Notebook Paper

UIC at TREC-2002: Web Track (Draft)

Shuang Liu, Clement Yu, *Wensheng Wu

Database and Information System Lab
Computer Science Department, University of Illinois at Chicago
{sliu, yu}@cs.uic.edu

*Computer Science Department
University of Illinois at Urban and Champion
wwu2@uiuc.edu

Abstract

This is the first year that members of the Database and Information System Lab (DBIS) at University of Illinois at Chicago (UIC) participate in TREC. We participate in two tasks for the Web track: topic distillation and named page finding. Linkage information among documents as well as content information about documents is used in some of our submitted runs.

We utilize the Okapi weighting scheme with some modification for documents and passages retrieval; the proximity of query terms in documents is also utilized for document ranking.

The PageRank of a document is combined with the similarity of the document with the query to obtain an overall ranking of documents. A local linkage and URL analysis algorithm is employed for topic distillation. In the named page finding task, we combine the surrogate similarity with the document similarity in one run.

0. Introduction

In TREC-2002 experiments, we carry out the topic distillation and the named page finding tasks. A modified Okapi weighting scheme is implemented in our search engine. The modification is to replace the original parameter *K* which is the length of a passage by the parameter *K'* which is the norm of the passage. We also propose a proximity re-ranking method. In this method, documents covering more query terms in a certain window size than documents having fewer query terms within the same window size are ranked higher. The PageRank of a document is combined with the similarity of the document before or after proximity re-ranking to obtain an overall ranking of the document. In topic distillation, a document is assigned a value which is the sum of its similarity and a weighted sum of the similarities of its descendents within the same host. For each host, the document that gets the

highest value among documents belonging to the same host is chosen for final ranking. In named page finding, the title and the anchor text are used to construct a surrogate index. We submit one run that combines the document similarity with the surrogate similarity.

In section 1 of this paper, we present the indexer, the weighting schemer, the proximity re-ranked feature and the linkage analysis. In the next part, we discuss the specific technique we used for the distillation task and the named page finding task. The paper is concluded in section 3.

1. System Description

1.1 System and Data Structure

In this section, an overview of our system is given.

System

(1) Document Parser

It is responsible for robust parsing of HTML/SGML documents. This includes HTML/SGML parsing, tokenization, lower case conversion and stemming.

(2) Indexer

The indexer reads collection files from disk, decompresses the documents, and parses them. For each document, the system assigns a document ID, stores the content words (word ID) and their positions within the document. For each appearance of the content word, its type (title, anchor or plain) is identified. The information is held in a forward index file, in descending order of document ID. The lexicon will be kept in a separate file. The lexicon file contains all the words and their corresponding IDs. The indexer also extracts linkage information between documents and maintains it by using two tables: one keeps track of, for each document, the set of documents it points to; the other keeps track of, for each document, the set of documents pointing to it.

(3) Inverse Indexer

The inverse indexer reads the forward index file and sorts it by word ID to generate an inverted index. An index of the inverted index is produced which contains, for each word, the word ID and its offset into the inverted index.

A document is partitioned into passages, each having at most 300 words including stop words.

Data Structure

(1) Forward Index

Documents in the forward index are in ascending order of document ID. One complete record is as follows:

Document ID, t

Word Id_1, tf_1, (position, type[, docID]), ….., (position, type[, docID])

Word Id_2, tf_2, (position, type[, docID]), ….., (position, type[, docID])

…

Word Id_t, tf_t, (position, type[, docID]), ….., (position, type[, docID])

where t is the number of distinct terms in the document; tf_i is the term frequency within the document; for $docID_{i1}$ in (position, type[, $docID_{i1}$]), if the word type is anchor, the word is logically associated with the document with $docID_{i1}$; otherwise the $docID_{i1}$ is omitted.

(2) Inverted Index

Ordered by word ID, the word record is as follows:

Word ID, df, (docID, normWeight) ….. (docID, normWeight)

where df is the document frequency of the word; docID and normWeight are the document and the normalized term weight of the word in the document respectively.

There are exactly two forward index files and two inverted index files, one for documents and another for passages.

(3) Outgoing Link File and Back Link File

Ordered by document ID, the outgoing link record is as follows:

Document ID, t, $docID_1$, $docID_2$, …, docIDt

where t is the number of outgoing links; documents with IDs: $docID_1$, $docID_2$, … $docID_t$ are pointed to by document ID.

The back link file has the same format, but the docIDs are associated with the back links of the given document.

(4) Document Index

Document index table, implemented as a fixed length ISAM file and ordered by document ID, contains, for each document, document ID, document name, its URL, its length, its norm, its

PageRank (see section 1.4) and entries to various files: forward index file, outgoing links file, and back links file.

Basic Search Engine

In the current system, once the search engine is fired up, several data structures are populated and resident in main memory. This includes (a) a hash table holding lexicon; (b) a set holding a list of stop words; and (c) a search map holding the index of the inverted index. The search map is implemented as sorted pairs of (word ID, offset), where the offset indicates the starting position where the inverted index for the word with word ID is located on the disk.

A memory resident lexicon and search map enables fast lookup of inverted file and is critical to the efficiency of our search engine.

All the processing is done on a Dell computer with 2.0G Pentium processor and 1GB RAM. This machine runs Linux.

1.2 Weighting Scheme

We adopt the Okapi weighting function in our system [RobertsonWalker00].

Basic Weight Function

$$\sum_{T \in Q} w^{(1)} \times \frac{(k_1+1)\times tf}{K+tf} \times \frac{(k_3+1)\times qtf}{k_3+qtf} \qquad (1)$$

$w^{(1)}$ is the Robertson/Spark Jones weight of T in Q [RobertsonSpark76], which is:

$$\log \frac{N-n+0.5}{n+0.5} \qquad (2)$$

where,

N is the number of items (documents/passages) in the collection

n is the number of documents/passages containing the term

tf is the frequency of occurrence of the term within a specific document/passage

qtf is the frequency of the term within the query

$$K = k_1 \times ((1-b) + b \times \frac{dl}{avgdl}), \qquad k_1 = 1.2 \;\; b = 0.75 \;\; k_3 = 1000$$

Documents retrieval uses the basic okapi function.

Modified Okapi Function for Passages Retrieval

Passages retrieval uses a modified okapi function where each passage has up to 300 words including stop words (the last passage of a document may have less than 300 words).

We modify Okapi function by replacing the original parameter K with K', where K' is:

$$K' = k_1 \times ((1-b) + b \times \frac{Norm}{AvgNorm})$$

Here the Norm is the norm of the passage, the AvgNorm is the average norm of a passage in the collection. k_1, b, k_3 are 1.2, 0.75 and 1000 respectively. We use the Norm and AvgNorm instead of the traditional length factor for the reason that passages differ more in norms than in lengths.

1.3 Proximity Feature

We take the proximity of query terms into consideration in retrieval. Specifically, if the largest number of query terms which can be found in a document within a certain window size (50 words) is m, then m is called the proximity factor. Supposed that the similarity of the document computed by the basic/modified Okapi function is sim, then an intermediate similarity of the document is (m, sim). Documents are arranged in descending order of (proximity factor, similarity) with proximity factor being the leading coefficient. That is, if document d_1 has intermediate similarity (p_1, sim_1), and document d_2 has intermediate similarity (p_2, sim_2), then document d1 will be ranked higher than document d_2 if $p_1 > p_2$, or $p_1 = p_2$ and $sim_1 > sim_2$.

After TREC competition, we do more experiments on proximity. We are investigating the effect of window size on retrieval efficiencies.

1.4 PageRank

We compute the PageRank of each document according to [BrinPage98] [PageBrinMotwani98] [Havelinwala99].

1.5 Retrieval

Documents/Passages Retrieval

Documents are retrieved in descending order of basic okapi similarity. Let the similarity between document d_i and query q be normalized to value between 0 and 1 and be denoted by sim(di, q).

The basic search engine also retrieves the passages in descending order of the modified okapi similarity. The similarity of a document containing a set of passages in the list is the maximum

similarity of a passage within the document. Let $sim(p_k, q)$ represent the similarity value between passage p_k and query q, $p_{i1}, p_{i2}, \ldots, p_{it}$ be passages belonging to the same document d_i. The similarity of document d_i between query q is normalized between 0 and 1 and is given by:

$$sim(d_i, q) = \max(sim(p_{i1}, q), sim(p_{i2}, q), \ldots, sim(p_{it}, q)) \quad (3)$$

Ranking Documents by Proximity

After the similarities of documents are computed according to formula (3), they are re-ranked in descending order of (proximity factor, similarity) as described in section 1.3. The (proximity factor, similarity) is normalized to a value between 0 and 1, and this value is denoted by inter-sim(d, q).

Combine PageRank

We combine the PageRank value of each document with sim(d, q) or inter-sim(d, q) to aim at the final similarity [SilvaRibeiro-Neto00].

Combine with PageRank:

$$\text{final-sim}(d, q) = a * sim(d, q) + (1-a) * \text{PageRank} \quad 0 < a < 1 \quad \text{or}$$

$$\text{final-sim}(d, q) = a * \text{inter-sim}(d, q) + (1-a) * \text{PageRank} \quad 0 < a < 1 \quad (4)$$

In the above expression, PageRank is normalized to a value between 0 and 1.

The BaseSet of documents are the top N documents obtained by formula (3), or re-ranking using proximity or formula (4).

2. Web TREC

2.1 Topic Distillation

"Topic distillation involves finding a list of key resources for a particular topic. A key resource is a page which, if someone built me a (short) list of key URLs in a topic area, I would like to see included." [TREC-2002Guidelines]

In TREC2002, we try to find the key resources by applying the following local link and URL analysis to the BaseSet.

Distillation Algorithm

In the BaseSet, the documents within each host are used to compute their qualities. This takes into considerations of:

a. The final similarity of each document within the host

b. The linkage information among the documents in the host.

(1) We first construct a graph as follows. A BaseSet of 1000 documents which have the largest similarities was obtained. This set is augmented by another set of documents S'. Each document, say d' in S' is a parent of a node d in S, and d' and d belong to the same host. Hyperlinks between documents on the same host form the directed edges of the graph.

(2) We now compute the quality of a document. It is essentially the sum of its final similarity and a weighted sum of the final similarities of its descendents within the same host. It is computed as follows.

Let N be the set of nodes in the parenthood graph; Let q[n] be the quality value of node n in N, and s[n] be its final similarity value.

a. Initialize q[n] to 0; s[n] = 0 if n does not belong to S, s[n] = its similarity value if n is in S.

b. For each node n in set N, compute its shortest path to each of its descendents, m. Let the length of the shortest path be d[n, m]. d[n, m] = 0 , if m = n.

c. $q[n] = \sum_{\substack{m \in N \\ descendant_of_n}} \frac{s(m)}{2^{d(n,m)}}$

(3) Rank the documents in descending order of quality. Return the 10 documents with the largest qualities, provided that no more than 1 document comes from the same host. In other words, if document d_1 and document d_2 come from the same host and d_1 has higher quality, then discard d_2.

Result

For topic distillation task, 4 BaseSets are constructed:

The first BaseSet contains documents retrieved using passage retrieval only.

The second BaseSet contains documents obtained using passage retrieval but with proximity take into consideration (section 1.3).

The third BaseSet contains documents obtained using passage retrieval, proximity and combined with PageRank.

The fourth BaseSet contains documents retrieved using document retrieval combined with PageRank.

Notice that this year's TREC topic distillation task only judges the top 10 retrieved documents. So in our submission, for each query, top 10 documents are from distillation, and the remaining documents are from BaseSet.

The best result is obtained by applying distillation algorithm to the third BaseSet, we get 52 key resources.

2.2 Named Page Finding

Surrogate Index and Retrieval

Since title and anchor text are very important in homepage finding, we treat them in a special way and generate separate index set of title and anchor text for afterward retrieving. The title and anchor text will be indexed as follows:

(1) For each HTML page p, extract its title T.

(2) For each HTML page p, construct a set S of anchor texts associated with the link in the Web pages that have link(s) pointing to p. For example, if page q has a link pointing to p, e.g. on page q there is <a> href=http://www.dbis.uic.edu>UIC database and information xxx Web site and page p is dbis's web site. Then "UIC ..." is the anchor texts for page p. Note that it is extracted from page q.

(3) For each HTML page, construct a document surrogate from all the tokens in T and S.

(4) Index surrogates as we described in section 1.1.

For a given query each retrieved document will have a surrogate similarity. Then it is combined with the original document similarity to get the final similarity. The procedure is as follows:

(1) Compute the similarity of query q with surrogate: sim(q, surrogate-of-doc)

(2) Interpolate its similarity with the original document p, to get final similarity:

$$c * sim(p, q) + (1-c)* sim(surrogate\text{-}of\text{-}p, q) \quad (5)$$

Result

We submit 3 runs for named page finding task.

The first run is obtained from the surrogate as described in formula (5).

The second run is obtained by combining document retrieval with PageRank.

The third rum is obtained using passage retrieval only.

The best is the third run, the average reciprocal over 150 topics is 0.564, the number of topics for which the named page found in top 10 is 114 (76%), the number of topics for which no named page was found is 20 (13.3%).

3. Conclusion

Our TREC-2002 experiments show that passage retrieval is beneficial to both topic distillation and named page finding. Proximity ranking can improve the precision of content retrieval. Linkage information (PageRank) helps both tasks. But unfortunately, the surrogate (title and anchor text) does not give us a good performance in named page finding task. Since this is the first time we participate in TREC, we do not have the time to try more sophisticated techniques. We are experimenting with new techniques to perform retrieval, which hopefully will yield much better effectiveness in the future.

Reference

[BrinPage98] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998

[ChakrabartiDom99] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, Mining the link structure of the World Wide web, *IEEE Computer*, 1999.

[Havelinwala99] Haveliwala, T. Efficient computation of pagerank. *Technical Report*, Stanford University, Stanford, CA., 1999

[PageBrinMotwani98] L. Page, S. Brin, R. Motwani, and Terry Winograd. The pagerank citation ranking: Bring order to the web. *Technical Report*, Stanford University, 1998

[RobertsonSparck76] S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129--146, 1976.

[RobertsonWalker00] Robertson S E, Walker S. Okapi/Keenbow at TREC-8, *The Eight Text REtrieval Conference*, NIST SP 500-264, pp151-161, 2000

[SilvaRibeiro-Neto00] IR Silva, B Ribeiro-Neto, P Calado, N Ziviani, and ES Moura. Link-based and Content-based Evidential Information in a Belief Network Model. *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96-103, 2000.

[SinghalKaszkiel01] Amit Singhal, Marcin Kaszkiel. A Case Study in Web Search Using TREC Algorithms. *Proceedings of the 10th International World Wide Web Conference*, pages 708-716, HongKong, May 2001

[TREC-2002Guidelines] TREC-2002 Web Track Guidelines, http://www.ted.cmis.csiro.au/TRECWeb/guidelines_2002, 2002