Reasoning about Similarity Queries in Text Retrieval Tasks

Xiaohui Yu
School of Information Technology
York University
Toronto, ON, Canada M3J 1P3
xhyu@yorku.ca

Yang Liu
Department of Computer Science & Engineering
York University
Toronto, ON, Canada M3J 1P3
yliu@cse.yorku.ca

ABSTRACT

In many text retrieval tasks, it is highly desirable to obtain a "similarity profile" of the document collection for a given query. We propose sampling-based techniques to address this need, using calibration techniques to improve the accuracy. Experimental results confirm the effectiveness of the proposed approaches.

Categories and Subject Descriptors

H3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Text Retrieval, Sampling, Calibration, Query Evaluation

1. INTRODUCTION

The problem we address in this paper stems from the Blog archiving system we are building at York University. The system collects and indexes blog postings related to movies as they become available, with individual indexes built for various fields, such as date of creation, header, body, and links. Consider the following the query: how many Blog entries in the collection (a) have a body highly similar (with similarity score greater than 0.9) to a given Blog entry, and (b) were created between 01/01/2007 and 02/01/2007? There are two ways to process this query, depending on whether condition (a) or (b) is evaluated first. Clearly, the most efficient way is to evaluate the more selective condition of the two, and then the other. To help planning the query evaluation, it is very important to obtain a quick estimate of the number of Blog entries satisfying each condition. This becomes more important for complex queries involving multiple data sources or predicates.

The need to obtain a "similarity profile" of the document collection for a given query also arises from a number of other applications. For example, in peer-to-peer text retrieval, an essential task is to guide the users to the sources that contain the most relevant documents in a fast and efficient way. A naive way to identify those sources is to ask each peer evaluates the query and reports the size of the query result, which are then compared to pick the best sources. The problem with this approach is that it can be very inefficient, as the document collections at some peers can be huge, and evaluating the query at all peers will therefore be quite expensive. In such cases, it will be very helpful if we could get a quick estimate of the "similarity profile" of the document collection at each peer without fully evaluating the query against the collections. For instance, the following information can be very important: how many documents are there in the collection whose similarities to the query (assuming a given similarity metric) exceed a given threshold? This information will allow us to quickly zoom in onto the most promising peers for answering the query, by choosing only the peers with the largest number of relevant documents.

The core problem we address in this paper is to how to efficiently and accurately estimate the sizes of similarity queries, i.e., the number of documents whose similarities to the query (under some metric) exceed a user-defined threshold. We start with the vanilla sampling-based approach, in which the estimate is obtained based on a sample of the documents. Our main contribution, however, is the calibration approach which can greatly improve the sampling-based estimation accuracy.

Our work is related to the well-studied problem of database selection ([3, 2, 1]). However, unlike algorithms such as *gGloss* [3] and *CORI* [2] which are specifically designed for database ranking, our approaches also take into consideration the need arising from query planning (as illustrated in the second example above), and we aim to provide a set of "general purpose" algorithms that can provide more detailed similarity profiles of collections.

2. ESTIMATION TECHNIQUES

Let us denote the set of documents to be queried upon by P, and the number of documents in P by N. For a given query q, denote the set of terms contained in q by $G_q = \{g_1, g_2, \ldots, g_m\}$. Let s be the number of documents whose similarities with q are greater than a given threshold θ, i.e., $s = |\{d \in P : sim(d, q) > \theta\}|$. Here we assume that the classic vector space model is used, in which every document d_i is represented by a weight vector w_i of index *terms*. The weight of the j-th term in document d_i, denoted by w_{ij}, can be calculated using the well-known *tf-idf* (term frequency-inverse document frequency) measure. The same weighting scheme applies to the query q. In this paper, we assume that the cosine similarity is used, where the similarity between the document d_i and the query q is defined as $sim(d_i, q) = w_i \cdot w_q/(\|w_i\| \cdot \|w_q\|)$. Nonetheless, our results can generalize to any similarity measures.

2.1 The sampling-based approach

One way to estimate s without examining each document is to employ sampling. A naive approach is to obtain a uniform random sample S from the whole document collection P. Note that, however, only those documents that share common terms with the given query can possibly have a similarity score (w.r.t. the query) higher than the threshold θ (assuming $\theta > 0$). It is therefore beneficial to narrow down the sample space (and thereby improve the sampling efficiency) by restricting our efforts to those "relevant" documents

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

only. Let's denote this set of relevant documents by P_R. Assuming that an inverted list L_i (of document IDs) is maintained for each term i, P_R can be constructed by computing the union all relevant L_i's. For example, suppose the query consists of terms "www", "Beijing", and "2008". The inverted lists corresponding to those three terms are identified, and the union of all document IDs contained in those lists are then computed. This will be the P_R for sampling.

Suppose a sample S of size n is taken from P_R in order to estimate the result size of q. S can be any type of sample, such as a simple random sample (SRS), or a weighted sample. Let us denote the weight of the sampled document d_i by v_i ($d_i \in S$), which is the inverse of the inclusion probability of document d_i in the sample. Intuitively, v_i indicates how many documents that this sampled document d_i can represent. For example, in the case of SRS, the inclusion probability of any document in the sample is n/N, so correspondingly the weight of any document in the sample is N/n. We use x_i as an indicator variable to indicate whether d_i satisfies the query q, i.e., $x_i = 1$ if d_i satisfies q, and 0 otherwise. An estimator of the result size s can be $\hat{s} = \sum_{d_i \in S} x_i v_i$. In the case of SRS, where the inclusion probabilities are all equal to n/N, the estimator simplifies to $\hat{s} = \frac{N}{n} \sum_{d_i \in S} x_i$.

2.2 Calibration

It is a common practice for text retrieval systems to maintain inverted lists for the index terms. For the purpose of result size estimation, such inverted lists represent valuable auxiliary information that we can utilize to further improve the accuracy of the sampling-based approach. We propose to adjust the weights of individual documents in a sample, such that the sample conforms better to the known auxiliary information.

Since we maintain an inverted list for each term in the document collection, we can safely assume that we also know the number of distinct document IDs in each inverted list (the length of the inverted list). Let the number of distinct IDs in the inverted list for term $g_j (1 \leq j \leq m)$ be t_j. Ideally, the number of documents in the sample that contain this term should be exactly proportional to the sample size. However, due to sample variations, this is rarely the case. Therefore, we would like to adjust the weights of the sampled documents, such that $t_j = \sum_{d_i \in S} y_i w_i$, where w_i is the new weight of document d_i, and y_i is an indicator variable to indicate whether g_j appears in d_i ($y_i = 1$) or not ($y_i = 0$). As a result, the set of known t_j's ($1 \leq j \leq m$) essentially present a set of constraints that we would like the new weights w_i's to satisfy. Once the new weights w_i's are obtained, we can simply plug them into the estimator described in Section 2.1 to replace the v_i's, and compute the new estimate.

When adjusting the weights of sampled documents, we also would like to keep the new weights w_i as close to the original weights v_i as possible. This is because we hope to keep certain properties of the original sample design. For example, if the sample was taken as a SRS, the estimate obtained through the estimator in Section 2.1 is unbiased. By requiring the new weights to be as close to the original weights as possible, we expect the estimate obtained using the new weights remain nearly unbiased. As such, calibration can be cast as a constrained optimization problem as follows: Minimize $\sum_{d_i \in S} v_i D(w_i/v_i)$, subject to $\sum_{d_i \in S} w_i y_j = t_j \quad (1 \leq j \leq m)$, where $D(w_i/v_i)$ is some distance function that measures the discrepancy between the new weights w_i and the original weights v_i.

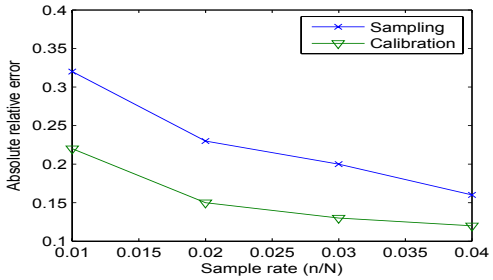
Various distance measures can be used, as long as they satisfy the following properties: (a) D is positive and strictly convex, (b) $D(1) = D'(1) = 0$, and (c) $D''(1) = 1$.

This optimization problem can be solved using numerical methods (e.g., Newton's method), and for certain distance measures (e.g., $D(x) = \frac{1}{2}(x-1)^2$), only one iteration is required, making the calibration process very efficient.

It is worth noting that when many inverted lists are involved in the estimation, the presence of too many constraints (posed by those inverted lists involved) can render the optimization process unstable when the sample size is small. Therefore, we cap the number of constraints in the equation to a certain limit. In our implementation, the limit is set at 10% of the sample size. The constraints used for optimization are randomly chosen from the set of all available constraints.

3. EXPERIMENTS

We evaluated our techniques on a movie data set as well as synthetic data sets. The movie data set consists of selected Blog entries collected for movies released in the United States during the period from May 1, 2006 to August 8, 2006. In total, the data set contains 45046 blog entries that comment on 30 different movies. In generating the synthetic data sets, we varied a number of factors such as the number of documents, the sample rate, and the distributions of the documents, to study the behavior of the techniques w.r.t. to those factors. Due to space limitation, only selected results on the movie data set are shown here. Errors of estimation are measured by the Absolute Relative Error (ARE), defined as $\frac{|\hat{s}-s|}{s}$. All results are averages of 30 runs. SRS is used in the sampling-based approach. For the calibration approach, the distance function use is $D(x) = \frac{1}{2}(x-1)^2$. The query set is generated by randomly selecting documents from the date set, and the similarity threshold is set to 0.1. The Figure below shows the accuracy of two approaches described in Section 2.



For both approaches, the accuracy improves as the sample rate increases. It is also evident that the calibration approach improves significantly upon the pure sampling-based approach.

4. CONCLUSIONS

We have presented two techniques to reason about the result sizes of similarity queries for text retrieval. Using the techniques, a text retrieval application can quickly determine the most promising data sources, without running the query against the document collections. The techniques can also help in query planning, and they can be easily extended to handle the estimation of other parameters of the query results. Experiments have confirmed the effectiveness of the proposed techniques.

5. REFERENCES

[1] A. Broder, M. Fontura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu. Estimating corpus size via queries. In *CIKM '06*, pages 594–603, 2006.

[2] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR '95*, pages 21–28, 1995.

[3] L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In *VLDB '95*, pages 78–89, 1995.