Reverted Indexing for Feedback and Expansion

Jeremy Pickens, Matthew Cooper, Gene Golovchinsky
FX Palo Alto Laboratory
Palo Alto, CA 94304 USA
jeremy@fxpal.com, cooper@fxpal.com, gene@fxpal.com

ABSTRACT

Traditional interactive information retrieval systems function by creating inverted lists, or term indexes. For every term in the vocabulary, a list is created that contains the documents in which that term occurs and its relative frequency within each document. Retrieval algorithms then use these term frequencies alongside other collection statistics to identify the matching documents for a query. In this paper, we turn the process around: instead of indexing documents, we index query result sets. First, queries are run through a chosen retrieval system. For each query, the resulting document IDs are treated as terms and the score or rank of the document is used as the frequency statistic. An index of documents retrieved by basis queries is created. We call this index a reverted index. With reverted indexes, standard retrieval algorithms can retrieve the matching queries (as results) for a set of documents (used as queries). These recovered queries can then be used to identify additional documents, or to aid the user in query formulation, selection, and feedback.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

1. INTRODUCTION

In *ad hoc* information retrieval, users describe their information needs by queries to search a document collection. Query terms are often used with an inverted index to rank documents by estimated relevance. In query expansion based on relevance feedback, users explicitly identify relevant documents to help refine a search iteratively. In this paper, we present and evaluate a general approach to indexing a collection for exploration using document-based queries, which we call *Reverted Indexing*. Reverted indexing is a variant of inverted indexing founded on the retrievability of documents by queries [5]. While traditional inverted indexing associates terms with documents in which they occur, reverted indexing relates documents with the queries that retrieve them.

To build the reverted index, we first assemble a large set of queries, and refer to its elements as *basis* queries to distinguish them from the *user* queries that represent users' information needs. Basis queries can be compiled from keywords extracted from query logs or from the documents comprising the collection, or by using other conjunctive mechanisms to form more complex queries (e.g. using metadata or facets). Once a set of basis queries is determined, we use a standard ranking function to order documents by their relevance to each basis query. For each document we construct the vector with elements corresponding to the basis queries and values determined by that document's estimated relevance to that basis query. This reverted index thus associates each document with the basis queries that retrieve it, as in Figure 1(b). This processing is performed off-line and does not require substantial computation for simple ranking functions.

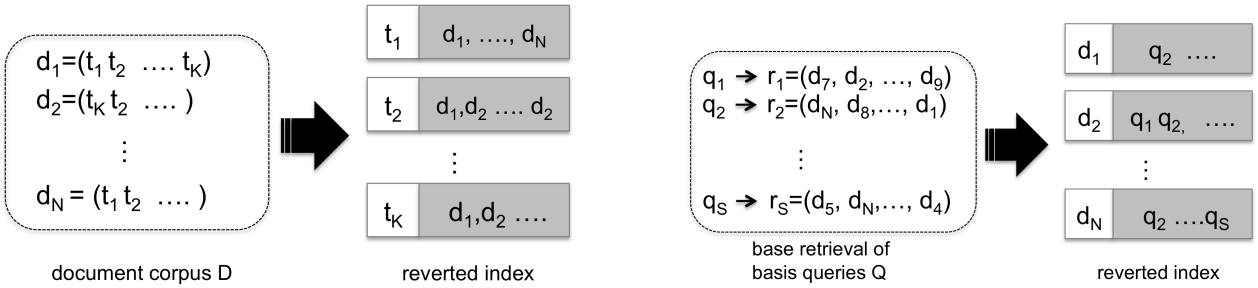
The process is analogous to standard term-based inverted indexing. A conventional inverted indexes is a set of lists, one for each term. A given term's list contains elements for each document in which that term occurs, as in Figure 1(a). Each element is weighted according to the term's importance within the corresponding document.

In sum, the basic conceptual foundation for conventional inverted indexes is term occurrence, while reverted indexes are founded on document *retrievability*. In the next section, we review related representations for document collections. We then describe our indexing scheme in detail and present query expansion experiments that demonstrate improved performance over well-regarded methods. We also show that our method reduces online computational costs, and conclude the paper with a discussion of our results and potential future extensions.

2. BACKGROUND

In this section, we summarize related work and discuss differences with our proposed framework. A primary source of inspiration was the work of Azzopardi *et al.* [5]. Adopting a document-centric view of information retrieval, they ask what portion of a document collection is retrieved by a large set of queries at a rank less than or equal to k. Building on the notion of retrievability, we index the set of documents retrieved by basis queries. We then apply established informa-

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.



(a) A depiction of traditional inverted indexing. t_i and d_n denote terms and document IDs, respectively.

(b) A depiction of reverted indexing. q_b and d_n denote basis queries and document IDs, respectively.

Figure 1: Traditional Inverted Indexing versus Reverted Indexing

tion retrieval algorithms to the reverted index to determine the basis queries most characteristic of a given document. The reverted index also identifies documents that are likely to not be retrievable in the sense used by Azzopardi *et al.* [5].

Robertson [18] explored symmetries in information retrieval within the classic vector space framework, examining the concept of duality in representing documents by terms, and *vice versa*. From this perspective, both queries and documents are treated as "bags of terms." In reverted indexing, on the other hand, basis queries are never processed according to their constituent terms, but rather treated as a atomic units distinct from other basis queries with which they may share common terms.

Craswell *et al.* [9] use a web-scale search engine to collect a large set of queries and clicked results. They combine these sets into a bi-partite graph and use random walks on the graph to infer likely end points from a single starting point. Knowing what documents many internet users clicked after issuing a query makes it possible to go backwards to find the most common queries related to a given document. Our work differs in key ways. First, by relating queries and clicks in a bipartite graph, Craswell *et al.* [9] is constrained to a single document starting point, i.e. you can initiate a random walk from only one document at a time. In contrast, reverted indexing can process a group of documents as a set. Furthermore, documents can be combined using boolean operators or other query constructs within reverted indexing. Craswell *et al.* [9] associate documents only with queries that users have issued. Our framework can use any automatically-extracted basis queries (n-grams, conjunctions, and such) that can be handled by the base ranking algorithm. Reverted indexing is broadly applicable to both large web collections with abundant user data, and to smaller enterprise or personal collections without available clickthrough statistics.

Query expansion is an established approach to improving users' search experience. Traditionally, query expansion is performed at search time using explicit relevance feedback (e.g. [19]) though other variants propose implicit relevance feedback (e.g. [17, 11]). Billerbeck *et al.* [6] present methods for query expansion based on processing user logs to determine which previously issued queries returned a given document. They define "document surrogates" comprised of the terms in issued queries for which a given document appeared among the top 19 results. The number of queries associated with any given document is limited to 39 to retain queries with maximum statistical similarity. Three variations on their general approach permute the representations used in the first and second rounds of retrieval. Either the full document collection or the collection of surrogates is used to perform retrieval or select expansion terms.

In federated information retrieval (FIR), query-based sampling [7] has been proposed to aid in database selection and query expansion. For this, a set of queries is assembled and executed and the set of returned documents are used to estimate collection word frequency statistics, typically to then produce a traditional inverted index. In this manner, multiple document collections are indexed more efficiently using partial information. Ogilvie and Callan [13] applied the approach to query expansion using blind (pseudo) relevance feedback [11] in the FIR setting to examine sampling density and performance tradeoffs with poor results. More recently, Shokouhi, *et al.*, [20] extended this approach by experimenting with database selection and local and global expansion to show improvements in FIR.

Puppin *et al.* [15, 16] use query logs to partition a web-scale document collection using a query-document matrix. They find the top 100 documents for each query from a training set (derived from a query log), and then discover query-document clusters. The text terms of the query set from each document cluster comprises a "document surrogate". A secondary inverted index is constructed from these proxy documents. In use, a query is first evaluated against the secondary index, and well-matching proxy documents are then used to select which primary subcollection index to search. Thus Puppin *et al.*'s two-tier term→document inverted indexes are close in spirit to Billerbeck *et al.* [6], who create proxy documents by augmenting documents with terms derived from queries that retrieved them. In Billerbeck *et al.*, surrogates are query-text proxies for individual documents, while Puppin *et al.*'s surrogates are query-text proxies for document clusters. In both cases, the entity being indexed is a text document.

We share with both Billerbeck *et al.* and Puppin *et al.* the notion of retrievability, but draw a strong distinction in our work relative to what is being indexed. Instead of indexing document proxies composed of query text, we index results sets directly. Our document proxies are constructed out of document identifiers (docids) from query results sets, a process that we call "reverted indexing". This allows us to query the reverted index using document ids to identify queries that were effective at retrieving the given documents. These retrieved queries can be used in a variety of ways: to expand the original query, to browse the collection, etc. Furthermore, reverted indexing should not be confused with direct files: whereas a direct file contains only the terms

KL		Bo1		Reverted_PL2		KL		Bo1		Reverted_PL2	
Term	df	Term	df	Term	df	Term	df	Term	df	Term	df
Topic 172 : "effectiveness of medical products and related programs utilized in the cessation of smoking" : **4 rels**						Topic 341 : "airport security" : **11 rels**					
smoking	2459	smoking	2459	cessation	627	airport	11769	airport	11769	detectors	379
cessation	627	cessation	627	birthweight	41	security	61748	security	61748	security	61748
smokers	722	smokers	722	smokers	722	faa	1213	faa	1213	airport	11769
nicotine	235	nicotine	235	nicotine	235	baggage	817	baggage	817	hijacker	108
health	33887	intervention	5485	smoking	2459	heathrow	993	heathrow	993	kean	66
intervention	5485	health	33887	bloodstream	196	passengers	4758	passengers	4758	knbc	182
study	48065	quit	4641	quitting	575	airports	2923	detectors	379	luggage	749
quit	4641	researchers	6725	scip	11	aviation	5289	airports	2923	outfitted	239
researchers	6725	patch	1960	quit	4641	detectors	379	aviation	5289	baggage	817
patch	1960	quitting	575	questiona..	22	screening	1762	screening	1762	bicolor	5
drug	22258	study	48065	pretesting	16	police	31862	boarding	787	unchall..	384
quitting	575	drug	22258	clonidine	20	persons	10777	kean	66	wuerenli..	3
state	108787	forms	14873	podiatry	22	metal	8948	metal	8948	boarding	787
forms	14873	disease	10224	gum	494	air	34988	persons	10777	lapse	741
disease	10224	birthweight	41	worksites	47	boarding	787	terrorist	4661	screeners	8
avg df = 17435		avg df = 10185		avg df = 675		avg df = 11934		avg df = 7793		avg df = 5199	
Topic 407 : "poaching wildlife preserves" : **10 rels**						Topic 419 : "recycle, automobile tires" : **2 rels**					
wildlife	2891	leakey	22	poaching	331	tires	1050	tires	1050	shredding	113
leakey	22	poaching	331	poachers	117	tire	910	tire	910	bulkiness	4
poaching	331	wildlife	2891	tsavo	12	waste	10022	landfills	542	pyrolysis	22
kenya	1163	kenya	1163	leakey	22	landfills	542	waste	10022	decompos..	194
ivory	1014	ivory	1014	tusks	42	recycling	2038	recycling	2038	spagnoli	3
elephants	356	elephants	356	elephants	356	million	51272	energy	24068	retreaded	9
elephant	743	elephant	743	wildlife	2891	energy	24068	nuisance	553	tires	1050
deer	748	deer	748	kws	9	disposal	7150	disposal	7150	lackawanna	32
conserva..	4298	poachers	117	kez	173	dump	1608	dump	1608	westley	12
poachers	117	conserva..	4298	ivory	1014	nuisance	553	million	51272	folkways	14
species	3479	species	3479	jealousies	56	new	200289	recycle	652	landfills	542
african	10636	tusks	42	elephant	743	recycle	652	dumps	639	nuisance	553
africa	20390	african	10636	conserv..ists	293	county	52281	illegal	10518	diapers	218
tusks	42	namibia	483	kenya	1163	old	61378	old	61378	gaddi	146
namibia	483	animals	3928	fiefdom	129	dumps	639	county	52281	incineration	303
avg df = 3114		avg df = 2016		avg df = 490		avg df = 27630		avg df = 14979		avg df = 214	

Table 1: Example query expansion terms and associated term frequencies of terms identified using KL, Bo1 and reverted indexing (PL2) at a judgment depth of 20. Terms are sorted in score order, though scores are not shown due to space. Note the number of found relevant documents used to construct these sets. The document frequency (*df*) associated with each expansion term is discussed in Section 4.4

of the corresponding document, a reverted index contains *queries* that retrieved the document. These queries may be simple terms, conjunctions, or more complex expressions available in the underlying search engine.

We are also not required to mine query logs for our set of system (basis) queries, Q. Billerbeck *et al.* and Puppin *et al.*, note that 25% of the documents in their collection have no associated queries, and thus no surrogates. This result is consistent with the experiments in [5]. Automatically generating a large set of basis queries is likely to associate more documents in the collection with at least one basis query. Additionally, the surrogates don't preserve each query as a distinct unit, in contrast to our approach (but similar to [18]). Because logged queries sharing terms are collapsed into terms within a surrogate, the richness (and size) of the representation of documents by surrogates is reduced. Normalizations for the distinctiveness of a system query (akin to inverse document frequency) will in turn be skewed. Also, the estimated relevance and ranking information associated with the logged queries is not retained in the document surrogates. We preserve this information and use it for normalizations that have proven beneficial in standard information retrieval settings. We expect them in turn to improve performance within our framework.

3. REVERTED INDEXING

We now describe the reverted index in detail. Whereas an inverted index associates terms with the documents in which they occur, a reverted index associates documents with the basis queries that retrieve them. The structure of the reverted index is identical to the structure of an inverted index, and retrieval algorithms created for inverted indexes are directly applicable to reverted indexes. In the following subsections, we detail the construction of a reverted index, and describe its use for retrieval. We finally present our experimental system and its results in Section 4.

3.1 Index Construction

We start with a collection of documents $D = \{d_1, \cdots, d_N\}$. The first step is to determine a set of basis queries. While this can be done in any number of ways, the most obvious technique is to employ the same tokenizer that was used to build the standard inverted index of the document collection. The terms of an inverted index can constitute the basis queries $q_b \in Q$ used to construct the reverted index. Retrieval algorithms that operate on the inverted index are used to evaluate each query.

In addition, Q may include bi-grams or larger n-grams, (contiguous and non-contiguous, ordered and unordered), window phrase queries, metadata (e.g. document geographic location, creation time and date, categorical classification, etc.), and arbitrary conjunctions and disjunctions of terms and metadata (e.g. a singleton term combined with a geographic location). One obvious source for basis queries is existing user query logs [6, 15, 9, 16], though it is an open question whether user logs offer the necessary coverage of a

collection. Ideally, Q should be sufficiently large to represent both the content of the corpus (depth) as well as the breadth of potential user queries. The choice of Q also depends on the target application or user model for which the retrieval system is being designed. In general, the only constraint on basis queries is that they can be evaluated by the base retrieval algorithm. For the query expansion experiments in this paper, Q contains all singleton (unigram) terms that appear in at least two documents (df ≥ 2) in the collection.

We next use a base retrieval algorithm to rank the documents with respect to each basis query. Examples include Best Match, Vector Space, Language Modeling [14], Divergence From Randomness, and so on. In this paper, we use PL2 [2, 10] as the base retrieval algorithm. Each basis query $q_b \in Q$ generates a result list $r_b \subseteq D$ containing documents and their corresponding relevance scores. This information is processed as a synthetic *reverted document*. Each element in the list corresponds to a document identifier n such that $d_n \in r_b$ with a value computed from the retrieval score or rank within r_b. This list may be normalized by either the number of relevant documents $|r_b|$ or by the sum of scores. The set of lists for all basis queries can be viewed as a retrievability index (cf. [5]). This index is inverted (using traditional IR techniques) to generate the "reverted" index, in which each document is represented by the basis queries that retrieve it. This is illustrated in Figure 1(b).

Results lists r_b are truncated at 1000; consistent with the retrievability [5] inspiration for this work, only these top ranked docids d_n are included. Taking a cue from [4], the results list for each basis query is then further normalized: We shift, ($minmax$) scale, and quantize the retrieval scores to the integer range $\{1, \cdots, 10\}$. The d_i with the highest score (at the top rank) gets a value of 10, and the lowest, 1.

3.2 Index Usage: Reverted Querying

A user (or system) can combine this reverted index with a standard retrieval algorithm to retrieve the best basis queries in response to document identifier queries, a process that we call *reverted querying*. Reverted queries may be best match, boolean, or may use standard query operators such as synonyms, distance windows (e.g. you can find two docids that were retrieved within a given window size of each other by a particular basis query. The retrieval algorithm to do reverted querying is not required to be the same algorithm used to construct the reverted index (i.e. to issue basis queries). For the sake of consistency, and to demonstrate how easily deployable reverted indexes are, we use the same algorithm, PL2. (See Table 2 for more details.)

The queries against a traditional inverted index consist of terms and are issued to retrieve lists of documents. In a reverted index, queries consist of document ids (d_i) that retrieve basis queries q_b. Once the best basis queries have been retrieved, the terms or other aspects of those basis queries may be used as query suggestions, as query expansions, and so forth. Queries enhanced or refined in this manner may then be issued back against the original inverted index to retrieve new documents.

We adapt global and local statistical machinery from conventional inverted indexing. Intuitively, suppose that a particular docid (d_i) is retrieved (or "retrievable" [5]) by only a few basis queries. The presence of d_i in a reverted query should be weighted more heavily than other more commonly retrieved documents. Similarly, the estimated relevance of d_i to a particular basis query q_b is a natural local statistic. Similarly, the sum of all relevance scores in a specific results list provides a per-basis query normalizing factor. These intuitions are rooted in established methods (and easily implemented using existing software packages). Inverse document frequency measures in a standard index correspond to the inverse retrievability statistic described above. The relevance score associating a document and a basis query is the local statistic analogous to traditional term frequency. This above normalization corresponds to common usage of document length (sum of term frequencies). Models other than tf·idf use concepts such as eliteness [2, 10] and risk [14]; those statistics are also applicable in the reverted setting. In this manner, decades of information retrieval research and algorithm development can be applied directly to reverted indexes.

Table 1 contains examples of generated query expansion terms. For each TREC topic, an initial query was run and documents were identified according to NIST relevance judgments at a ranked depth of 20. To establish the baseline for comparison, we applied the KL and Bo1 [2, 3] query expansion techniques to identify candidate terms. The columns labeled Reverted_PL2 represent the results of using these same identified relevant documents as reverted queries, the basis query q_b results of which are ranked using the PL2 retrieval algorithm. Again, while other basis queries are possible, the direct correspondence between terms in the inverted index and basis queries in the reverted index is employed to insure fair comparison in the experimental section.

While the KL and Bo1 method reasonably capture the gist of the topic, they does so using relatively generic terms. For example, Topic 172's description and narrative require not only information about quitting smoking, but specific products and their effectiveness. From the four identified relevant documents, KL and Bo1 extract terms such as "study" and "patch", while Reverted_PL2 mentions a specific product by name ("Clonidine"). Similarly, even though only two relevant documents were identified in Topic 419's (finding new uses for old tires) initial retrieval list, reverted querying produces very reasonable results. The KL and Bo1 terms for Topic 419 generically mention recycling, waste, and disposal. But the reverted methods select more specific methods for recycling, such as shredding and pyrolysis, as well as specific uses such as retread[ing] and [turning into] diaper[s].

3.3 Index Application: Query Expansion

After a reverted query (constructed from the appropriate relevant document identifiers) is issued, query expansion is done thusly:

(a) The top m basis query q_b results are selected

(b) The raw reverted query retrieval scores of these m results are shifted and scaled to $[0, 1]$

(c) The results are then treated as regular terms and the $|m|$ vector is added back to the original (title-only TREC query)

Table 3 shows an example of Step (c) using information from Table 1, the top few basis queries that were retrieved by the Reverted_PL2 algorithm for Topic 407, "poaching wildlife preserves". The residual average precision (not counting documents already examined) for Topic 407 using all 15

Step	Description	Baseline		Our Method		
		Index	Algorithm	Index	Algorithm	Calculation
0	Reverted Index Construction	<not applicable>		Inverted	PL2	**Offline**
1	Initial Title-Only Retrieval	Inverted	PL2	Inverted	PL2	Online
2	Performance of Relevance Judgments	"User"				Online
3	Term Selection and Weighting ($m = 500$)	Inverted	Bo1 or KL	**Reverted**	PL2	Online
4	Final Expanded Query	Inverted	PL2	Inverted	PL2	Online

Table 2: Experimental Setup. Conditions are held constant across every step, except Step (3). In Step (3), the Bo1 (or KL) technique for selecting and weighting expansion terms is compared against the PL2 algorithm on a reverted index (Reverted_PL2).

expansion terms (and their associated weights, not shown due to space constraints) from Table 1 is [0.188, 0.231, 0.248] for KL, Bo1, and Reverted_PL2 respectively. For Topic 172 it is: [0.138, 0.141, 0.202] For Topic 419 the improvement was even greater: [0.0065, 0.0065, 0.088].

Table 3: Application of reverted results (retrieved basis queries) to query expansion.

	Original	Reverted	Expanded
poaching	1.0	1.0	2.0
poachers	0	0.565	0.565
tsavo	0	0.563	0.563
leakey	0	0.413	0.413
tusks	0	0.394	0.394
elephants	0	0.340	0.340
wildlife	1.0	0.243	1.243
kws	0	0.197	0.197
...	...	...	...
preserves	1.0	0	1.0

4. EXPERIMENTS

4.1 Experimental setup

We evaluate our approach using a common scenario: query expansion. Such work has a long history, from Rocchio in the early 1970s [19] to today. Query expansion typically proceeds in the following manner (see also Table 2):

(1) User issues query, system returns ranked documents

(2) User judges the top n documents, and marks $1 \leq k \leq n$ of them as relevant

(3) System selects m terms from these k relevant documents and assigns a weight to each term

(4) System runs the expanded, weighted query against the collection and returns *as-yet unseen* results to the user

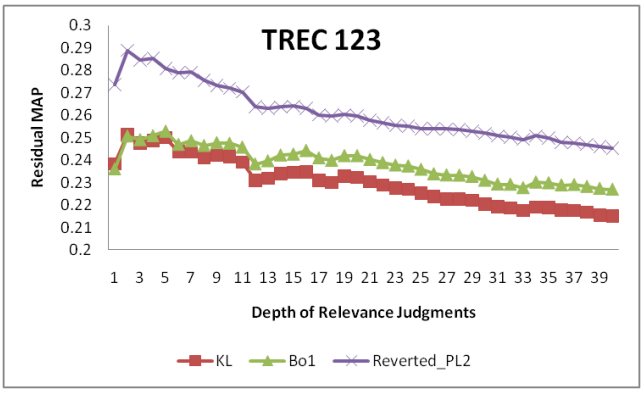
Variations on these steps interchange system and user involvement. For example, in Step (2) if the user were to make document relevance judgments to a depth in the ranked list of n, marking $k \leq n$ documents as relevant, this is called relevance feedback (RF). If the system instead automatically assumes that all n documents are relevant, this is called pseudo-relevance feedback (PRF). In Step (4) if the user rather than the system were to choose which of the top m terms to add back to the query, this is called manual query expansion. In this paper we focus on automatic query expansion, which means that the system is wholly responsible for constructing and issuing the expanded query based on the (relevant or pseudo-relevant) documents supplied to it.

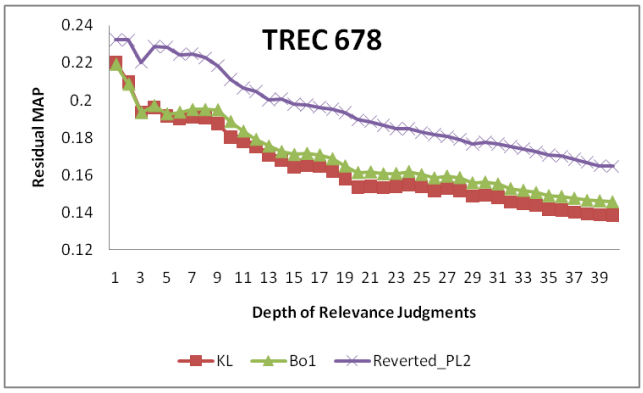
To compare several approaches to automatic query expansion algorithms, we fix steps (1), (2), and (4) across all conditions and only vary Step (3). In Step (1) the topic title-only query is issued using the PL2 retrieval model and the system returns a ranked list of documents. For RF, in Step (2), NIST relevance judgments are used to simulate user judgment on the top n documents returned by Step (1). (Note that we are applying n *judgments of relevance* rather than trying to identify n relevant documents.) For PRF, all n documents are simply assumed relevant. Documents identified in Step (2) are fed to the various automated query expansion algorithms of Step (3). In Step (4), PL2 is again used to retrieve documents using the expanded and weighted queries from Step (3).

After an initial comparison of several well-regarded algorithms, we chose a Divergence from Randomness model, PL2, for Steps (1) and (4) as it exhibits high precision at low recall, a necessary condition for obtaining many high-ranked relevant documents for RF or PRF. In both conditions, we use the PL2 algorithm with the same model-specific parameters on the same standard index. The only difference comes at Step (3), how the expansion terms are selected and weighted. For the control condition, we use two established expansion algorithms, Kullback-Leibler Divergence (KL) and Bose-Einstein (Bo1), for expansion term selection and weighting, as implemented in the Terrier open source retrieval platform [1]. For comparison, we use PL2 ranking on our reverted indexes to select and weight expansion terms as in Section3.2. The number of expansion terms is $m = 500$ for all conditions, but this choice was based solely on a parameter sweep over the baseline KL and Bo1 expansion algorithms. While fifty expansion terms is a more typical setting, we found that baseline effectiveness continued to improve as more terms were added. We chose to compare against the strongest possible baseline. For our experiments we use two primary TREC collections: (a) Topics 51-200 from TREC 1-3 on Disks 1 and 2, and (b) Topics 301-450 from TREC 6-8 on Disks 4 and 5. We label each of these collections TREC123 and TREC678, respectively.

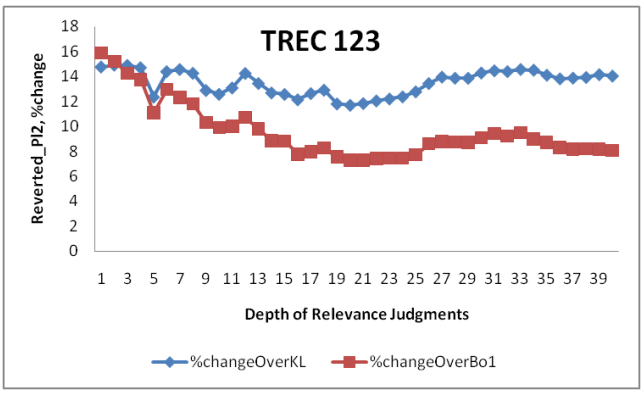
4.2 Relevance Feedback

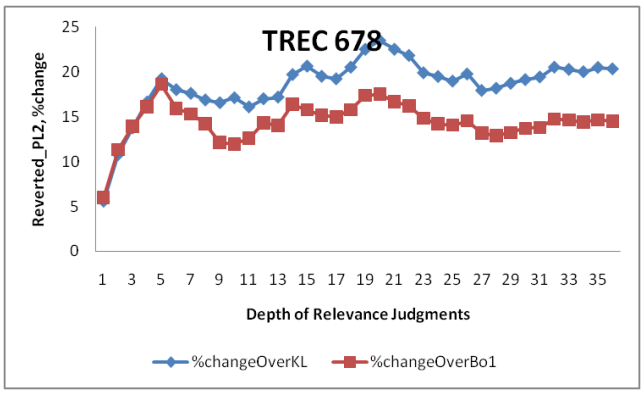
If a certain topic has not yielded any relevant documents at judgment depth n, there is no relevance information for expansion. We drop such topics from the evaluation as they do not differentiate the algorithms' performance. Naturally, as the judgment depth deepens, more topics retrieve at least one relevant document. Nevertheless, even at a rather shallow depth of 5 judgments, 119 of the 150 available queries for TREC123, and 115 of the 150 available queries for





Residual MAP results for 123 (left) and 678 (right).





Percentage difference of Reverted_PL2 over both KL and Bo1 baselines for 123 (left) and 678 (right).

Figure 2: Relevance Feedback: Residual MAP

TREC678, contain at least one relevant document. Importantly, for TREC123 improvements for Reverted_PL2 are statistically significant at a 0.01 value using t-test at all depths $n \geq 1$; for TREC678 the same significance holds at all depths $n \geq 3$.

The primary metric of query expansion performance is *residual* mean average precision (MAP) [12]. When an expanded query is run, it is of little use to the user to repeatedly see documents that she has already judged. We operate under the assumption that any document that the user has already judged, whether relevant or not, is removed from the results list of the subsequent, expanded query. Only documents that the user has not yet seen matter for evaluation. Residual MAP captures that idea; for a given judgment depth of n, all n documents that were judged are removed, and average precision is calculated across only the remaining documents.

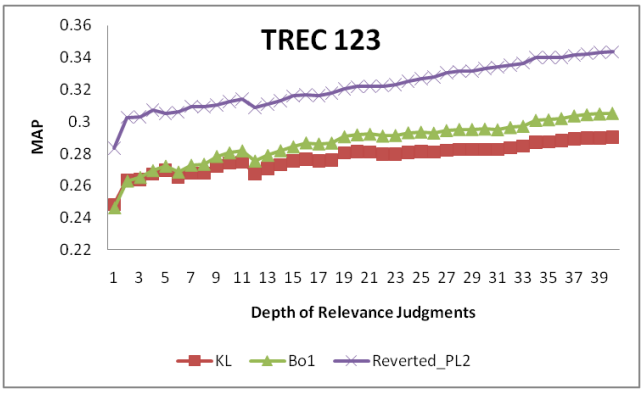
Figure 2 shows residual MAP results for both collections as a function of judgment depth. Results are presented for three algorithms. KL and Bo1 are our baselines; Reverted_PL2 is our experimental system. As the judgment depth increases, fewer relevant documents remain in the expanded results list and residual MAP values decrease. This accounts for the downward-sloping MAP curves. Note that residual MAP results are not comparable at different judgment depths.

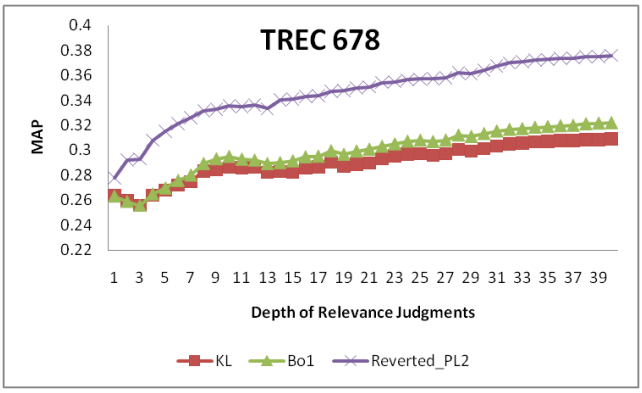
The key comparison is between Reverted_PL2 and KL or Bo1 at each judgment depth. Figure 2 also shows graphs of the percentage difference between Reverted_PL2 and KL or Bo1. For each fixed judgment depth (i.e. for the same set of found relevant documents), Reverted_PL2 outperforms both of these baselines. Performance varies slightly by collection, but even with fewer than five relevance judgments, Reverted_PL2 does 10-20% better than either KL or Bo1. For TREC123 that difference narrows as the judgment depth increases, while for TREC678 it holds steady in the mid teens. These results demonstrate the clear superiority of Reverted_PL2 over both the KL and Bo1 strong baselines.

While full MAP as an evaluation metric suffers from flaws described above, it does permit performance comparisons between judgment depths. Figure 3 shows these same experimental runs without the removal of judged documents. We see that the percentage improvement of Reverted_PL2 over both KL and Bo1 continues to rise. While the residual MAP results show that the reverted method improves precision for unseen documents, these full MAP results show that the reverted method also does an overall better job of "memorizing" seen relevant documents. While residual MAP results provide the more valuable comparison, the full MAP results demonstrate the robustness of the method.

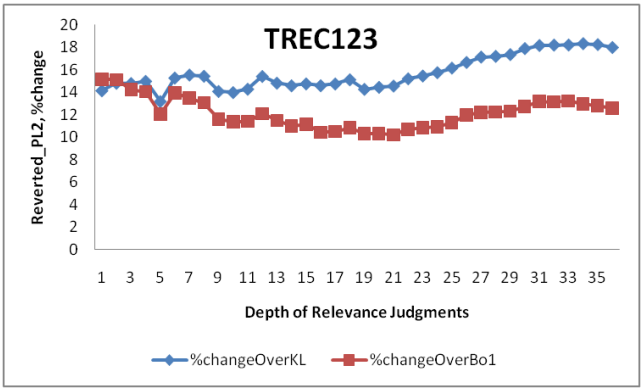
4.3 Pseudo Relevance Feedback

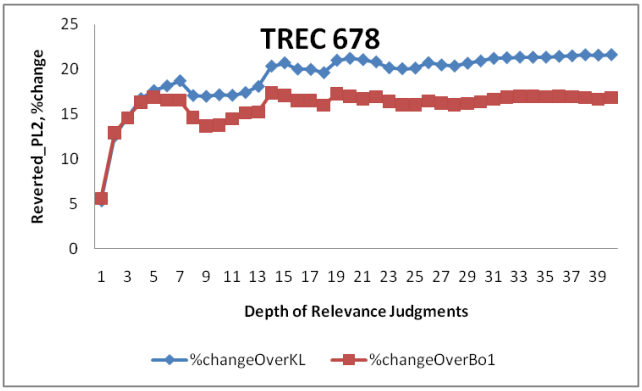
We designed reverted indexing to improve interactive information retrieval, but it is also applicable for other types of feedback. For non-interactive applications, a system may run an initial *pseudo*-relevance feedback step in which the





Full MAP results for 123 (left) and 678 (right).





Percentage difference of Reverted_PL2 over both KL and Bo1 baselines for 123 (left) and 678 (right).

Figure 3: Relevance Feedback: Full MAP

top n documents are assumed relevant and the query automatically expanded. Figure 4 contain MAP and percentage difference results for Reverted_PL2 over both baselines. Again, results are given at increasing judgment depths n. Because the user does not actually examine any documents, we report full MAP rather than residual MAP. Results shown are for all 150 queries for each collection, whether or not there are true relevant documents within the top n, as the discrimination made in the previous section is not possible in a pseudo-relevance context.

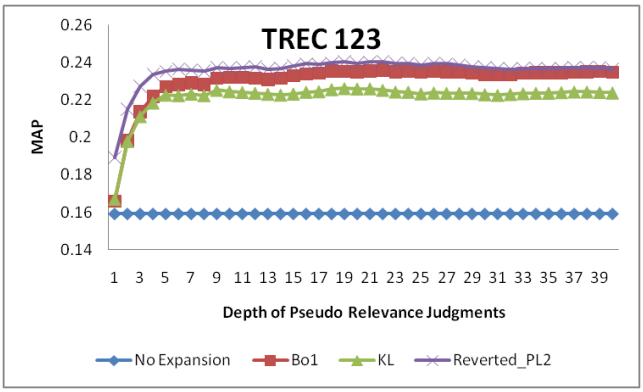
All three techniques (Reverted_PL2, KL and Bo1) either equal (at pseudo judgment depth $n = 1$) or outperform ($n > 1$) doing no query expansion, on average. As n increases, the improvement stabilizes (on TREC123) or worsens (on TREC678), but this reflects the number of available relevant documents in each collection. TREC123 tends to be much more plentiful than 678, so the pseudo-relevance judgments contain many more true relevant documents. This keeps the expansion from drifting too much. However, the primary comparison is between Reverted_PL2 and each of the baselines. At low n, reverted indexing outperforms both KL and Bo1 on both collections, but generally the differences are not statistically significant. Bo1 slowly narrows the gap as n increases. At the optimal value of n for each collection (around 20 on TREC123, around 14 on TREC678) there is only about a 2% difference, that is not statistically significant.

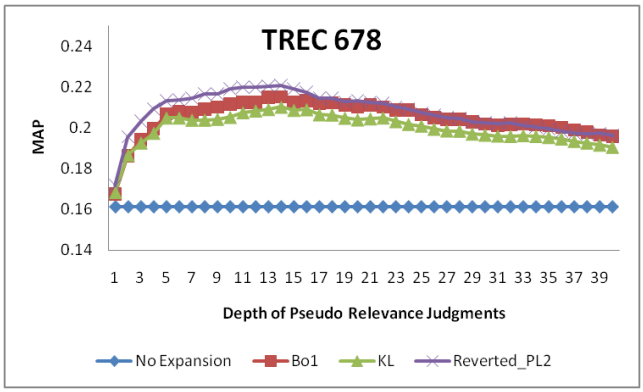
In terms of PRF effectiveness, Reverted_PL2 is at least as good as state-of-the-art query expansion methods. However, we will demonstrate in the following section that PRF under reverted indexing is an order of magnitude more efficient. Though not developed with PRF in mind, reverted indexing is widely applicable.

4.4 Efficiency

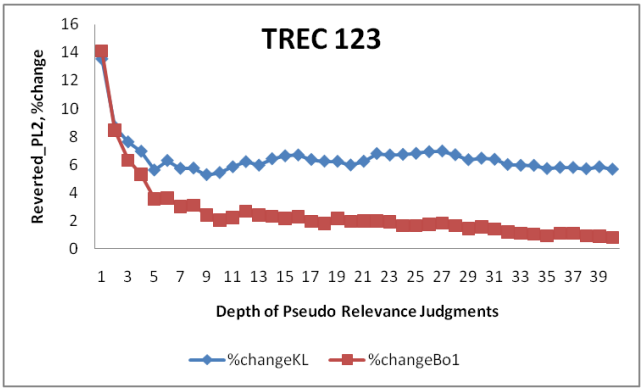
Reverted indexing is more effective than the baselines, but it is also significantly more efficient. Inspecting Table 2, there are two stages at which complexity differences may occur: In Step (3) the selection and weighting of the most informative expansion terms, and in Step (4) the execution of the expanded query. We examine these separately.

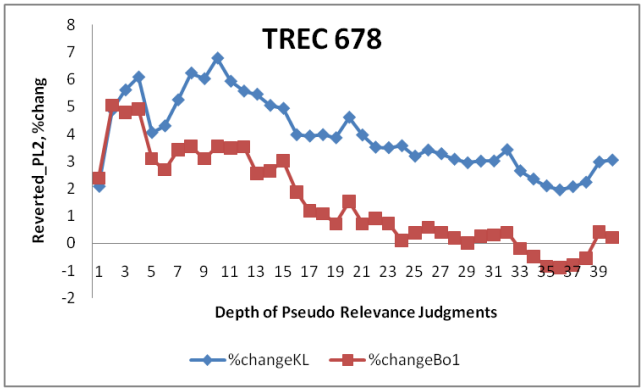
All of the following experiments were run under the same operating environment on a dual core, 2.83 GHz Intel machine with 3 GB of RAM. The codebase (Terrier[1]) and standard indexes were also shared, so implementation issues do not account for differences in efficiency. In the interest of space, we show results for the TREC678 collection only, although the same patterns of improvement were observed for both collections. We also note that building the reverted index for TREC678 takes approximately eight hours on this single machine. This includes both basis query execution time as well as reverted (results set) indexing time for all 295k basis queries, or approximately 97 ms per basis query. This is an offline process that is trivially parallelizable, so the rest of our analysis will focus on the online computation costs.





Pseudo-relevance MAP results for 123 (left) and 678 (right).





Percentage difference of Reverted_PL2 over both KL and Bo1 baselines for 123 (left) and 678 (right).

Figure 4: Pseudo-Relevance Feedback: Full MAP

4.4.1 *Selection and Weighting Time*

The first potential difference in efficiency (overall running time) is the selection and weighting of expansion terms. The top half of Figure 5 shows the term selection and weighting time data for Reverted_PL2, KL, and Bo1 for both RF and PRF. Reverted indexing is an order of magnitude faster than both KL and Bo1. Average time for RF (reverted) ranges from 3.4 milliseconds at a single document, to 4.8 ms at 40 documents. Under PRF the range is 8.3 to 10.7 ms. For both the KL and Bo1 methods, RF ranges from 12.0 to 47.2 ms, and PRF from 10.7 to 106.7 ms.

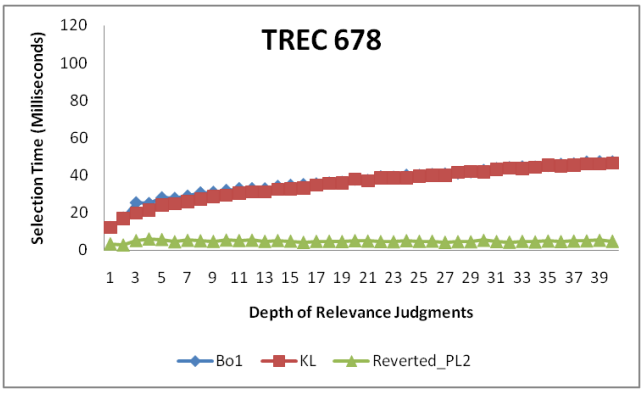
The explanation for this difference is simple. For KL and Bo1, a score or weight needs to be calculated online for every unique term in the set of feedback documents (whether RF or PRF). With reverted indexing, selection and weighting is as efficient as running a reverted query using the relevant docids. The weights are partially precomputed, and selection is simply a matter of choosing the top m results from the reverted query ranked list. This accelerates selection time immensely without sacrificing effectiveness. But because of this offline processing, selection time is perhaps not the fairest comparison as it may be possible to precompute Bo1 or KL weights for each document and achieve similar speedups. We turn our attention to the second factor: execution time.

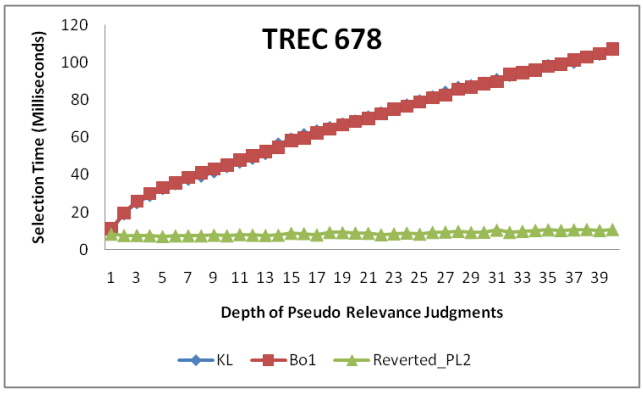
4.4.2 *Execution Time*

In the bottom half of Figure 5, the average execution time for the expanded query is slightly more than an order of magnitude faster for Reverted_PL2. For RF, reverted indexing execution times range from 107 ms (at 1 document) to 323 ms (at 40), while Bo1 and KL execution times range from 3394 to 4857 milliseconds. The difference is greater for PRF wherein the highest Reverted_PL2 execution time across any judgments depth was 556 ms, compared to 5387 ms and 5915 ms for KL and Bo1, respectively.

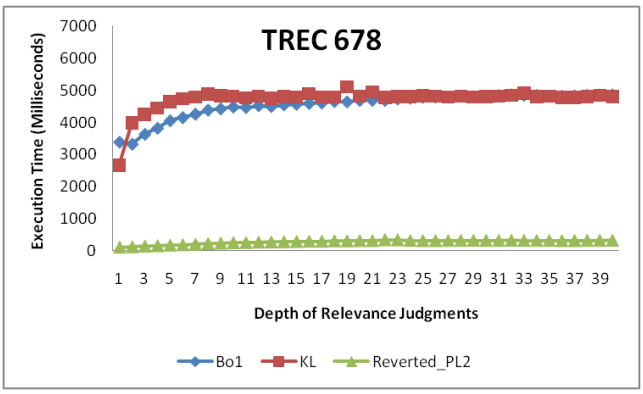
Recall from Step (4) in Table 2 that the same underlying algorithm (PL2) is used in all conditions for the final, expanded query. The retrieval algorithm is parameterized by the actual terms that were selected in Step (3), so differences in execution time are due to the different document frequencies *df* of the expansion terms. Higher *df* means a longer inverted list, which means a longer execution time when that term is part of the expanded query. Even with optimizations (e.g. optimal skips [8]), an optimized shorter list runs faster than an optimized longer list. A detailed analysis of document frequencies is beyond the scope of this paper, but the examples in Table 1 demonstrate the differences: with as few as $m = 15$ expansion terms, the average *df* of the highest weighted (best) expansion terms is in the thousands for KL and Bo1, as opposed to the hundreds for Reverted_PL2.

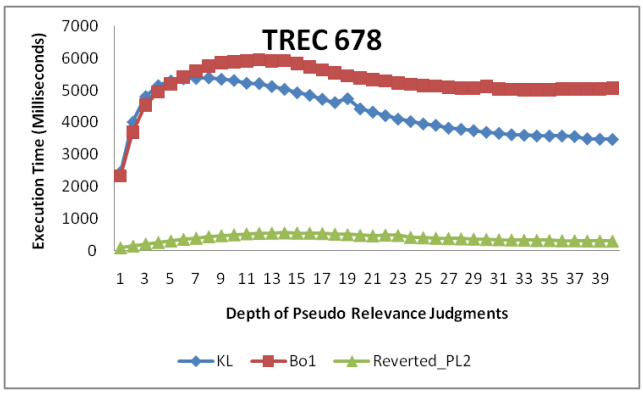
In Section 4.4.1, we suggested that it might be possible





Selection Time (in milliseconds) for TREC678; Relevance Feedback (left) and Pseudo Relevance Feedback (right)





Execution Time (in milliseconds) for TREC678; Relevance Feedback (left) and Pseudo Relevance Feedback (right)

Figure 5: Selection (top) and execution (bottom) timing plots for RF and PRF experiments on TREC678.

to precompute some Bo1 and KL weights or at least partial weights to achieve parity with reverted indexing's enhanced selection time. However, (from Figure 5) since execution time dominates selection time by *two* orders of magnitude (10s to 1000s of milliseconds), the overall efficiency of the reverted approach would still dominate. In summary, the reverted indexing approach achieves equal (PRF) or greater (RF) effectiveness with an order of magnitude improvement in efficiency.

5. FUTURE WORK

This paper introduces a framework for *ad hoc* retrieval of basis queries (as results) using document-ids (as queries). A basis query set and a base retrieval function combine to construct the reverted index. By choosing specific reverted retrieval functions and a reverted query language, we use the reverted index for retrieval focusing on the problem of automated query expansion. There are two major directions for future work. One is to invent better methods for constructing and querying the reverted index; the other is to create applications and interactions for the *ad hoc* retrieved basis queries.

For reverted index construction, extensions include automatically extracting bi-grams and longer n-grams, adding facets and other metadata to the basis queries, and incorporating effective queries mined from large scale search logs. When creating the reverted index using base retrieval functions, one is not limited to traditional IR models; any function that can produce a ranking can be used. For example, to bring this approach more in line with Craswell *et al.* [9], one could use relative click rates as a ranking function. A reverted index would then allow users to construct *ad hoc* multi-docid queries to retrieve the most likely basis queries to have been responsible for producing clicks on those multiple documents.

Another interesting idea is to construct the index by running the same set of basis queries across multiple retrieval algorithms or (web) search engines, and storing that algorithmic choice as part of the basis query. When doing a reverted query, one would then be able to retrieve or discover not only the best basis queries, but also the algorithm or engine to use to produce those results. Integration of multiple media types (images and image queries, music and music queries) is also possible; a reverted song ID query could retrieve textual, audio, and image basis queries.

For reverted index querying, one interesting approach relates to synonyms. If a user's information need can be decomposed into multiple aspects, then relevant documents identified as belonging to each aspect could be treated as synonyms. For example, if d_w and d_x are relevant to one aspect, and d_y and d_z are relevant to another, the reverted query could be $[syn(d_w\ d_x)\ syn(d_y\ d_z)]$. This conflation has an effect on global and local statistics and could produce better results. Boolean constructs for integrating non-relevant documents may be useful as well, e.g. if d_w and d_y are rel-

evant, and d_x and d_z are non-relevant, a possible *ad hoc* reverted query might be: $[(d_w \wedge \neg\, d_x) \vee (d_y \wedge \neg\, d_z)]$.

6. CONCLUSION

In this paper, we described reverted indexes as a representation for document collections. Using the retrievability of documents, reverted indexes complement term-based inverted indexes. Index construction involves issuing a broad set of basis queries to establish retrievability within the collection. Document identifiers are then used as generalized *ad hoc* queries against the reverted index to retrieve "relevant" basis queries. Because the reverted index is wholly analogous to standard inverted indexing, most Information Retrieval techniques for ranking and retrieval (models, query languages and operators, et cetera) are directly applicable.

Our experiments demonstrate high performance query expansion using reverted indexing in combination with proven ranking techniques. Results show that this approach outperforms two strong, established baselines for query expansion for on test collections with consistent and significant improvements over a range of judgment depths and types. Also, the computational costs of using reverted indexing are substantially lower at retrieval time relative to the baselines.

7. REFERENCES

[1] Terrier information retrieval platform, http://ir.dcs.gla.ac.uk/terrier/.

[2] G. Amati. *Probability Models for Information Retrieval based on Divergence from Randomness*. PhD thesis, Department of Computing Science, University of Glasgow, June 2003.

[3] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, 2002.

[4] V. N. Anh and A. Moffat. Simplified similarity scoring using term ranks. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 226–233, New York, NY, USA, 2005. ACM.

[5] L. Azzopardi and V. Vinay. Retrievability: an evaluation measure for higher order information access tasks. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 561–570, New York, NY, USA, 2008. ACM.

[6] B. Billerbeck, F. Scholer, H. E. Williams, and J. Zobel. Query expansion using associated queries. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 2–9, New York, NY, USA, 2003. ACM.

[7] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Trans. Inf. Syst.*, 19(2):97–130, 2001.

[8] F. Chierichetti, S. Lattanzi, F. Mari, and A. Panconesi. On placing skips optimally in expectation. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 15–24, New York, NY, USA, 2008. ACM.

[9] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246, New York, NY, USA, 2007. ACM.

[10] B. He and I. Ounis. Term frequency normalisation tuning for bm25 and dfr model. In *Proceedings of the 27th European Conference on Information Retrieval (ECIR'05)*, pages 200–14. Springer, 2005.

[11] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127, New York, NY, USA, 2001. ACM.

[12] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, 2008.

[13] P. Ogilvie and J. Callan. The effectiveness of query expansion for distributed information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 183–190, New York, NY, USA, 2001. ACM.

[14] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, New York, NY, USA, 1998. ACM.

[15] D. Puppin, F. Silvestri, and D. Laforenza. Query-driven document partitioning and collection selection. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 34, New York, NY, USA, 2006. ACM.

[16] D. Puppin, F. Silvestri, R. Perego, and R. Baeza-Yates. Tuning the capacity of search engines: Load-driven routing and incremental caching to reduce and balance the load. *ACM Trans. Inf. Syst.*, 28(2):1–36, 2010.

[17] Y. Qiu and H.-P. Frei. Concept based query expansion. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–169, New York, NY, USA, 1993. ACM.

[18] S. E. Robertson. Query-document symmetry and dual models. *Journal of Documentation*, 50(3):233–238, September 1994.

[19] J. J. Rocchio. Relevance feedback in information retrieval. *SMART Retrieval System Experimens in Automatic Document Processing*, 1971.

[20] M. Shokouhi, L. Azzopardi, and P. Thomas. Effective query expansion for federated search. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 427–434, New York, NY, USA, 2009. ACM.