Query modeling for classifieds retrieval

Varvara Tzika

August, 2012

Abstract

To do...

1 Motivation- Introduction

2 Related Work

3 Method

In this thesis we want to improve retrievals system effectiveness. Creating a retrieval system evolves the creation of the following modules:

. Document collection preprocessing and creation

. Indexing

. Query modeling

. Retrieval model

3.1 Preprocessing

Preprocessing is a good approach to improve the effectiveness of system like ours. The document collection consists of documents created by regular users and contains a lot of noise which should be extracted before indexing takes place.

Preprocessing can be done using several methods:

. Parsing

. Stemming

. Stop words

. Storing the information on file with special structure

In the following parts we will extend these methods.

3.1.1 Parsing

Parsing is the process of analyzing a piece of text, made of a sequence of tokens(for example, words), to identify tokens in a stream of text [2]. For a string This is a book we can agree that the tokens are this, is,a,book. In some languages like Chinese this isnt trivial since words are not separated by spaces, but in our case this doesn't constitute a problem.

Difficulties arise in an attempt to parse sentences like This is a book for F1-score.One simple and comfortable way to parse this sentence is to remove punctuations and this will solve the problem. But should all hyphenated terms be treated as a single term? For example, suppose we parse the text and then he took the oh-so-long-awaited-jump. If we remove punctuations and treat it as a single word, the result would be ohsolongawaitedjump. Another option is to split the word based on the punctuation. This will solve the problem with oh-so-long-awaited-jump. But in case the text F1-scores gets split into f1 and score, the meaning of the each word will be treated separately thus losing the meaning of combining both words.

Given the previous situations,we can conclude that parsing is of a great importance since it affects the query accuracy. Thus decisions have to be made carefully, rules have to be created and every term of every document has to obey them. Also,parsing time can have a dramatic impact on the entire indexing process. The time taken to index documents directly impacts the delay between when a document is created to when it can be retrieved. For a static collection, this is not a big issue, but for a dynamic collection lengthy indexing times are unacceptable.

3.1.2 Stop Words

In computing, stop words are words which are filtered out prior to, or after, processing of natural language data (text). There is not one definite list of stopwords which all tools use, if even used. Some tools specifically avoid removing them to support phrase search [4]. Some search engines use stop word lists for short function words such as a, the, on etc. which occurs very often. Since these terms occur very often in common texts, indexing and retrieving them is probably meaningless. The odds of a term the randomly occurring in a document and a query is much higher than the odds of a more infrequent term. These frequently used terms are merely noise and can be eliminated from an information retrieval system. Typically, an information retrieval system contains a list of stop words.After the parser is done with the manipulation of characters as described above,it will then remove stop words.

3.1.3 Stemming

Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base or root form,generally a written word form. In projects likes ours is the process of removing suffixes and prefixes in every term in the document collection and in every term of the query. For example, cars will be stemmed in car thus if we have a document related to car,we can retrieve documents related to cars also. However, stemming sometimes negatively affects a queries accuracy. For instance, if we search for organization and our stemmer removes the reasonably common suffix ization we end up with documents about organ.The other advantage to stemmers is efficiency. They dramatically reduce the number of distinct terms in the collection. This affects the size of the entire inverted index [5].

3.2 Indexing

Having discussed preprocessing of the collection of documents, the next step is to explain how and why we will use inverted index and what are the limitations.

The point of using an index is to increase the speed and efficiency of searches of the document collection. Without some sort of index, a query must sequentially scan the document collection, finding those documents containing the search terms. This costs a lot of time for some thousands of files. Given this situation, a data structure called an inverted index or indexing is commonly used by search engines. Its basic structure inverts the text so that instead of the view obtained from scanning documents where a document is found and then its terms are seen, an index is built that maps terms to documents. Instead of listing each document once (and each term repeated for each document that contains the term), an inverted index lists each term in the collection only once and then shows a list of all documents that contain the given term. Each document identifier is repeated for each term that is found in the document. An inverted index contains two parts: an index of terms, which stores a distinct list of terms found in the collection and, for each term, a posting list, a list of documents that contain the term. In particular, for each term t, the index table contains an inverted list It consisting of a number of index postings. Each posting consist of details for the term in this document as occurrence-frequency of this term in this specific document, document id, etc.

While constructing an index consumes a lot of time,retrieval time is significantly faster with the use of index. Furthermore,since index construction is an off-line process and it requires to be done only once, shorter query processing times at the expense of lengthier index construction times is an appropriate trade off.

Also, many techniques have been proposed to improve inverted index construction such as the compression of inverted index in [6]. Their goal was to decrease inverted index size and improve query through-compression. They follow document reordering approach. Actually they perform some form of text clustering on the collection to find similar documents, and then they assign document IDs based on the similarity of documents. However, in some cases,it is difficult or impossible to change the way document IDs are assigned. In our case document IDs are the unique identifiers of documents created by the system and used by many modules of the existing system in Marktplaats.

Finally, inverted index repositories can exceed the storage demands of the document collection itself. However, for many systems, the inverted index can be compressed to around ten percent of the original document collection. Given the alternative (of twenty minute searches), search engine developers are happy to trade index construction time and storage for query efficiency [3].

3.3 Query models

Query modeling is the procedure to create a bag of words or a word to represent the information need. This bags of words or word is the query. Query modeling creation involves the preprocessing step and the identification of tokens. Even if all previous steps are chosen and implemented to improve the accuracy of relevant results, the right query modeling will find highly related documents. As it is described in the related work, query modeling has been researched multiple times in the past and is of great importance in various projects.

To create a good query model,the query has to contain the most important information of the classified. But how can we find words that contribute the most important information from a classified? In the case of classifieds we have several parts that important information can be found. In figure 1 the important information is in title while in figure 2 is in description. For this reason multiple query models will be created and they will be experimented to find which of them is more accurate. In section 4.4 list of multiple queries is provided.

Listing 1: TREC formated classified

```
<doc>
  <docno> 20000000 </docno>
  <title> HM top zwart 36 5 euro </title>
  <description>
    alles 2 euro excl!!
    behalve merken 5 euro extra
    zie ook mijn andere advertenties
  </description>
  <category> dames </category>
  <price>  5 </price>
  <attributes> </attributes>
</doc>
```

Listing 2: To do

3.4 Retrieval Models

A retrieval model takes a query and a document as input and identifies a measure of relevance between the query and the document. Different retrieval models have different retrieval strategies thus resulted documents differs as well.

Retrieval model or ranking function used by search engines mostly. A search engine except of finding the relevant document, has to rank and order them by relevance. This is typically done by assigning a numerical score to each document based on a ranking function, which incorporates features of the document, the query, and the overall document collection.

The study of retrieval models is central to information retrieval. Many different retrieval models have been proposed and tested, including vector space models,probabilistic models and logic-based model.

The ranking functions-retrieval models that we will use for this project are the following:

. Tf.Idf

. Okapi BM25

. Probabilistic LM

Also, we will use Pseudo relevance feedback to improve our results.

3.4.1 Tf.Idf

Tf.Idf (term frequencyinverse document frequency) is a kind of common methods used to select the text feature. This retrieval method ranks documents based on the characteristic terms. Characteristic terms for a document are those who only frequently appears in the possible relevant document while infrequently in the rest documents of data collection [7].

TF is words frequency and Idf is inverse document frequency. Term frequency in the given document is the number of times a given term appears in that document. The inverse document frequency is a measure of whether the term is common or rare across all documents.

Tfidf is calculated as:

$$TfIdf = tf * idf \tag{1}$$

$$Idf = log\frac{d}{dt} \tag{2}$$

Where
d:total number of documents in the collection
dt:total number of documents where term t occurs

However if the term t does not occur in the document collection idf, then dt will be equal to zero. Therefore the formula is adjusted to 1+dt

Tf.Idf advantage is that it tends to filter out common terms. When a document has high term frequency while the term appears rarely in the whole collection of documents then it has high weight in Tf.Idf scoring.

3.4.2 Okapi BM25

In information retrieval, Okapi Best match 25 (BM25) is a ranking function used by search engines to rank matching documents according to their relevance to a given search query [9] . Okapi ranking function is based on the probabilistic retrieval framework. It makes an estimation of the probability of finding if a document dj is relevant to a query q. Three factors affects Okapi s score. First is the query terms frequency,second is the inverted frequency of query terms and finally, the length of the document. With this way, it scores higher a short document that mention all query terms.

Given a query , containing keywords , the BM25 score of a document is:

$$BM25(dj, qi : N) = \frac{Idf(qi) * Tf(qi, dj) * (k+1)}{(Tf(qi, dj) + k * (1 - b + (b * |dj|/L))} \tag{3}$$

Where N : total number of documents

Tf(qi,dj) : the frequency of qi word in dj document

Idf(qi) : is the inverse document frequency of word given by:

$$Idf(qi) = log\frac{N - DF(qi) + 0.5}{DF(qi) + 0.5} \tag{4}$$

dj : is the length of document dj in words L : is the average document length in the corpus

3.4.3 Language Model

Language Modeling is the task of estimating the probability distribution of linguistic units such as words, sentences, queries, utterances, or even complete documents. The probability distribution itself is referred to as a language model [10].

Given the query q and the user U , we want to find the most probable documents. That is, we want to rank the documents by p(d—q, U).

Using Bayes theorem,

$$p(d|q, U) = \frac{p(d|U)p(q|d, U)}{p(q|U)} \tag{5}$$

For the purpose of ranking, we can ignore the denominator and define the relevance of a document as:

$$pq(d) = p(d|U) * p(q|d, U) \tag{6}$$

The query likelihood p(q—d) is calculated by assuming that the query terms are independent, and then multiplying the probabilities for the individual terms. If the query q = (q1 q2 . . . qm) , then:

$$p(q|d) = \prod_{i->1}^{m} qi$$

Furthermore,suppose that we have the query This is a great book for retrieval and evaluation in IR created by the description of a book and also we have as candidate document with description This is a book for evaluation in IR . The candidate document does not contain the query word retrieval. Now, if we estimate p(retrieval—d) ,then this probability will be zero and the query likelihood will vanish. Thus, the language model for a document has to distribute some probability mass among words that are not in the document too. This task is called smoothing [11].

In our case Dirichlet smoothing is used to solve the zero probability and data sparseness problems.

$$p(q|d) = \frac{Tf + m * p(q|C)}{|D| + m} \tag{7}$$

3.4.4 Pseudo Relevance Feedback

The general idea behind relevance feedback is to take the results that are initially returned from a given query and to use information about whether or not those results are relevant to perform a new query [12]. Pseudo relevance feedback provides an automated way to have feedback for a query. With the creation of a new query based on this feedback the retrieval performance improves.

The method we must follow in order to use relevance pseudo feedback is the same method used in normal retrieval. The system will use the results from the original query and extend it with the feedback. Then the system assumes that the top "k" ranked documents are relevant, and finally to do relevance feedback as before under this assumption with an expanded query.

Although, pseudo relevance feedback improves the efficiency of the system, it is dependable to the original query since it assumes that the top k results are relevant. However,through a query expansion, some relevant documents missed in the initial round can then be retrieved to improve the overall performance. Clearly, the effect of this method strongly relies on the quality of selected expansion terms.

3.5 Log Likelihood Ratio

One of query models is Log Likelihood Ratio(LLR).

'A likelihood ratio test is a statistical test used to compare the fit of two models, one of which (the null model) is a special case of the other (the alternative model). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one model than the other.'

As it is described in [14] we can use LLR to compare corpus and find the terms of a corpus that are more characteristic. There are two main types of corpus comparison:

. Comparison of a sample corpus to a larger corpus (normative)

. Comparison of two (roughly-) equal sized corpora

These two main types of comparison can be extended to the comparison of more than two corpora. For example, we may compare one normative corpus to several smaller corpora at the same time, or compare three or more equal sized corpora to each other. In general, however, this makes the results more difficult to interpret.

This first type of comparison is intended to discover features in the sample corpus with significantly different usage (i.e. frequency) to that found in general language. While second type aims to discover features in the corpora that distinguish one from another. In our case, first type is more appropriate since we need to find a way to distinguish model for a classified against a large corpora that will give us enough feedback for every word in the classified. We refer to the larger corpus as a normative corpus since it provides a text norm (or standard) against which we can compare. We will use as first corpora(sample or null corpora) the visited classified and as second(normative corpora or alternative), a big data set that will remain the same for all visited classifieds.

The representativeness of the big corpora needs to be considered when comparing two corpora. It should contain samples of all major text types and if possible in some way proportional to their usage in the natural writing of a classified in case we want features(in our case frequencies) to make sense . In the case of classifieds created by users, we need a corpora with a data set of classifieds really created by users and big to contain almost all different words a user will write in his classified [14].

3.6 Late Data Fusion

Based on [15], data fusion, is the process of integration of multiple data and knowledge representing the same real-world object into a consistent, accurate, and useful representation.

In our case there are two approaches for the combination of data known as early fusion or late fusion. Early fusion is the combination of data prior to indexing. Thus the data aggregated and then retrieval model use these aggregated data as input. While late fusion assumes each source of data has associated with it some form of a ranking function, each of which can be independently queried. Once each source has been queried, the outputs of each of these queries can be aggregated together to form a final response to the initial query [16].

4 Evaluation

5 Results

6 Discussion and Conclusions

7 Future Work

8 Appendix

9 References

References

[1] Valentin Jijkoun , Gilad Mishne , Maarten de Rijke, *Preprocessing Documents to Answer Dutch Questions*, In proceedings of the 15th belgian-dutch conference on artificial intelligence BNAIC03, pp.487-497

[2] Parsing

[3] Ir book

[4] Stopwords

[5] Stemming

[6] Hao Yang,Shuaib Ding, *Inverted Index Compression and Query Processing with Optimized Document Ordering*, In proceedings of the 18 International conference of World wide web, WWW '09, pp. 401-410

[7] Shouning Qu,Sujuan Wang,Yann Zou, *Improvement of Text Feature Selection Method based on TFIDF*, In proceedings of the 2008 International Seminar on Future, Information Technology and Management Engineering, FITME '08, pp. 79-81

[8] Fang Hui,Tao Tao,Zhai Chengxiang, *Diagnostic Evaluation of Information Retrieval Models*, ACM Transactions on Information Systems (TOIS) - Special issue on research and development in information retrieval TOIS Homepage archive, July 1991, pp 187 - 222

[9] Okapi

[10] W. Bruce Croft,John Lafferty, *Language Modeling for Information Retrieval*, Springer Publishing Company,2010

[11] Chengxiang Zhai,John Lafferty, *A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval*, Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '01, pp 334 - 342

[12] Relevance Feedback

[13] Amit Singhal, *Modern Information Retrieval: a brief overview by Amit Singhal*, Bulletin of the IEEE computer society technical committee on data engineering

[14] Paul Rayson,Roger Garside, *Comparing Corpora using Frequency Profiling*, In proceedings of the workshop on Comparing Corpora, held in conjunction ACL 2000, October 2000, Hong Kong, pp 1-6

[15] Wilkins Peter, *An Investigation Into Weighted Data Fusion for Content-Based Multimedia Information Retrieval* PhD thesis, Dublin City University, Nov 2009

[16] Data fusion

[17] Christian Middleton , Ricardo Baeza-yates, *A Comparison of Open Source Search Engines*, SIGIR 2007

[18] lemur

[19] Trevor Strohman *Dynamic collections in Indri*

[20] Stemmer

[21] LLR

[22] *Comparing Corpora using Frequency Profiling*