STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 1, 2010

CLUSTERING, TIERED INDEXES AND TERM PROXIMITY WEIGHTING IN TEXT-BASED RETRIEVAL

IOAN BADARINZA AND ADRIAN STERCA

Abstract. In this paper we present a textual retrieval system based on clustering and tiered indexes. Our system can be used for exact phrase matching and also for improved keyword search by employing term proximity weighting in the similarity measure. The document retrieval process is constructed in an efficient way, so that not all the documents in the database need to be compared against the searched query.

1. Introduction

Textual-based web search accounts for a large part of the traffic in the Internet nowadays. The majority of the Internet traffic no longer flows through core routers, but through edge distribution networks like the one of the Google search engine. Although there are several forms of information retrieval (i.e. textual information retrieval, sound-based information retrieval, video information retrieval etc.) the one that evolved the most is text-based information retrieval and this is reflected in the commercially web search engines available today. In this paper we present an information retrieval systems which offers besides keyword search functionality also exact phrase matching. Our system uses a similarity measure which favors documents that contain large portions of consecutive terms from the query, so it can be used in detecting plagiarism in a scientific paper. If the candidate document does not contain groups of consecutive terms from the query, the similarity measure defaults to a classical cosine similarity and the search is a keyword search (not an exact phrase search). The index structure is based on clustering the saved documents and regular term-frequency/inverse-document-frequency indexes.

The rest of the paper is structured as follows. In section 2, the general structure of an IR system is outlined and work related to ours is mentioned.

2010 *Mathematics Subject Classification.* 68U35, 68M11.

1998 *CR Categories and Descriptors.* H.3.3 [**Information Systems**]: Information Storage and Retrieval – *Information Search and Retrieval*; H.3.1 [**Information Systems**]: Information Storage and Retrieval – *Content Analysis and Indexing*.

Key words and phrases. textual information retrieval, tiered indexes, document clustering, term proximity.

Then, section 3 presents the first part of our IR system, the inverted index, followed by section 4 which presents the second part, the retrieval algorithm of our system. In section 5 we present the results of some preliminary evaluations of our system and the paper ends with conclusions in section 6.

2. Information retrieval fundamentals and related work

Every information retrieval system is build from two main parts: the index structure and the retrieval/ranking algorithm. There are several space models for an IR system [1]: the boolean model, the probabilistic model [2], [3], vector space model, linguistic model.

Most IR systems extract keywords from the documents after an initial prefiltering phase (which includes stop words elimination, stemming and lemmantization) and build an inverted index. Most IR systems assign to each keyword t from document d a weight like the following [4]:

$$tf\_idf_{t,d} = tf_{t,d} \times idf_t$$

where $tf_{t,d}$ is the number of occurrences of term t in document d (i.e. term frequency) and idf_t is the inverse term frequency (i.e. number of occurrences of term t in all indexed documents). There are several variants for the term weight formula, but most of them use in some form the term frequency and the inverse document frequency.

The ranked retrieval algorithm compares the query given by the user to all or most documents in the collection and based on some similarity measure between the query and a document it returns the top k relevant documents. A very used similarity measure is the cosine similarity used in the vector space model. For exact phrase matching, a positional index (i.e. an index holding positions in a document for each term) must be used and the similarity measure should include some form of term proximity scoring [5], [6], [7].

3. The index structure of the system

Our system represents documents in the vector space model where each document is viewed as a vector with different document terms and for each term, the system maintains a regular term-frequency/inverse-document-frequency weight [1]. More specifically, for each term t, the index structure holds a postings list with an entry for each document from the collection in which t occurs. Each entry stores the document ID, the term weight in that document and a list of positions were the term appears in that document. The term weight for term t in document d is:

$$wf\_idf_{t,d} = (1 + log(tf_{t,d})) \times idf_t$$

All documents indexed by the system are grouped in clusters/groups based on the similarity between their representative vectors and each group has a leader which is chosen in a random way. When a query is submitted to the IR system, it first checks with the groups' leaders and then it continues the search within the group with the leader most similar to our query. In this way all other non similar documents are excluded from the search and only the most relevant ones are considered which decreases the runtime execution of the query. This cluster pruning heuristic is very useful when new documents are added to the collection. In an IR system, this is done by a crawler. The algorithm used for building the index structure of our system is described in the following lines:

The crawler indexing algorithm:
Input: R = {url_1, url_2, ..., url_n} // crawler's repository
L = {l_1, l_2, ..., l_m} // the existing leaders of the indexed clusters where
// l_i is the representation of a document in the
// vector space model

For all $r \in R$ do
d ← getHtml(r);
d ← filter(d);
init v; // v is the representation of document d in vector space model
for all $t \in d$ do //for each term t from document d
$v[t] = wf\_idf_{t,d}$;
end for;
init Sim;
for all $l \in L$ do
Sim(l) ← Similarity(v,l); // computing similarity between
// document d and cluster leader l
end for;
l' ← MAX(Sim);
insert d in CLUSTER(l');
end for;

The cluster based crawler first takes from the repository an url and gets the html source code of that page. The next step is text formatting, deleting the html tags, stop words elimination (e.g. and, or etc.), deleting javascript and css code etc. The following step is index creation based on the term weight $wf\_idf_{t,d}$ for all the terms that appear in the document. In order to find the most appropriate cluster to which this document should be added, the most

similar cluster leader from the collection is found and the new document is added to the cluster of this leader. When measuring the similarity between the vectors v and l, the representations of the document d and a cluster leader, the classical cosine similarity metric is used:

$$Similarity(v, l) = \frac{v \circ l}{\|v\| \cdot \|l\|}$$

where the numerator is the dot product of vectors v and l and $\|.\|$ symbolizes the Euclidean norm.

4. The retrieval algorithm of the system

Our information retrieval system uses a combination of clustering and tiered indexes for document searches. When using tiered indexes we set a similarity threshold at a higher value when we search for a document at the first tier and decrease that value at tier two and so on until we find the desired number of documents. This means that the user can search for a whole document and the system will return the most similar indexed documents.

Because of our similarity metric, the retrieval algorithm of our system is a combination between exact phrase retrieval and keyword based retrieval. This means that although the search is essentially a keyword search based on cosine similarity between vectors containing term weights, the vector representation of the query and the vector representation of a candidate document, documents that contain large groups of consecutive terms from the query are favored when returning the results (thus, considered more relevant than documents that do not contain groups of consecutive terms from the query).

The retrieval algorithm returns the top k documents most similar to our query (off course, the query is represented in vector space, in order to be compared to other documents) from the document collection. The algorithm is the following:

The document retrieval algorithm:
Input: L = {l_1, l_2, ..., l_m} // the leaders of indexed clusters where l_i is the
// vector representation in the vector space model
q // q is the vector representation of the query

Score ← [];
init minimum_threshold;
init similarity_threshold;
index ← 0;
while (index ≤ k) or (similarity_threshold > minimum_threshold) do
L' ← first_three_similar_leaders(q, L, similarity_threshold);

```
            L ← L - L';
            for all l ∈ L' do
                  for all d' ∈ CLUSTER(l) do
                        if (similar(q,d') ≥ similarity_threshold)
                              Score[d'] ← similar(q, d');
                              index ← index +1;
                        end if;
                  end for;
            end for;
            similarity_threshold ← similarity_threshold - 1;
end while;
for all d' in Score[] do
            Score[d'] ← Score[d'] + title_metadata_url_score(d');
end for;
Sort(Score);
return Score
```

The first step is the search of the most similar leaders from the clusters, which add some speed to the algorithm because the document is compared only to the leaders and not to all documents from the collection. After getting the first three most similar clusters, the document is searched in these leaders' clusters. The function that is used for similarity computation between the vector representations of 2 documents, d_1 and d_2, is a modified cosine similarity function that takes into account matching groups of consecutive words:

$$similar(d_1, d_2) = \frac{d_1 \circ d_2}{\|d_1\| \cdot \|d_2\|} + \left(1 - \frac{1}{N_{d_1,d_2}}\right) \tag{1}$$

where the denominator represents the dot product between vectors d_1 and d_2, $\|.\|$ represents the Euclidean norm of a vector and N_{d_1,d_2} is the length (in terms) of the largest group of consecutive terms that occurs both in d_1 and d_2. Two documents that have a large group of consecutive terms occurring in both will have a value close to 1 for the second part of the $similar(d_1, d_2)$ formula. If documents d_1 and d_2 have no terms in common or if they have terms in common, but no groups of consecutive terms in common, N_{d_1,d_2} is set to 1 and the formula $similar(d_1, d_2)$ defaults to a classical cosine similarity metric. So the two documents, d_1 and d_2, are more similar when the value of the $similar()$ function is higher and less similar when its value is lower. Please note that the formula (1) is not a metric in the mathematical sense since it does not satisfy the triangle inequality property, but it is a semimetric. The values of the semimetric (1) will be between 0.0 and 2.0. The reason that the

semimetric contains the N_{d_1,d_2} term is to implement a flexible form of exact phrase matching.

After the first leaders most similar to the query were found, the next step takes place which contains the actual extraction of the k-th most relevant documents that have a similarity value at least as higher as the threshold. The relevant documents that would be returned to the user are searched in the clusters of the selected leaders. The extraction of the first k documents is based on tiered indexes and the following heuristics were used:

- In the first tier, the document will be searched in the first 3 most similar leaders' clusters and the extracted documents must be at least 50% similar with the searched document;
- If the number of returned documents after the first tier is lower than k, than the search goes to tier 2 where the similarity threshold is set to 40%;
- If after tier 2 the number of returned documents is lower than k than it goes to tier 3 where the similarity threshold is set to 20%;
- If the tier 3 search is done and there still aren't k returned documents, than the found documents are returned.

The last step in the algorithm is the rank and score computation for the extracted documents. For score computation, the following factors are taken in consideration: similarity percentage calculated with the formula (1), the words from documents title, key words from meta tags and the words from the url as follows: the score increases with 1 if words from the query are found in the meta data, with 2 if words from the query are found in the document's title and with 2 if words from the query are found in the document's url. Considering the score computation, we can say that this algorithm has support for web pages that were optimized for searched engines.

5. Evaluation

In order to evaluate our text retrieval system we performed initial tests on a rather small document collection consisting of 100 documents, most of them crawled and indexed from the wikipedia.org website. The tests showed that our systems retrieves relevant documents to a large degree of the returned results. We detail in the following lines the results of two tests. In the first test we used a long query of about 70 terms and in the second query we used a smaller query of about 20 terms. Let this query be referred to by Q in both tests. In order to test the efficiency of our modified cosine similarity measure, 7 documents from our 100 documents collection were artificially created:

- Document D_1 contains just the query, Q
- Document D_2 contains the query Q, repeated 3 times

- Document D_3 contains query Q, then some random text, then another occurrence of Q, then other random text
- Document D_4 contains half of Q, followed by some random text, then the other half of Q, followed by another random text
- Document D_5 contains a large portion of random text followed by Q and followed by another random text
- Document D_6 containing some text which resembles Q, but is not the text from Q
- Document D_7 which contains the first third of Q followed by some random text, then the second third of Q, then followed by another random text, then the final third of Q and some random text

For both tests, we set the parameter k of the retrieval algorithm to 10. When the longer query was given to the system, the system retrieved the following documents in the specified order and with the specified similarity score:

Document	Similarity score
D_1	1.99
D_2	1.98
D_3	1.63
D_4	1.44
D_5	1.39
D_7	1.32
Da (irrelevant)	0.0304
Db (irrelevant)	0.0293
Dc (irrelevant)	0.0265
Dd (irrelevant)	0.0248
Precision = 6/10 = 0.6 Recall = 6/6 = 1.0	

The retrieved documents for the short query of about 20 terms are:

Document	Similarity score
D_1	1.89
D_2	1.88
D_3	1.53
D_5	1.46
D_4	1.35
D_7	1.23
D_6 (irrelevant)	0.26
Da (irrelevant)	0.035
Db (irrelevant)	0.028
Dc (irrelevant)	0.026
Precision = 6/10 = 0.6 Recall = 6/6 = 1.0	

We can see from both tables that the relevant documents were returned, the documents containing large portion of Q have higher similarity score and there is a significant distance between the similarity score for relevant documents and the similarity score for irrelevant documents.

6. Conclusions and future work

In this paper we have presented an information retrieval system based on clusters and tiered indexes that combines exact phrase search with (non-phrase) keyword based search. The system should scale well with a large document collection because it uses clustering in the retrieval process. Initial tests on a rather small sized document collection show that the precision and recall measures of our system have reasonable good values. Of course, in order to assess the full efficiency of such a retrieval system, we need to test it on large collections of documents like the Ad hoc track from the TREC collections [8].

7. Acknowledgments

This work was partially supported by the CNCSIS-UEFISCSU unit of the Romanian Government, through project PN II-RU 444/2010.

References

[1] Manning C.D., Raghavan P., Schutze H, *An introduction to Information Retrieval*, Cambridge University Press, 2009.

[2] Crestani F., Lalmas M., Van Rijsbergen C. J., Campbell I., *Is this document relevant? ... probably: A survey of probabilistic models in information retrieval*, in ACM Computing Surveys, vol 30, no.4, pp.528552, 1998.

[3] Fuhr N., *Probabilistic models in information retrieval*, in The Computer Journal, vol. 35, no.3, pp. 243255, 1992.

[4] Papineni K., *Why inverse document frequency?*, In Proc. North American Chapter of the Association for Computational Linguistics, pp. 18, 2001.
[5] Sadakane K., Imai H., *Text retrieval by using k-word proximity search*, in International Symposium on Database Applications in Non-Traditional Environments, pp.183-188, 1999.
[6] Buttcher S., Clarke C. L. A., Lushman B., *Term proximity scoring for ad-hoc retrieval on very large text collections*, in Proceedings of the 29th annual international ACM SIGIR conference on Research and development in IR, pp. 621622, 2006.
[7] Rasolofo Y., Savoy J., *Term proximity scoring for keyword-based retrieval systems*, in Proceedings of the 25th European Conference on IR Research, pp. 207218, 2003.
[8] *The Text Retrieval Conference*, http://trec.nist.gov .

Babes-Bolyai University, Computer Science Department
E-mail address: ionutb@cs.ubbcluj.ro

Babes-Bolyai University, Computer Science Department
E-mail address: forest@cs.ubbcluj.ro