop on Web Information and Data Management*, (pp. 61-67).

[18] Robertson, S. E., & Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science,* 27(3), 129-146.

[19] Robertson, S. E., Walker, S., Jones, S., Beaulieu, M. M., & Gatford, M. (1994). Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, (pp. 109-126).

[20] Roussinov, D., & Chen, H. (1998). A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation. *Communication and Cognition,* 15(1-2), 81-112.

[21] Ruthven, I., & Lalmas, M. (2003). A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review,* 18(2), 95 - 145.

[22] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM,* 18(11), 613-620.

[23] Schamber, L. (1994). Relevance and information behavior. *Annual Review of Information Science and Technology,* 29, 3-48.

[24] Shaw, W. M., Wood, J. B., Wood, R. E., & Tibbo, H. R. (1991). The cystic fibrosis database: Content and research opportunities. *Library and Information Science Research,* 13, 347-366.

[25] Trotman, A. (2003). Choosing document structure weights. *Information Processing & Management,* to appear.

[26] Wilkinson, R. (1994). Effective retrieval of structured documents. In *Proceedings of the 17th ACM SIGIR Conference on Information Retrieval*, (pp. 311-317).

[27] Yang, J., Korfhage, R., & Rasmussen, E. (1992). Query improvement in information retrieval using genetic algorithms - a report on the experiments of the TREC project. In *Proceedings of the 1st Text REtrieval Conference (TREC-1)*, (pp. 31-58).

[28] Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st ACM SIGIR Conference on Information Retrieval*, (pp. 307-314).

[29] Zobel, J., & Moffat, A. (1998). Exploring the similarity space. *SIGIR Forum,* 32(1), 18-34.