# Collection Selection with Highly Discriminative Keys

Sander Bockting
Avanade Netherlands B.V.
Versterkerstraat 6
1322 AP, Almere, Netherlands
sander.bockting@avanade.com

Djoerd Hiemstra
University of Twente
P.O. Box 217
7500 AE, Enschede, Netherlands
d.hiemstra@utwente.nl

## ABSTRACT

The centralized web search paradigm introduces several problems, such as large data traffic requirements for crawling, index freshness problems and problems to index everything. In this study, we look at collection selection using highly discriminative keys and query-driven indexing as part of a distributed web search system. The approach is evaluated on different splits of the TREC WT10g corpus. Experimental results show that the approach outperforms a Dirichlet smoothing language modeling approach for collection selection, if we assume that web servers index their local content.

## 1. INTRODUCTION

The web search approach of major search engines, like Google, Yahoo! and Bing, amounts to crawling, indexing and searching. We call this approach *centralized search*, because all operations are controlled by the search engines themselves, be it from a relatively limited number of locations on large clusters of thousands of machines. The centralized web search paradigm poses several problems.

The amount of web data is estimated to grow exponentially [34]. The changing and growing data requires frequent visits by crawlers, just to keep the index fresh. Crawling should be done often, but generates a huge amount of traffic, making it impossible to do frequent crawls of all pages. With an estimated four weeks update interval, updates are performed relatively slow [25, 35]. Also, it is impossible to index everything, as the search engine accessible *visible* web is only a fraction of the total number of web pages [16].

Callan [5] identified the distributed information retrieval problem set, consisting of resource description, resource selection and result merging. We believe that distributing the search efforts may be an approach to solve the problems described above. This research focuses on resource description and resource selection [10, 22]. Resource description, i.e. indexing of the peers, is distributed; resource selection, or *collection selection*, is centralized in our research.


*LSDS-IR* Workshop. July 2009. Boston, USA.

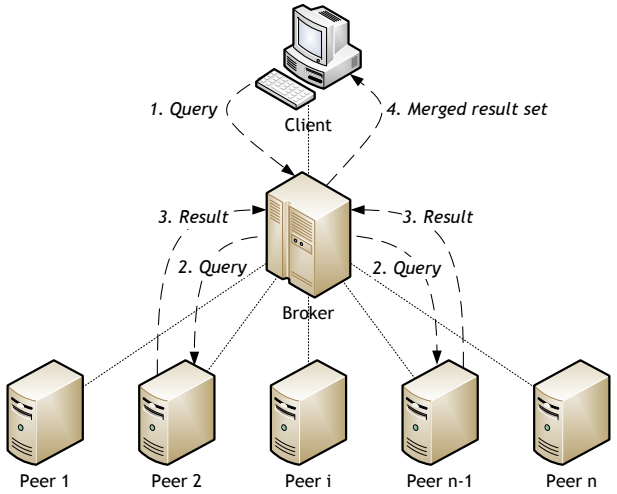


**Figure 1: Peers are accessible via a broker to answer queries from clients**

Figure 1 shows three types of entities: peers, brokers and clients. We assume that peers are collaborative. Every peer runs a search engine enabled web server. The search engine indexes the local website(s), but it may also index other websites. In this scenario, there can be many millions of peers. When a user has an information need, he can pose a query at the client. The client sends the query to the broker. In a response, the broker tries to identify the most promising peers to answer the query. This has to be a small amount of peers, e.g. five to ten peers, so not a lot of traffic is generated. The query is routed to those peers and the results are returned to the broker. The broker merges the results and sends the merged list to the client.

Peers and brokers cooperate to enable brokers to identify most promising peers. Therefore, every peer sends a small part of its index to the broker. This part cannot be too small, to still allow for proper judging about the peers' ability to satisfactory answer queries. The index part cannot be too large due to index data traffic scalability.

Techniques have been proposed to manage the index size. Podnar *et al* [20] used the concept of highly discriminative keys (HDKs) for document retrieval in distributed information retrieval. An HDK is a set of terms that is highly discriminative, i.e., that only match a few documents in the collection. Because the terms are pre-coordinated (they are combined at index time, not at search time) and because

only a few document match all terms in a pre-coordinated set, the HDK approach is able to very efficiently retrieve the top documents for a query. Although searching can be efficient, the HDK indexing process, described in more detail in Section 3.1, has the negative side-effect that a huge amount of highly discriminative keys are generated. To reduce the number of keys, Skobeltsyn [32] proposed a query-driven indexing strategy that uses caching techniques to adapt to the changing querying behavior of users. The combination of HDK with query-driven indexing allows for completely distributed document retrieval that in theory grows to web scale proportions [31].

This paper contributes to the field of distributed information retrieval by the applying HDKs and query-driven indexing to select collections, instead of documents. Such an approach would in theory scale the distributed search scenario described above to millions of peers: The broker lists for every HDK a small number of peers to send the query to, and the peers retrieve the documents; possibly many. Unlike a traditional inverted file index that typically consists of huge posting lists and a, in comparison, tiny dictionary [36], our HDK index consists of a huge dictionary and, in comparison, tiny posting lists. The system is fitted into the previously sketched scenario, which allows for control at the broker. This control can for example be used to prevent misuse or to allow for domain-specific search.

This paper is organized as follows: the next section discusses earlier collection selection methods, Section 3 introduces our collection selection system and Section 4 describes the evaluation. The paper concludes with results and conclusions.

# 2. EARLIER WORK ON COLLECTION SELECTION

Collection selection systems have been developed to select collections containing documents that are relevant to a user's query. The generalized Glossary-Of-Servers Server (gGlOSS) is such a system [13]. It uses a vector space model representing index items (document collections) and user queries as weight vectors in a high dimensional Euclidean space to calculate the distance (or similarity) between document collections and queries [24].

Another well-known approach is CVV, which exploits the variation in cue validity to select collections [38]. The *cue validity* $CV_{i,j}$ of query term $t_j$ for collection $c_i$ measures the extent that $t_j$ discriminates $c_i$ from the other collections, by comparing the ratio of documents in $c_i$ containing $t_j$ to the ratios of documents in other collections containing $t_j$. The larger the variation in cue validities for collections with respect to a term, the better the term is for selecting collections.

This section will describe two collection selection methods in more detail: inference networks and language modeling.

## 2.1 Inference networks

CORI [7] is a collection ranking algorithm for the INQUERY retrieval system [6], and uses an inference network to rank collections. A simple document inference network has leafs $d$ representing the document collections. The terms that occur in those collections are represented by representation nodes $r$. Flowing along the arcs between the leaves and nodes are probabilities based on document collection statistics. Opposed to TF.IDF, the probabilities are calculated using document frequencies *df* and inverse collection frequencies *icf* (DF.ICF). The inverse collection frequency is the number of collections that contain the term. An inference network with these properties is called a collection retrieval inference network (CORI net).

Given query $q$ with terms $r_k$, the network is used to obtain a ranked list of collections by calculating the belief $p(r_k|c_i)$ in collection $c_i$ due to the observation of $r_k$. The collection ranking score of $c_i$ for query $q$ is the sum of all beliefs $p(r_k|c_i)$, where $r \in q$. The belief is calculated using Formula 1. In this formula, $b$ and $l$ are constants, $cw_i$ is the number of words in $c_i$, $\overline{cw}$ is the mean number of words in the collections and $|C|$ is the number of collections. *df* and *cf* respectively are the number of documents and collections that contain $r_k$. Finally, $d_t$ and $d_b$ respectively are the minimum term frequency component and minimum belief component when $r_k$ occurs in $c_i$.

To improve retrieval, the component $L$ is used to scale the document frequency in the calculation of $T$ [6, 23]. $L$ is made sensitive to the number of documents that contain term $r_k$ (using $b$) and is made large using $l$. $L$ should be large, because *df* is generally large. Proper tuning of these two parameters is required for every data set, but deemed impossible as the correct settings are highly sensitive to data set variations [12]; the value of $l$ should be varied largely even when keeping the data set constant while varying the query type.

Further research showed that it is not justified to use standard CORI as a collection selection benchmark. D'Souza *et al.* showed that HIGHSIM outperformed CORI in 15 of 21 cases [11]. Si and Callan [28] found limitations with different collection sizes. Large collections are not often ranked high by CORI, although they often are the most promising collections.

$$
\begin{aligned}
L &= l \cdot ((1-b) + b \cdot cw_i/\overline{cw}) \\
T &= d_t + (1-d_t) \cdot \frac{\log(df)}{\log(df+L)} \\
I &= \frac{\log\left(\frac{|C|+0.5}{cf}\right)}{\log|C| + 1.0} \\
p(r_k|c_i) &= d_b + (1-d_b) \cdot T \cdot I
\end{aligned} \tag{1}
$$

## 2.2 Language modeling

A language model is a statistical model to assign a probability to a sequence of words (e.g. a query) being generated by a particular document or document collection. Language models can be used for collection selection in distributed search, by creating a language model for each collection [29, 37]. They have also been used for collection selection for other tasks, for instance for blog search [2, 26].

INDRI is an improved version of the INQUERY retrieval system [33], as it is capable of dealing with larger collections, allows for more complex queries due to new query constructs and uses formal probabilistic document representations that use language models. The combined model of inference networks and language modeling is capable of achieving more favorable retrieval results than INQUERY [18]. Due to these differences, term representation beliefs are calculated in an-

other way than with CORI (as described in Section 2.1):

$$P(r|D) = \frac{tf_{r,D} + \alpha_r}{|D| + \alpha_r + \beta_r}$$

The belief is calculated for representation concept $r$ of document node $D$ (in document collection $C$). Examples of representation concepts are a term in the body or title of a document. $D$ and $r$ are nodes in the belief network. The term frequency of representation node $r$ in $D$ is denoted by $tf_{r,D}$. $\alpha_r$ and $\beta_r$ are smoothing parameters. Smoothing is used to make the maximum likelihood estimations of a language model more accurate, which could have been less accurate due to data sparseness, because not every term occurs in a document [39]. Smoothing ensures that terms that are unseen in the document, are not assigned zero probability. The default smoothing model for INDRI is Dirichlet smoothing, which affects the smoothing parameter choices [17]. In setting the smoothing parameters, it was assumed that the likeliness of observing representation concept $r$ is equal to the probability of observing it in collection C, given by $P(r|C)$. This means that $\alpha_r/(\alpha_r + \beta_r) = P(r|C)$. The following parameter values were chosen:

$$\alpha_r = \mu P(r|C)$$
$$\beta_r = \mu(1 - P(r|C))$$

This results in the following term representation belief, where the free parameter $\mu$ has a default value of 2500:

$$P(r|D) = \frac{tf_{r,D} + \mu P(r|C)}{|D| + \mu}$$

## 2.3 Discussion

The language modeling approach of INDRI has a better formal probabilistic document representation than CORI and INDRI is an improved version of INQUERY (which is the foundation of CORI). We will use the language model on document collections as implemented by INDRI as our baseline collection selection system. Si *et al.* [29] showed that a language modeling approach for collection selection outperforms CORI. Furthermore, CORI outperforms algorithms like CVV and GGLOSS in several studies [9, 21].

# 3. SOPHOS

Sophos is a collection selection prototype that uses HDKs to index collections. The keys are used to assign scores to the collections. Using a scoring function, collection scores can be calculated to rank collections for a particular query. A general overview is depicted in Figure 2. This section describes how the broker index is created, explains index size reduction using a query-driven indexing approach, identifies query result parameters, and concludes with a collection ranking formula (ScoreFunction in Figure 2).

## 3.1 Highly discriminative keys

Building the index is done in two phases. First, every peer builds an index of its document collection and sends that index to the broker. Second, the broker constructs a broker index from all peer indices.

### 3.1.1 Peer indexing

Peer indexing starts with the generation of term set statistics. First, single term statistics are generated by counting

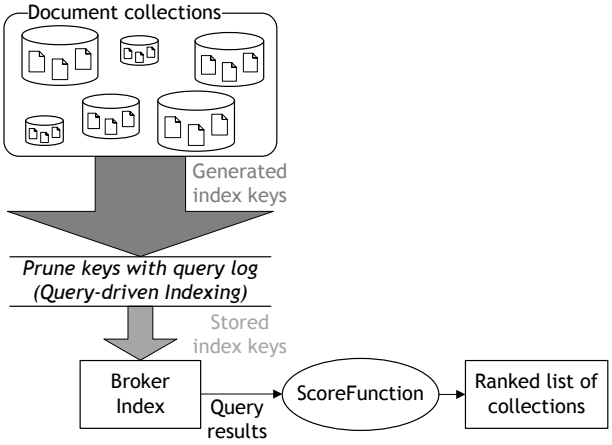


**Figure 2: General overview of the indexing and collection selection system Sophos**

term frequencies of every word in the collection, without looking at document boundaries in the collection. A term set is called *frequent* when it occurs more times than a term set frequency maximum $tf_{\max}$. Every infrequent single term is added to the peer index together with its frequency count. The frequent keys are stored for further analysis.

The next step consists of obtaining double term set statistics. For every frequent term in the collection, frequency statistics are created for term combinations that consist of the frequent term and a term that appears after that term within a window size $ws$. The result is a list of double terms with corresponding frequency counts. Once again, the term set frequency maximum defines which term sets are frequent and will be used for further analysis, and which term sets are infrequent and will be stored in the peer index. This procedure can be repeated as long as the generated term sets do not contain more than $ws$ terms, or when a predefined maximum number of terms in a term set, $h_{\max}$, has been reached.

Summarizing, the peer index consists of tuples with term sets and corresponding frequency counts.

### 3.1.2 Broker indexing

The infrequent term sets from the peer indices are sent to the broker. The broker index contains term sets with associated sets of collection identifier counters. A *collection identifier counter* is a tuple of a collection identifier and a term set frequency counter. A collection identifier is a short representation of a collection where the term set occurs.

When a term set is inserted into the broker index, it is called a highly discriminative key (HDK). The broker index will contain a maximum number of collections per HDK, denoted by the collection maximum $cm$. As soon as the maximum number of collections is reached for a particular HDK, the $cm$ collections with the largest term set frequencies will be stored in the broker index.

## 3.2 Query-driven indexing

A list of popular keys can be created by extracting all unique queries from a query log. Every key that is infrequent at a peer, and which is present in the unique query list, will be sent to the broker; the other keys are filtered out to reduce the broker index size and to reduce traffic.

| | |
|---|---|
| $c$ | sum of query term set occurrence within one collections (grouped by sets with $h$ terms) |
| $h$ | #terms in an HDK |
| $h_{\max}$ | maximum #terms that HDK can consist of |
| $q$ | #terms in query |
| $n$ | #distinct query terms found in a collection |
| $tf_{\max}$ | maximum frequency of a term set in a collection |

**Table 1: Parameter definitions for query result handling**

This index pruning strategy was used before by Shokouhi *et al.* [27]. It is not the most desirable strategy for query-driven indexing, because it is unable to deal with unseen query terms. However, it will give a good idea about the possible index size reduction and the loss of retrieval performance.

## 3.3 Identifying query result parameters

Once the broker index has been built, the system is ready to be queried. The broker index contains HDKs with $h$ terms, where $h$ varies from 1 to $h_{\max}$. In Sophos, $h_{\max}$ is set to 3. Every key has an associated posting list, which contains tuples of collection identifiers and counters. A counter indicates the number of occurrences of a certain key within the corresponding collection. The counter value cannot exceed the term set frequency maximum, $tf_{\max}$, as a key would otherwise have been locally frequent and new HDKs would have been generated when the maximum number of terms, $h_{\max}$, was not yet reached.

When a user poses a query with $q$ terms, e.g. abcde with $q = 5$, the query is first decomposed in query term combinations with length $h_{\max}$ (i.e. abc, abd, ..., bde, cde). This results in $\binom{q}{h}$ combinations. Each combination is looked up in the broker index. Note that this introduces additional load on the broker, but these lookups do not require network access to the peers. The results of each of those smaller queries are merged by summing the number of occurrences per collection. The sum of all counters, $c$, has a potential maximum of $\binom{q}{h} \cdot tf_{\max}$. It may happen that this procedure results in little or no collections where an HDK occurs. The procedure is then repeated for smaller term sets; first term sets of two terms will be looked up in the index. When even this gives too few results, single terms will be looked up in the index. In the case that one collection contains two different combinations, e.g. both abc and bce occur in collection X, the number of occurrences are summed (this is $c$ that was just introduced). However, it also interesting to note that 4 out of 5 query terms can be found in collection X. The number of query terms that can be found using HDKs of a particular length is indicated by $n$. The different parameters are displayed in Table 1.

## 3.4 ScoreFunction: ranking query results

ScoreFunction, given in Formula 2, is a ranking formula that uses the query result parameters to enforce a collection ranking conforming to our desired ranking properties. It calculates a score $s$ for a collection for a given query

$$s = \log_{10}\left(\left[h - 1 + \frac{n-1}{q} + \frac{\sqrt{\frac{c}{(h_{\max}+1-h)\cdot\binom{n}{h}\cdot tf_{\max}}\cdot\alpha^{(q-n)}}}{q}\right]/h_{\max}\right) \quad (2)$$

It consists of three components; one component per desired ranking property. The properties, and corresponding components, are listed below in order of importance.

1. Collections that contain longer HDKs should be ranked higher. Component 1: $h - 1$.
2. A collection should be ranked higher if it contains more distinct query terms. Component 2: $(n-1)/q$.
3. More occurrences of query term sets within a collection should result in a higher collection ranking. Component 3: $\frac{\sqrt{\frac{c}{(h_{\max}+1-h)\cdot\binom{n}{h}\cdot tf_{\max}}\cdot\alpha^{(q-n)}}}{q}$.

   The general intuition behind this component is that a collection is more important when it contains more query terms. This is controlled by a damping factor $\alpha$. The term set counts have less impact when they get closer to $tf_{\max}$, because longer keys would be generated for a term set in a collection when its frequency exceeds $tf_{\max}$. We refer to earlier work for a more detailed explanation about ScoreFunction [4].

An important detail to mention about querying and ranking is that collection X can be found after looking for HDKs with length $h$. When the same collection X is found after looking for HDKs with length $h - 1$, those results are discarded as the collection was already found using larger HDKs. Counts $c$ are only compared with other counts that are retrieved after looking for HDKs of the same length. The motivation for this behavior is that smaller HDKs tend to have higher counts.

Each of the three components has a share in the collection score. The component share of a less desired property never exceeds that smallest possible share of a more desired property's component value.

# 4. EXPERIMENT SETUP

This section describes the corpus, query set and query logs that were used in the evaluation process, and continues to describe how the collection selection effectiveness of Sophos was measured.

## 4.1 Data collections

### *4.1.1 WT10G corpus splits*

The Web Track 10GB corpus (WT10g) was developed for the Text REtrieval Conference[1] and consists of 10GB of web pages (1,692,096 documents on 11,680 servers). Compared to regular TREC corpora, WT10g should be more suited for distributed information retrieval experiments, due to the existence of hyperlinks, differences in topics, variation in quality and presence of duplicates [3, 8].

[1] http://trec.nist.gov/

Four splits of the WT10G document corpus were made to look at the effect of document corpora on collection selection. Every split is a set of collections; every collection is a set of documents. The numbers 100 and 11512 indicate the amount of collections in the corpus split.

**IP Split:** Documents are put in collections based on the IP addresses of the site where a document was residing. This results in 11,512 collections.

**IP Merge 100:** A cluster is created by grouping up to 116 collections, which results in 100 collections. Grouping is done in order of IP address. This split simulates the scenario of indexing the search engines that index many servers.

**Random 100 and Random 11512:** Two random splits with 100 collections and with 11,512 collections were created. Documents were randomly assigned to a collection. The number of 11,512 collections was chosen to be able to compare a random split with the IP Split.

The number of documents in random splits is relatively constant, but varies in IP based collections from less than 10 up to more than 1000 documents. This is typical for the size of sites on the Internet; the number of documents per server follows a Zipf distribution on the Internet [1]. The IP based splits show signs of conformance to a Zipf distribution [4]. This means that the IP based splits can be compared to the Internet in terms of distribution of the number of documents and the sizes of the collections.

The *merit* of a collection is the number of relevant documents in a collection for a particular query. The IP based corpus splits have a larger deviation in merit among the collections. This contrasts with random splits, which by approximation have equal merit for each collection [4].

### 4.1.2 WT10g retrieval tasks

Fifty WT10g informational 'ad-hoc' queries were used for evaluation (query numbers 501–550). The queries have a query number, title, description and a narrative description of the result that is considered relevant. The title is a small set of words which was used as the query text. The narrative descriptions were used by humans to assign relevance judgments to documents. The relevance judgments can be used to count the number of relevant documents in the collections, which in turn can be used to measure collection selection effectiveness.

There are three types of relevance judgments: not relevant (0), relevant (1) and authoritative (2). There can be multiple authoritative documents in the document corpus for a query, but for some queries there are no authoritative documents. All authoritative judgments are converted to 1, so documents are either relevant or not relevant. This allows for evaluation of the collected merit.

### 4.1.3 Query logs

AOL published a query log with 21,011,340 queries [19]. The log has been anonymized and consists of several data fields: the actual query issued and the query submission date and time, and an anonymous user ID number. The release of the anonymized data set was controversial at the time because it was proven possible to link an ID to a real person. To respect the anonymity of the users, we used a stripped version of the query log that only contains the actual queries issued in random order (and none of the other metadata).

We also used two query logs that were published by Excite[2] that were stripped in the same way. One query log from 1997 contains 1,025,907 queries and another query log from 1999 contains 1,777,251 queries.

Finally, a fourth query log with 3,512,320 unique queries was created by removing all queries from the AOL query log that were issued only once. This query log will be referred to as AOL2. The other logs are called AOL, Excite97 and Excite99.

## 4.2 Method

To evaluate the performance of our collection selection system, we adopted the precision and recall metrics for collection selection as defined by Gravano *et al.* [13]. We start by obtaining the *retrieval system ranking* ($S$) for a query, which contains up to 1,000 collections. We also create the *best ranking* ($B$) which is the best possible ranking for a particular query; collections are ranked by their amount of merit with respect to a query.

Given query $q$ and collection $c_i$, the merit within a collection can be expressed using $merit(q, c_i)$. The merit of the $i^{\text{th}}$ ranked collection in rankings $S$ and $B$ is given by $S_i = merit(q, c_{s_i})$ and $B_i = merit(q, c_{b_i})$.

The obtained recall after selecting $n$ collections can be calculated by dividing the merit selected by the best possible retrieval system:

$$R_n = \frac{\sum_{i=1}^{n} S_i}{\sum_{i=1}^{n} B_i} \tag{3}$$

Precision $P_n$ is the fraction of top $n$ collections that have non-zero merit:

$$P_n = \frac{|\{\text{sc} \in \text{Top}_n(S) | \text{merit}(q, \text{sc}) > 0\}|}{|\text{Top}_n(S)|} \tag{4}$$

The precision and recall obtained by Sophos is compared to the collection selection results from a baseline of Language Modeling with Dirichlet Smoothing (LMDS) as implemented by INDRI. The baseline system has one parameter $\mu$ that is set to 2500. Krovetz word stemming is applied to the collections.

Section 3 introduced Sophos and its parameters. There are three parameters for peers: the maximum key length $h_{\max}$, the maximum key frequency before calling a key frequent ($tf_{\max}$) and the window size $ws$. Based on numbers used by Luu *et al.* [15], we use the following settings: $tf_{\max} = \{250, 500, 750\}$ and $ws = \{6, 12, 18\}$. The average query length on the Internet is 2.3 [14, 30]. We use this observation to set $\text{h}_{\max}$ to 3. Setting it smaller would require many intersections of term sets (with associated collections) to answer queries. Setting it larger would result in many term sets that are rarely queried for. For the broker, the collection maximum $cm$ is tested for values 5, 10, 20 and 50.

To evaluate Sophos, the collections are processed by removing stop words – drastically reducing the number of invaluable keys and speeding up term set generation – and applying Porter word stemming.

Finally, we will look at the precision and recall with query-driven indexing using four different query logs. By pruning the keys from the peer indices that do not occur in a query

[2] `http://www.excite.com/`

log. At the same time, we will look at the number of term sets (keys) in the broker index to get an idea about its size.

# 5. RESULTS

## 5.1 Index size

Figure 3 shows the number of collection identifier counters within the broker index for different indexing settings of indexing the IP Split. Spread over single, double and triple term set collection identifier counters, the number of counters are a good indication for the actual broker index size. The figure shows that the number of counters decreases when the term set frequency maximum is increased.

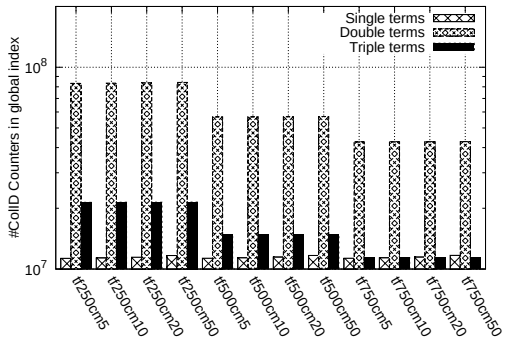


**Figure 3: Number of collection ID counters with IP Split.**

A more substantial reduction of collection identifier counters – of roughly 70% – can be achieved by using query-driven indexing, as shown in Figure 4. The figure shows the number of collection identifier counters after indexing the Random 11,512 corpus split with or without query-driven indexing. Figure 5 depicts the obtainable index size reduction by using different query logs. The Excite query logs contain significantly less query term sets than the AOL query log. The figure shows that more keys are pruned from the peer indices, resulting in a smaller broker index. The figure shows that using Excite query logs instead of the standard AOL query log can result in roughly 75% less collection identifier counters.

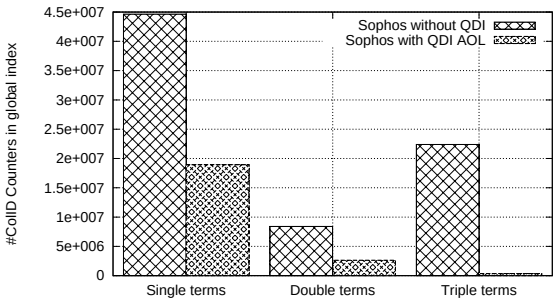


**Figure 4: Number of collection identifier counters stored per QDI strategy for Random 11512 corpus split**

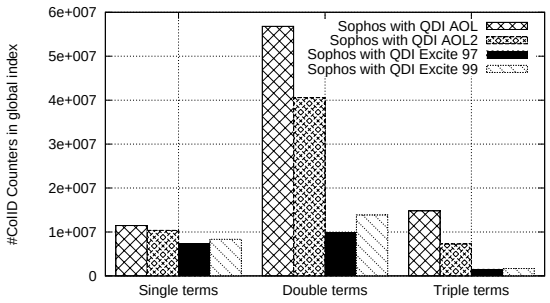


**Figure 5: Number of collection identifier counters stored per QDI strategy for IP Split**

## 5.2 Collection selection performance

Table 2 shows the average precision and recall over 50 queries that were calculated with Formula 3 and Formula 4. The numbers are calculated for four different corpus splits with which the baseline (LMDS) and Sophos were tested. Sophos was used with the following settings: query-driven indexing with the AOL query log, $tf_{\max} = 250$, $cm = 20$ and $ws = 6$. Due to memory constraints, we were unable to run Sophos with a window size larger than 6. The table shows that the baseline outperforms Sophos on the Random 11,512 corpus split, but Sophos outperforms the baseline on the IP split.

This is displayed in more detail in Figure 6, which shows the average recall of Sophos and the baseline after selecting $n$ collections. Sophos was tested using different query logs, $tf_{\max} = 250$ and $cm = 50$. Interestingly, pruning with the smallest Excite query logs results in the best recall figures, possibly because the queries were logged at the same time as when the corpus was crawled.

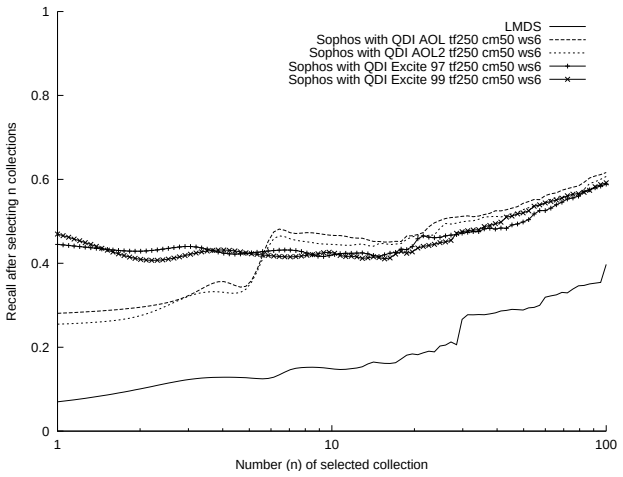


**Figure 6: Recall of different collections selection methods on IP Split**

# 6. CONCLUSIONS

We introduced the collection selection system Sophos that uses highly discriminative keys in peer indices to construct a broker index. The broker index contains keys that are good discriminators to select collections (or peers). To limit

| Corpus split | Collection selection method | P@1 | P@10 | P@20 | P@50 | R@1 | R@10 | R@20 | R@50 |
|---|---|---|---|---|---|---|---|---|---|
| Random 100 | LMDS | 0.290 | 0.251 | **0.249** | **0.217** | 0.314 | 0.435 | **0.526** | **0.683** |
| | Sophos QDI AOL tf250 cm20 | **0.330** | **0.254** | 0.237 | 0.208 | **0.379** | **0.436** | 0.493 | 0.644 |
| Random 11512 | LMDS | **0.140** | **0.096** | **0.084** | **0.069** | **0.233** | **0.196** | **0.186** | **0.198** |
| | Sophos QDI AOL tf250 cm20 | 0.040 | 0.036 | 0.037 | 0.026 | 0.067 | 0.073 | 0.082 | 0.073 |
| IP Merge 100 | LMDS | 0.280 | 0.202 | 0.188 | 0.154 | **0.489** | **0.489** | 0.567 | 0.755 |
| | Sophos QDI AOL tf250 cm20 | **0.300** | **0.254** | **0.214** | **0.160** | 0.211 | 0.485 | **0.626** | **0.825** |
| IP Split | LMDS | **0.170** | 0.110 | 0.083 | 0.056 | 0.070 | 0.149 | 0.183 | 0.289 |
| | Sophos QDI AOL tf250 cm20 | **0.170** | **0.147** | **0.121** | **0.091** | **0.280** | **0.466** | **0.466** | **0.548** |

**Table 2: Average precision and recall over 50 queries after selecting $n$ collections, high scores shown in bold**

the number of keys transferred to the broker, and to reduce the broker index size, we employed query-driven indexing to only store the keys that are queried for by users. Similar studies were performed for document retrieval [15], but to the best of our knowledge, we are the first to apply this approach for collection selection.

Precision and recall was measured using 50 queries on the WT10g TREC test corpus and compared to a state-of-the-art baseline that uses language modeling with Dirichlet smoothing (LMDS). The results showed that our system outperformed the baseline with the IP split as test corpus. This is promising, because the IP based splits most closely resemble the information structure on the Internet. LMDS was better capable of selecting information in random based splits, because it stores all available information about the collections. In random based splits, relevant documents (and their corresponding terms) are mixed over many collections, making it hard for our approach to select highly discriminative keys that can discriminate collections with relevant documents.

Query-driven indexing is required to keep the broker index size manageable; a 70% index size reduction can be obtained by pruning keys using the AOL query log, another 75% reduction is possible by using a smaller query log. Our results on the IP split showed that pruning using the smaller Excite query logs resulted in higher precision and recall than with AOL query logs. The use of any query log resulted in higher precision and recall than the baseline results. This motivates further research using more advanced query-driven indexing strategies, such as described by Skobeltsyn [32], to further reduce the index size while improving the performance. It would also be interesting to make $tf_{\max}$ depending on the collection size.

## Acknowledgements

We are grateful to the Yahoo Research Faculty Grant programme and to the Netherlands Organisation for Scientific Research (NWO, Grant 639.022.809) for supporting this work. We would like to thank Berkant Barla Cambazoglu and the anonymous reviewers for their valuable comments that improved this paper.

## 7. REFERENCES

[1] L. A. Adamic and B. A. Huberman. Zipf's law and the internet. *Glottometrics*, 3:143–150, 2002.

[2] J. Arguello, J. L. Elsas, J. Callan, and J. G. Carbonell. Document representation and query expansion models for blog recommendation. In *Proc. of the 2nd Intl. Conf. on Weblogs and Social Media (ICWSM)*, 2008.

[3] P. Bailey, N. Craswell, and D. Hawking. Engineering a multi-purpose test collection for web retrieval experiments. *Inf. Process. Manage.*, 39(6):853–871, 2003.

[4] S. Bockting. Collection selection for distributed web search. Master's thesis, University of Twente, Feb. 2009.

[5] J. Callan. Distributed information retrieval. *Advances in Information Retrieval*, pages 127–150, 2000.

[6] J. Callan, W. B. Croft, and S. M. Harding. The inquery retrieval system. In *Proc. of the 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[7] J. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR '95: Proc. of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 21–28, New York, NY, USA, 1995. ACM.

[8] N. Craswell, P. Bailey, and D. Hawking. Is it fair to evaluate web systems using trec ad hoc methods. In *Workshop on Web Evaluation*. SIGIR'99, 1999.

[9] N. Craswell, P. Bailey, and D. Hawking. Server selection on the world wide web. In *DL '00: Proc. of the 5th ACM conference on Digital libraries*, pages 37–46, New York, NY, USA, 2000. ACM.

[10] D. D'Souza, J. Thom, and J. Zobel. A comparison of techniques for selecting text collections. In *ADC '00: Proceedings of the Australasian Database Conference*, page 28, Washington, DC, USA, 2000. IEEE Computer Society.

[11] D. D'Souza, J. A. Thom, and J. Zobel. Collection selection for managed distributed document databases. *Information Processing & Management*, 40(3):527–546, May 2004.

[12] D. D'Souza, J. Zobel, and J. A. Thom. Is cori effective for collection selection? an exploration of parameters, queries, and data. In *Proc. of the 9th Australasian Document Computing Symposium*, pages 41–46, Dec. 2004.

[13] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *International Conference on Very Large Databases, VLDB*, pages 78–89, 1995.

[14] T. Lau and E. Horvitz. Patterns of search: analyzing and modeling web query refinement. In *UM '99: Proc. of the 7th international conference on User modeling*, pages 119–128, Secaucus, NJ, USA, 1999.

Springer-Verlag New York, Inc.

[15] T. Luu, F. Klemm, M. Rajman, and K. Aberer. Using highly discriminative keys for indexing in a peer-to-peer full-text retrieval system. Technical report, TR: 2005041, EPFL Lausanne, 2005.

[16] P. Lyman and H. R. Varian. How much information, 2003. `http://www.sims.berkeley.edu/how-much-info-2003`, retrieved on April 23, 2008.

[17] D. Metzler. Indri retrieval model overview, July 2005. `http://ciir.cs.umass.edu/~metzler/indriretmodel.html`, retrieved on January 20, 2008.

[18] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735–750, 2004.

[19] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06: Proc. of the 1st international conference on Scalable information systems*, page 1, New York, NY, USA, 2006. ACM.

[20] I. Podnar Zarko, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Scalable peer-to-peer web retrieval with highly discriminative keys. *IEEE 23rd International Conference on Data Engineering (ICDE)*, 2007.

[21] A. L. Powell and J. C. French. Comparing the performance of collection selection algorithms. *ACM Trans. Inf. Syst.*, 21(4):412–456, 2003.

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

[23] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-beaulieu, and M. Gatford. Okapi at trec-3. In *TREC-3 Proc.*, pages 109–126, 1995.

[24] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[25] N. Sato, M. Udagawa, M. Uehara, Y. Sakai, and H. Mori. Query based site selection for distributed search engines. *Distributed Computing Systems Workshops, 2003. Proc.. 23rd International Conference on*, pages 556–561, 2003.

[26] J. Seo and W. B. Croft. Umass at trec 2007 blog distillation task. In *Proc. of the 2008 Text REtrieval Conference*. NIST, 2007.

[27] M. Shokouhi, J. Zobel, S. Tahaghoghi, and F. Scholer. Using query logs to establish vocabularies in distributed information retrieval. *Information Processing and Management*, 43(1):169–180, 2007.

[28] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR '03: Proc. of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 298–305, New York, NY, USA, 2003. ACM.

[29] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. *Proc. of the 11th international conference on Information and knowledge management*, pages 391–397, 2002.

[30] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[31] G. Skobeltsyn. *Query-driven indexing in large-scale distributed systems*. PhD thesis, EPFL, Lausanne, 2009.

[32] G. Skobeltsyn, T. Luu, I. Podnar Zarko, M. Rajman, and K. Aberer. Query-driven indexing for scalable peer-to-peer text retrieval. *Future Generation Computer Systems*, 25(1):89–99, June 2009.

[33] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proc. of the International Conference on Intelligence Analysis*, 2004.

[34] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. *Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 175–186, 2003.

[35] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search system. In Proc. of the International Conference on Very Large Databases (VLDB), Aug. 2004.

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

[37] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. *Proc. of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 254–261, 1999.

[38] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the internet. In *Proc. of the 5th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–50. World Scientific Press, 1997.

[39] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proc. of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, New York, NY, USA, 2001. ACM.