

Simplified Similarity Scoring Using Term Ranks

Vo Ngoc Anh

Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia
vo@cs.mu.oz.au

Alistair Moffat

Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia
alistair@cs.mu.oz.au

ABSTRACT

We propose a method for document ranking that combines a simple document-centric view of text, and fast evaluation strategies that have been developed in connection with the vector space model. The new method defines the importance of a term within a document qualitatively rather than quantitatively, and in doing so reduces the need for tuning parameters. In addition, the method supports very fast query processing, with most of the computation carried out on small integers, and dynamic pruning an effective option. Experiments on a wide range of TREC data show that the new method provides retrieval effectiveness as good as or better than the Okapi BM25 formulation, and variants of language models.

Categories and Subject Descriptors

H.3.1 [Information Storage and Retrieval]: Content analysis and indexing – *indexing methods*; H.3.2 [Information Storage and Retrieval]: Information storage – *file organization*; H.3.3 [Information Storage and Retrieval]: Information search and retrieval – *search process*; H.3.4 [Information Storage and Retrieval]: Systems and software – *performance evaluation*.

General Terms

Efficiency/scale: architectures, compression, efficient query evaluation; Text representation and indexing.

1. INTRODUCTION

Two of the most important features of a document ranking mechanism are its *retrieval effectiveness* and *querying efficiency*. The former reflects the ability of the mechanism to retrieve relevant documents at the top of the ranking, while the latter represents the desire to control per-query processing time. Other features such as index size, index building speed, or memory usage during indexing and querying are also important, but, to some extent, less crucial.

A large body of research work has been devoted to improving retrieval effectiveness, and has led to the development of a range of innovative retrieval models and weighting schemes. The effectiveness of ranking mechanisms is demonstrated through empirical

studies, using resources such as the ones offered by the ad-hoc and web tracks of the annual TREC conference, see trec.nist.gov for details. The figures from these tracks demonstrate the ongoing strength of three mechanisms: the vector space model with pivoted document length normalization [Singhal et al., 1996]; the Okapi BM25 formula (BM25) [Robertson et al., 1998]; and the language modeling approach (LM) [Ponte and Croft, 1998].

Implementations of document ranking systems typically make use of an inverted index to support efficient computation of the set of similarity scores $S_{d,q}$ between a query q and the N documents d in a collection D . Each distinct term t in D is represented in the index by its *document frequency* f_t ; its *collection frequency* F_t ; and an *inverted list* storing pointers of the form $\langle t, f_{d,t} \rangle$ where the *term frequency* of t in document d is $f_{d,t}$. Similarly, $f_{q,t}$ is used to denote the term frequency of t in the query q . Each document is also associated with a *document length* W_d , a quantity that is pre-calculated at indexing time and does not affect efficiency.

For example, one variant of the vector space model calculates $S_{d,q}$ as

$$\frac{1}{W'_d} \cdot \sum_{t \in d \cap q} (1 + \log_e f_{d,t}) \cdot (1 + \log_e f_{q,t}) \cdot \log_e (1 + \frac{f_t^a}{f_t}) \quad (1)$$

where f_t^a is the average value of f_t over D , and $W'_d = (1 - s) + s \cdot W_d/W_d^a$ is the normalized weight of d and can be pre-calculated at indexing time with the *slope* $s = 0.7$ [Singhal et al., 1996] (see also the work of Chowdhury et al. [2002]). In operation this scheme is simple: the inverted lists of all terms $t \in q$ are traversed, and partial scores for indicated documents are accumulated; then the values W'_d are factored in; and finally the highest scores are extracted.

An alternative similarity formulation is the Okapi BM25 mechanism [Robertson et al., 1998]. One simple implementation (see, for example, Jin et al. [2002]) defines $S_{d,q}$ as

$$\sum_{t \in d \cap q} \ln \frac{N - f_t + 0.5}{f_t + 0.5} \cdot \frac{f_{d,t}}{0.5 + 1.5 \cdot W_d/W_d^a + f_{d,t}} \quad (2)$$

This computation can be implemented using the same information as is used in the vector space approach, but in an operational sense is potentially less efficient than the vector space computation, because of the use of the normalized document weight each time a pointer in an inverted list is examined.

In language models, the similarity score $S_{d,q}$ is replaced by the conditional probability $P(q | M_d)$ of generating the query q given a model M_d of d . Ponte and Croft [1998] calculate this probability as:

$$P(q | M_d) = \prod_{t \in q} P(t | M_d) \cdot \prod_{t \notin q} (1 - P(t | M_d))$$

where the probability of producing and not producing the terms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '05, August 15–19, 2005, Salvador, Brazil.

Copyright 2005 ACM 1-59593-034-5/05/0008 ...\$5.00.

in the query are taken into account. However, the computation of $P(q \mid M_d)$ is less efficient than that of $S_{d,q}$ in the BM25 scheme described above. Experiments with language model approaches show that it gives good retrieval effectiveness, and the joint statement by Allan et al. [2002] is indicative of the confidence with which a significant section of the community view this approach.

In all three approaches, the computation of $S_{d,q}$ for each document d is followed by a process that selects a number of highly-scoring documents, and presents them or some summary of them to the user.

In this paper we describe another way in which term frequency evidence can be used to generate similarity scores. The new method has elements in common with all of these three approaches, but also has a critical distinguishing feature – we make use of term frequency *ranks* in the scoring process, rather than numerically calculated term frequency *values*. The end result of our scoring regime is still a number $S_{q,d}$ – there is no alternative to that if we wish to score documents – but the reduction in the number of tuning factors and transformation functions during the computation means that there is much less room for ambiguity when implementing it.

We have carried out extensive experiments using the new approach, using a wide range of TREC data. Despite the simplicity of the scoring regime, it performs well when compared to other similarity computations, and its retrieval effectiveness on a range of TREC tasks is good. The new method also requires *less* query-time information than other approaches, and has a simple evaluation process. The result is good retrieval effectiveness at high query processing rates, an attractive blend of attributes. The query-processing regime is also amenable to a dynamic pruning process that further enhances query throughput rates.

A brief word on nomenclature is necessary. In this work we concentrate on individual documents d rather than the whole collection D , and it is useful to slightly abuse some of the usual notation. In particular, a document d is supposed to have n_d distinct terms t_1, t_2, \dots, t_{n_d} , and $T_d = (t_1, t_2, \dots, t_{n_d})$ is referred to as the *term list* of d . In any formulation of $S_{d,q}$, the value derived solely from a certain term t and the document d is called *impact* of term t in d , and is denoted by $\omega_{d,t}$. If necessary, a superscript letter is used to indicate precisely which type of impact. Thus, in Formula 1 we have, with respect to t , the *cosine* document impact as $\omega_{d,t}^c = (1/W_d) \cdot (1 + \log_e f_{d,t}) \cdot \log_e(1 + f_t^a/f_t)$, and the cosine query impact $\omega_{q,t}^c = 1 + \log_e f_{q,t}$. We also refer to $\Omega_d = (\omega_{d,t_1}, \omega_{d,t_2}, \dots, \omega_{d,t_{n_d}})$ as the *impact vector* for d . Abusing mathematical definitions, we say that, in the case of the vector space model and BM25, the similarity score $S_{d,q}$ is the scalar vector product of the document impact vector Ω_d and the query impact vector Ω_q . The abuse lies on the projection of both the document term list T_d and the query term list T_q to $T_d \cap T_q$ to allow the mathematical scalar vector product. That is, we write

$$S_{d,q} = \Omega_d \cdot \Omega_q = \sum_{t \in d \cap q} \omega_{d,t} \cdot \omega_{q,t}.$$

One classical interpretation of these formulations, for document impacts in particular, is that they are the product of a term frequency *TF* component and an inverse document frequency component *IDF*, representing respectively the importance of the term in the document, and the importance of the term in the collection. We will return to the TF-IDF notion below.

2. APPROACH

Anh and Moffat [2002] describe an implementation of the vector space model that supports more efficient query evaluation than the standard mechanism. At indexing time, for each document d and

term $t \in d$, the cosine impact $\omega_{d,t}^c$ in Formula 1 is calculated, adjusted via a scaling process to give it certain characteristics, and then quantized to an integer in the range from 1 to $k = 32$. In this method the impact values are assigned in the context of the whole document collection, since statistics for the whole collection are used in the computations. In particular, the transformation of $\omega_{d,t}^c$ makes use of minimum and maximum values over the whole collection:

$$L = \min_{d \in D, t \in T_d} \omega_{d,t}^c,$$

and

$$U = \max_{d \in D, t \in T_d} \omega_{d,t}^c.$$

In this work the transformed integer values suggested by Anh and Moffat are denoted by $\omega_{d,t}^g$, and their scoring method as *Global-By-Value*, summarizing its key attributes – the evaluation is global, and the assignment of impacts is accomplished using a transformation based on numeric values.

Impact transformation improves on the standard cosine mechanisms in two key areas. First, since the impacts $\omega_{d,t}^g$ are small integers, they can be stored directly, rather than $f_{d,t}$ values, thereby reducing the amount of computation required as queries are being processed. Second, there is no need to exhaustively evaluate the standard vector space similarity score. The inverted lists can be *impact-sorted*, with each inverted list structured as a sequence of equal-impact blocks, with the blocks stored in decreasing impact order. The blocks of the relevant inverted lists can then be processed in an interleaved manner in decreasing order of the product between query impact and block impact. Not all the blocks need to be processed, and effective pruning can be achieved. Moreover, since all of the impacts are small integers, the accumulation process can be combined in parallel with the selection and sort process by using a score-indexed priority queue data structure.

Anh and Moffat [2002] give experimental evidence that impact transformation can be just as effective as the standard vector space model, especially for content search in web data, where queries are typically just a few words long. They show that for the collection *wt10g* and TREC queries 451–500, the *Global-By-Value* impact transformation gives excellent retrieval effectiveness.

Here we adopt the main idea of impact transformation: the use of vector space mechanism in conjunction with precomputed integral impacts. The difference is that we eliminate the global normalization step, and instead use a qualitative ranking assignment of impacts based on localized document information, rather than a quantitative score-based one built on global collection statistics. We also seek to make this process as simple as possible. This lineage of ideas ensures that the new ranking method is efficient.

In summary then, our goal in this project was to establish the details of an impact-based similarity scoring mechanism that:

- makes use only of localized “one document at a time” information when setting the document impacts, and minimizes the role of formula tweaks and tuning factors;
- provides fast ranking, by virtue of its use of integer computations;
- is amenable to dynamic pruning; and
- provides a high level of retrieval effectiveness.

The next few sections address these desiderata.

3. QUANTITATIVE IMPACTS

The most direct way of incorporating a document-centric viewpoint into the impact definition is to employ the same process of impact transformation described by Anh and Moffat [2002], but locally within each document.

In this quantitative *score-based* approach, the cosine impacts $\omega_{d,t}^c$ are used as input values for a document-by-document transformation. The requirement of document locality means that the factor W'_d (and hence W_d) in $\omega_{d,t}^c$ is unavailable, and that we cannot incorporate the document length factor which is crucial to the original cosine similarity concept [Singhal et al., 1996]. On the other hand, nor does it necessarily reflect an abandonment of the vector space ranking principle – by using each individual document as the model for transformation, a document length factor is implicitly employed.

In operational terms, for each document $d \in D$ and term $t \in d$, $\omega_{d,t}^c$ is redefined as

$$\omega_{d,t}^c = (1 + \log_e f_{d,t}) \cdot \log_e \left(1 + \frac{f_t^m}{f_t}\right).$$

The population of $\omega_{d,t}^c$ for each $d \in D$ is then used to compute

$$L_d = \min_{t \in T_d} \omega_{d,t}^c$$

and

$$U_d = \max_{t \in T_d} \omega_{d,t}^c.$$

Next, each value $\omega_{d,t}^c$ of that population is transformed to

$$\omega_{d,t}^d = \left\lceil k \cdot \frac{\log \omega_{d,t}^c - \log L_d}{\log U_d - \log L_d + \epsilon} \right\rceil + 1,$$

where ϵ is an arbitrarily small positive number which ensures that the impacts $\omega_{d,t}^d$ do not exceed k , and k is the number of distinct impact scores, and controls the fidelity of the approximation.

Within each document d , the transformation reduces the gaps between the low and high original impacts in a logarithmic way, and allows the collective effort of a number of low impacts to have an increased chance of overcoming a single high impact value. In other words, a document that shares a larger number of common terms with the query now has a higher similarity score than previously. This transformation scheme is referred to as *Local-By-Value* – compared to *Global-By-Value*, the difference is that the initial scoring (prior to the conversion to integer impacts) is based on attributes local to each document.

It is interesting to compare the two approaches. With the *Global-By-Value* method, the largest impact value in a document d_1 might still be small compared to the largest impact in document d_2 . By way of contrast, the *Local-By-Value* method forces a re-evaluation process that adjusts the impact values so that, over the set of documents D , the largest and smallest impact values of each document are roughly the same. As an intuitive justification, consider two documents which have about the same term appearance statistics, but one of which uses many rare words while the other tends to use more “conventional” words. In this case the *Global-By-Value* method assigns a greater number of high impacts to the first document than it does to the second one. On the other hand, the *Local-By-Value* method can assign high impact values to the “conventional” words in the second document. Consequently, when a query with “conventional” words is issued, the second document is more likely to be retrieved by *Local-By-Value* than by *Global-By-Value*.

4. QUALITATIVE IMPACTS

Now suppose that we are required to manually assign an integral impact between 1 and k to each distinct term of some document. We would instinctively expect to assign a relatively small number of high values, and a relatively large number of low values – in human terms, a reflection of the observation that “ordinary” is shared among many, while “outstanding” is less frequent.

The *Local-By-Value* method, unfortunately, does not guarantee a distribution in which only a few things are deemed to be “outstanding”, and provides little control over the distribution of the assigned impact values in any given document. That is, the precise value of cosine impacts is perhaps not a good basis for retrieval, and the set of impact *ranks* (ordered positions when sorted into order) might be a better choice, thereby ensuring that the complete range of impact scores is used in every document. In this alternative regime, the term in a document with the highest score is given an impact of k , and a group of terms with low scores is given an impact of 1, with the assignment in between based on the relative position of each term in the sorted list of scores for that document.

One problem with using ranks rather than scores is the need to choose a partitioning of the n_d impacts into k groups so that the number of elements in the groups grows in the desired way. We minimize this problem by choosing a mechanism that does not employ any further tuning parameters, but does embody the “many ordinary” principle. Taking x_i to be the number of impacts of value i (where i is between 1 and k inclusive), we build a geometric sequence with $x_i \approx B \cdot x_{i+1}$. To anchor the values, we then further set $x_k = B - 1$. The solution to these two constraints is $B = (n_d + 1)^{1/k}$. All of the x_i values are then appropriately rounded to integers, with carry-forward of residual discrepancies. In practice, for typical values of k and n_d , B is between 1 and 3, meaning that zero, one, or two terms in each document are identified as being the “most important” ones, and assigned an impact of k . Table 1 gives a worked example of the computation.

Use of ranks rather than numeric scores is rather liberating, and allows *any* ordering rule on terms to be applied, of which those that result in a numeric term weight are just a small subset. To create an ordering that is mapped to ranks all that is really required is “more important than” and “less important than” guidelines, rather “this important” numeric comparators. That is, to assign impacts, a first *sorting* phase orders the term list T_d of document d in decreasing order of term contribution. Then a second *mapping* phase converts each ordered position to an integer in the range 1 to k . As the end of the process, these integer values serve as term impacts.

Once this freedom of choice is recognized, several options for ordering the terms are immediately apparent. For example, any

i	Raw target	Incoming residual	Revised target	x_i
6	1.158	0.000	1.158	1
5	2.499	0.158	2.657	3
4	5.393	−0.343	5.050	5
3	11.638	0.050	11.688	12
2	25.115	−0.312	24.802	25
1	54.198	−0.198	54.000	54

Table 1: Computing x_i , the number of impact scores of value i , for an assumed $k = 6$ and $n_d = 100$. In this example $B = 101^{1/6} \approx 2.158$, and hence a target of $B - 1 \approx 1.158$ terms are assigned an impact of 6; a target of $B(B - 1) \approx 2.499$ are assigned an impact of 5; and so on.

cosine formulation could be used to calculate a numeric sort key, as was proposed in the previous section. That method is the first one in the list below. In addition, other variants that step right away from numeric computation have been tested:

- *Local-By-Rank-(TF × IDF)*: The term list is sorted in increasing order of the localized impact used in the *Local-By-Value* method, and then converted to rank-based impact scores.
- *Local-By-Rank-(IDF, TF)*: The term list is sorted in increasing order of f_t , with ties on f_t broken using decreasing order on $f_{d,t}$ as a secondary key. In this arrangement, the *IDF* component is presumed to dominate the *TF* component; the nomenclature reflects the lexicographic sort ordering.
- *Local-By-Rank-(TF, IDF)*: The *IDF* component is a statistic of the collection, rather than of a particular document. The *Local-By-Rank-(IDF, TF)* method, in that sense, does not reflect our “one document at a time” theme. From the point of view of a single document, the *TF* component should dominate. In this third arrangement, the term list is sorted in decreasing order of $f_{d,t}$, with ties broken by using increasing order on f_t as a secondary key.
- *Local-By-Rank-(TF)*: This variant eliminates collection dependent values completely, and orders the term list solely based on $f_{d,t}$. Terms with the same $f_{d,t}$ value are all allocated the same impact value, namely the impact of the median term with this $f_{d,t}$. Of the four *Local-By-Rank* alternatives, this one might result in the final distribution of impact values within the document varying from the geometric target distribution.

One slightly complication with our schemes is that of common words. In most English prose the word “the” is the most frequent, but it probably should not be assigned an impact of k in every document, since the appearance of just two or three common words in the term list for each document then means that only $k' < k$ different impact values are available to express the importance of the more meaningful terms. To bypass this dilemma, and ensure that the high impacts are not subverted by common words, all stop words are given an artificial $f_{d,t}$ score of one during the sorting phase of the impact assignment process. The file `stoplist.orig` at the site <http://goanna.cs.rmit.edu.au/~jz/resources/stopping.zip> provides a suitable list of 600 common words, and we used that list as the basis for this adjustment to the $f_{d,t}$ values.

Finally in this section, note that the number of different f_t values is high in any non-trivial document collection. In all but the last of these four sorting rules there is only a small chance that two different terms in the same document have the same sort key value, and further tie-breaking is not critical. In the *Local-By-Rank-(TF)* method, ties are handled explicitly.

5. TRAINING

The discussion and various conjectures of the previous section can be summarized as a multi-part hypothesis:

- localized statistics provide similarity estimates that are not inferior to those derived from global statistics;
- using relative term orderings (*Local-By-Rank* methods) to determine the impacts is as good as using calculated scores (*Local-By-Value* methods);

- *TF* is more important than *IDF* in the *Local-By-Rank* methods;
- improving the impact ordering by putting stop words at the end is plausible; and
- retrieval effectiveness is consistent except when k is very small.

In this section we examine these claims, and explore the behavior of the ranking mechanism in a training environment. In particular, an appropriate value for k needs to be set. Taking k too small allows compact indexes and fast evaluation, but may harm retrieval effectiveness. Conversely, taking k too large provides more fidelity in the scoring process, but increases the size of the compressed index. The bigger hypothesis – that the new approach provides excellent retrieval effectiveness compared to the vector space model, BM25, and language model alternatives – is considered in the next section, using different collections, and once the details of our document-centric qualitative-impact system are finalized.

The dataset used in this section is *WSJ2* – a subset of *Disk2* of the TREC corpus, see trec.nist.gov. Collection *WSJ2* is a homogeneous set of documents, and contains the text of the *Wall Street Journal* for the period 1990–1992. It has around 75,000 documents totaling approximately 240 MB. To measure effectiveness, 150 short queries were formed by taking the “title” fields only from TREC topics 051–200.

We also need to specify how query impacts are computed, since they are also part of the similarity estimation process. All of the results in this paper make use of the following process:

1. For each $t \in q$, the *weight* of t in q is defined as

$$(1 + \log_e f_{q,t}) \times (\log_e (1 + f^m / f_t)) ,$$

where f^m is the maximum value of f_t over the collection, and $f_{q,t}$ is the number of times term t appears in the query.

2. The set of weights is transformed to a set of integer query impacts by linear scaling so that the maximum query term impact is exactly k .

The different procedure (to document impact computation) is required because, in general, queries cannot be expected to contain sufficiently many terms for ranks to be sensibly employed. Also worth noting is that this calculation still includes f_t in an *IDF* factor, but that it is not required until the index has been completed and queries are being processed.

The first set of experiment in this series, shown in Figure 1, compare the *Local-By-Value* and four *Local-By-Rank* methods across values of k . Three standard metrics for assessing retrieval effectiveness are in common use. The most reliable metric is average precision [Buckley and Voorhees, 2000]. However, precision at 10 documents retrieved is also included, as it is important in some searching situations. The corresponding graph for mean reciprocal rank showed a similar trend to the two shown, and is omitted.

There are several conclusions to be drawn from the graphs in Figure 1. First, three of the methods are clearly superior, except when k is small, and all three are relatively stable when k is greater than about 6. This compares well with the $k = 32$ (or $b = 5$) value noted by Anh and Moffat [2002] to obtain stability when using globally normalized impacts. As already noted, when k is small, each inverted list has only a few blocks, and the inverted index is compact.

Second, *Local-By-Rank-(IDF, TF)* performs poorly in all situations. This outcome is also somewhat surprising, since at face value

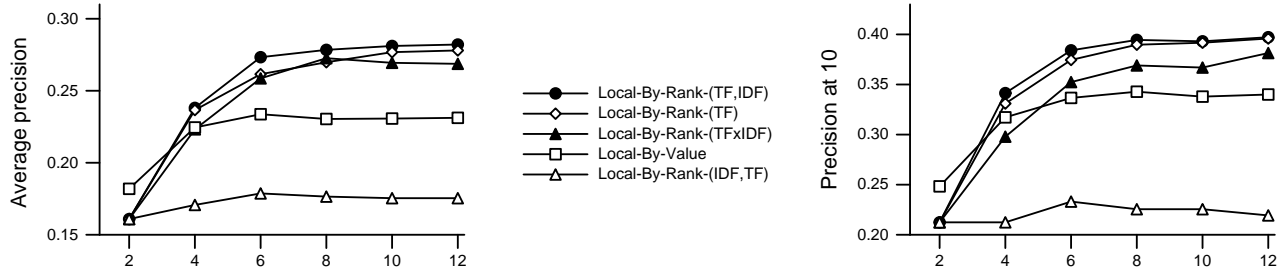


Figure 1: Retrieval effectiveness as a function of the number of distinct impact values employed, using two different effectiveness metrics, and dataset *WSJ2*. The queries are taken from the title field of TREC topics 051–200, and effectiveness values are averaged over the set of queries.

Metric	No adjustment	With $f_{d,t} = 1$
Av. Prec.	0.2781	0.2784
Prec. 10	0.3855	0.3945
R. Rank	0.6167	0.6288

Table 2: Changes in retrieval effectiveness caused by the adjustment in frequency of 600 common words, using the *Local-By-Rank-(TF, IDF)* strategy, $k = 8$, and the same collection and query combination as in Figure 1.

it means that the *IDF* factor performs only a minor role in determining the contribution made by a term. One possible explanation for the poor performance is that *IDF* takes on many more distinct values than does *TF*, and so there is only limited opportunity for *TF* to be used in a tie-breaking role. On the other hand, when the primary sort key is given by *TF*, then *IDF* can be expected to be used relatively often to break ties. That is, in *Local-By-Rank-(TF, IDF)* both of *TF* and *IDF* are likely to influence the ordering. If this is the case, then there may be further benefit to be gained by quantizing either or both of the *TF* and *IDF* components before applying the ordering step.

Third, and most satisfying, is that the methods that employ ranks to determine impacts are consistently slightly better than *Local-By-Value*, a validation of our suggestion that ranks are the more intuitive assessment.

The second set of experiments then take the *Local-By-Rank-(TF, IDF)* method with $k = 8$, and explore the suitability of the process described above for handling common words. Table 2 shows the outcomes, for the same collection and query set.

As can be seen, the stopping version has a slight (but not significant) edge in terms of effectiveness, and in the remainder of the experiments reported here we applied the $f_{d,t} = 1$ stopping scheme for common words. Note that queries containing common words can still be processed, so that the decision to stop in this way is not as irrevocable as if they were simply not indexed at all.

6. COMPARATIVE EFFECTIVENESS

In this section we compare the performance of the new retrieval mechanism to the established vector space approach using pivoted document normalization (equation 1), and the BM25 mechanism (equation 2). We have also tested the *Global-By-Value* mechanism of Anh and Moffat [2002]. All settings, including the parser used and the treatment of stop words, were identical between the three approaches; and each of these implementations was validated by comparing against other publicly-available effectiveness scores in

preliminary testing, to ensure that the implementations are comparable with those of other authors. In particular, the pivoted vector space results were compared with the BD-ACI-BCA mechanism of Zobel and Moffat [1998]; the BM25 results with those given by Jin et al. [2002]; and the *Global-By-Value* mechanism against the work of Anh and Moffat [2002]. In all of the cases we obtained closely matching (but not quite identical) levels of effectiveness

Three datasets were employed in this testing: *TREC12* (composed of TREC disk 1 and disk 2) with “title” queries from topics 051–200; *TREC45-CR* (TREC disk 4 and 5, minus the Congressional Record) with queries taken from the titles of TREC topics 351–450; and *wt10g* with “title” queries from topics 451–550. These datasets were chosen because they include some of the largest collections for TREC ad-hoc-style retrieval, and because each is accompanied by a relatively large number of topics of a diverse type and length. Information about the test data appears in the various TREC overview papers available at trec.nist.gov.

A detailed comparison is given in Table 3. Effectiveness outcomes for the *Local-By-Rank-(TF, IDF)* mechanism, for the three data sets, and for three different evaluation metrics, are presented in the first column headed “Score”, in all cases as averages over the set of queries. The three pairs of columns that follow give the effectiveness for the three competing similarity formulations, presenting first of all a comparable average score, and then a flag to indicate whether the difference between the score for the new method differs significantly from the score for that previous method. Statistical significance was carried out using the Wilcoxon signed-rank test over the effectiveness results from the individual queries, with the 95% confidence level used. A “+” in any of the three columns headed “Sig.” indicates that the new method is reliably better than the corresponding previous technique. The largest value in each row of the table is highlighted in bold to assist in the comparison.

Quite remarkably, the new method outperforms (our implementations of) both the pivoted vector space model, and also the BM25 scoring scheme. It also performs well compared to the *Global-By-Value* scheme of Anh and Moffat [2002].

We also compared the new *Local-By-Rank-(TF, IDF)* method with the published performance of the *BM25J* scoring regime (a BM25 implementation), the *LMJ* regime (a language model implementation), and the *TLM* mechanism (the best of the “title language models”), taking results from the work of Jin et al. [2002]. Four data collections were used in Jin et al.’s experiments: *AP2* (the AP sub-collection of TREC disk 2); *AP3* (the AP sub-collection of TREC disk 3); *SJM* (the sub-collection SJM of TREC data), and *WSJ2* (the sub-collection WSJ of TREC disk 2). For all four collections, TREC topics 201–250 are used as queries.

Collection	Query set	Metric	Score	Pivoted		BM25		<i>Global-By-Value</i>	
				Score	Sig.	Score	Sig.	Score	Sig.
<i>TREC12</i>	051–200	Av. Prec.	0.2196	0.1914	+	0.2021	+	0.2032	+
		Prec. 10.	0.4773	0.4497	+	0.4748		0.4480	+
		R. Rank	0.6452	0.6091	+	0.6246		0.6509	
<i>TREC45–CR</i>	351–450	Av. Prec.	0.2180	0.1409	+	0.2112		0.1995	+
		Prec. 10.	0.4470	0.3160	+	0.4460		0.4260	+
		R. Rank	0.6873	0.4552	+	0.6281	+	0.5412	+
<i>wt10g</i>	451–550	Av. Prec.	0.1907	0.1113	+	0.1667	+	0.1458	+
		Prec. 10.	0.3020	0.2051	+	0.2657	+	0.2450	+
		R. Rank	0.5429	0.3615	+	0.5060	+	0.4552	+

Table 3: Performance of different similarity scoring mechanisms. Values in the first column headed “Score” are for the *Local-By-Rank* (*TF*, *IDF*) method, with $k = 8$. They are compared against three other similarity heuristics: the pivoted cosine implementation described in equation 1; the BM25 computation described in equation 2; and the *Global-By-Value* method described by Anh and Moffat [2002]. The columns headed “Sig” contain a “+” if the new method is significantly better (at the 95% confidence level, using a Wilcoxon sign-rank test) than the comparator system; contain a “–” if the comparator is significantly better; and is blank if neither is significantly better. Three different data collections and query sets are used; the values in bold are the largest in each row.

Collection	<i>BM25J</i>	<i>LMJ</i>	<i>TLM</i>	<i>Local-By-Rank</i> (<i>TF</i> , <i>IDF</i>)
<i>AP2</i>	0.2463	0.2238	0.2667	0.3007
<i>AP3</i>	0.2511	0.2411	0.2711	0.2838
<i>SJM</i>	0.1727	0.1845	0.2081	0.2257
<i>WSJ2</i>	0.1719	0.1844	0.1950	0.2240

Table 4: *Local-By-Rank*-(*TF*, *IDF*) versus BM25 and some Language Modeling variants. The numbers reflect the mean average precision over the query set. The last column represents the new method, *Local-By-Rank*-(*TF*, *IDF*) with $k = 8$; all other values are taken from the work of Jin et al. [2002]. Numbers in bold are the largest in each row.

Results from the comparison are given in Table 4. In general, the table shows that the new ranking method give the same or better level of effectiveness as the best version of the title language model, and is also consistently better than the two other methods tested by Jin et al. – a further validation of the claims made in Table 3. Note that statistical significance testing of these results is not possible, since the query-by-query details underlying the averages for the *BM25J*, *LMJ*, and *TLM* methods are not available to us. It should further be noted that the differences between our results and those of Jin et al. might be in part or entirety a consequence of altered parsing or stemming regimes, since it is not possible for us to be certain that precisely the same experiment has been carried out.

Table 5 shows how the localized-ranking schemes would have fitted against two of the 2004 TREC Terabyte track runs. The two TREC runs listed were chosen by position from an ordering based on mean average precision of all 56 submitted runs for short queries, so that each row of the table reflects a single system. While it is clearly much easier to be competitive in arrears than at the time, we can observe that, like the 2004 TREC participants, we have trained only on earlier years’ data, and simply present, without any further iteration, the results obtained by our system on the new data. The fact that the *Local-By-Rank* approach gives effectiveness scores well into the top quartile, and within a few positions of the top of the listing, is extremely positive.

Method	Metric		
	Av. Prec.	Prec. 10	R. Rank
<i>Local-By-Rank</i> (<i>TF</i> , <i>IDF</i>)	0.2722	0.5816	0.7568
(<i>TF</i>)	0.2660	0.5878	0.7563
(<i>TF</i> \times <i>IDF</i>)	0.2508	0.5673	0.6779
Best TREC run	0.2838	0.5510	0.7156
Quartile TREC run	0.2209	0.4612	0.6641

Table 5: The various *Local-By-Rank* versus submitted TREC runs on the 2004 Terabyte collection and queries. The row labeled “TREC best” gives the effectiveness scores for the TREC run that achieved the best score for mean average precision, over all 56 TREC short query runs; and the run labeled “TREC Quartile” is the one that was 25% of the way down the same list of TREC runs when ordered by mean average precision.

7. EFFICIENCY

There are two primary aspects to retrieval efficiency: the speed at which queries can be resolved; and the volume of index space occupied by the compressed inverted file. In order to provide a baseline for measurements of these attributes, we make use of the *mg* system, available from www.cs.mu.oz.au/mg/. It employs a document-sorted inverted index storing d -gaps alternating with $f_{d,t}$ values, and processes queries using floating-point computations based around equation 1. In particular, *mg* provided the basis for the Pivoted and BM25 results shown in Table 3. The *mg* system uses a Golomb code to represent the d -gaps in the index, and an Elias gamma code for the $f_{d,t}$ values, see Witten et al. [1999] for details of these codes. While *mg* is by no means a state-of-the-art system in terms of retrieval effectiveness, it does provide a reference point against which relative retrieval speed and index storage costs can be evaluated. The *mg* system was compiled and executed using the default options for ranked querying.

The non-interleaved nature of the impact-sorted indexes used in the *Local-By-Rank* methods allows efficient representations to be used [Anh and Moffat, 2005]. Impact computations are carried out strictly on integers, and the process of computing the highest r scoring documents is also significantly eased [Anh et al., 2001].

Table 6 shows, as a fraction of the collection size, the cost of

Collection	Index size (%)	
	mg	New
<i>TREC12</i>	7.04	6.89
<i>TREC45-CR</i>	6.45	6.54
<i>wt10g</i>	4.56	4.91

Table 6: Sizes of inverted index, including vocabulary files, as a percentage of the text of the collection. The column labeled “New” is for the *Local-By-Rank-(TF, IDF)* system using $k = 8$, and with the index stored using the carry-12 method of Anh and Moffat [2005].

Collection	Queries/sec	
	mg	New
<i>TREC12</i>	11.7	37.6
<i>TREC45-CR</i>	13.3	40.4
<i>wt10g</i>	4.6	14.8

Table 7: Query throughput rates, in queries per second measured over a sequence of 10,000 random queries, using a dual 2.8 GHz Intel Xeon with 2 GB RAM. All other details are as for Table 6.

storing the impact-sorted index for some of the document collections used in the experiments reported in this paper. The small differences in storage cost arises because of two different factors: the mg system stores a document-ordered index, and changing to an impact-sorted index means that different data is stored; and because a different integer representation is used. In particular, the carry-12 method used in our implementation obtains better compression effectiveness than the Golomb code when terms have localized appearance patterns. The real message of Table 6 is that the new arrangement has no great effect on storage cost compared to what can be regarded as a benchmark system in terms of economy of storage.

Table 7 compares the same two implementations in terms of querying speed, with all experiments carried out on a dual 2.8 GHz Intel Xeon with 2 GB RAM running Debian GNU/Linux (sarge), with a 73 GB SCSI disk for system files and twelve 146 GB SCSI disks for data in a RAID-5 configuration. To construct the table, each of the systems executed a stream of 10,000 random queries containing an average of three terms each, using a single processor. To form a query, words were chosen at random from a randomly selected document in the collection, with stop words (using the same stop list) not permitted in queries. Times reflect the cost of doing TREC-style experiments, in which the top 1,000 matching documents are identified, but not retrieved.

The uniform gain in query throughput shown in Table 7 arises from three factors: the use of integer computations in the new method; the consequent use of a digital priority queue when extracting the highest-ranking answers [Anh et al., 2001]; and the carry-12 mechanism used to represent the impact-ordered index, which is faster to decode than are Golomb codes [Anh and Moffat, 2005]. As a total package, we have improved upon the mg system by a factor of around three.

The fact that the index is stored in index-sorted order allows a further benefit: dynamic pruning techniques can be easily applied to accelerate query processing. Figure 2 illustrates the benefits that are possible, by comparing the effect of three different approaches. To create the graph, the total number of document pointers involved in each of the queries was calculated, by summing the sizes of the inverted lists for the terms. The query was then executed multiple

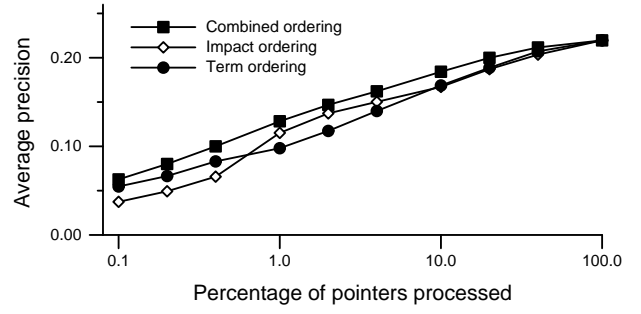


Figure 2: Effectiveness of three dynamic pruning techniques, using *Local-By-Rank-(TF, IDF)* with $k = 8$, and collection *TREC12*. The three pruning techniques are: (a) ordering the processing of pointers purely by decreasing query-term impact; (b) ordering the processing of pointers purely by decreasing document-term impact; and (c) by decreasing product of query-term impact and document-term impact.

times, stopping the evaluation after 0.1%, 0.2%, 0.4%, and so on, of the total pointers in the query had been incorporated into the ranking. At each partial evaluation level an overall ranking was produced, and the relevance judgments used to assess the retrieval effectiveness.

Three different pruning strategies were employed, where a “strategy” is an ordering of the blocks of pointers in the inverted lists for the terms in the query. The first strategy, denoted “term ordering” in the graph, is the simplest. It orders blocks strictly by the query term impact score, processing pointers in decreasing order without any interleaving of blocks from different terms. This strategy closely (but not exactly) corresponds to the usual heuristic of processing terms in a document sorted index in decreasing f_t order.

The second strategy, labeled in the graph as “impact ordering”, processes pointers from terms in decreasing document impact order. That is, all blocks of pointers with impact k are processed, multiplied by their corresponding query impacts; then all blocks with impact $k - 1$; and so on.

The third combined strategy processes blocks of pointer in decreasing order of the product of query impact and document impact – that is, in decreasing order of similarity contribution.

The monotonic growth of all three curves in the graph shows that all of the available pointers contribute to the evaluation of queries. However, around half of the available mean average precision score arises from the first just 1% of the pointers, suggesting that there is a definite role for dynamic pruning techniques. Of the three strategies, the combined one works best, in which blocks of pointers are processed in decreasing order of the product of the document and query term impacts. When effectiveness is quantified by precision at 10 documents retrieved, the bias towards early pointers is even more marked.

8. CONCLUDING REMARKS

Local impacts, assigned as a function of the term ranks rather than as a function of their scores, have several advantages.

The most striking of their benefits is the excellent retrieval effectiveness that can be obtained – outperforming our implementations of three competitive methods on a broad set of TREC ad-hoc tasks, and also outperforming published results from other authors. Their simplicity is not at the expense of quality.

In terms of querying speed, the low complexity of the *Local-By-*

Rank methods pays off handsomely. The fact that all calculations are performed on small integers, and that there is no need for query-time incorporation of document weights, allows very fast query processing. Index costs are close to what is possible with good compression schemes applied to standard document ordered indexes. That is, the localized by-rank methods described in this paper represent a genuine all-round improvement over current methods. The required index structures are also easy to build.

There are several areas in which we still hope to make progress. One is in the area of pruning strategies – the results in Figure 2 are encouraging, but not yet compelling, and we plan further experiments on long queries, with a view to devising improved dynamic pruning heuristics. A second area of future investigation is the formulation of the query-term impacts – there is still an “formula” used in that part of the computation, and it would be interesting to try and bypass that requirement too, to make the entire similarity computation a qualitative one, rather than numeric.

Acknowledgment. This work was supported by the Australian Research Council, and by the ARC Center for Perceptive and Intelligent Machines in Complex Environments.

References

- J. Allan, J. Aslam, N. Belkin, C. Buckley, J. Callan, B. Croft, S. Dumais, N. Fuhr, D. Harman, D. J. Harper, D. Hiemstra, T. Hofmann, E. Hovy, W. Kraaij, J. Lafferty, V. Lavrenko, D. Lewis, L. Liddy, R. Manmatha, A. McCallum, J. Ponte, J. Prager, D. Radev, P. Resnik, S. Robertson, R. Rosenfeld, S. Roukos, M. Sanderson, R. Schwartz, A. Singhal, A. Smeaton, H. Turtle, E. Voorhees, R. Weischedel, J. Xu, and C. Zhai. Challenges in information retrieval and language modeling: Report of a workshop held at the center for intelligent information retrieval, University of Massachusetts Amherst, September 2002. *SIGIR Forum*, 37(1): 31–47, 2002.
- V. N. Anh, O. de Kretser, and A. Moffat. Vector space ranking with effective early termination. In W.B. Croft, D.J. Harper, D.H. Kraft, and J. Zobel, editors, *Proc. 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35–42, New Orleans, Louisiana, USA, September 2001. ACM Press, New York.
- V. N. Anh and A. Moffat. Impact transformation: effective and efficient web retrieval. In Beaulieu et al. [2002], pages 3–10.
- V. N. Anh and A. Moffat. Inverted index compression using word-aligned binary codes. *Information Retrieval*, 8(1):151–166, January 2005. Source code available from www.cs.mu.oz.au/~alistair/carry/.
- M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, editors. *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, August 2002. ACM Press, New York.
- C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In N.J. Belkin, P. Ingwersen, and M. Leong, editors, *Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, Athens, Greece, September 2000. ACM Press, New York.
- A. Chowdhury, M. C. McCabe, D. Grossman, and O. Frieder. Document normalization revisited. In Beaulieu et al. [2002], pages 381–382.
- R. Jin, A. G. Hauptmann, and C. Zhai. Title language model for information retrieval. In Beaulieu et al. [2002].
- J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proc. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, Melbourne, Australia, August 1998. ACM Press, New York.
- S. E. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC–7: automatic ad hoc, filtering, VLC and filtering tracks. In Voorhees and Harman [1998], pages 253–261. URL http://trec.nist.gov/pubs/trec7/t7_proceedings.html. NIST Special Publication SP 500-242.
- A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In H. P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proc. 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. ACM Press, August 1996.
- E.M. Voorhees and D. Harman, editors. *Proc. Seventh Text REtrieval Conference (TREC–7)*, Gaithersburg, MD, November 1998. National Institute of Standards and Technology. URL http://trec.nist.gov/pubs/trec7/t7_proceedings.html. NIST Special Publication SP 500-242.
- I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco, second edition, 1999.
- J. Zobel and A. Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1):18–34, Spring 1998.