Collection-Independent Document-Centric Impacts

Vo Ngoc Anh

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010 Australia

vo@cs.mu.oz.au

Alistair Moffat

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010 Australia

alistair@cs.mu.oz.au

Abstract *An information retrieval system employs a similarity heuristic to estimate the probability that documents and queries match each other. The heuristic is usually formulated in the context of a collection, so that the relationship between each document and the collection that contains it affects the scoring used to provide the ranked set of answers in response to a query. In this paper we continue our study of document-centric similarity measures, but seek to eliminate the reliance on collection statistics in setting the document-related components of the measure. There is a direct implementation benefit of being able to do this – it means that impact-sorted inverted indexes can be built with just a single parse of the source text.*

Keywords Information Retrieval.

1 Introduction

An information retrieval system employs a similarity heuristic to estimate the probability that documents and queries match each other. The answers presented to the user are the documents that have the highest probability of being related to the query, in the hope that they are relevant to the user's information need.

The similarity heuristic is usually formulated in the context of a collection. For example, many similarity computations include a factor derived from the inverse document frequency, $1/f_t$, where f_t is the number of documents in the collection that contain the term t. This means that the relationship between each document and the collection affects the scoring used to provide the ranked set of answers, and that similarity cannot be assessed between a single document and a query.

In this paper we continue our study of document-centric similarity measures [Anh and Moffat, 2002b], but seek to eliminate the reliance on collection statistics in setting the document-related components of the measure. Reducing the reliance on collection statistics brings a direct implementation benefit – it means that impact-sorted inverted indexes can be built with just a single parse of the source text.

Proceedings of the 9th Australasian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

Previously, single-parse indexing mechanisms have been used to construct document-sorted indexes, but not the impact-sorted indexes that we have been using in our work. With the similarity formulations described in this paper, we are now able to achieve the same savings at index construction time, plus retain all of the other benefits associated with the use of impact-ordered indexes, including high ranking effectiveness, and fast processing of ranked queries.

A particular benefit of making all parts of the index independent of collection statistics is that retrieval on distributed collections can more readily be accommodated, since there is no need for any collation of global parameters for the super-collection. That is, the index for any composition of indexed collections can be built by simply concatenating the various partial indexes.

The rest of the paper is organized as follows. Section 2 reviews the local reordering technique. Section 3 then describes a number of heuristics to remove the IDF component and make the document impacts dependent solely on localized information. Section 4 presents experimental results showing that the proposed techniques work well in practice.

2 Similarity computation using impacts

The *local reordering* technique, introduced by Anh and Moffat [2002b], improves both the effectiveness and the efficiency of the ranking process. In this technique, an integer *impact* between 1 and k is associated with each term t in a document d, to indicate the "strength" or "importance" of t in d. The limit k is a small integer fixed in advance, and in our experiments is typically taken to be 10 (see [Anh and Moffat, 2002b]). The impact of t in d is denoted by $\omega_{d,t}$.

For a text collection $\mathbf{D}$ containing N documents and n distinct terms, a n-dimensional vector space is formed, with each dimension associated with one of the terms. Every document $d \in \mathbf{D}$ is represented by a *document impact vector*,

$$\Omega_d = \{\omega_{d,1}, \omega_{d,2}, \ldots, \omega_{d,n}\} ,$$

and every query q by a *query impact vector*,

$$\Omega'_q = \{\omega'_{q,1}, \omega'_{q,2}, \ldots, \omega'_{q,n}\} .$$

Document impacts and query impacts might be defined differently, which is why different notation has been used for them. If a term $t \in q$ does not have a corresponding dimension in the n-space, it is ignored.

Once the impact vectors for a document d and a query q have been formed, the similarity score $S(d, q)$ between d and q is defined as

$$\begin{aligned} S(d,q) &= \Omega_d \times \Omega'_q \\ &= \textstyle\sum_t \omega_{d,t} \cdot \omega'_{q,t} . \end{aligned}$$

In practical implementations an incomplete version of this computation is often performed, with only a partial evaluation of the inner product computed, using additional heuristics that are generically referred to as *pruning* techniques. Anh and Moffat [2002a] describe several pruning methods that can be used with impact-based similarity scores.

The local impacts introduced by Anh and Moffat are *document-centric*, since the impact $\omega_{d,t}$ of a term t in a document d is defined by comparing statistics of t with other terms that appear in the same document d. The process of assigning impacts to the terms of d consists of three phases, shown in Figure 1.

In the first *parsing* phase, documents are processed by identifying the various terms that appear in them, and recording information about them, primarily $f_{d,t}$, the number of times term t appears in document d.

Then, in the *sorting* phase, the list of distinct terms of d is rearranged into decreasing order of a *primary sort key*, with ties broken by a *secondary sort key*. In this paper the primary sort key value is always $f_{d,t}$, which represents the use of the well known TF factor. One possible secondary sort key value – as in Figure 1 – is the corresponding term IDF factor, calculated as $1/f_t$, where f_t is the collection frequency of t.

In the third *mapping* phase of assigning impacts, the ordered list of terms is partitioned used some pre-defined scheme, and divided into k consecutive non-overlapping segments numbered from k down to 1. The number of elements x_i in a segment i ($i = k \ldots 1$) is

$$x_i = (B-1) \cdot B^{k-i} ,$$

where

$$B = (n_d + 1)^{1/k} ,$$

and n_d is the number of elements in the list being partitioned – in other words, the number of distinct terms in document d. At the end of the mapping phase, each term t is assigned the integer impact i corresponding to its position in the ranked list, shown as step 3 in Figure 1. The result of this step is that a small number of terms are deemed to be of high impact in the document, and a much greater number of terms are deemed to be of low impact.

Once the impacts have been computed for each term in each document, the set of inverted lists required for the collection index can be formed. This is the fourth step shown in Figure 1. Each inverted list is ordered by decreasing impact, and because there are only k possible different impact scores for each term, the inverted list can be thought of as a sequence of k (or fewer) blocks, each of which contains document numbers d in which that term has the same impact. The result is an *impact-sorted index* that allows very fast query processing.

To process a query q, the first step is to parse the query and assign a query term impact value to each distinct term t of q. Anh [2004] described a number of schemes for determining query term impacts. Here we make use of a simpler scheme which also gives good retrieval effectiveness:

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

   $$w'_{q,t} = (1 + \log_e f_{q,t}) \times (\log_e(1 + f^{\max}/f_t)) ,$$

   where $f^{\max}$ is the maximum value of f_t over the collection.

2. The set of term weights is transformed to a set of integer impacts $\omega'_{q,t}$, by linear scaling so that the maximum query term impact is exactly k.

It is this scheme that is applied for all the experiments in this paper. Worth noting is that this calculation does still include f_t in an IDF factor, but that it is not required until the index has been completed and queries are being processed.

After query term impacts have been decided, the inverted lists for the query terms are retrieved, and their blocks are processed an interleaved manner so that blocks with high product $\omega_{d,t} \cdot \omega'_{q,t}$ are processed first. The combination of pruning techniques `Continue`, `TermFine`, and `BlockFine` (see Anh and Moffat [2002a] for details) is applied to suppress the contributions from index blocks with a low product $\omega_{d,t} \cdot \omega'_{q,t}$, and allow the process to execute quickly.

In addition to these processes, special treatment is given to stop-words so that they are always assigned the lowest impact of 1. For this purpose, a list of 600 stop-words is employed, taken from the file `stoplist.orig` that is publicly available at the site `http://goanna.cs.rmit.edu.au/~jz/resources/stopping.zip`. The whole scheme of defining impacts in this way is, as in our previous work, denoted `(TF,IDF)`.

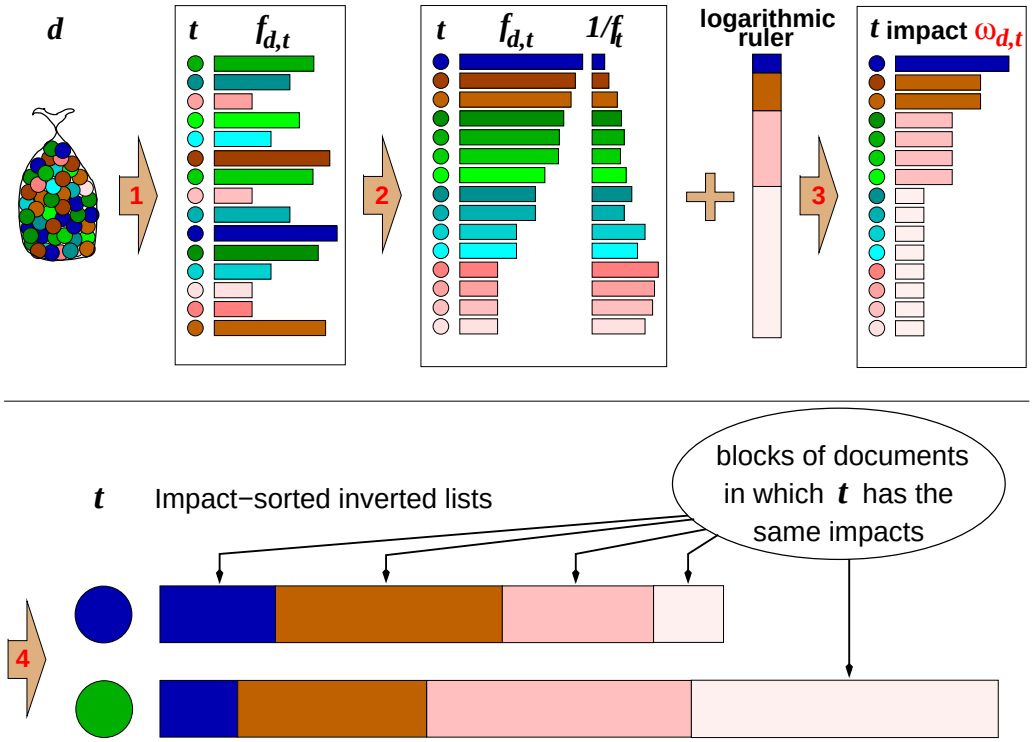

Figure 1: *Impact-sorted indexing.* Building a static-pruned impact-sorted inverted index with $k = 4$. The set of f_t values is used in step three as part of the sort key used to generate term ranks.

3 Non-IDF Impacts

One of the key factors contributing to the retrieval effectiveness of document-centric impacts is how well the sorting phase is able to arrange the terms in decreasing order of term importance. The `(TF,IDF)` technique reported above relies on the TF.IDF principle (see, for example, Salton [1989] and Witten et al. [1999]), and uses both the TF and the IDF factors as part of the sort key.

In this section we explore other heuristics for setting the document impacts. Our goal throughout is to avoid any reliance on collection-wide statistics. While this makes a partial violation of the TF.IDF principle inevitable, it is an attractive target for practical reasons, including ease of index construction.

Hence, we are interested in alternative quantities that are available while a single pass through the collection is being performed, and can stand in for IDF as an indication of a term's importance.

Independent Factors An obvious solution is to remove IDF from the computation, and replace it by a factor that is independent of not only the collection, but also the term and the document themselves. The simplest solution is sorting the term list on the decreasing order of the term frequencies alone.

The complete removal of the IDF component means that there is no secondary sorting key, which gives rise to a problem when assigning impacts to equal-frequency terms. In considering this situation, the following options are explored.

- `(TF)`: This is the simplest approach. Terms are sorted in decreasing order of their within-document frequencies. Then to deal with ties, two principles are applied. First, all terms that have the same frequency must have the same impact assigned. Second, if, according to the mapping scheme, an equal-frequency set of terms would be scattered across more than one impact segment, the average value (or the closest integer that is not less than the average) is chosen as the impact for all the terms in the set.

  This strategy means that the impact groups are not guaranteed to be as defined by the partition scheme of the mapping phase. Some groups may be larger than they should be, others smaller.

- `(TF,Random)`: This policy represents a simple and elegant way to deal with the tie problem. Distinct random numbers are generated for distinct terms and serve as the secondary key for the sorting process. In comparison with `(TF)`, the `(TF,Random)` policy does not guarantee that equal-frequency terms are assigned equal impacts. On the other hand, the number of elements in each impact group is kept precisely as was defined by the partition scheme of the mapping phase.

- (ITF): This is the dual of (TF), in that it sorts the terms in increasing (rather than decreasing) order of term frequency. It flagrantly violates the TF criterion of the TF.IDF principle, and there is no reason to believe that it will perform well. Nevertheless, inclusion of it in the experiments provides a useful litmus test – if the (ITF) method does not perform badly, then the whole notion of TF-based ordering is in doubt.

Term and Word Lengths Compared with using the IDF component as the secondary sort key, method (TF) may not provide enough discrimination to the terms, and while method (TF,Random) ensures the complete range is used, it is in a rather arbitrary way. A term attribute that can be a direct surrogate for the IDF factor might be a better option. Ideally, such an attribute would be highly correlated with IDF, while being collection-independent.

The term length (that is, number of characters in the term) is one possible candidate for this task. Terms that are used frequently (such as "sea" and "food") tend to be short, and rare terms (such as "microbiology" and "pedestrian") are normally long.

One potential issue is that in many document ranking systems, words in documents are stemmed before indexing (see, for example, Baeza-Yates and Ribeiro-Neto [1999]), and it is the stemmed words that are indexed. As a result of stemming, vocabulary terms are generally shorter than any of the words they represent.

Stemming complicates the choice of term or word length for the secondary sort key. In all the experiments conducted for this paper, a light stemming scheme has been applied. It is a variant of Porter's stemmer, but with the reduction rules condensed to cover only the regular cases of plural nouns; adverbs ending with "ly"; and verb forms ending with "s", "ed", and "ing".

Assuming some stemming policy is in place, the following options employing term lengths are considered:

- (TF,TL): The length of each stemmed term is used as the secondary sort key. The greater the length, the more important the associated term is presumed to be.

- (TF,WL): As in the case of (TF,TL), but the average unstemmed length over all appearances of words sharing the same stem is taken as the secondary sort key. Note that the average is taken locally for each document. Again, terms with longer average length are considered more important, but now words that in some variants are heavily pruned by the stemming are given extra weight.

- (TF,ITL): The inverse length of stemmed terms is used as secondary sort key. Poor performance from this option would support the idea of using term length as a surrogate for IDF.

Term Positions and Density It is likely that an ordering that better reflects our human perception of term importance can improve retrieval effectiveness, and human perception for terms in an isolated document cannot reflect any IDF component. Hence it is interesting to consider replacement of the IDF factor by a sort key that is perhaps completely uncorrelated.

One such factor is the position of the term in the document. Another factor is the degree of concentration of terms. The concentration of a certain term in a small area of text can heuristically show that this area is relevant to the term.

Based on these assumptions, three further options were explored, all based on term positions:

- (TF,ITFP): The assumption here is that important terms are likely to appear early in each document. So, the inverse position of the first appearance of terms (counting word appearances from the beginning of the document to the first appearance of this term) is used as the secondary sort key. There is one nice property of this policy – there are absolutely no ties amongst the sort key values.

- (TF,TLP): This policy is, to some extent, a dual of the previous one. Now the position of the last appearance of a term is taken as the secondary sort key. It reflects the intuition that, at least sometimes, important words reappear at the end of documents, for example, in conclusions.

- (TF,ITD): As a measure of term density, we use the inverse distance between the position of the two closest term appearances. If a term appears only once, the distance is set to the number of words in the document. If a term appears two or more times the distance is set to the number of words between its two closest appearances, with small distances deemed to be evidence of term importance. Ties are treated as in the case of the (TF) method.

Tags Many electronic documents are provided in a structured or semi-structured form incorporating SGML-like tags. Exploring this document structure might also lead to better term ordering.

A difficulty in using text structure is the great diversity of styles and markup used. A general solution for a heterogeneous collection or homogeneous collections might be hard to achieve, and careful study of a new homogeneous collection might produce better performance than can be attained by generic rules.

Figure 2 shows a typical document from the WSJ collection, which is very similar to other documents of other TREC data except for the Web data (see trec.nist.gov for descriptions of the TREC project and the TREC data).

For easy presentation, when a term appears under a tag, the tag is referred to as a "category" of the term. We can suppose that each appearance of any

```
<DOC>
     <DOCNO>
          WSJ870512-0150
     </DOCNO>
     < HL>
          Collins Foods International Sells ...
     </HL>
     <DD>
          05/12/87
     </DD>
     <SO>
          WALL STREET JOURNAL (J)
     </SO>
     <IN>
          CF TENDER OFFERS, MERGERS, ACQUISITIONS (TNM)
     </IN>
     <DATELINE>
          LOS ANGELES
     </DATELINE>
     <TEXT>
          Collins Foods International Inc. said it ...
     </TEXT>
</DOC>
```

Figure 2: *Sample* WSJ *document.* The document is an article in one of the *Wall Street Journal* editions in 1987. Most of the content part of the article has been omitted from the presentation.

term is associated with at least one category (category "DOC" in this case). In general, a term appearance is associated with a number of categories. For example, in our document model, several term appearances belong to both "DOC" and "TEXT" categories. The number of appearances of a term in a category (including the disjoint occurrences of the category) is called the term frequency in that category. The total number of word appearances in a category is called the cardinality of the category. The number of distinct categories associated with a term is called the term's diversity degree.

Various heuristics can be used with the category statistics. We explore the following simple strategies:

- `(DenseTag)`: The terms in a document are sorted in decreasing order of a complex key defined by term diversity degree and the term frequencies in the categories, where the categories are arranged in the decreasing order of their cardinality. Note that in this strategy, the primary sort key remains the within-document term frequency.

  The underlying assumptions of this strategy are: (a) the higher the diversity degree, the more valuable the term is – if it appears in the "HL" category and also in the "TEXT" category, it is more important than it would be if it appears only in one of them; and (b) the higher the cardinality, the more important the category is.

  The second assumption may not be correct for other collections with rich SGML markup, but reflects the level of markup in `WSJ`.

- `(RareTag)`: This policy is similar to `(DenseTag)`, except that the categories with low cardinality are considered as more important than the ones with higher cardinality. This policy is distinguished from all others explored in this paper in that the within-document term frequency stops being the primary sort key. Rather, it serves at the last element in the complex sort key.

- `(TF,RareTag)`: In this policy, the within-document term frequency is the primary sort key. The secondary sort key is the whole sort key which is defined for `(RareTag)`.

4 Experiments

To test the different sorting schemes, a series of experiments have been conducted. The desired outcome is an assessment of the extent to which any of the proposed heuristics are worthwhile in terms of maintaining the level of retrieval effectiveness achieved by the `(TF,IDF)` method. That is, we wish to establish whether any of the proposed methods is capable of producing retrieval effectiveness comparable with the level produced by the standard document-centric impacts which rely on collection-wide statistics.

Data The data collections and queries used for the experiments are derived from the TREC resources (see `trec.nist.gov`). The three text collections employed are (a) `WSJ` which is a set of 173,252 *Wall Street Journal* articles, supplied in Disk 1 and Disk 2 of the `TIPSTER` corpus; (b) `TREC12` which is all of the 741,856 documents on Disk 1 and Disk 2 of the `TIPSTER` corpus; and (c) `wt10g` which is a set of 1,692,096 web documents. Note that the three text collections represent three different types: homogeneous officially-printed, heterogeneous officially-printed, and web documents.

Each TREC collection is accompanied by a set of topics and relevance judgements. The topics are used to generate queries. In all of the experiments here, short queries derived from the "TITLE" field of the topics are used. A data set is then defined by the text collection name and the range of topic number. For example, the dataset `TREC12.051-200` reflects use of the collection `TREC12` and queries taken from the TREC topics numbered 051 to 200. More details of the text collections and topics can be found in the papers by Harman [1995] and Hawking [2001].

Four metrics are employed to compare the retrieval effectiveness of the different techniques: `Av.Prec`, which is the mean value (over the tested queries) of the average precision at 1,000 retrieved documents; `Prec.10`, which is the mean precision at 10 documents retrieved; `Recp.Rank`, which is the mean reciprocal rank; and `Recall`, which is the recall at 1,000 retrieved documents. More details of these effectiveness metrics, as well as their stability, can be found in Baeza-Yates and Ribeiro-Neto [1999] and Buckley and Voorhees [2000].

Retrieval Effectiveness In the first set of experiments, all of the methods were compared to the `(TF,IDF)` baseline on a query-by-query basis, over

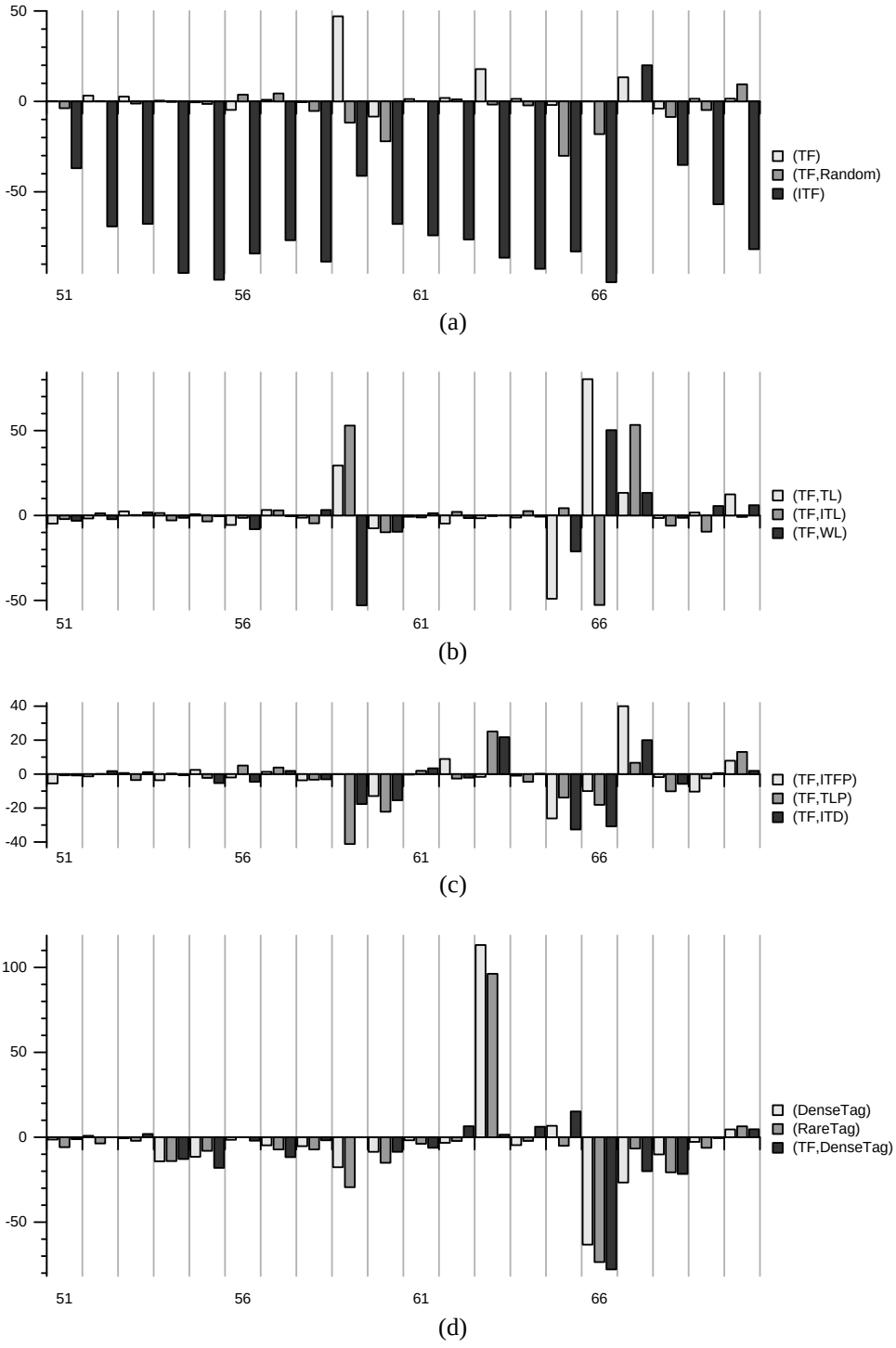

Figure 3: *Query by query relative performance.* Relative average precision for the data set `WSJ.051-070`, for variants of each of the sorting groups: (a) Independent Factors; (b) Term Lengths; (c) Term Positions; and (d) Tags. In each of the graphs, a bar for a method shows the difference between the score obtained by the method on that query and the score of the `(TF,IDF)` baseline method on that query, measured as percentage of the latter. For example, the third bar in graph (a), which has dark color, shows that for query `051`, the `(ITF)` method attains average precision 35% lower than that obtained by the baseline method.

a relatively small collection and a set of 20 queries – WSJ.051-070. The only effectiveness metric used in this preliminary testing was Av.Prec.

The outcome of the experiments is depicted in Figure 3, in which the methods are presented in their various families. The first graph in the figure shows that the method (TF) performs comparably to the baseline, and outperforms it by a considerable margin on query 059. Using the TF factor alone when determining impacts, and retaining ties rather than breaking them, is a good choice. The power of the TF factor is also demonstrated by the extremely poor performance of the (ITF) method, which reverses the (TF) sort ordering. The first graph also shows that the (TF,Random) method provides decreased effectiveness on a quarter of these twenty queries, and is somewhat inconsistent.

The bottom three graphs tell a slightly different story. For each of the methods there is at least one query in which effectiveness decreases by more than 20% relative to the baseline; and another for which effectiveness increases by 20% or more. This inconsistent behavior suggests that reliance on any single process is risky. But, as a general observation, we note that the term length heuristics are better than the term position ones, and that the heuristics based on markup tags appear to be ineffective.

Adding more queries Table 1 shows retrieval effectiveness on the same WSJ collection, but with a full set of 150 queries incorporated in the average effectiveness values. Also included as an additional reference point in Table 1 is an implementation of the pivoted cosine method [Singhal et al., 1996]. In each cluster in the table, the best results in each of the three metrics are shown in bold.

Table 1 confirms that, in an average sense, the (TF) strategy – and also several of the other approaches – compares well to the previous (TF,IDF) mechanism. Moreover, the impact-sorted mechanism outperforms the pivoted cosine approach, confirming our claims in Anh and Moffat [2002b].

Other data sets The final set of experiments measured retrieval effectiveness for all three text collections, using in each case all of the available queries; that is, WSJ.051-200, TREC12.051-200, and wt10g.451-550. The methods taken to be compared include the (TF,IDF) starting point for this investigation; (TF), which performed best in the previous experiments; (TF,WL), which represents the group of methods based on the average matching word length; (TF,ITD), which is about the best of the term position methods; and finally (DenseTag), which is a reasonable choice from the tag based methods. As an additional baseline, Figure 4 also plots a standard pivoted cosine mechanism.

Figures 4(a) and 4(b) show that all of the impact-based methods give similar effectiveness on the properly-published WSJ and TREC12 collections, regardless of whether the collections are homogeneous or heterogeneous.

Method	Effectiveness Scores		
	Av.Prec.	*Prec.10*	*Recp.Rank*
Baselines			
(Pivot)	0.2384	0.4160	0.6428
(TF,IDF)	**0.2471**	**0.4440**	**0.6572**
Independent Factors			
(TF)	**0.2459**	**0.4440**	**0.6752**
(TF,Random)	0.2399	0.4360	0.6385
(ITF)	0.0655	0.1247	0.2583
Term and Word Length			
(TF,TL)	0.2419	**0.4433**	0.6479
(TF,WL)	**0.2425**	0.4413	**0.6693**
(TF,ITL)	0.2409	0.4347	0.6416
Term Position and Density			
(TF,ITFP)	0.2432	**0.4440**	**0.6669**
(TF,TLP)	0.2393	0.4300	0.6607
(TF,ITD)	**0.2436**	0.4413	0.6509
Tags			
(DenseTag)	**0.2410**	0.4400	0.7013
(RareTag)	0.2403	**0.4493**	**0.7052**
(TF,DenseTag)	0.2331	0.4427	0.6798

Table 1: *Relative performance of different impact schemes.* Document-centric impact variants compared to two baselines on the data set WSJ.051-200. The baselines are the well-known pivoted cosine vector space measure [Singhal et al., 1996], and the document-centric impact version (TF,IDF) [Anh and Moffat, 2002b]. The maximum score attained in each group and for each effectiveness metric is shown in bold.

Figure 4(c) shows more variability. The most significant observation is that method (DenseTag) gives inferior results, when compared to the other approaches. As described earlier, (DenseTag) was designed for the traditional TREC data, and not for web documents in which the use of SGML markup is more intense.

On the other hand, Figure 4(c) shows that, except (DenseTag), all other methods (including the (TF,IDF) baseline) perform similarly for the web data set wt10g.451-550, and dominate the pivoted cosine arrangement. The (TF) method is fractionally weaker than the other three if Recp.Rank is taken as the criterion. However, this metric is often unstable, and this slight weakness may not be significant.

As an overall assessment, Figure 4 confirms the suitability of the (TF) method as a replacement for the (TF,IDF) method we previously proposed. A number of other factors can also be taken as the secondary sort key without greatly affecting average effectiveness.

5 Conclusion

Retrieval approaches based on document-centric impacts are good candidates for use in large systems, because of their simple implementation, fast execution, and compact index requirements.

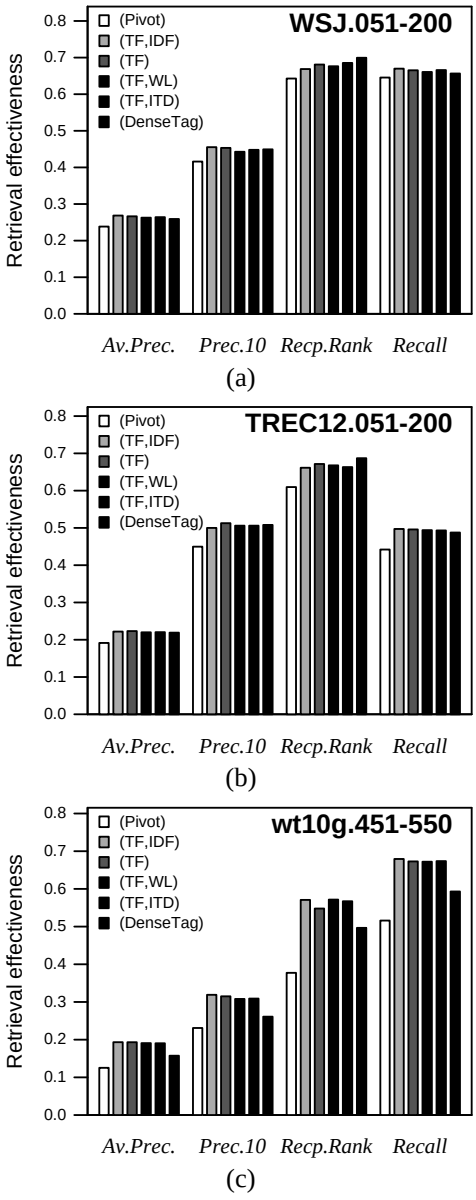

Figure 4: *Overall performance of selected document-centric impacts.* Retrieval effectiveness performance in terms of `Av.Prec`, `Prec.10`, `Recp.Rank`, and `Recall`, for (a) `WSJ.051-200`, (b) `TREC12.051-200`, and (c) `wt10g.451-500`.

This paper shows that a further desirable property can be added – reduced index construction times, achieved through the use of document impacts that are independent of the collection statistic f_t, thereby eliminating one of the two document parsing passes previously required when constructing indexes for the `(TF,IDF)` mechanism.

Collection statistics are, however, still required in the formulation of query impacts. To be truly collection independent, that reliance also needs to be circumvented. Whether it is possible to define the similarity score between a document and a query using nothing more than the document and the query themselves remains as an open – and very interesting – question.

Figure 3 also raises a further intriguing possibility – the variability of effectiveness shown in the lower three sections of the graph suggests that some form of combination of evidence may be able to outperform all of the methods we have used so far. We plan to explore that possibility as we further develop the document-centric approach.

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

References

V. Anh. *Impact-Based Document Retrieval*. PhD thesis, Department of Computer Science, The University of Melbourne, 2004.

V. Anh and A. Moffat. Impact transformation: effective and efficient web retrieval. In M. Beaulieu, R. Baeza-Yates, and S. Myaeng, editors, *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–10, Tampere, Finland, August 2002a. ACM Press, New York.

V. Anh and A. Moffat. Vector space ranking: Can we keep it simple? In J. Kay and J. Thom, editors, *Proc. 2002 Australian Document Computing Symposium*, pages 7–12, Sydney, Australia, December 2002b.

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, Addison Wesley, New York, 1999.

C. Buckley and E. Voorhees. Evaluating evaluation measure stability. In N. 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.

D. Harman. Overview of the second Text REtrieval Conference (TREC-2). *Information Processing and Management*, 31(3):271–289, May 1995.

D. Hawking. Overview of the TREC-9 Web Track. In E. Voorhees and D. Harman, editors, *The Tenth Text REtrieval Conference (TREC 2001)*, pages 87–102, Gaithersburg, MD, November 2001. National Institute of Standards and Technology (Special Publication 500-250). URL `http://trec.nist.gov/pubs/trec10/t10_proceedings.html`.

G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison Wesley, Reading, Massachusetts, 1989.

A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In H. 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.

I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco, second edition, 1999.