Sketch-based Indexing of n-Words

Samuel Huston
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
Amherst, MA, 01002, USA
sjh@cs.umass.edu

J. Shane Culpepper
School of Computer Science & Information Technology
RMIT University
Melbourne, VIC, 3001, Australia
shane.culpepper@rmit.edu.au

W. Bruce Croft
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
Amherst, MA, 01002, USA
croft@cs.umass.edu

ABSTRACT

Formulating and processing phrases and other term dependencies to improve query effectiveness is an important problem in information retrieval. However, accessing these types of statistics using standard inverted indexes requires unreasonable processing time or incurs a substantial space overhead. Establishing a balance between these competing space and time trade-offs can dramatically improve system performance.

In this paper, we present and analyze a new index structure designed to improve query efficiency in term dependency retrieval models, with bounded space requirements. By adapting a class of (ϵ, δ)-approximation algorithms originally proposed for sketch summarization in networking applications, we show how to accurately estimate various statistics important in term dependency models with low, probabilistically bounded error rates. The space requirements of the sketch index structure is largely independent of this size and the number of phrase term dependencies.

Empirically, we show that the sketch index can reduce the space requirements of the vocabulary component of an index of all n-grams consisting of between 1 and 5 words extracted from the Clueweb-Part-B collection to less than 0.2% of the requirements of an equivalent full index. We show that n-gram queries of 5 words can be processed more efficiently than in current alternatives, such as next-word indexes. We show retrieval using the sketch index to be up to 400 times faster than with positional indexes, and 15 times faster than next-word indexes.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and software—*performance evaluation*.; H.3.1 [**Content Analysis and Indexing**]: Indexing methods; H.3.3 [**Information Search and Retrieval**]: Information filtering

Keywords

Sketching, Indexing, Scalability, Term Dependency Models

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

1. INTRODUCTION

Term dependency models are a compelling new approach to improving the effectiveness in ranked document retrieval. A term dependency is any relationship between two or more terms. Examples of term dependencies include noun phrases, verb phrases, ordered windows, unordered windows, spans of text, or any sequence of n words. Many recently developed retrieval models depend on statistics extracted for query term dependencies [1, 7, 9, 10, 16]. While these methods have been shown to significantly improve the *effectiveness* of the retrieval model, prior work has not addressed how to *efficiently* generate the necessary statistics at query time.

We investigate the problem of providing access to document level statistics for *all* n-words in the collection in a space and time efficient manner. We define the term n-*word* as any sequence of n sequential words. In order to avoid confusion with character level n-grams, or linguistic definitions of various types of phrases, we will use the term n-*word* throughout this paper. This type of term dependency is often used as a surrogate for more complex, linguistic term dependencies. Improving the efficiency of the calculation of this type of retrieval model feature can improve the efficiency of each of the above retrieval models.

Previous approaches have used a variety of different index structures. Positional indexes [15] support the calculation of n-word statistics by processing n term posting lists simultaneously, and comparing position offset information for each term in each document. Next-word indexing [14] was proposed as an attractive trade-off between index space and retrieval efficiency for processing phrase queries. A next-word index stores position data for pairs of terms. Similar to the positional index, statistics for longer phrases are computed at query time by comparing positional data.

Direct indexes of all n-words dramatically reduces the cost of processing positional information at query time, however these indexes have unacceptably large space requirements. Filtered indexes [8] reduces this problem by discarding a large fraction of the vocabulary. Alternatively, query-log analysis could be used to guide the filtering of direct indexes, similar to a cache of intersected posting lists [12].

In this paper, we present an indexing structure using data stream sketching techniques to estimate n-word statistics. Our sketch index is derived from a **CountMin** sketch [5], and designed to minimize space usage while still producing accurate statistical estimates. This strategy also ensures that the space required by the index is independent of the number of indexed n-word terms, while still supporting efficient query processing.

Conceptually, our summary sketch is an (ϵ, δ)-approximation of a full inverted index structure. The index representation is capable of estimating statistics for specific n-word with bounded performance. We show that the relative error of the term dependency

statistics being estimated is within the theoretic bounds of similar data streaming applications, and describe how the bounds minimize the space requirement in practice. We show that the retrieval efficiency of the sketch index is comparable to full indexes, and many times faster than positional and next-word indexes.

This paper is structured as follows: Section 2 presents the necessary background on data stream sketching techniques; Section 3 presents the algorithmic framework for our term dependency statistics estimator, and outlines the probabilistic error bounds ensured by the representation; Section 4 evaluates the performance of our new estimator empirically. We conclude in Section 5.

2. FREQUENCY-BASED SKETCHING

Algorithms for approximating frequency moments have advanced dramatically in the last twenty years [3, 4]. This line of research is based on the streaming model of computation, and has widespread applications in networking, databases, and data mining [11]. Much of the work in the networking community using these tools has focused on identifying "heavy-hitters", or top-k items (see [2] or [3]). If only the k most frequent items must be accurately estimated, counter-based approaches work well in practice. However, counter-based methods are generally not sufficient if estimates for *all* the items in a stream are desirable, since the number of counters must be significantly fewer than the number of unique items in the stream. For frequency estimation of any item in a stream, various "sketching" methods are an appropriate alternative.

We focus our discussion on the **CountMin** sketch, recently proposed by Cormode and Muthukrishnan [5]. The key idea of a **CountMin** sketch is to create an array of $r \times w$ counters, with independent hash functions for each row r. The hash functions map each update to set of counters, one in each row r. In streams that do not support deletions, this ensures that the frequency of any item $f(i)$ in the sketch is an overestimate. The collisions for i on any row is $\sum_{1 \leq i' \leq \sigma, i' \neq i} f(i')/w$. **CountMin** can be used to estimate $\hat{f}_i$ with error at most ϵn with probability at least $1 - \delta$ using $\mathcal{O}(\frac{1}{\epsilon} \log \frac{1}{\delta})$. The time per update is $\mathcal{O}(r)$ where $r = \log \frac{1}{\delta}$ and $w = \mathcal{O}(\frac{1}{\epsilon})$.

1	0	7	0	0
0	5	3	0	0
0	0	0	2	6

Figure 1: Example **CountMin** sketch containing frequency data for 3 items: $f(i_1) = 1$, $f(i_2) = 5$ and $f(i_3) = 2$. Where i_2 is hashed to each of the highlighted cells. The frequency of i_2 can be estimated as the minimum value of the highlighted cells $f(i_2) = 5$.

An example **CountMin** sketch is shown in Figure 1. Updates are performed as follows: when an item, i, is added to the sketch, one counter is incremented in each row of the **CountMin** sketch. The correct cell is determined by the corresponding hash function. Formally:

$$\forall_{j<r} : count[j, h_j(i)] = count[j, h_j(i)] + 1$$

If the stream contains only positive frequencies, the frequency i can be estimated by returning the minimum count in the set.

$$\hat{a}_i = \min_j \ count[j, h_j(x_i)]$$

For streams allowing positive or negative frequencies, the frequency of i is estimated by returning the median of the r counts.

$$\hat{a}_i = \text{median}_j \ count[j, h_j(x_i)]$$

3. SKETCHING STATISTICS FOR N-WORDS

Let $\mathcal{C}$ be a text collection partitioned into d documents $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_d\}$ containing at most σ_n terms. Here, a term t can be n-words, a sequence of n adjacent "words". So, σ_1 represents the total number of 1-words in a collection. An *inverted index*, $\mathcal{I}$ counts the number of times each term t appears in each document $\mathcal{D}_j$, for $1 \leq t \leq \sigma_n$ and $1 \leq j \leq d$. Conceptually, this can be represented as a $\sigma_n \times d$ matrix, $\mathcal{M}$, as every term t appears in all d documents in the worst case.

The following statistical notation will apply for our discussion:

- $f_{d,t}$, the frequency of term t in document d;
- $f_{q,t}$, the frequency of term t in the query;
- f_t, the number of documents containing at least one occurrence of term t;
- F_t, the number of occurrences of term t in the collection;
- d, the number of documents in the collection;
- σ_n, the number of indexed terms in the collection; and
- $|\mathcal{C}| = \sum_{i=0}^{\sigma_1} F_{t_i}$.

In practice, $\mathcal{M}$ is sparse, and $\mathcal{I}$ is often represented as a compressed adjacency list. An inverted index is a mapping of keys to a list of document counters. For each document identifier d, $f_{d,t}$ is maintained. Traditionally, each term in the vocabulary is stored explicitly in a lookup table.

Now, consider the case of constructing an inverted index of n-words. The collection $\mathcal{C}$ can contain at most $(|\mathcal{C}| - n$ distinct n-words. This number is often less than the σ_1^n possibilities, but still much larger than σ_1, thus increasing the number of rows in $\mathcal{M}$.

In this paper, we investigate how to apply the ideas presented by Cormode and Muthukrishnan [5] to fix the number of rows in $\mathcal{M}$ and still provide accurate statistical information. Interestingly, d is already static for a given collection, and $d \ll |\mathcal{C}|$. But, the number of σ_n rows increases with n, and we would like to minimize this overhead. Note that the total number of rows required in the sketch is proportional to $r \cdot f_t$. So, if we reduce $\mathcal{M}$ to a linear projection of f_t, we can use **CountMin** to accurately approximate $\hat{f}_t$. Recall that the number collisions for t on any row in the sketch is $\sum_{1 \leq t' \leq \sigma, t' \neq t} f'_t/w$. Using a Markov inequality argument, Cormode and Muthukrishnan [5] show that by setting $w = 2/\epsilon$ and $r = \log 1/\delta$ in the sketch, the estimate $\hat{f}_t$ is at most ϵF_1 with probability at least $1 - \delta$, where F_1 is the first frequency moment $\sum_{1 \leq t' \leq \sigma_n} f'_t$, the sum of all of the frequencies.

In a sketch representation of an inverted index, each distinct term is replaced with a hash value where each hash value *may* represent more that one term. This reduction means that the vocabulary of n-words no longer needs to be stored with the index. If a simple hashing representation were used, then there is no mechanism available to resolve collisions unless each term string is accessible to the table. However, using the collision mitigation strategy of a sketch, such as the method described for **CountMin**, we are able to reduce the probability that hashing collisions will result in incorrect results.

Our new indexing structure is composed of an $r \times w$ matrix of pointers to $r \times w$ postings lists. Conceptually, this matrix is equivalent to a **CountMin** sketch designed to estimate $\hat{f}_t$ with one twist: we do not simply use a single counter to aggregate $\hat{f}_t$, but rather

$(0, h_0(a\ b\ c))$	$(\mathcal{D}_1, 1)$	$(\mathcal{D}_2, 1)$	$(\mathcal{D}_3, 5)$	$(\mathcal{D}_5, 8)$
$(1, h_1(a\ b\ c))$	$(\mathcal{D}_1, 2)$	$(\mathcal{D}_3, 3)$	$(\mathcal{D}_5, 2)$	$(\mathcal{D}_6, 5)$
$a\ b\ c$	$(\mathcal{D}_1, 1)$	$(\mathcal{D}_3, 3)$	$(\mathcal{D}_5, 2)$	–

Figure 2: Example extraction of the statistics for a single term dependency in our sketch representation. The first two sketch posting lists are processed to produce the intersected posting list for the term $a\ b\ c$. Colors are used for each output document posting.

	Robust-04	ClueWeb-B
Disk Space	1.9 GB	1460 GB
Collection Length	$252 \cdot 10^6$	$39.8 \cdot 10^9$
Document Count	$0.5 \cdot 10^6$	$50 \cdot 10^6$
Term Vocab.	$0.775 \cdot 10^6$	$0.0984 \cdot 10^9$
2-word Vocab.	$24.5 \cdot 10^6$	$1.37 \cdot 10^9$
3-word Vocab.	$95.1 \cdot 10^6$	$5.96 \cdot 10^9$
4-word Vocab.	$166 \cdot 10^6$	$11.6 \cdot 10^9$
5-word Vocab.	$204 \cdot 10^6$	$15.8 \cdot 10^9$

Table 1: Global statistics for TREC Collections Robust-04 and ClueWeb-B.

allow multiple document counters attached in list-wise fashion to each cell in the **CountMin** sketch. These document counters are then used to aggregate $\hat{f}_{d,t}$.

This approach allows us to fix the size of the lookup table independent of the vocabulary of n-words being indexed. We do not attempt to fix the number of $\hat{f}_{d,t}$ counters. As in a standard inverted index, every term could appear in every document, producing a maximum of $d \cdot |\sigma_n|$ counters in the worst case. But, in practice, the distribution is skewed, and many terms have very few non-zero $\hat{f}_{d,t}$ counters. Note that since the width of $|\sigma_n|$ is fixed in our approach, the number of counters is largely independent of the order of n, but rather some percentage of the counters are redistributed in the redundant postings lists.

We now discuss how to estimate point queries using our approach. By using the biased estimation of a **CountMin** sketch of only positive counts, our estimates of $\hat{f}_t$, and subsequently $\hat{f}_{d,t}$, are guaranteed to be an overestimate of the true term counts. Furthermore, the same formal arguments using the Markov inequality and Chernoff bounds can be made for bounding $\hat{f}_t$, and subsequently $\hat{f}_{d,t}$, we could reasonably expect for each cell. So, to estimate $\hat{f}_t$ using **CountMin**, we would take $\min_j\ count[j, h_j(x_i)]$. But, each counter $count[\ ]$ is actually a pointer to a postings list, containing approximately $\hat{f}_t$ counters. When the posting list for any t is requested, the *intersection* of the r posting lists is performed using the minimum frequency from the w counters representing each sketched posting list. Figure 2 shows an example of the intersection process, which represents the $\min_j$ for a given t.

4. EXPERIMENTS

4.1 Experimental Setup

We compare the performance of our new term dependency statistical estimator over two TREC collections: Robust-04 and Clueweb-B. Statistical properties for each collection is shown in Table 1. As a pre-processing step, all terms in each collection were mapped to 32-bit integers, where an integer corresponds to a unique word in the collection. This is a standard technique to improve index and vocabulary compression, and ensures uniform space requirements across all of the baselines used in this study.

In each of our experiments, we measure index properties and retrieval performance on n-word data. An n-word is defined as any sequence of n sequential words. In the literature, each distinct sequence is sometimes referred to as a phrase, a shingle, or an n-gram.

We compare the performance of our statistical n-word estimator with five other index structures capable of generating the true statistics of n-word term dependencies. We compare our approach with positional indexes [15], full indexes of n-words, filtered indexes [8], query-log-based indexes [12] and next-word indexes [14]. The query-log-based indexes are created by ordering the query log by timestamps, then indexing all n-words extracted from the first 90% of queries. The remaining 10% are used to test the retrieval efficiency.

To ensure a fair comparison, all index structures are implemented as disk based b-tree indexes that implement a variety of modern index compression techniques, including d-gap and *vbyte* integer compression for posting list data, and prefix-based vocabulary compression for b-tree blocks. We use 32 kB b-tree blocks. Witten et al. [15] provides a good description of standard compression techniques amenable to text indexing and retrieval. Hash functions used in the sketch indexes are pair-wise universal hash functions.

Our experiments focus on three key aspects of our new statistical term dependency estimator: relative statistical error, space usage, and retrieval efficiency. The sketch index can only estimate various statistical values, and so we assess the relationship between ϵ and δ parameter selection and the resulting relative error. We measure statistical error in the collection for a random sample of term dependencies. Space requirements of our approach are compared with each comparable index structure. We compare retrieval efficiency of our estimator with the same set of comparable index structures using n-words extracted from the last 10% of the AOL query log.

Due to space constraits, we omit the details of experiments testing the effect of using a sketch index on information retrieval effectiveness. These experiments show that using sketch index with conservative parameters resulted in no change in the retrieval effectiveness.

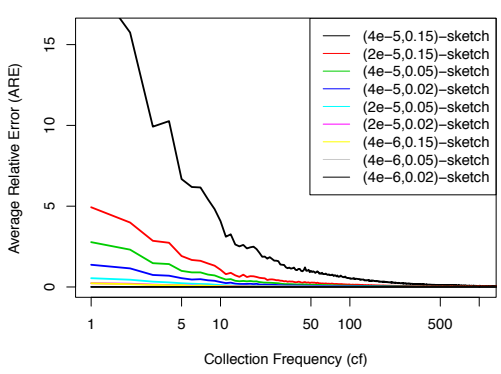
Each index structure we investigate in this section is implemented as an extension to the Galago package, provided by the Lemur Toolkit [6]. All timing experiments were run on a machine with 8-core Intel Xeon processors, with 16 GB of RAM, running the Centos distribution of Linux, Using a distributed, network-attached, 4-node Luster file system to store index data.

4.2 Estimation of Collection Frequency

As discussed previously, sketch indexes provide an attractive trade-off between space usage and accuracy. In this section, we investigate the relationship between the (ϵ, δ) parameters and the quality of approximation by comparing the relative error of our approach to the true collection statistics. We show results computed on indexes created over n-words, extracted from the TREC Robust-04 collection. Average Relative Error (ARE) is defined as the average of the absolute difference between the true value and the estimated value. In our case: $ARE = \frac{1}{|C|} \sum_{t \in C} \frac{|f_t - \hat{f}_t|}{f_t}$.

Figure 3 shows ARE values grouped by f_t for 3-words using our approach with a variety of parameters. Other sized n-words were tested, and produce similar results. Data shown in this graph is aggregated from 10 instances of sketch indexes with each parameter setting.

The graph shows that conservative settings for (ϵ, δ) can ensure



3-word sketch index

Figure 3: Average relative error of 3-word frequency statistics extracted from 10 instances of sketch indexes over Robust-04 data, using each set of parameters. Sketch index parameters shown are $\delta \in \{0.15, 0.05, 0.02\}$, and $\epsilon \in \{4 \cdot 10^{-5}, 2 \cdot 10^{-5}, 4 \cdot 10^{-6}\} \approx \{1{,}000/|C|, 5{,}000/|C|, 10{,}000/|C|\}$.

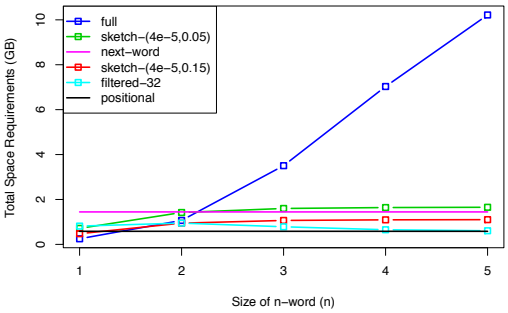


Figure 4: Space requirements for sketch and full indexes across a range of n-word sizes over Robust-04 data. The *full* index shown on this graph is an inverted index storing frequency data for all n-words. The filtered index stores frequency data for n-words that occur at least 32 times. The parameters used for the sketch indexes are $\epsilon = 4 \cdot 10^{-5} \approx 10{,}000/|C|$ and $\delta \in \{0.05, 0.15\}$.

a very low error rate in the estimation of collection statistics. Additionally, we can see that using overly restrictive values of ϵ and δ can degrade our estimates, particularly for infrequent items. This insight is not surprising, since summary sketching is primarily used in networking scenarios that require only the top-k items in a set to be accurately estimated.

Our experiments suggest that $\epsilon \leq 4 \cdot 10^{-6}$ and $\delta \leq 0.05$ produce a very low relative error for all n-words tested in the Robust-04 collection. All other experiments are based on these parameters.

4.3 Space Requirements

We now compare space requirements for our estimator with the space requirements of the baseline index structures. Each index structure presented is capable of producing statistics for n-words present in the collection. Figure 4 shows space requirements for the sketch index, as compared to the baselines using the Robust04 Collection. As expected, the sketch estimator requires only a fraction of the space of a full index. For $n = 5$, a sketch index requires less than 20% of the space required by a full five-word index. We can also see from the graph that sketching representations for any n require only slightly more space than other baseline indexes: positional, next-word, and filtered index structures. Query log cache structures are omitted from this graph as we do not have an appropriate query log.

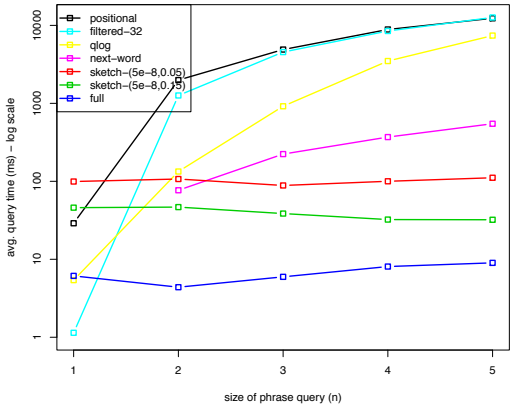


Figure 5: Query processing times for indexes over the Clueweb-B collection. Each data point is the average of 5 runs of 10,000 phrase queries.

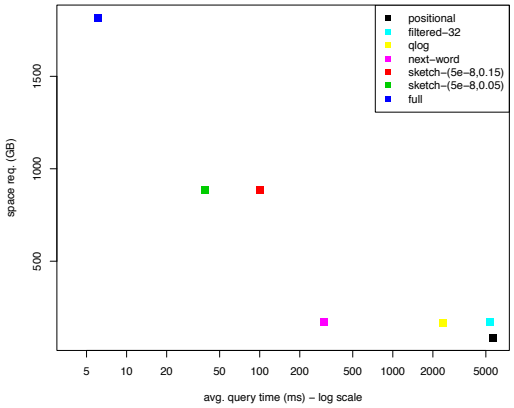


Figure 6: Query processing time versus space usage. Query processing time as a function of index space requirements. Each data point is the average of 5 runs of 10,000 of n-word queries. The size of the query for each data point is indicated on the graph. Colored lines are used to link data points for each index structure.

An important prediction for the sketch index is supported by this data; the space requirement for sketch indexes do not increase significantly as n increases. This is because the number of rows in our estimator is not bound by n or by the size of the vocabulary, but rather by ϵ. This means that a sketch representation of much larger term dependencies are feasible on commodity computing hardware.

4.4 Retrieval Efficiency

We now evaluate the retrieval efficiency of our statistical estimator relative to the other index structures for n-word queries. We

use the ClueWeb-B collection and queries sampled from the AOL query log. By using a web collection to target the queries, no query translation [13] methods are required.

Recall that we use the first 90% of the time-ordered query log to create the query log cache index structure. Test queries are sampled from the remaining 10% of the log. From this subset of the query log, we uniformly at random sample $10,000$ n-word queries for each size $1 \leq n \leq 5$.

The query processing speed for each index structure over a fixed query length is measured as the average of 5 timed runs of the corresponding sample of queries. In each run the query order is randomized. The retrieval system is initialized for each experiment by running a randomly selected sub-sample of 2,000 queries. Initialization ensures that a portion of the index data is held in memory-based file buffers, as it would be in a live retrieval system. Figure 5 shows query processing time as the length of the query increases for each index structure. Note that times shown in this graph are displayed in log scale. All data points in the graph are significantly different from each of the other index structures, ($p < 0.05$ using the unpaired t-test).

We observe that query processing time of the sketch index data structure is significantly faster that each of the positional, filtered, query-log and next-word indexes. Unlike position based indexes, we can see that the sketch index is scalable as n increases, the time to process n-word queries does not increase with n. This data shows that sketch indexes are over 300 times faster than a positional index, and 15 times faster than next-word indexes, for processing 5-word indexes.

Figure 6 shows the trade-off between query processing speed and space usage. Data shown represents the total space requirements to process 1-to-5-word queries for each index structure, and the average time to process all of the sampled queries used in the timed experiments above. Query processing times are shown in log scale. Data structures that provide the best trade-off between space requirement and retrieval efficiency will approach the origin. This graph clearly demonstrates that the our n-word statistical estimator offers an attractive trade-off between space usage and query efficiency when compared with all other baselines.

5. CONCLUSION

In this paper, we have investigated the problem of accurately estimating n-word statistics in large data collections. Existing solutions for this problem require large space, or are inefficient for query processing in practice. We have presented a novel approach to estimating n-word statistics for information retrieval tasks. By using frequency sketching techniques developed for data streaming applications, we can accurately estimate collection statistics, and provide an attractive trade-off between space and relative error. Furthermore, we show how to bound the space usage of the data structure. Importantly, the number of distinct n-words does not directly influence the space bounds, allowing us to accurately estimate a wide variety statistics efficiently.

We have demonstrated in this paper that the sketch index data structure provides a new and useful trade-off between query processing time and space requirements for n-word queries. Importantly, we also have shown that this index structure is both scalable in both query processing time and space requirements for the size of the query, n.

6. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF CLUE IIS-0844226, in part by NSF grant #CNS-0934322 and in part by the Australian Research Council. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

[1] M . Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proc. of the 31st ACM SIGIR conf.*, pages 491–498, 2008.

[2] R. Berinde, G. Cormode, P. Indyk, and M. J. Strauss. Space-optimal heavy hitters with strong error bounds. In *PODS*, pages 157–166, 2009.

[3] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. volume 1, pages 1530–1541, 2008.

[4] G. Cormode and M. Hadjieleftheriou. Methods for finding frequent items in data streams. *The VLDB Journal*, 19(1): 3–20, 2010.

[5] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. In *LATIN*, pages 29–38, 2004.

[6] W.B. Croft, J. Callan, and http://lemurproject.org/contrib.php. The lemur project. http://www.lemurproject.org/, 2001-2012. Lemur Toolkit, indri, galago, ClueWeb09.

[7] B. He, Huang J. X., and Zhou X. Modeling term proximity for probabilistic information retrieval models. *Journal of Information Sciences*, 181(14):3017 – 3031, 2011.

[8] S. Huston, A. Moffat, and W. B. Croft. Efficient indexing of repeated n-grams. In *Proc. of the 4th ACM WSDM*, pages 127–136, 2011.

[9] W. Lu, S. Robertson, and A. MacFarlane. Field-weighted xml retrieval based on bm25. In *Proc. of the 4th INEX*, pages 161–171, 2006.

[10] D. Metzler and W.B. Croft. A markov random field model for term dependencies. In *Proc. of the 28th ACM SIGIR*, pages 472–479, 2005.

[11] S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers, 2005.

[12] R. Ozcan, I.S. Altingovde, B.B. Cambazoglu, F.P. Junqueira, and O. Ulusoy. A five-level static cache architecture for web search engines. *Information Processing & Management*, 2011.

[13] W. Webber and A. Moffat. In search of reliable retrieval experiments. In *Proc. 10th ADCS*, pages 26–33, 2005.

[14] H. E. Williams, J. Zobel, and P. Anderson. What's next? Index structures for efficient phrase querying. In *Proc. of ADC*, pages 141–152, 1999.

[15] I.H. Witten, A. Moffat, and T.C. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999.

[16] X. Xue and W.B. Croft. Representing queries as distributions. In *Proc. of 33rd ACM SIGIR*, pages 9–12, 2010.