Index Ordering by Query-Independent Measures

Paul Ferguson and Alan F. Smeaton

CLARITY: Centre for Sensor Web Technologies,
Dublin City University

Abstract

Conventional approaches to information retrieval search through all applicable entries in an inverted file for a particular collection in order to find those documents with the highest scores. For particularly large collections this may be extremely time consuming.

A solution to this problem is to only search a limited amount of the collection at query-time, in order to speed up the retrieval process. In doing this we can also limit the loss in retrieval efficacy (in terms of accuracy of results). The way we achieve this is to firstly identify the most "important" documents within the collection, and sort documents within inverted file lists in order of this "importance". In this way we limit the amount of information to be searched at query time by eliminating documents of lesser importance, which not only makes the search more efficient, but also limits loss in retrieval accuracy. Our experiments, carried out on the TREC Terabyte collection, report significant savings, in terms of number of postings examined, without significant loss of effectiveness when based on several measures of importance used in isolation, and in combination. Our results point to several ways in which the computation cost of searching large collections of documents can be significantly reduced.

Key words:
indexing, efficiency/effectiveness tradeoffs, query-independent search

1. Introduction

As we continue to amass large collections of data, text, image, audio, video etc., for work and for leisure reasons, if we wish to search through these collections *effectively* and *efficiently* we make large demands on our search engines. An effective search engine is one that can present only relevant documents to a user in response to their search query, while an efficient search engine is one that can make the most out

of the system resources available to it, i.e. memory, hard-disk space and CPU cycles, and return results quickly. Although the effectiveness and efficiency of a system are often in conflict as it is difficult to make an increase in one without degrading the other and for the most part the two are treated independently, in our work we consider both in order to strike a balance between the two.

Web search engines have to balance many parameters including the size and growth rate of the web and the quality of the documents they index. Much of the success of the Google search engine has been attributed to the incorporation of the PageRank formula, which identifies "important" documents in the Web (based on the web's linkage structure) and gives these a prioritised ranking. We can view this importance estimation of a document as one indicator of its overall quality, and if we can discriminate between documents of higher and lower quality with a degree of confidence, we may promote high quality documents and demote documents of lower quality, in a search ranking. Furthermore, if we can identify high quality documents and search only among these at query time, then not only should we be able to improve the effectiveness of the search, but we should also reduce the amount of processing required to perform a search.

In this paper we investigate a number of measures (including PageRank) which may indicate the quality of a document in a query-independent manner. We use these to filter out documents that may not be of high quality and to promote higher quality documents. This means that the search process may be both more efficient and more effective.

We begin by describing the specific stage of the information retrieval process that we are interested in as well as detailing related research that has been done elsewhere to date, and we do this in the next section. In Section 3 we discuss the need for quality assessment in search systems and we introduce some quality assessment measures, and then in Section 4 we describe a dozen query-independent measures that may be used in detecting high quality documents in a collection.

Our experiments in Section 5 then use these measures as parameters and we assess a number of these (both taken together and in combination), in order to investigate their usefulness in reducing the collection at search time where usefulness is measured in the reduced percentage of postings list entries to be searched. We experiment with a number of different combination methods for these parameters at indexing and at retrieval time. Our findings show that the Support Vector Machine (SVM) approach provides an effective means by which to combine the query-independent measures into a sorted inverted index. Also, with the use of our most effective sorted index we can achieve at least the same retrieval performance in terms of precision at rank 10 by only examining the top 15% of the postings in the index.

The experiments are then followed by a short discussion of the impact of our contribution and we then present our conclusions and as some directions for future work in Section 6.

2. Related Work

Within an information retrieval (IR) system, the process that we are chiefly concerned with is the *document retrieval* component of the system. This process receives a query, and using this query consults an inverted index in order to generate a list of highly-scored documents which may be relevant to the query. The inverted index consists of a lexicon, which holds each of the terms in the index of the system as well as the location of the postings list associated with each of these terms. When the system executes a query, the retrieval process consults the lexicon for each term in the query. This provides the locations of each query terms' postings list, and so allows the postings to be retrieved for each term. Postings lists are then weighted and combined according to a retrieval strategy providing a list of potentially relevant documents to be returned to the user. This typically involves the examination of all postings entries associated with each of the query terms; our area of interest is in reducting of the number of these postings that need to be examined in order to produce the final list of ranked results, while at the same time still retaining a high quality list for retrieval. So, although our chief concern is in improving the performance of the document retrieval component of the IR system, in order to best support this we need to address how the inverted index is built, to allow the retrieval component to perform more effectively.

We now describe some related work on approaches to reduce the "search space", while simultaneously aiming to maintain the same level of accuracy.

2.1. Nearest Neighbour Search

In the context of information retrieval, finding the "nearest neighbours" corresponds to finding the n closest documents to a query, where "closeness" is measured by a similarity measure. This can be done naively by performing a sequential search through the complete collection and then selecting the top n documents or to use an inverted file where the record keys are index terms such as t, and record entries are the f_t individual document identifiers containing the keyword index. In order to reduce the search time, it has been suggested that query terms be evaluated in a given order, with terms with the lowest value of f_t starting first. These are the index terms with the lowest number of occurrences in the collection, and therefore (according to most term-weighted retrieval strategies) of most influence in the document ranking

process. Using this approach, inverted file records containing query term(s) with the highest f_t may not searched at all once an *upperbound* is reached [44].

An algorithm for efficiently calculating the nearest neighbours maintains two sets: R and S, where R contains all documents which at any point are candidates for the final set of nearest neighbours, and S contains all documents that have been examined so far. When processing each of the postings, if the document is in the set S it is ignored, otherwise it is added to S and a similarity value is calculated between the document and the query. If this value is higher than any of the values in R, then the document is also added to R. After processing all documents for a given term, a maximum possible similarity value of the documents containing the unprocessed query term(s) can be calculated, referred to as the *upperbound*. If this upperbound is lower than the current best value, then the process can be terminated. This means that whole postings lists for a particular term, or terms, may not need to be processed in order to find the exact nearest neighbours, so allowing faster query throughput. Although this has been shown to be effective, updating the collection by adding new documents carries an additional overhead since the maximum term frequency in each postings list must be maintained.

2.2. Self Indexing

For a conjunctive Boolean query, the maximum number of accumulators or registers that is required[1] is the frequency of the least common query term. If the query is processed in order of least to most common query term, only the documents that are common to all need to be evaluated. This self-indexing approach [32] allows inverted lists to be scanned quickly in order to retrieve these required postings. It also allows postings that are not common to all query terms processed at that point to be skipped (enabling faster evaluation of Boolean queries).

Moffat and Zobel [32] identify "the principal costs of ranking" as "the space in random access memory, and the time required to process inverted lists". In an attempt to reduce the number of accumulators necessary to process a query, they proposed two strategies: *quit* and *continue*: where an upper-bound is set on the number of accumulators that are to be used in the processing of each query. The quit strategy is carried out in a similar way to that of the nearest neighbour search: ordering query terms by decreasing weight, and then stopping once a fixed number of accumulators have been filled An alternative to the quit strategy is to continue processing after the accumulators have been filled, allowing existing accumulators to

[1]Each accumulator keeps the score of a given document as the query is being processed

be updated but not allowing new ones to be created: this is known as the *continue* strategy.

2.3. Sorted indices

As an alternative to the document-ordered index, posting lists have been sorted by various metrics in order to reduce the time and resources necessary to produce a ranked set of results in response to a query. Some of these are now described.

2.3.1. Frequency sorted

Persin [38] introduced the idea of frequency-sorted indexes, where the posting lists are each sorted by decreasing within-document term frequency $d_{t,d}$. He suggests that the majority of query time and memory is spent processing the most commonly occurring terms, which are the least informative in terms of their contribution to overall query-document similarity.

In order to filter documents at query time, two thresholds are calculated for each query term: "an insertion threshold k_{ins} and an addition threshold k_{add}, where $k_{ins} \geq k_{add}$". During processing, a $tf.idf$ score is calculated for each document in the postings list, and if this $tf.idf$ score is more than k_{ins}, the document is considered as a candidate for inclusion in the final ranking. If the document is already present as a candidate then this score is added to its previous score, otherwise a new accumulator is created for this document. If $tf.idf < k_{ins}$, but $tf.idf > k_{add}$, and if this document has an accumulator, $tf.idf$ is added to the accumulator's value. Finally, if the $tf.idf$ score is less than k_{add} this entry is ignored and the next posting is processed. The skipped documents are considered as unimportant as if a large enough number of documents are found with high similarity to the query, then those documents with small partial similarities are unlikely to have a major impact on the outcome of the final ranking. However if each of the postings lists are sorted in decreasing order of $d_{t,d}$, then there is no need to keep processing more postings once $tf.idf < k_{add}$, as all postings below this point will have the same or lower $tf.idf$.

Using this type of index it has been shown that CPU time and disk traffic can be reduced to approximately one third versus the case of an unsorted index. However, it should also be noted that Persin's experiments were carried out using queries ranging in length from 35 to 130 terms, which would allow greater gains to be achieved using this approach, rather than the much shorter queries that a user would typically use in interactive searching.

2.3.2. Impact sorted

Impact-ordered indexes (introduced by Anh et al.) [3], are a form of inverted index ordered by quantised weights (or *impacts*). This provides the facility to only

decode postings with highest impact, thereby reducing query processing time while at the same time improving retrieval effectiveness. Moffat and Zobel [32] proposed filtering documents based on the *idf* component of the *tf.idf* term-weighting scheme, similar to Persin [38] and Persin et al. [39] who sorted documents based on term frequency as mentioned above, thus allowing filtering based on the $tf.idf$. Anh et al. suggest sorting using a normalised $tf.idf$ term weighting scheme, or based on the *impact* of a term in a document, similar to Buckley and Lewit [9]. This is known as the *term impact*. In further work on the impact sorted index, [4] explored the use of ranks of the term frequencies in each document (rather than the frequencies themselves). In [2] Anh et al. explored the use of document impacts in the BM25 method, showing that for short queries an improvement in retrieval effectiveness could be achieved, while also reducing the number of tuning parameters necessary for its calculation.

Anh and Moffat also experimented with early termination schemes, such as the *quit* and *continue* schemes introduced by Moffat and Zobel [32]. Also, the use of the quantised components of their *tf.idf* similarity measure allows for faster and more memory-efficient calculation of similarity scores. However in [3], their term impact sorted index did not perform as well (on a per postings processed basis), when compared to a term frequency sorted index (as in [39]) when low numbers of postings had been processed.

2.4. Document-Centric Pruning

The Document-Centric Pruning (DCP) approach proposed by Büttcher and Clarke [10] retains only the terms in each document that are most likely to be queried. The importance of a term to a document is determined by its contribution to the document's Kullback-Leibler divergence (KLD) from the entire collection. In addition to the DCP, Büttcher and Clarke prune the most frequent terms of the (top 1,000,000 terms in [10]), resulting in a very small index which can be held in memory and is supplemented by an unpruned index which can deal with rare query terms.

Ann and Moffat [4] also calculate their impact scores in a document-centric manner, which essentially assigns impact scores to the terms within each document. Similarly Carmel and Amita [12] have explored the used of document-at-a-time processing, with successful results, using their *WAND* algorithm.

2.4.1. Access ordered

A further form of sorted index (introduced by Garcia et al. [22]), is an Access Ordered Index, in which the postings lists are ordered based on previous queries. Founded on the idea that even with a large number of different queries, the same

documents are ranked highly, and postings are ordered so that the documents returned by the retrieval system for most queries are towards the top of the lists and the less highly ranked documents are stored towards the bottom. Again, using this approach, not all postings need to be decoded as the most frequently accessed postings will be evaluated first, and so effective retrieval can be achieved using much less resources than in a conventional approach.

In order to generate the *access counts*, Garcia et al. firstly processed 1.9 million web queries from an Excite search engine log, on a collection of 1.6 million documents. For each of these queries they generated a ranked list of documents, using the Okapi BM25 formula. For every time a document occurred within the top 1000 of this ranked list for a query, its *access count* was incremented. Following the execution of these queries, each document had an associated *access count*, so allowing each postings list to be sorted in descending order, based on the document's access count.

They experimented with various query pruning techniques, again including the *quit* and *continue* strategies from [32]. They also introduced a number of other strategies to take advantage of the access ordered index, in order to limit the number of postings being processed. The most successful of these was their *MAXPOST* scheme, in which a maximum number of postings are processed from each of the postings lists for each of the terms in the query. This is the same scheme developed by us independently and reported in [19, 20], although we referred to it as *top subset* retrieval.

2.5. Index pruning

Another area to *reduce the search space* in retrieval is called *index pruning*, where pruning methods are used to remove the least important postings from the index. Pruning methods include stopword removal and static index pruning, both of which are explained below.

Since frequently occurring words do not offer much in terms of discrimination among documents they are often removed from the index. These are known as *stopwords*, and since these are the most frequently occurring terms, their elimination can result in a significant reduction in index size, up to 40%, according to [6].

The aim behind *static index pruning* is to effectively reduce the size of the inverted index in such a way that retrieval performance is not affected, or at least is limited, and it can essentially be viewed as a type of lossy compression. The way in which this is done is to identify the postings that are unlikely to affect the overall accuracy of the system.

Based on the fact that a *scoring function* assigns a score to each document for a query so that the highest scores are the most relevant, Carmel et al. [14] proposed

two methods for static index pruning. The *uniform pruning* method removes all postings entries from the index whose score falls below a threshold, while the *term-based* approach assigns thresholds to each term. Based on experiments in [14] it was found that the *term-based* approach to pruning performed best, however in [13] Carmel et al. found that the *uniform-based* approach performed better in terms of similarity with the top 10 results of an unpruned index. Also in the area of index pruning Long and Suel [31] investigate the use of pruned indexes with global page ordering using PageRank scores for documents; although their work focusses on query throughput, rather than on result quality.

2.6. Query Caching and Two-Tier Indexes

There has been a substantial amount of research carried out in the area of query caching as well as two-tier index architectures. Both these approaches attempt to maximise the queries that the search engine can respond to with information that it stores in memory, rather than resorting to a second tier which generally contains information regarding a larger number of documents than the first tier index or cache, although it is generally slower to search. Therefore the aim is to answer the user's search by only consulting the top tier or cache.

In the area of intelligent caching Lempel and Moran [30] present a probability driven approach to query caching, while Fagni et al. [18] uses a two levels of caching, with results stored in either a top-level static section or a dynamic second-level section. [51] takes the approach of improving both the query caching as well as the compression techniques used in the index to increase query throughput.

Skobeltsyn et al. [43] propose the use of static index pruning in conjunction with caching to improve performance: because a cache is employed a smaller pruned index is required for queries not dealt with by the cache. The approach used by Skobeltsyn et al. to generate a pruned index is similar to Ntoulas and Cho [34], which aims to reduce the size of the index without degrading the overall results. In an approach that is similar to the skipping used by Moffat and Zobel [32] (discussed in Section 2.2) Broder et al. [8] provide an efficient skipping of query words using their WAND operator in a two-level architecture.

2.7. Summary of Existing Search Space Reduction Techniques

From our examination of all these index modification methods it is obvious that they essentially all rely on different term weighting schemes to determine the importance of a posting entry and then choose whether to eliminate that posting from further processing based on this importance. Long and Suel [31] does offer an alternative to this by using Pagerank as a method for ordering posting lists, however as

they state, their work is focussed "on query throughput and response time, and in our experiments we try to bypass the issue of result quality which is of course very important". Alternatively our approach is focussed on result quality, looking directly at the impact that various different global sorting measures have on the result quality. This aims to produce a global measure that is more effective than standalone measures such as Pagerank and so can then be used by approaches such as [31].

Another exception to using only a term-weighting approach is the access count measure – however, if these counts are generated by a search system that relies on a term-weighting approach, as was the case in original paper by Garcia et al. [22], it could also be argued that this too may be reliant on a term-weighting strategy. Although term-weighting information is important in determining the importance of a document, we suggest that the quality of a document itself should also be considered in a query-independent manner. With regard to the related work on query caching, although some caching techniques will utilise certain quality measures such as click-through information, what we focus on is the affect that different query-independent quality measures can have on a sorted index. Although we do not discount the usefulness of caching approaches, we investigate the benefit of incorporating quality measures into a sorted index. In the next section we introduce a number of such measures.

Also to clarify, here we concentrate only on term-based pruning, rather than document centric approaches such as those described previously [10, 4, 12]. Although these have demonstrated the ability to significantly reduce the size of the inverted index, for our own purposes of examining the global importance of documents, this fits more naturally with a term-centric approach which we focus on in this work.

3. Search Quality

Information retrieval systems were originally developed to manage and retrieve information in libraries, where not only would the information have been carefully selected, but good document quality could be assumed from the reviewing process used by the publishers of documents in those libraries. Because all information would be of a high quality, retrieval approaches based purely on term distribution statistics were reliable. However this is not the case with large collections of unedited documents such as on the web, in blogs, in some enterprise collections or any of the many fora where no reviewing process takes place when publishing a document. Not only that, but as many of the queries issued to search engines may have hundreds of thousands of returned documents (that may vary greatly in terms of quality), it seems intuitive that the use of information regarding the quality of documents should be of benefit in a ranking process in order to promote higher quality retrieval.

In an effort to have their web documents ranked more highly by search engines, some malicious users create *spam* documents. Since 80% of web searchers view no more than the top 10 to 20 results [25], it is important for a web site to achieve a high ranking for user searches. Not surprisingly many operators make use of *Search Engine Optimisation* (SEO) techniques in order to gain a higher ranking. Given that the amount of spam on the web is estimated at 13.8% for English-language pages [35], the identification and elimination of such pages can save a large amount of search engine resources (in terms of indexing and retrieval), as well as improving the quality of the search results produced for the user.

The notions of quality and authority have been discussed in a number of studies into relevance criteria including "goodness" [16], "perceived quality" [37], "actual quality", "expected quality" [47] and "authority" [16], although any assessment of quality is dependent upon the needs of the individual seeking the information, as well as on the nature of the source being evaluated.

While the quality of a web site is a matter of human judgement, there are major factors influencing the notion of quality. In the late 1990s, a lot of work focused on the use of link analysis algorithms to identify high quality documents, with [36], [29] and [15], all exploiting the linkage structure of web documents. The aim of such linkage analysis measures as PageRank [36] is to "measure the relative importance of web pages", or in other words their relative quality.

Zhu and Gauch [52] identify and investigate six quality metrics to incorporate into the retrieval process: currency, availability, information-to-noise, authority, popularity and cohesiveness. They evaluated performance using a small corpus, with 40 queries taken from a query log file. They observed some improvement in mean precision, based on all the quality metrics although not all improvements were significant. The smallest individual improvements were for Popularity and Authority (both non-significant). The improvements obtained through the use of all other metrics were significant and the largest individual improvement was observed for the Information-to-noise ratio. Using all quality metrics, apart from Popularity and Authority, resulted in a 24% increase in performance over the baseline document ranking.

Amento et al. [1] evaluated a number of link and content-based algorithms, using a dataset of web documents rated for quality by human topic experts. They found that "three link-based metrics and a simple content metric do a very good job of identifying high quality items." However, surprisingly, they found that indegree performed at least as well as the more advanced HITS and PageRank algorithms, as well as a simple count of the pages on a site proving to be as successful as any link-based methods.

We now discuss in more detail some of the *query-independent* evidence that can be used to measure the quality of retrievable information.

4. Index Ordering by Query-Independent Measures

In order to effectively judge the importance of a document in a query-independent manner the use of term-frequency information alone is not ideal, as it is susceptible to the basic spamming techniques described above. We investigate the usefulness of query-independent measures for sorting an inverted index, both when used alone, and in combination. Our goal is to produce a sorting of postings entries which minimises the drop-off in retrieval accuracy as we process fewer postings in order to retrieve documents in response to a user's query maximising both *efficiency* and *effectiveness*.

We judge effectiveness by the number of document postings that the system processes rather than the time necessary to process each query. This is because we are working with an uncompressed inverted index, as the traditional means of producing a compressed inverted index is to utilise the $d-gaps$ or $run-lengths$ in the document identifiers [48], which are more suitable for compression using schemes such as those of [17] or [23]. These types of compression schemes (as well as others) have shown to have a considerable impact on retrieval speed and so comparing an uncompressed index with a compressed one would not allow for meaningful comparisons. Rather we make our evaluations based on the number of postings processed assuming that the further savings in processing speed obtainable by compression can also be applied to our reduced index.

We now describe some query-independent methods useful in identifying a document's importance and also in sorting the index.

4.1. Linkage Analysis

Two well known measures of the popularity of a document, based on the hyperlink structure among documents are Indegree and Pagerank. The *indegree* of a document is a count of the number of documents that link to it and is founded on the assumption that a link between two documents carries an implicit vote between them. PageRank [36] is a more advanced method to calculate the popularity of a document, based on the linkage structure between documents. Both these measures provide a query-independent measure of popularity or quality based on analysis of the linkage structure connecting documents.

4.2. Access Counts

As discussed in Section 2.4.1, access counts can be used as a means of determining the likelihood that a document would be returned in response to a query. Similar to Garcia et al. [22], we provide an access count measure generated from a different *Excite* query log of 2.4 million queries (from December 1999), again incrementing the access count of any document returned within the top 1,000 for each of the queries.

4.3. Information-to-noise ratio

The information-to-noise ratio can be computed as the number of terms in the document after pre-processing, divided by the total length of the document [52]. It is believed that a document with a lower information-to-noise ratio score is generally of lower quality than a document with a higher score. This may (for example) be a result of a small number of words in a document that contains a table, as well as other HTML mark-up, and consequently this document would not contain as much information for a user.

4.4. Document Cohesiveness

Document cohesiveness measures how closely related the information within a document is. Overall, information may become incoherent with the incorporation of unnecessary details, which may lead to confusion for a reader. Although incoherence may occur naturally within a document, a high level may indicate that a document is not of high quality, and a user may find it difficult to find relevant information when it is surrounded by unrelated information.

4.5. Spam

Not all web document authors are truthful about the content of their web pages, and some insert false content in order to gain increased exposure on the web. Such spam documents are unlikely to be relevant to most users' information needs thus a static spam feature, which can give a spam score to each document based on the prospect of it being spam, should be of use in order to promote non-spam documents.

4.6. Click-Through Data

For a typical query issued to a web search engine the user may only be interested in one, or a very small number of the potentially millions of documents returned. As "users do not click on links at random, but make a (somewhat) informed choice" [27], and although these clicked-on documents may not provide a perfect annotation of relevance (between a query and a document) they do provide information that may be used in order to improve the results presented to a user. A click-through is

recorded when a document in the result list (returned by an IR system) is clicked on by a user indicating that the user is making an informed decision that the document is at least of some interest.

4.7. Visitation Count

The visitation count of a document is a count of the number of times that a document is visited. A visitation count corresponds to the total number of times that a document is viewed, and not necessarily coming as a result of a search. This is quite a valuable source of information regarding the popularity of documents, however is also quite difficult to obtain as visitation counts can generally only be obtained from internet providers, proxy servers, web browsers or operating systems, and due to the potentially sensitive nature of this type of private information it is generally kept confidential.

4.8. Document Structure and Layout

One other important aspect of query-independent document importance (that is often overlooked from a IR point of view) is the structure and layout of a page, and how this relates to the quality of a document. We consider documents that follow certain text layouts to be higher quality than those with the same content in a less well organised fashion.

4.9. HTML Correctness

The level of detail contained within the HTML structure of a web document may provide an insight into the level of quality of that document, in a query-independent way. If a document is compatible with the HTML specifications [45], this gives an indication of a higher quality document. If a document contains errors, such as wrongly named tags, this could indicate that the document is of lower quality.

In order to investigate how useful the correctness of a document's HTML is at indicating the quality of that document, we processed each document in the collection, looking for compliance with the HTML specifications as defined by the World Wide Web Consortium (W3C). This generated two lists, the number of errors and the number of warnings for each document in the collection, where an error corresponds to a serious error such as an HTML tag in the document that is not in the W3C specifications, whereas a warning corresponds to something less severe such as an absent closing tag for a specific element.

4.10. Document Currency

Document currency deals with how current, or up-to-date a document is. In its simplest form this can take into account the date a document was last updated.

4.11. URL Information

Important information can be extracted from the actual URL of a web document. The hierarchal structure that is inherent in the URL is often ignored, even though this is quite important and is often viewed as being closely related to the link structure of the Web [49]. Due to the hierarchal structure that is used to organise documents, those at the root of directories are usually more general (in terms of content) than those found at the lower levels.

One simple query-independent measure derived from a web document, is its URL depth. We view documents with a shorter URL depth as being more authorative (within the overall context of the web). In our experiments we use this to assign an importance measure to each document, derived from the *URL depth* as well as the *URL length* (with the length measured in characters).

4.12. Document Length

In calculating the likelihood that a document is relevant to a query, the length of the document may have an impact on this, taking for example ranking methods such as BM25, although the fact that one document is longer than another is not necessarily an indication that one is of higher quality.

4.13. Term-Specific Sorting

There are other query-independent measures for document importance based on the index term occurring within them. For instance, a simple measure of significance for a document, on a term-by-term basis is the number of times that term occurs in the document. Persin [38] and Persin et al. [39] have used this measure to sort the postings for each term. A more sophisticated way to measure the importance of a document (on a term-by-term basis) is to use a term weighting approach, such as BM25. For an adhoc retrieval task and ignoring any repetition of terms in the query (as is the case for the vast majority of web queries) this function can be written as:

$$BM25(q,d) = \sum_{t \epsilon q} log\left(\frac{N - df_i + 0.5}{df_i + 0.5}\right) \times \frac{(k_1+1)tf_i}{k_1((1-b) + b\frac{dl}{avdl}) + tf_i} \qquad (1)$$

where df_i is the number of documents in the collection that contain the term i, and k_1 , k_3 and b are parameters. When applied to each document (in each term's postings list separately) it provides a pre-calculated BM25 score for each posting list. For a single term query this will provide the same ranking of documents as that generated by BM25 at query time. The only missing element from calculating a full query-dependent score (for multi-term queries) is the summation of each query term's scores. This of course is something that can only be done at query execution

time. Therefore it is as close as we can get to a query-dependent estimation of the query-dependent BM25 score.

If we are only concerned with calculating a score that is only to be used as a means by which to sort postings and this is done on a term-by-term basis, there is then no need to include any global term information, such as df_i (the number of documents in the collection that contain the term i). This leaves the simplified calculation as:

$$BM25(d,i) = \frac{(k_1+1)tf_i}{k_1((1-b)+b\frac{dl}{avdl})+tf_i} \tag{2}$$

which gives a relative measure of importance for each document in each postings list.

4.13.1. Global BM25 Sorting

Although the previous BM25 sorting is carried out on a term-by-term basis, we can also provide a static BM25 score of all the documents in the collection. These scores are calculated similarly to the way in which term-specific scores are generated, however we also keep track of the scores accumulated by each document and increment their score after each term is processed. For this we use the BM25 formula, as in equation 1, which takes into account the relative importance of each of the terms, unlike the scores calculated on a single term basis.

However, this basic *global BM25* measure is likely to underachieve due to its up-weighting of infrequently occurring terms in the collection, due to the idf component of the BM25 formula. This gives a higher weight to terms that occur less frequently within a collection, which is beneficial when we do not want to give high weights to very frequently occurring documents, however it also gives abnormally high weights to very infrequently occurring terms within a collection. As described by Zipf's Law [53] the most important terms are those that occur a moderate number of times. Usually the case where the terms occur too frequently are taken into account with the removal of stopwords, however identifying a pre-defined list of infrequently occurring terms to be removed is more difficult to generate in a collection-independent manner at least. In order to investigate the effect of this on the generation of our global BM25 measure, we no longer consider terms that do not occur more than a specified number of times within the collection. Varying this threshold gives us a clearer indication of the effect of including infrequent terms on the generation of a static BM25 measure.

We considered at what point would it be reasonable to impose such a threshold, so that unimportant terms are not considered yet important rare terms are still considered. If we consider the terms that users are searching for as the most important

in the collection, we can use these to determine and impose a similar threshold as before. In order to do this we used a large query log (of 2.4 million queries) from the *Excite* search engine from December 1999. We gathered all terms from the query log and generated two alternative global BM25 measures based on the term statistics of these queries.

- **global BM25 Query Log Terms (QLT):** with this approach we generate the global BM25 scores by including only terms used in the query log. This is a similar approach to the previous approach of using a threshold – except that rather than the threshold being imposed based on the global term frequency, it is imposed based on the occurrence of the term in the query log.
- **global BM25 Query Log Term Frequency (QLTF):** having removed the most commonly occurring terms, we then consider the frequency that each term occurs within the query log when calculating term importance (i.e. idf component of the BM25 calculation).

We now report our experimental findings when using different measures to sort inverted file postings entries.

5. Experiments

We evaluated the effectiveness of sorting an inverted index by a number of query-independent measures in a set of experiments which we now describe. We first use single measures, then later combine these together. We isolated the impact of using a sorted index by comparing its results with that of a standard inverted index. We first describe the experimental setup that we used, including the search engine, the document collection and queries. In Section 5.2 we compare the effectiveness of different query-independent measures, both static and term-based. Section 5.3 examines how to evaluate the effectiveness of different sorted indexes. Section 5.4 demonstrates the effect of combining different query-independent measures. Section 5.5 examines the trade-offs that are made by eliminating postings with this sorted index approach. Section 5.6 investigates the impact that a sorted index has on the effectiveness of the system, as well as the efficiency.

5.1. Experimental Setup

Our experiments are carried out with the Físréal search engine [19] using a single search engine architecture in order to alleviate any additional factors that might come

into play while using a distributed setup. The indexing and retrieval processing was carried out a Pentium 4, 2.6 GHz PC with 1.5 GB of RAM.

The experiments were carried out on the GOV2 test collection from TREC, using the queries (topic title only) from the TREC Terabyte track from 2004, 2005 and 2006. In order to compare sorting metrics, we provide a *baseline* measure to sort the postings within each postings list in the inverted index by the BM25 formula (as shown in equation 2) as well as a random sorting generated by assigning to each of the documents in the collection a unique and randomly generated number.

In previous experiments we found that the most effective way to select postings entries to process for each query term was to select the same (maximum) number of postings from each query term, regardless of the overall size of that term's list. As the postings lists would be sorted, based on some measure, then those most likely to be relevant (according to that measure) would be processed. We found this approach worked better than choosing a variable number of postings to process based a percentage or the overall size of the postings list, or based on terminating at a certain score. This is the same approach that we used previously in [19, 20], which was referred to as "top subset retrieval", and is essentially the same as the *maxpost* approach which Garcia et al. [22] also found to work best for early termination. In an effort to consolidate the two, we will refer to the maximum number of postings to be processed (for any one term), as the "maximum postings size".

5.2. Query-Independent Sorting

We now present experiments using query-independent sorting measures introduced previously. We first show how these measures perform individually, presenting the performance of both the BM25 sorted index, as well as an inverted index whose postings are sorted in a random order, to provide upper and lower baseline sortings respectively. We present the performance of each of the indexes in terms of MAP and P10 as we increase the *maximum postings size* from 10,000 per query term up to the point where all postings are processed for all query-terms. This effectively allows a sorting measure to show its performance as the number of postings are increased, allowing for comparison of different sorted indexes based on their retrieval accuracy versus the number of postings processed.

We evaluate the efficacy of BM25 as well as other query-independent measures in this section using the topics 701-850 from the TREC Terabyte ad hoc search task. Note that although each of the inverted indexes are sorted using different measures, at retrieval time we use the same BM25 retrieval strategy.

5.2.1. Term-Specific Sorting

Term-specific methods of sorting postings lists provide scores for documents on a term-by-term basis, rather than providing a single static score for each document. BM25 sorting, as previously discussed, is an example of a term-specific means of sorting these posting lists. In order to compare the performance of this against other term-specific methods we look at the performance of an index sorted based on the within-document term frequency (TF): i.e. TF being the number of times that the term occurs within a document. This was the type of index proposed by Persin et al. [39]. The performance of this measure is shown in Table 1 where once again we see excellent performance, though slightly below that of sorting by BM25.

		10k	**50k**	**100k**	**200k**	**300k**	**400k**	**all**
MAP	BM25	**0.2028**	**0.2558**	**0.2734**	**0.2869**	**0.2915**	**0.2965**	0.3106
	TF	0.1864	0.2402	0.2613	0.2775	0.2847	0.2891	0.3106
P10	BM25	**0.4839**	**0.5450**	0.5470	0.5597	**0.5691**	**0.5718**	0.5732
	TF	0.4812	0.5315	**0.5503**	**0.5658**	0.5671	0.5664	0.5732

Table 1: MAP and P10 performance of BM25 and TF sorting.

5.2.2. Static Sorting

In contrast to the term-specific methods of sorting, the static (or global) measures provide a score associated with each document and so remain static across different terms. Because of this they are unlikely to prove as effective as the term-specific means of sorting, however the aim is to eventually combine these with the term-specific measures.

5.2.3. Comparing Standalone Measures

By comparing the results in Tables 1 and 2 we can clearly see that the term-specific sortings perform better than any of the static measures. This is not particularly surprising, as we expect that a sorting that is tailored for each specific term should out-perform a single global sorting. In general from these results we can see that some of these, such as sorting by access counts, offer real tradeoff possibilities of performance against effectiveness. The next step is to see how these perform when used in combination

From Table 1 we observe that high P10 scores can be achieved processing relatively few of the postings; Table 3 shows the percentage of all postings processed at each cut-off point and looking at the cut-off point of 200,000 for example, we see that

		10k	**50k**	**100k**	**200k**	**300k**	**400k**	**all**
MAP	Random	0.0727	0.142	0.1587	0.1878	0.2055	0.2204	0.3106
	Glob. BM25	0.0789	0.1406	0.1742	0.2113	0.2373	0.2477	0.3106
	Glob. BM25/QLT	0.0878	0.1579	0.1837	0.2237	0.2424	0.2554	0.3106
	Glob. BM25/QLTF	0.0904	0.1519	0.1806	0.2171	0.2367	0.2486	0.3106
	Indegree	0.0721	0.1338	0.1579	0.1850	0.1970	0.2071	0.3106
	PageRank	0.0786	0.1433	0.1700	0.2078	0.2275	0.2429	0.3106
	Access counts	**0.0980**	**0.1710**	**0.2074**	**0.2405**	**0.2633**	**0.2763**	0.3106
	URL depth	0.0841	0.1432	0.1756	0.2139	0.2319	0.2434	0.3106
	URL length	0.0864	0.1492	0.1802	0.2138	0.2323	0.2491	0.3106
	Info-to-noise	0.0844	0.1467	0.1721	0.2154	0.2392	0.2531	0.3106
	Warning no.	0.0795	0.1398	0.1687	0.2014	0.2210	0.2357	0.3106
	Error no.	0.0616	0.1243	0.1515	0.1728	0.1906	0.2020	0.3106
	Doc. length	0.0803	0.1426	0.1713	0.2086	0.2280	0.2394	0.3106
P10	Random	0.2517	0.392	0.4376	0.4805	0.5128	0.5248	0.5732
	Glob. BM25	0.2295	0.4027	0.4705	0.5470	0.5564	0.5503	0.5732
	Glob. BM25/QLT	**0.2980**	0.4564	0.4993	**0.5577**	0.5604	**0.5705**	0.5732
	Glob. BM25/QLTF	0.2879	0.3966	0.4711	0.5195	0.5470	0.5497	0.5732
	Indegree	0.2584	0.3624	0.3933	0.4309	0.4416	0.4517	0.5732
	PageRank	0.2866	0.4134	0.4664	0.5101	0.5430	0.5537	0.5732
	Access counts	0.2826	**0.4570**	**0.5107**	0.5490	**0.5611**	0.5678	0.5732
	URL depth	0.2490	0.3805	0.4208	0.4839	0.5047	0.5235	0.5732
	URL length	0.2906	0.4020	0.4537	0.5067	0.5215	0.5329	0.5732
	Info-to-noise	0.2557	0.4054	0.4611	0.5450	0.5497	0.5597	0.5732
	Warning no.	0.2557	0.3859	0.4356	0.4987	0.5315	0.5456	0.5732
	Error no.	0.1309	0.2517	0.3013	0.3235	0.3705	0.3987	0.5732
	Doc. length	0.2403	0.4074	0.4570	0.5242	0.5409	0.5470	0.5732

Table 2: MAP and P10 performance of static sorting measures.

less than 14% of all postings are evaluated, As can be seen from Table 1, this results in a drop-off in MAP performance of only 0.0238 (7.6%) vs. MAP performance when all postings are processed. Looking at the trade-off in P10 at the same cut-off point results in an even smaller degradation in performance of only 0.0135 (2.3%).

	10k	**50k**	**100k**	**200k**	**300k**	**400k**	**all**
% Processed	0.86%	3.92%	7.34%	13.48%	19.01%	24.02%	100%

Table 3: Percentage of postings processed at each cut-off point.

As there is a significant saving in terms of the number of postings entries evaluated, we must look at the trade-off in terms of retrieval accuracy and decide if this trade-off is worthwhile. For web search applications in particular, where users are often only concerned with the top ranked documents and want results returned quickly, we suggest that a general web user would prefer to receive faster results from a search engine, and be willing to accept a small degradation in accuracy. Also looking at Table 2 we observe that for some of the top performing measures such as Global BM25/QLT, again from the 200,000 threshold onwards there is only a relatively small gain in P10, even though much more postings are being evaluated for each query. This indicates that with an effective combination of static measures it maybe possible to reduce the number of posting that are being examined without degrading the accuracy of the results.

5.3. Index Creation and Evaluation

In order to produce an evaluation of a measure for sorting posting indices using MAP and P10 we first must generate a complete sorted inverted index and then run a set of queries through the search system which are then evaluated using the query relevance judgements (QRELs) provided by TREC. This process is quite time consuming due, in most part, to the creation of the inverted index, which also takes up a considerable amount of disk-space for large collections like the Terabyte collection from TREC. This may be acceptable in evaluating a limited number of query-independent measures, such as those shown in the previous section, however when we experiment with the combination of these measures we have a large number of variants to be evaluated, and it is not feasible to generate a new inverted index for each experiment even though this is the strategy used by Garcia et al. [22] and Ferguson et al. [20].

On closer analysis we find that the problem with this approach is that although the index is sorted by some measure (X), this sorting is then re-sorted by BM25 at

query-time in order to produce the ranked set of results. The sorting by measure X therefore becomes more and more diluted as the maximum number of postings being evaluated for each term is increased. Also, if for example the sorting produced by X is effective at promoting highly relevant documents, and if these documents are ranked lowly by BM25, then these will be pushed down its ranked list and so will not allow for a proper evaluation of the sorting produced by measure X.

An alternative approach suggests that the system looks only for the known relevant documents for particular queries in the postings lists of the query terms and evaluates how highly measure X ranks those documents. Using this approach eliminates the dilution of the sorting produced by measure X when combined with an additional sorting algorithm at query-time. We call this evaluation *Index Precision (IP)*, as it calculates a precision score by analysing the postings lists of the inverted index. This is similar to the way in which traditional precision is calculated for a list of search results, except that now the sorting is calculated on the ranked postings instead. This score is calculated over each query-term's postings list as follows:

$$IP = \frac{\sum_{r=1}^{n} \frac{nrf_r}{r}}{n} \tag{3}$$

where n is the number of postings associated with the term being processed, r is the current position (or rank) in the postings list and nrf_r is the number of relevant documents that have been found up to the current rank r. To get the average IP score for a query we divide by the number of postings lists that were evaluated for that query.

In addition to the benefit of providing a precision score directly correlating to how highly a sorting measure ranks relevant documents, this also dramatically reduces the time required to evaluate a postings sorting measure because we can evaluate without creating a new inverted index.

Before we combine index sorting measures we reduced them to the following on the basis that these are the strongest (based on IP scores) and the ones eliminated are variants of the ones retained anyway: Access counts, Document length, Error number, Global BM25/QLTF, Indegree, Information-to-noise ratio, PageRank, URL length, Warning number.

5.4. Combination of Measures for Ranking Postings Entries

We experimented with different types of combination approaches, including linear combination, as well as combination using a Dempster-Shafer (D-S) approach, Here we present the results of combination using Support Vector Machines (SVMs), which seems to work best.

SVMs are a set of machine learning algorithms that have been used in the areas of *classification* and *regression*. SVMs were first suggested by Vapnik in the 1960s for classification, and were formally introduced by Boser et al. [7].

The binary independence model, introduced by Robertson and Spärck Jones [42], viewed IR as a classification problem. They considered retrieval in terms of classifying documents into two classes: relevant or non-relevant. Their approach essentially classified documents into these two classes at query-time, basing their classification on the query. Adopting this viewpoint of classifying documents as relevant or non-relevant in a query-independent manner, we use SVMs in order to solve this binary classification problem.

Providing suitable training examples for both relevant and non-relevant documents allows us to extract the scores for these documents from suitable query-independent features for use in estimating quality documents (e.g. PageRank and info-to-noise ratio, etc.). The SVM can then use these training examples to generate a model that can be used to estimate a document's likelihood to belong to either the class of relevant or non-relevant documents. We use the SVM^{light} software package [26], which is an implementation of the SVM described in [46].

Focusing on the use of SVMs for two-class classification, here the SVM learns by example to assign labels, e.g. either positive (+1), or negative (-1), to data. In order to do this, the SVM is firstly given a set of l *training* examples in the form $(\mathbf{x}_i, y_i)$, where i = 1, 2, ..., l, each $\mathbf{x}_i$ is a p-dimensional real vector (list of p numbers), and $y \in \{1, -1\}$. The task for the SVM is to learn the mapping from $\mathbf{x} \rightarrow y$, choosing from the set of all possible hypotheses the one that minimises the risk of error in the classification of a new (unseen) example.

We use the SVM to classify documents into relevant and non-relevant sets. One of the major difficulties to overcome when using this approach is how to generate representative sets of relevant and non-relevant documents in order to generate an effective SVM model to classify documents. For this we must choose a source of representative high quality documents, as well as a source of low quality documents. Then from these two sources we then take two different sets of documents to build the SVM's training and testing files.

The problem of choosing representative high quality documents can be solved by using the known-relevant documents associated with a set of TREC topics. Using these we generated a set of *positive training* examples from the TREC topics (701-750) and a different set of *positive testing* examples from a different set of TREC topics (751-800). The problem of generating a set of *negative* (i.e. low quality) documents is a more difficult challenge. As a first attempt we could use the documents returned by TREC for each topic and marked as non-relevant, however these would

not be representative because they were returned by at least one TREC participating group's system in response to that topic. So although these documents are not relevant for the specified topic they should not be considered as the least likely to be relevant (of all documents) to any topic. One possible way of generating more representative negative examples would be to look again at the access count measure though now we are looking for documents with the least number of access counts. There are a large number of documents that do not receive any access count for the 2.4 million queries that we used and so from these we randomly generate two different lists (*negative training* and *negative testing*) from the set of documents with zero access counts. As an alternative to these negative examples we also generated *negative training* and *negative testing* sets in a similar way from the lowest-scored documents in the global BM25/QLTF measure. In order to assess the effectiveness of these negative examples we also generated alternative negative sets for training and testing using a random selection of documents from the entire collection. Note that all these negative example sets are the same size as the positive example sets to give an equal distribution of positive and negative documents – in order to prevent the class imbalance problem [41]. This leaves us with four alternative training and testing sets, where the positive examples are the same but the negative documents differ:

- Positive + Negative QRELs (PN)
- Positive + Low Access counts (PLA)
- Positive + Global BM25 QLTF (PBM)
- Positive + Random (PR)

We can now generate an SVM model and assess the models' effectiveness in correctly classifying documents in the testing set.

5.4.1. Classification Training

The conventional approach to training an SVM for the task of classification is to generate an SVM model using the training set of positive and negative example documents, and then use this model to classify the documents in the testing sets. Based on the accuracy of the model at classifying the documents in the testing set we can assess the accuracy of the model (for classification purposes) and tune the SVM's parameters accordingly – using SVM^{light} [26] with a linear kernel.

Using each of these SVM models we can then *classify* all the documents in the collection, which gives us a likelihood score for each document that they should

belong to the positive or negative class. We then use these scores as a new query-independent measure, which we can use to re-sort an index and evaluate their IP scores.

In order to evaluate these we then calculated the average IP scores for each of these over the same topics that we used in the testing phase for the SVMs (topics 751-800) – these average IP scores are displayed in Table 4.

Measure	**Average IP Scores**
PN	0.0155
PLA	0.0181
PBM	0.0167
PR	0.0180

Table 4: IP Scores trained using classification accuracy.

To allow us to create more effective SVM models we should turn our attention from tuning based on the classification performance, to tuning based on the sorting performance, and in order to measure this (sorting performance) we use the IP measure.

5.4.2. IP Training

This is similar to the previous experiment in that we use the same SVM training files, however, instead of running the SVM classification process on the appropriate testing set we classify all the documents in the collection and generate a query-independent measure. We then evaluate the effectiveness of this measure by re-sorting the inverted index using this measure and calculate an average IP score over a set of topics. This is obviously a much more computationally expensive process as we generate several query-independent measures for each approach (PLA, PBM and PR) in order to tune the SVM's parameters, however, we believe that this will give us a more effective sorting measure.

As can be seen from Table 5, the performance of the different approaches differ somewhat from the previous approach of tuning parameters based on the classification accuracy. In particular PR performs quite well in comparison to both the PBM and the PLA approaches.

We now introduce the RBF kernel into the SVM, so that the input vectors may be mapped into a higher dimensional space where separation between positive and negative examples may be more successful. For this we tuned the parameters C (trade-off between training error and margin) and g (gamma in rbf kernel) using the

IP score as previously described. Table 5 shows a comparison between the three approaches using the linear kernel (as before) and now using a RBF kernel.

Measure	**Average IP Scores**
PLA (linear)	0.0175
PLA (RBF)	**0.0184**
PBM (linear)	0.0167
PBM (RBF)	0.0183
PR (linear)	0.0180
PR (RBF)	0.0188

Table 5: Training SVM using IP scores with a RBF kernel.

It is clear from Table 5 that the RBF kernel improves results significantly over the linear kernel and the PR approach performs the best, followed by PLA and then PBM.

Although the PR approach produces the best results here, we believe it has a distinct disadvantage, in that it is tuned very specifically to a certain set of positive and negative examples and for the (randomly selected) negative examples this is probably an untrue reflection of a correct selection of more accurate negative examples. When we evaluate the average IP scores on a set of unseen topics (801-850) as in Table 12, we see that the performance of the PR approach degrades significantly in comparison to both PLA and PBM, which is as expected, since it was trained to an unrealistic set of negative examples.

Measure	**Average IP Scores**
PLA	**0.0073**
PBM	0.0067
PR	0.0055

Table 6: IP scores with a RBF kernel on an unseen set of topics.

5.4.3. Combining at Index Creation Time

Having combined the static measures into a single measure we now combine this with a high-performing term-specfic measure such as BM25. In order to combine these, like combining the static measures, we can choose from a number of combination methods, but we focus on the use of the D-S method of combination to combine

the term-specific and static measures. We do this not only because of its prior success, but also as a new inverted index would need to be created for each combination that we investigate, and as this is a very time consuming process. Our D-S approach is applied to combining two sources in essentially the same way proposed by Jose et al. [28] as well as Plachouras and Ounis [40]

We combined the static measure PLA with BM25 for each postings list in the inverted index using the D-S approach, where we varied the weights given to each measure from 0.1 to 0.9 (in increments of 0.1). We then calculated the average IP scores for the topics 801-850 and selected the highest performing index, which is shown in Table 7.

Measure	**IP Score**
BM25	0.0351
PLA + BM25	**0.0356**

Table 7: Combining static and term-specific measures.

As the results show, the combination of the PLA measure with BM25 gains an increase in performance over that of the BM25 measure alone (with statistical significance value of 0.0028, using a paired t-test).

Having produced a method that outperforms the BM25 measure using the average IP, we now revert back to using more conventional measures such as MAP and P10 to see if the use of the IP measure that we adopted will also perform well with these.

5.4.4. Conventional IR Evaluation

Here we set out to evaluate the inverted index created using our optimal combination of BM25 with the PLA measure, and evaluate this using MAP and P10. This requires us to revert back to our previous approach of ranking documents using BM25, while limiting the number of postings that are evaluated for each query-term, using the *maximum postings size cut-off* approach (using the same cut-off thresholds). As discussed previously we feel that the cut-off point of 200,000 (postings per query-term) is a reasonable cut-off point that eliminates a large number of postings from processing, while allowing a satisfactory level of accuracy. Table 8 shows that by selecting the 200,000 cut-off point the system processes less than 15% of all the postings for each of the query-terms. This will save our system from examining a large number of postings, which will lead to a faster query-time as well as reducing the necessary system resources, such as memory and CPU clock cycles. The goal is then to limit the trade-off made between system accuracy and processing more

postings, as we process only a small number (e.g. 15%) of the overall postings for each query term.

	10k	**50k**	**100k**	**200k**	**300k**	**400k**	**all**
% Processed	1.04%	4.6%	8.4%	14.9%	20.7%	25.9%	100%

Table 8: Percentage of postings processed at each cut-off point on topics 801-850.

Table 9 shows the performance of the index in terms of MAP and P10 by using the same optimal combination of BM25 with the PLA measure (from Section 5.4.3), when compared with the standalone BM25 measure. Although the combined index does prove to be more effective, both for MAP and P10 at the 200,000 cut-off, overall there is little to separate performance from the two indexes. For MAP the two indexes perform quite similarly, while for P10 the BM25 index performs above the combined index at earlier cut-off points.

		10k	**50k**	**100k**	**200k**	**300k**	**400k**
MAP	BM25	0.1873	**0.2546**	0.2694	0.2785	0.2834	0.2903
	BM25 + PLA	**0.1926**	0.2513	**0.2702**	**0.2844**	**0.2909**	**0.2929**
P10	BM25	0.4980	**0.5460**	**0.5480**	0.5500	0.5560	0.5620
	BM25 + PLA	**0.5060**	0.5240	0.5400	**0.5700**	**0.5640**	**0.5640**

Table 9: Evaluating BM25 + PLA index using MAP and P10 (topics 801-850).

However there may be an unfair bias towards BM25 with this evaluation because in the creation of the index we re-rank the postings using a combination of the BM25 and PLA measures, and at retrieval we rank documents using only BM25 so diluting the effect of the PLA measure as we increase the size of the cut-off. In order to fairly evaluate the contribution of both measures we believe that the measures must also be combined at retrieval time.

5.4.5. Combining at Retrieval Time

At index creation stage the BM25 scores were calculated in isolation for each term, while at retrieval time BM25 combines the scores for each query term. So rather than combining after each term's scores are calculated, the combination is carried out after all BM25 calculations have been made. We use D-S for combination, and we evaluate different weights for each of the elements being combined, choosing two

weights so that they sum to 1, starting at 0 and incrementing by 0.05. In optimising these weights we favour the P10 performance because of the nature of our type of retrieval; we are trying to speed up query time while presenting the best top 10 results that we can, as for example, in a Web search scenario. Table 10 presents the MAP and P10 performance figures for this approach, where the combination occurred only in the index (*index combination*) and now in this approach where the combination is done at both the index and retrieval stages (*full combination*).

		10k	**50k**	**100k**	**200k**	**300k**	**400k**
MAP	BM25	0.1873	**0.2546**	0.2694	0.2785	0.2834	0.2903
	Index Combination	0.1926	0.2513	0.2702	**0.2844**	**0.2909**	**0.2929**
	Full Combination	**0.1930**	0.2514	**0.2707**	0.2817	0.2833	0.283
P10	BM25	0.498	**0.546**	0.548	0.55	0.556	0.562
	Index Combination	**0.506**	0.524	0.540	0.57	0.564	0.564
	Full Combination	0.504	0.534	**0.556**	**0.594**	**0.596**	**0.592**

Table 10: Evaluating BM25 + PLA at both the index and retrieval stages (topics 801-850).

Here we can see that although there is relatively little change between the scores for MAP, in the case of P10 there is a clear improvement, and yet another gain is made over standalone BM25 index. For our 200,000 cut-off point there is a 8% increase in P10 and the difference between the P10 values is also statistically significant with a value of 0.014.

5.5. Examining Performance Trade-offs

Table 11 compares the performance of a conventional IR system, where all postings are examined and then ranked using BM25, with a sorted index created using our approach which combines the SVM PLA measure with BM25 at both index creation and retrieval time and has a cutoff of 200,000 which is 15% of the postings. While this equates to a 6.9% loss in MAP, it is also an increase of 6.1% in P10 performance – even though the system is processing only 15% of the postings. This is an attractive trade-off in MAP for a search system where users are primarily concerned with the accuracy of the top 10 results, as well as the responsiveness of the system, and our approach provides both these traits. A drop in MAP performance is to be expected since we are processing far fewer documents and inevitably a small number of relevant documents will be missed, so the drop is most likely due to a

Index Type	**MAP**	**P10**
Conventional Index	**0.304**	0.560
Sorted Index (only processing 15%)	0.282	**0.594**

Table 11: Trade-off between a conventional index and a "pruned" sorted index (topics 801-850).

drop in recall. For a Web search scenario, where the typical user does not look past the top ten results this drop in recall will not be noticed.

As we are primarily interested in judging the effectiveness of our P10 performance for our fully combined sorted index, we compare this with the results of another search system. For this we look to the results produced from the publicly available *Zettair* search engine which was used by the RMIT group in the TREC terabyte track [21]. Their *Zetabm* run from their TREC 2006 submission, uses the same Okapi BM25 ranking which should be comparable to our baseline BM25 performance, while we also include their more effective *Zetadir* run which uses a language model with Dirichlet smoothing [50]. We also provide the results from three additional (officially submitted) runs from the University of Waterloo (*UwmtFadDS*) [11], Melbourne University (*MU06TBa2*) [5], IBM Haifa Research Lab (*JuruT*) [12] . These give a comparison between our proposed approach and other systems, using the same queries on the GOV2 collection: note that we selected runs that we selected runs comprising of topic title queries and without the enhancement of relevance feedback mechanisms.

Table 12 compares our baseline (using BM25 ranking, from Table 11), as well as our sorted index at the 200,000 cut-off point, with the other TREC Terabyte submissions. As we can see from this that our sorted index yield an increase in P10 performance.

Table 11 gives an objective view of the effectiveness of our system, and shows that our ranking provides a relatively strong baseline on which we have been able to subsequently improve performance. Although our MAP score is not our primary concern, for completeness Table 13 shows a comparison of our MAP performance against the same TREC runs. Again we can see that our both our baseline and our truncated sorted index perform quite competitively.

Carmel et al. [14] experienced a similar increase in P@10 for their term-based pruning method, although for them their results degraded after removed less than 40% of their index, in comparison to our reduction of 85%. Since the term-based approach that we use is designed to minimise any adverse impact to the results at the

Index Type	**P10**
Our BM25 baseline	0.560
Sorted Index (only processing 15%)	**0.594**
Zetabm	0.498
Zetadir	0.532
UwmtFadDS	0.518
MU06TBa2	0.530
JuruT	0.572

Table 12: Comparing performance with different search systems (P10) (topics 801-850)

top of the result list, by removing low-quality documents and those that are unlikely to appear in the result list, it is not necessarily surprising to see stable P10 figures with the use of a pruned index. However to see an increase in P10 performance with the use query-independent measures is encouraging, next we show how our query-independent measures can also be applied to a conventional index at query-time to improve P10 performance.

Index Type	**MAP**
Our BM25 baseline	0.304
Sorted Index (only processing 15%)	0.282
Zetabm	0.241
Zetadir	0.305
UwmtFadDS	0.288
MU06TBa2	0.289
JuruT	**0.329**

Table 13: Comparing performance with a different search systems (MAP) (topics 801-850).

5.6. Our Contribution and the Impact of the Sorted Index

In order to measure the impact that a sorted index has on improving the system performance, we combined our best static measure (PLA) with our baseline BM25 search (which processes all postings) at retrieval, so we can see if the resulting system

can achieve the same effectiveness (in terms of P10), even though it will be much less efficient as it processes all postings for each query-term. This will show us if the use of a sorted index contributes to the effectiveness of the system as well as to its efficiency.

For this we combined the BM25 scores with our PLA measure using the D-S approach in the same way as we combined our sorted index's results with PLA previously. We determined the top-performing combination using a number of weights combinations and Table 14 shows the P10 performance for the baseline BM25 ranking, as well this same baseline combined with our PLA measure at retrieval time, and our fully combined sorted index (using PLA and BM25).

Index Type	**P10 Score**
Conventional Index	0.560
Conventional Index + PLA measure	0.584
Sorted Index (processing only 15%)	**0.594**

Table 14: Impact of using a sorted index (P10) (topics 801-850).

This shows that even though the incorporation of the static measure improves performance over using BM25 alone, it still does not achieve the same level of performance as the sorted index which uses the same measure, highlighting a further usefulness of the sorted index. We believe this is because the sorted index filters out lower quality documents that a term-weighting ranking function such as BM25 does not account for and so by processing fewer postings using the sorted index, those that are not selected are less likely to be relevant and retrieval accuracy and efficiency are both improved. However, the improvement is gained by combining the PLA measure with the conventional index (only at query-time) illustrates the usefulness of effective query-independent measures, which could offer potential benefits to retrieval performance regardless of the underlying index or ranking method.

6. Conclusions and Future Work

In this paper we have presented a case for the use of query-independent measures as a means of determining the more important documents in a collection and using only those in calculating the top-ranked for retrieval. We proposed a number of static measures, and experimented with several of these in order to investigate their effectiveness in eliminating candidate documents for retrieval from inverted file postings, through the use of a sorted inverted index.

We also proposed the use of an *Index Precision* (IP) measure to calculate the effectiveness of using different query-independent measures to sort the inverted index. This is a more direct way of assessing the performance of a sorted index compared with analysing its performance in a weighted term ranking such as BM25 at various cut-off points. The Index Precision approach also eliminates the dilution effect caused by assessing a ranking after applying another query-time ranking such as BM25, and it also allows combinations to be evaluated faster, and without the need for creating additional inverted indexes.

Our experiments found that there are static measures that can be combined with a term-based measure to provide an increase in effectiveness in terms of IP. When we combined the static measures at both indexing and retrieval time, the system was also able to gain a significant increase in P10 performance, even though it processed less than 15% of all available postings for the terms in the queries compared with the conventional approach of processing all postings for each query-term.

Although the Access Count measure proved to be quite an effective measure, if this were to be used in a live search engine an up to date query log would also be needed to generate the access counts. This would attempt to ensure that currently hot or trending topics are not removed from the index. Also it may be possible to use additional sources such, Twitter for example, to supplement query logs and identify new topics of interest.

As a means of comparing the efficiency of different variants of sorted indexes that we produce, we do this based on the number of postings that are processed at query-time. Traditional *document-ordered* inverted indexes gain significant compression by taking the difference between adjacent document numbers, resulting in faster processing. With our inverted indexes the posting lists are ordered based on different query-independent measures and so these same compression schemes are no longer possible. However, once an effective measure is chosen based on the number of number of postings that are being processed this index can be pruned (for example with a *max posting size* of 200,000) and then sorted in the traditional document-ordered manner. Garcia et al. [22] have shown this same process with their *access-pruned* index, which allows time savings of 25% - 40% over a conventional index while maintaining almost the same accuracy. Clearly the same approach can be applied to our measure to generate a fully compressed pruned index to demonstrate meaningful query throughput measures based on query-time as well as saving substantial disk-space. The focus of this paper has been on the generation of a more effective query-independent measure, which we have demonstrated to outperform a number of standalone measures (such as access counts and PageRank), although the generation of standalone pruned index, or its incorporation into a two-tiered index system is

planned as part of our future work.

Similar the work by Büttcher and Clarke [10], who propose a method for pruning the index by over 90% so that it can fit into main memory with only a minor decrease in effectiveness, our approach would allow similar reductions in index size. From Table 8 we can see that at the 100,000 posting level only 8.4% of postings are being processed, then from comparing the results in Table 10 that there is only a small drop in P10 performance. It may also be possible that as an extension to document-centric pruning (such as [10]) may allow a more dynamic type of pruning, so that more information may be retained for what is perceived as being an important document, as proposed by our approach.

One of the advantages of using a sorted index (without pruning), is that it allows the cut-off point for termination to be changed. This may be of benefit when answering potentially difficult queries, where there are few relevant documents, and so using a larger (or no) cut-off point may be of benefit. This would require us to pre-determine the potential difficulty of a query and then dynamically choose the number of postings to process for each query term. There has been a significant amount of work carried out in this area: Hauff et al. [24] provide an assessment of 22 pre-retrieval query performance predictors as well as assessing these on three different TREC collections, while Mothe and Tangui [33] analyse the correlation between query difficulty and certain linguistic features. It seems likely that some of these query prediction mechanisms could be used in addition with a sorted index to provide a more adaptive search system.

We will also explore the scenario where the user may wish to specify the amount of information that is to be processed in response to the query. For example, at one extreme, for certain searches it may be important that all available documents are returned (i.e. high recall) or for some applications retrieval speed may be more important in which case pruning may be used.

Acknowledgments

This work was funded by Science Foundation Ireland as part of the CLARITY CSET, under grant numbers 03/IN.3/I361 and 07/CE/I1147.

References

[1] Amento, B., Terveen, L., and Hill, W. (2000). Does "authority" mean quality ? Predicting expert quality ratings of web documents. In *SIGIR 2000: Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*, pages 296–303. ACM Press.

[2] Anh, V., Wan, R., and Moffat, A. (2009). Term impacts as normalized term frequencies for bm25 similarity scoring. In *String Processing and Information Retrieval*, volume 5280 of *Lecture Notes in Computer Science*, pages 51–62. Springer Berlin / Heidelberg.

[3] Anh, V. N., de Kretser, O., and Moffat, A. (2001). Vector-space ranking with effective early termination. In *SIGIR 2001: Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 35–42. ACM Press.

[4] Anh, V. N. and Moffat, A. (2005). Simplified similarity scoring using term ranks. In *SIGIR 2005: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 226–233. ACM Press.

[5] Anh, V. N., Webber, W., and Moffat, A. (2006). Melbourne University at the 2006 Terabyte Track. In *TREC 2006: Proceedings of The Text REtrieval Conference*.

[6] Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

[7] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT 1992: Proceedings of the 5th annual workshop on Computational learning theory*, pages 144–152. ACM Press.

[8] Broder, A. Z., Carmel, D., Herscovici, M., Soffer, A., and Zien, J. (2003). Efficient query evaluation using a two-level retrieval process. In *CIKM 2003: Proceedings of the 12th international conference on Information and knowledge management*, pages 426–434, New York, NY, USA. ACM.

[9] Buckley, C. and Lewit, A. F. (1985). Optimization of inverted vector searches. In *SIGIR 1985: Proceedings of the 8th annual international ACM SIGIR conference on research and development in information retrieval*, pages 97 – 110.

[10] Büttcher, S. and Clarke, C. L. A. (2006). A document-centric approach to static index pruning in text retrieval systems. In *CIKM 2006: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 182–189, New York, NY, USA. ACM.

[11] Büttcher, S., Clarke, C. L. A., and Yeung, P. C. K. (2006). Index pruning and result reranking: Effects on ad-hoc retrieval and named page finding. In *TREC 2006: Proceedings of The Text REtrieval Conference*.

[12] Carmel, D. and Amitay, E. (2006). Juru at trec 2006: Taat versus daat in the terabyte track. In *TREC 2006: Proceedings of The Text REtrieval Conference*. National Institute of Standards and Technology.

[13] Carmel, D., Amitay, E., Hersovici, M., and Maarek, Y. (2004). Juru at TREC 10 — Experiments with Index Pruning. In *TREC 2004: Proceedings of The Text REtrieval Conference*. National Institute of Standards and Technology.

[14] Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y., and Soffer, A. (2001). Static index pruning for information retrieval systems. In *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50.

[15] Chakrabarti, S., Dom, B. E., Kumar, S. R., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D., and Kleinberg, J. (1999). Mining the web's link structure. *Computer*, 32(8):60–67.

[16] Cool, C., Belkin, N., Frieder, O., and Kantor, P. (1993). Characteristics of text affecting relevance judgments. In *Proceedings of the 14th National Online Meeting*, pages 77–84.

[17] Elias, P. (1975). Universal codeword sets and representations of the integers. In *IEEE Transactions on Information Theory*, volume IT-21(2), pages 194–203.

[18] Fagni, T., Perego, R., Silvestri, F., and Orlando, S. (2006). Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM Transactions on Information Systems*, 24(1):51–78.

[19] Ferguson, P., Gurrin, C., Wilkins, P., and Smeaton., A. F. (2005a). Físréal: A Low Cost Terabyte Search Engine. In *ECIR 2005: Proceedings of the 27th European Conference on Information Retrieval*, pages 520–522.

[20] Ferguson, P., Smeaton, A. F., Gurrin, C., and Wilkins, P. (2005b). Top Subset Retrieval on Large Collections using Sorted Indices. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 599–600. ACM.

[21] Garcia, S., Lester, N., Scholer, F., and Shokouhi, M. (2006). RMIT University at TREC 2006: Terabyte Track. In *TREC 2006: Proceedings of The Text REtrieval Conference*.

[22] Garcia, S., Williams, H. E., and Cannane, A. (2004). Access-ordered indexes. In *CRPIT 2004: Proceedings of the 27th conference on Australasian computer science*, pages 7–14, Darlinghurst, Australia. Australian Computer Society, Inc.

[23] Golomb, S. W. (1966). Run-length encoding. In *IEEE Transactions on Information Theory*, volume IT-12(3).

[24] Hauff, C., Hiemstra, D., and de Jong, F. (2008). A survey of pre-retrieval query performance predictors. In *CIKM 2008: Proceedings of the 17th ACM international conference on Information and knowledge management*, pages 1419–1420.

[25] Jansen, B. J. and Spink, A. (2003). An analysis of web documents retrieved and viewed. In *The 4th International Conference on Internet Computing*, pages 65–69.

[26] Joachims, T. (1999). Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184.

[27] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *KDD 2002: Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA. ACM Press.

[28] Jose, J. M., Furner, J., and Harper, D. J. (1998). Spatial querying for image retrieval: A user oriented evaluation. In *SIGIR 1998: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval*, pages 232–240.

[29] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[30] Lempel, R. and Moran, S. (2003). Predictive caching and prefetching of query results in search engines. *WWW 2003: Proceedings of the 12th international conference on World Wide Web*.

[31] Long, X. and Suel, T. (2003). Optimized query execution in large search engines with global page ordering. *VLDB 2003: Proceedings of the 29th international conference on Very large data bases - Volume 29*.

[32] Moffat, A. and Zobel, J. (1996). Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems*, 14(4):349–379.

[33] Mothe, J. and Tanguy, L. (2007). Linguistic analysis of users' queries: Towards an adaptive information retrieval system. In *IEEE SITIS 2007: International Conference on Signal Image Technology and Internet-Based Systems*.

[34] Ntoulas, A. and Cho, J. (2007). Pruning policies for two-tiered inverted index with correctness guarantee. In *SIGIR 2007: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–198, New York, NY, USA. ACM.

[35] Ntoulas, A., Najork, M., Manasse, M., and Fetterly, D. (2006). Detecting spam web pages through content analysis. In *WWW 2006: Proceedings of the 15th international conference on World Wide Web*, pages 83–92.

[36] Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

[37] Park, T. (1993). The nature of relevance in information retrieval: An empirical study. *Library Quarterly*, 63(3):318–351.

[38] Persin, M. (1994). Document filtering for fast ranking. In *SIGIR 1994: Proceedings of the 17th annual international ACM SIGIR conference on research and development in information retrieval*, pages 339 – 348.

[39] Persin, M., Zobel, J., and Ron-Sacks-Davis (1996). Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society of Information Science*, 47:749–764.

[40] Plachouras, V. and Ounis, I. (2005). Dempster-shafer theory for a query-biased combination of evidence on the web. *Information Retrieval*, 8:197–218.

[41] Probost, F. (2000). Machine learning from imbalanced data sets 101. In *AAAI 2000: Proceedings of the Workshop on Imbalanced Data Sets*.

[42] Robertson, S. E. and Spärk Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society of Information Science*, 27:129–146.

[43] Skobeltsyn, G., Junqueira, F., and Plachouras, V. (2008). Resin: a combination of results caching and index pruning for high-performance web search engines. In *SIGIR 2008: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 131–138.

[44] Smeaton, A. F. and van Rijsbergen, C. J. (1981). The nearest neighbour problem in information retrieval: an algorithm using upperbounds. In *SIGIR 1981: Proceedings of the 4th annual international ACM SIGIR conference on research and development in information retrieval*, pages 83–87.

[45] The World Wide Web Consortium (2011). The World Wide Web Consortium (W3C) HTML Specifications - http://www.w3.org/MarkUp/.

[46] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc.

[47] Wang, P. and White, M. D. (1999). A cognitive model of document use during a research pro ject. study ii. decisions at the reading and citing stages. *Journal of the American Society for Information Science*, 50(2):98–114.

[48] Witten, I. H., Moffat, A., and Bell, T. C. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann.

[49] Xue, G.-R., Yang, Q., Zeng, H.-J., Yu, Y., and Chen, Z. (2005). Exploiting the hierarchical structure for link analysis. In *SIGIR 2005: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*. ACM.

[50] Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR 2001: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, USA. ACM.

[51] Zhang, J., Long, X., and Suel, T. (2008). Performance of compressed inverted list caching in search engines. In *WWW 2008: Proceeding of the 17th international conference on World Wide Web*, pages 387–396, New York, NY, USA. ACM.

[52] Zhu, X. and Gauch, S. (2000). Incorporating quality metrics in centralized/distributed information retrieval on the world wide web. In *SIGIR 2000: Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval*, pages 288–295.

[53] Zipf, G. K. (1932). *Selective Studies and the Principle of Relative Frequency in Language*. Harvard University Press.