Building a Search Engine Model with Morphological Normalization Support

Jure Mijić[1], Bojana Dalbelo Bašić[1], Jan Šnajder[1]

[1]*Faculty of Electrical Engineering and Computing, University of Zagreb*
Unska 3, 10000 Zagreb, Croatia
{*jure.mijic, bojana.dalbelo, jan.snajder*}*@fer.hr*

Abstract. *Searching a collection of documents can seem like an easy task, but manipulating textual data can be difficult because the data are mostly unstructured. We undertook the task of building an effective search engine for a collection of Croatian legislative documents. The developed search engine model supports multiple modules for information retrieval. To improve the effectiveness of the retrieval, we used a morphological normalization module that uses an inflectional lexicon automatically acquired from a document corpus. As we do not have a gold standard for our legislative document collection, we evaluated our search engine on three English test collections, explored the effects of stemming, and compared the results to the vector space model.*

Keywords. Information need, Information retrieval, Morphological normalization, Search engine.

1. Introduction

The need for correct information is becoming a part of everyday life, and, with the abundance of information available, especially on the Internet, satisfying that information need has become a challenging task.

The benefits of using machine processing for purposes of information retrieval are self-evident. The recent increase in computer power enables us to use more complex algorithms and to process data faster than ever before. The problem is, however, that algorithms used in computer processing are often low level in terms of understanding the data they are processing; i.e., the meanings of the words and the relations among words are unknown. It is up to the programmer to develop better algorithms that can represent the data in a way that enables manipulation of those data on a higher level. The form of the user query can greatly influence the results of the retrieval. The form of the user query can greatly influence the results of the retrieval. The same query can be formulated in many different ways, so it is logical to expect that different users will formulate different queries for the same information need.

In Section 2, we will give a brief view of the process of building an information retrieval system and describe some systems that have already been developed. The implementation of our search engine is described in Section 3, and the results of the evaluation are presented in Section 4. Conclusions are given in Section 5.

2. Information retrieval systems

To overcome the technical difficulties of efficient computer resource usage, the major factors in an effective information retrieval (IR) system are the algorithms for information processing. In general, information can be structured or unstructured. Here, we will limit the field of information retrieval to the textual data type, as we are building a search engine for text documents. Most of the documents are represented in some sort of marked-up format and are regarded as "semi-structured". These documents are usually divided into sections, paragraphs, and the signature, and the title of the document is almost always marked. Documents can also be interlinked, as in the case of web pages. This document link structure can also be exploited in the ranking of the retrieval results. Our search engine is designed for use in a limited document collection, so we will not focus our research on web retrieval.

Documents are represented by document features, and the most logical choice for features are the words contained in the documents. Using the words as document features results in a high dimensionality of the document collection, as there are many words and word forms, especially for morphologically rich languages such as Croatian. The high dimensionality can be reduced by word normalization, i.e., lemmatization or stemming. Along with the words themselves, some other word features can be used for document representation, i.e., the part-of-speech tag of the word, capitalization of the word, and position of the word in the sentence. Sequences of characters or character n-grams can also be used as document features. The dimensionality when using character n-grams depends on the length of the n-grams, as there can be more character combinations in longer character sequences. The logic behind character n-grams is that they encompass the roots of words and outnumber the odd character combinations that appear on word boundaries, such as commas, punctuations, and other special characters. This approach has its advantages; for instance, the frequencies of n-grams that encompass typographic errors in the document are low compared to the n-grams from correctly spelled words, making the model more resilient to typographic errors. Another advantage is that the model can eliminate language-dependent features such as grammar for stemming, stopwords, or even matters as simple as where to break individual words. The disadvantage of this model is the higher memory needs because of the higher number of n-grams. The use of character n-grams has been explored for the classification of textual documents [2] and their use as document features showed similar classification performance to word features, but at the cost of larger memory consumption.

The SMART system [7] was developed by Gerard Salton and his students at Cornell University. The system uses a vector space model for representing documents, and it performs automatic indexing by removing stopwords, stemming, and term weighting. Queries are also converted into vectors and compared with each document. The similarity measure of query vector and document vector is used for ranking. The system returns the top n documents, where n is a number defined by the user.

The Indri retrieval system [8] was developed at the University of Massachusetts and is based on inference networks. It combines the advantages of the inference net framework with the language modeling approach to retrieval. The Indri search engine is designed to support complex queries and retrieval at various levels of granularity (e.g., sentence, passage, XML field, document, and multi-document). The system is also designed to support very large databases, optimized query execution, and fast and concurrent indexing and querying. The effectiveness of the system was proven on the TREC Terabyte Tracks [5, 9]. An Indri index is actually composed of a set of smaller self-contained indexes. The system is therefore able to evaluate a query against many indexes simultaneously, and the indexes do not need to reside on the same machine.

3. Our search engine model

Our goal is to develop an intelligent search engine for a limited collection of legislative documents. As we are planning to experiment with various information extraction methods, we decided to develop our own search engine model instead of using existing software.

The document collection can be considered small by today's standards. The collection currently consists of 10000 documents and will eventually grow to about 15000 documents. The collection contains legislative documents of the Republic of Croatia written in Croatian. All the documents have a similar structure, which consists of a title, introduction, body, and signature. Furthermore, the body is divided into articles and each article into paragraphs. This document structure is useful, as it can be exploited in the retrieval procedures. So far we have considered the title of the document by assigning higher weights to the terms of the document title. Legal documents also reference other documents from the collection, and that document link structure can also be used to enhance the retrieval results. The link structure of legal documents resembles a hierarchy, where documents of law amendments reference the document of that law. This structure is different from the link structure of web documents, where there are many more random document connections. Web page links are based on the author's opinion on the value of that link to the given document. This is what the PageRank algorithm exploits, but that algorithm would not be very useful in our case because the documents at the top of the link hierarchy would always get higher scores and the documents at the bottom would not benefit at all because they have no or few documents linked to them.

For the purpose of text processing, we use the Text mining tools (TMT) library [1]. The most basic text processing operation is the tokenization procedure, which is implemented for use with the UTF-8 character set, which we use for internal text representation. Input documents are in XML format, and some characters can be encoded using XML entities, so we added an additional procedure for converting the character entities to their respective UTF-8 codes. We also use the procedures of morphological normalization, as the Croatian language is morphologically complex. The normalization procedures are implemented using an inflectional lexicon acquired using an innovative method [6]. The lexicon is automatically acquired from a given document corpus using the morphology description of the Croatian language, i.e., a set of inflectional rules. Unlike stemming, lexicon-based normalization allows for precise normalization of the inflectionally complex Croatian language. The acquired lexicon is of large coverage (over 99%) and allows for good normalisation performance ((the understemming index is less than 7%, and the overstemming index is less than 3%); cf. [6] for a detailed evaluation. The use of morphological normalization for the Croatian language, although based on a different approach, has already been applied in one Croatian web search engine [3], and the improvements of using normalization are significant [4].

Generally, our search engine model can support multiple search engine implementations, each with its own parameters. At the core of every implementation is an index database, containing all words found in the document collection, along with their respective positions in the documents. The main part of the index is a tree of a specified depth, where each level of the tree branches according to one character. As characters are represented in the UTF-8 code page, the index can support words from multiple languages that may contain special or language specific characters. Additional data can also be added to the implementation, i.e., a document-word matrix for the vector space model. For the time being, we have implemented a basic full-text search module. A document collection index database is built using an index builder tool, which processes the documents from the collection and stores the words and their respective positions in the documents. By using the morphological normalization module, the index builder tool can add the normalized forms of words instead of the original word forms. Additionally, by specifying a stopwords file, we can omit those words from the index database. A summary of each document is also stored in the index database, but in a compressed form. The index database is saved to a file in binary format and later loaded with the search engine module. Serialization and deserialization procedures used are also implemented in the TMT library.

The input for our basic full-text search is a simple search string containing the keywords we want to search with. The search procedure finds the locations of all the keywords from the query, using the index database. The documents are then ranked using a heuristic algorithm that we implemented. The algorithm gives higher scores to phrases containing many different keywords from the query; i.e., longer phrases get higher scores. The score for the particular phrase will be higher if the phrase contains different keywords and if the keywords are closer together; i.e., some other words can be between the keywords. The order of the keywords in the phrase is ignored. At the end, scores of all the keywords and phrases for each document are summed, and a score is assigned for each document. This algorithm is actually very intuitive, as the user usually searches for a particular phrase or phrases. Documents with longer phrases will have higher rankings. The document score will also be higher if it contains more keywords and phrases, but a limit is necessary to avoid higher ranking for very long documents. Exact phrase matching is also supported; phrases in the query must be marked with quotation marks. In that case, the order of the keywords in a phrase must match the order of the keywords in the query. The search procedure returns documents ranked by the relevance score. As some documents can be very large, we could also try to find the minimal part of the document that satisfies the information need, for instance, an article or perhaps even a paragraph, but we will leave that feature to future implementations.

4. Experimental evaluation and discussion

Due to the time constraints for this paper, we did not develop a gold standard for our legislative document collection, and we also did not find any other Croatian test collection that could be used to evaluate the effects of morphological normalization in the retrieval process. For the evaluation of the ranking procedure, we used three English test collections and explored the effects of stemming and the removal of stopwords.

The test collections used were Medline, CACM, and CISI developed for the SMART system. Table 1 shows the information for these three collections, including the number of documents and the number of queries for the collection. Table 2 shows more information about the distribution of relevant documents for the queries, such as the average number of relevant documents for all queries. We also added the median, minimum, and maximum numbers of relevant documents. For the CACM and CISI collections, some of the queries have a small number of relevant documents, and there are a few queries that have a very high number of relevant documents. The Medline collection, however, has a distribution that is more dense and uniform.

We compared the performance of our search engine (SE) against a vector space model that uses the bag-of-words (BOW) representation. Evaluation was performed with three standard measurements: recall, precision, and F1 measure. We also compared the influence of the removal of stopwords and the use of stemming, i.e., Porters stemming algorithm for the English language. In the vector space model, all the stopwords were removed, but stemming was not used.

A first look at the results reveals low levels of recall and precision for all tests on both the CACM and CISI collections. This could be explained by the fact that half of the documents in the CACM collection have only the title of the document, which is, in most cases, very short. In the CISI collection, the queries are mostly in the form of a question, and some queries are quite long and complex. The queries for the Medline collection, however, are relatively short and full of relevant keywords, and the documents are larger than the documents in the CACM and CISI collections.

From Fig. 1 we can see that the recall for the vector space model is higher than the recall

Table 1: Number of documents and queries for document collections

collection	documents	queries
Medline	1033	30
CACM	3204	52
CISI	1460	76

Table 2: Number of relevant documents per query

collection	average	median	min	max
Medline	23	23	9	39
CACM	15	12	1	51
CISI	41	32	1	155

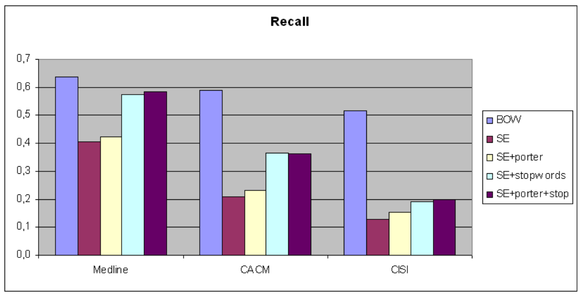

Figure 1: Recall

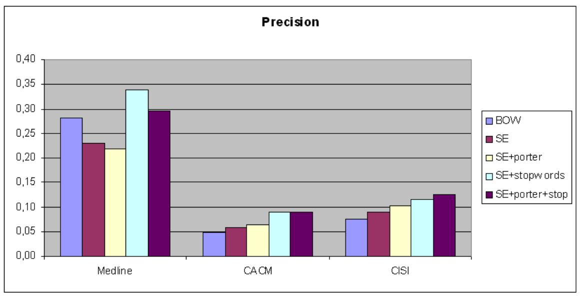

Figure 2: Precision

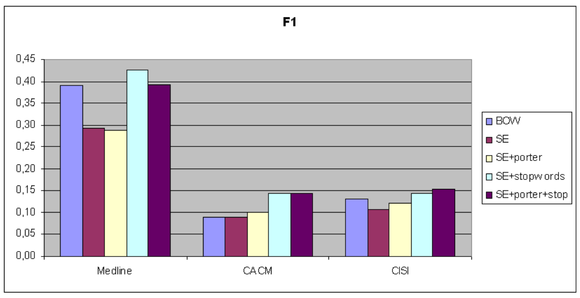

Figure 3: F1 measure

of our search engine, i.e., 0.63 for the vector space model versus 0.58 as the best result of our search engine. The reason for this is that the vector space model ranks the document based on the similarity of the query vector and the document vector in the vector space. The document vectors in the vector space can be similar to the query vector not only by the keywords from the query but also by other features. This means that the documents retrieved in the search results do not have to contain the keywords from the query but that they could contain other words that have similar meanings to some of the keywords.

The downside of the vector space model is the lower precision because the keywords from the query can have different semantic meanings. The precision for the vector space model is 0.28, while the best case precision of our search engine is 0.34. Fig. 2 shows that the drop in precision is more pronounced for the vector space model, so the F1 measure is lower than the best result of our search engine.

The removal of stopwords and the use of stemming can give different results for different collections. We compared their use with our search engine. In Fig. 1 and Fig. 2, we can see that the removal of stopwords increased both recall and precision on all collections; i.e., on the Medline collection recall increased from 0.4 to 0.57 and precision increased from 0.23 to 0.34. This was expected because stopwords do not carry any semantical meaning. For the use of stemming, the results on the Medline collection indicate that it tends to increase recall at the cost of precision; i.e., recall increased from 0.57 to 0.59, and precision decreased from 0.34 to 0.29. On other collections, the difference is not significant. This is explained by the fact that some words of different meanings are reduced to the same stem; thus, more documents will be returned. The larger number of documents returned by the search engine contributes to the increase of recall, but some documents can be returned only because they contain some words that have the same stem as the words from the query even though they have totally different meanings and contexts. The decrease of the precision for the Medline collection was significantly higher than the increase of recall, which is visible by the F1 measure in Fig. 3. The F1 measure shows a good balance between precision and recall, and the results show better performance with the removal of stopwords. Slightly better performance was achieved with the use of stemming, but only on the Medline collection. We expect that the use of morphological normalization would yield significantly better results for a Croatian document collection.

5. Conclusion

We have developed a search engine model and implemented a simple full-text search algorithm. The process of indexing uses the procedures of stopword removal and morphological normalization for the Croatian language. As we did not have a Croatian test collection, the evaluation was done on three English test collections, and the results indicate better performance than the vector space model using the bag-of-words representation. The removal of stopwords proved to enhance the retrieval results, while the use of stemming did not yield significantly better results. The use of morphological normalization for the Croatian language could prove to be more useful, as the Croatian language is more morphologically complex than the English language.

Further development will be in the areas of information extraction, and methods such as named entity recognition and coreference resolution will be researched and evaluated. Taking advantage of the document structure and with the use query expansion, we hope to further refine the search procedure.

Acknowledgments

This work has been jointly supported by the Ministry of Science, Education and Sports, Republic of Croatia and the Government of Flanders under the grant No. 036-1300646-1986 and KRO/009/06 (CADIAL).

References

[1] Artur Šilić, Frane Šarić, Bojana Dalbelo Bašić, and Jan Šnajder. TMT: Object-oriented text classification library. In *ITI 2007 Proceedings of the 29th International Conference on INFORMATION TECHNOLOGY INTERFACES*, pages 559–566, 2007.

[2] Artur Šilić, Jean-Hugues Chauchat, Bojana Dalbelo Bašić, and Annie Morin. N-grams and morphological normalization in text classification: A comparison on a croatian-english parallel corpus. In *Lecture Notes in Artificial Intelligence vol. 4874, Progress in Artificial Intelligence: 13th International Conference EPIA 2007; Proceedings*, pages 671–682, 2007.

[3] Damir Krstinić and Ivan Slapničar. Web indexing and search with local language support. In *Proceedings of SoftCOM 2003*, pages 488–492, 2003.

[4] Damir Krstinić and Ivan Slapničar. Improving text search performance with grammar support. In *Workshop on Information and Communication Technologies*, pages 71–75, 2004.

[5] Donald Metzler, Trevor Strohman, and W. Bruce Croft. Indri at TREC 2006: Lessons learned from three terabyte tracks. 2006. In online Proceedings of Text REtrieval Conference.

[6] Jan Šnajder, Bojana Dalbelo Bašić, and Marko Tadić. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing & Management*, 2008. In press. DOI: 10.1016/j.ipm.2008.03.006.

[7] G. Salton. *The SMART Retrieval System–Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[8] Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. Indri: A language model-based search engine for complex queries, 2005. poster presentation.

[9] Xing Yi and James Allan. Indri at TREC 2007: Million query (1mq) track. NIST, 2007.