MAI4CAREU

Master programmes in Artificial Intelligence 4 Careers in Europe

University of Ruse

Information Retrieval

Yordan Kalmukov

May 2023

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Content-based document retrieval

Text pre-processing
Boolean Retrieval

I. Text pre-processing

Usually, before proceeding with an algebraic or probabilistic text analysis model, it is necessary to pre-process the text to remove certain words and symbols that can introduce a lot of noise into the analysis and greatly affect the reliability of the the results. The five basic text preprocessing operations are listed below. It is desirable that at least the first four are implemented.

1. ***Remove punctuation***. Generally, there are many punctuation symbols in documents - periods, commas, dashes, quotation marks, apostrophes, etc. They are important for the construction of sentences and their syntax, but from a semantic point of view they do not carry any additional information. They are not telling us what a document is about. Just because they do not give us any information, does not mean they should be removed. But they actually cause problems. Especially when they touch on words. In this way they interfere with word recognition. For example, a word immediately followed by a dash or period is recognized as different from the same word without such a symbol. If, on the other hand, these symbols are separated from the words by spaces, then they are recognized as unique words, which unnecessarily increases the dimensionality of the vectors. Therefore, it is best to remove them. This is most easily done with a regular expression (eg. the following perl-compatible regex /[\p{P}]/u) or with a “string replace“ function.
2. ***Convert all letters to lowercase***. Whether a word is written in lowercase, uppercase, or a mixture of lowercase and uppercase does not change its meaning in any way. But in word recognition, letters matter, unless converted to all lowercase (preferably) or uppercase. It is desirable to do such a conversion for both all documents and the query. This will ensure that the same words will always be detected and recognized, no matter what letters they are written in.
3. ***Tokenization of the text***. I.e. converting the text into an array (or list) of unique words.
4. ***Removing semantically insignificant words*** (stop words). These are mainly prepositions, conjunctions, pronouns, adverbs, and etc. But not only. They also have an important syntactic meaning, but in no way represent the meaning and subject domain of the documents. They are also extremely common, which is a problem - they will have a disproportionately large *tf* value compared to semantically significant words that really carry information about the meaning and content of the documents. I.e. semantically insignificant words may overtake semantically significant ones. This can be somewhat solved by using *idf*, but only to a certain extent and not in the case of multiple documents in a single subject area. Therefore, it is best to simply remove these words. They are usually predefined as a

list, and are of course language dependent. There are ones for English, another for Bulgarian, another for Greek, and so on. The list is loaded as an array. Subsequently, when a document is processed, one simply checks to see if the current word appears in this array, and if so, ignores it.

5. ***Word stemming***. This is an operation of separating word endings from the root. The goal is to use only the roots in the analysis. For example, if different documents contain the words *beauty*, *beautiful*, *beautifully* and they are not stemmed, they will all be recognized as different words. However, it is clear that it is the same word, just in different forms. Therefore, it makes sense to recognize them as one word, not as different ones. For this purpose, the root of the word must be separated or the endings removed. This is, of course, an operation that is highly language-dependent, because the rules for forming endings in different natural languages are different. Various stemming algorithms exist. For the English language, the most used stemmer is the one proposed by Martin Porter [3]. Implementations of this algorithm can be found on the Internet in many different programming languages. Of course, there are other stemming algorithms for English. For example, the one proposed by Chris Paice called Lancaster Stemming Algorithm [2]. There is a stemming algorithm for Bulgarian as well - BulStem [1], proposed by Preslav Nakov.

II. Boolean retrieval

The most elementary method of searching by content is the so-called Boolean model (Boolean retrieval). It allows searching for words that are interconnected by AND and OR logical expressions. NOT inversion could be also used.

For example, if a user is searching for airplanes manufactured by Boeing, he/she should type the following query:

airplane AND *Boeing*

If searching for all documents that mention Boeing or any other aircraft, the query would be:

airplane OR *Boeing*

If searching for all airplanes, which <u>are not</u> Boeing or Airbus, they the query would be:

airplane AND (NOT(*Boeing* OR *Airbus*))

The Boolean searching model is largely used. Virtually all database management systems (DBMS), especially those managing relational databases (RDBMS), rely on it. Regardless of whether searching requires full match (ie *attribute="value"*) or partial match (searching for a substring within a string, ie *attribute LIKE "%value%"*), the DBMS implements exactly this search model.

The Boolean Retrieval's popularity is motivated by its following advantages:

- Easy to understand.
- Easy to implement.
- Computationally efficient.

However, database administrators probably also know how many limitations it has, which determine its disadvantages:

- Queries must be represented as Boolean expressions, which is not always easy. In fact, with a large number of words it is extremely difficult. Therefore, it is usually only used for simple queries with few words.
- In the case of many words, the search returns either a *huge amount of found documents*, if the words are connected with OR (low precision), or *just a few or even no documents*, if the words are connected with AND (low recall).
- Its behavior is truly Boolean - the document is either found or not. No similarity coefficient is calculated, therefore the results are not ranked, i.e. order by degree of similarity. The lack of ordering means that it is possible for some very similar/relevant documents to be pushed far back in the results list, and others that are less relevant to the query may be brought up at the beginning of the list.

As a conclusion, the Boolean retrieval is only suitable for searching by single words, not by entire sentences, paragraphs, and documents. The lack of calculated similarity to the query and ordering of the results does not guarantee that less relevant documents will not be incorrectly brought forward in the results, while relevant ones could be pushed at the end of the list.

References:

1. Nakov, Preslav. "BulStem: Design and evaluation of inflectional stemmer for Bulgarian." In Workshop on Balkan Language Resources and Tools (Balkan Conference in Informatics). 2003.
2. Paice, Chris D. "Another Stemmer." ACM SIGIR Forum 24.3 (1990): 56-61
3. Porter, Matrin F. An algorithm for suffix stripping. In J. S. Karen and P. Willet, editors, Readings in information retrieval, pages 313-316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997