Syntactic Indexes for Text Retrieval

Ioan Badarinza, Adrian Ioan Sterca, Maria Ionescu
Faculty of Mathematics and Computer Science
Babes-Bolyai University
Cluj- Napoca, Romania
ionutb@cs.ubbcluj.ro, forest@cs.ubbcluj.ro, maria.ionescu.or@gmail.com

***Abstract*—In this paper, we present three techniques for incorporating syntactic metadata in a textual retrieval system. The first technique involves just a syntactic analysis of the query and it generates a different weight for each term of the query, depending on its grammar category in the query phrase. These weights will be used for each term in the retrieval process. The second technique involves a storage optimization of the system's inverted index that is the inverse index will store only terms that are subjects or predicates in the document they appear in. Finally, the third technique builds a full syntactic index, meaning that for each term in the term collection, the inverse index stores besides the term-frequency and the inverse-document-frequency, also the grammar category of the term for each of its occurrences in a document.**

Keywords*— *textual search; syntactic metadata; query term; natural language processing

I. Introduction

Information retrieval (IR) is the process of finding material, usually with unstructured content, that has a relevant meaning for the information need, from within big collections of documents. There are many types of information retrieval, taking into consideration its broad meaning. We can classify them into three main categories [1], according to the scale at which they operate. In the first category, web searching, the system must be capable of satisfying the information need over a collection of billions of documents. The second category, personal information retrieval, refers to operating systems search, text classification done by email programs (for example a spam filter) or other types of IR in a collection of personal data. In the last category, enterprise, institutional and domain-specific search, the information need is related to the development of the business process. Each information retrieval system is composed of two parts: the indexer and the ranking algorithm. The indexer (or crawler) is responsible for scanning the collection in order to build an inverted index of terms. The most complex crawlers are used in web search where the content of the collection grows exponentially and also changes over time. When a query is performed, the inverted index will be used to get the set of documents which best satisfy the information need. In order to do this and also to sort (rank) the resulted documents, a ranking algorithm is applied.

In order to construct the index, the crawler starts with one or more initial URLs from a seed set and fetch the web page at that URL. The content is passed to a text indexer and the extracted links are added to an URL frontier which stores URLs of pages to be crawled. This process can be interpreted as traversing the web graph [2] - a graph where each page is a node and a hyperlink from one page to another is an edge from the corresponding node to the other.

For each web page in turn, the crawler obtains relevant text (eliminate HTML tags and scripts), performs tokenization to obtain the set of words/tokens, eliminates stop words (most frequent words which have very little relevance), normalizes the tokens and then applies stemming and lemmatization to obtain the root word.

The purpose of the ranking algorithm is to retrieve from the collection of documents the most relevant ones in response to a query. This is done by computing, for each matching document, a score which should reflect the relevance of that document with respect to the given query. Then, the documents are sorted descending according to the computed score, and only a subset of them is returned as a result set.

The rest of the paper is structured as follows. Section II outlines related work. The main contribution of the paper is described in Section III, where we first present the components of our information retrieval system, the crawler and the ranking algorithm, followed by the three techniques for building a syntactic index. Also in this section, an experimental evaluation of these techniques is performed. The paper ends with conclusions and future work ideas in section IV.

II. Syntactic Indexes in IR Related Work

Traditional IR systems do not use sophisticated Natural Language Processing (NLP) techniques, they only perform tokenization, stemming and lemmatization on the indexed text. They do not attempt, in the crawling phase, to find a syntactic structure in the natural language text, because this is time-consuming. They just extract terms from tokens, without any inter-relation between them. Nevertheless, IR systems could improve their search result by employing simple or full-fledged syntactic indexes. In this direction, the main idea is to capture term dependencies for representing the meaning of documents and to improve the efficiency of a searching system. The linguistic analyzing should be done in the indexing phase, by an automatic indexer which parses the content of documents and memorizes information about the term frequencies and also about the syntactic relations between terms, which should be provided by a syntactic parser. The linguistic inter-relationships that are established between terms can have different forms and definitions, depending on the parser, but the idea is to use these relationships in order to determine the relevance of terms inside a phrase and moreover, inside the document. Then the ranking algorithm should use the new information that the index contains and adopt new weighting schemes that take into account this linguistic information. The new methods of scoring can use syntactic information extracted from the syntactic index as well as syntactic information about the query terms if the query is represented as a phrase.

Reference [3] uses a linguistic rule-based dependency parser to add different annotations for the words in a document and index these words along with these annotations. These annotations will carry information about the linguistic features of the individual words from the document (e.g. noun, singular, verb etc.).

Another way of using NLP (Natural Language Processing) in IR systems is the one described in [4] where the authors try to use corpus statistics and linguistic heuristics in order to extract meaningful sub-compounds from complex noun phrases. Using these sub-compounds instead of the whole noun phrases as indexing terms will try to solve the phrase normalization problem encountered in phrase-based IR systems. They also represent the improved values obtained from the precision and recall measurements.

Since the use of single words as keywords in IR systems is not accurate enough to represent the documents, a lot of focus is shown in NLP for extracting different semantic information from documents. Reference [5] describes how to use complex terms, more precisely noun phrases, to represent a document and how these noun phrases are extracted using linguistic analysis and syntactic patterns. These noun phrases will allow integrating dependencies between words, dependencies that do not exist in the “bag of words” paradigm. This method is performed in two steps: the first one, similar to what is done in the first method, requires a linguistic analyzer with a tagger which generates a tagged collection (a tag corresponds to the syntactic category of a word: noun, verb, adjective) and the second step is to use the tagged collection to extract the so-called ‘noun phrases’, which are complex terms extracted from the phrase

Also, there are lots of Statistical Language Models (LM) that have been used in many natural language processing tasks including speech recognition and machine translation [6, 7]. Recently language models are explored as a framework for information retrieval [8, 9, 10]. The basic idea is to view each document as if it has its own language model.

III. Syntactic Indexes For Textual Retrieval

Our information retrieval system, which aims to construct and use a syntactic index, has two components: SynCrawler, a parallel syntactic crawler for the web, and SynSearch which implements the ranking algorithms and retrieval functions. Both components are implemented in Java. The crawler starts a new thread for each new web document that needs to be indexed, traversing the web graph in a breadth-first search. To obtain the syntactic metadata that is used in the creation of the index, it uses the Stanford Parser [11].

The Stanford Parser is a program that works out the grammatical structure of sentences, for instance, which groups of words go together (as "phrases") and which words are the subject or object of a verb. It can be used as a lexicalized probabilistic parser with separate probabilistic context-free grammar (PCFG) phrase structure and lexical dependency experts, or as an un-lexicalized stochastic context-free grammar parser. The parser provides Stanford Dependencies output as well as phrase structure trees. For example, for the phrase ‘My dog also likes eating sausage.’ it provides a typed dependency representation, which we will use in our crawler:

- *poss*(dog-2, My-1)
- *nsubj*(likes-4, dog-2)
- *advmod*(likes-4, also-3)
- *root*(ROOT-0, likes-4)
- *xcomp*(likes-4, eating-5)
- *dobj*(eating-5, sausage-6)

In the example above *nsubj* is a typed dependency between subject and predicate.

The crawling thread divides the raw text into phrases and then analyses each phrase with the Stanford Parser in order to obtain the dependencies list. Then it uses the information provided by the types of dependencies that exist between words to identify the subjects and predicates of each phrase. We will then use the syntactic information to enrich the information of a posting (corresponding to a term appearing in a document) from the postings list. The crawling algorithm is shown in Fig. 1.

The ranking algorithm, shown in Fig. 2, has the role in finding the top ten most relevant documents, having an array of query terms as input. The weighting scheme used starts from the following formula [12], which computes the rank of a document relative to a query:

$$Score(Q, d) = \sum_{q \in Q} weight(q, d) \tag{1}$$

where $Q = (q_1, q_2, \ldots, q_n)$ is the array of query terms, d is the document, for which we compute the rank, and $weight(q, d)$ is the weight of the query term q inside document d.

In the following 3 subsections we present three methods (weighting schemes) for incorporating lexical-syntactical analysis into an IR system for an increased search efficiency. All three methods are based on the following hypothesis: A document in which a term occurs as subject or predicate is more relevant than a document in which the term occurs as other syntactical categories.

A. *Method 1: Syntactic Analysis of the Query Phrases*

The main idea is to use the syntactic parser to analyze the query and use this syntactic metadata in weighting the query terms. But, to be able to do this, the user must provide the query in the form of a phrase. This cannot be applied when the user enters queries containing just keywords.

```
**Crawl** (initialURL, maxDepthLevel)
1.  if maxDepthLevel > 0
2.  then htmlText = fetchWebPage(initialURL);
3.  rawText = eliminateTagsAndScripts(htmlText);
4.  taggedTerms[ ] = useStanfordParser(rawText);
5.  doc = saveDocumentToDB(initialURL);
6.  for each term t in taggedTerms do
7.      rootTerm = stemm(t);
8.      saveTermToDB(rootTerm);
9.      savePostingToDB(doc, t);
10. URLs[ ] = extractLinksOnCurrentPage(htmlText);
11. for each url in URLs
```

Fig. 1. The crawler algorithm.

```
**RankingAlgorithm**(Q[ ])
1.  for each q in Q
2.  do s = stemm(q);
3.     look for s in the inverted index
4.  D[]=getAllDocumentsContainingAtLeastOneTerm(Q);
5.  for each d in D
6.  do rank = 0;
7.    for each q in Q
8.    do rank = rank + weight(q, d);
9.    assign rank to d
10. sortDesc(D);
11. return topK(D);
```

Fig. 2. The ranking algorithm.

Let us consider the following example: *"Where does silk come from?"*

We syntactically analyze the query phrase in order to determine the subject and the predicate of the query. According to the Stanford parser typed dependencies list, we have a subject – predicate relationship between the words 'silk' and 'come': *nsubj*(come-4, silk-3).

The idea is to consider the words which appear as subject or predicate in the query phrase more relevant to the information need and to increase the weights of these terms when computing the score of a document relative to the query using the *tf-idf* weighting scheme. In the above example, the words 'silk' and 'come' are obviously more relevant than the words 'does' or 'from'. The following heuristic rules are applied:

- artificially increase the weight *(idf)* of the subject term by 20% * *(maxIDF – minIDF)*, where *maxIDF* = the maximum *idf* of a term from the vocabulary, and minIDF = the minimum *idf* of a term.
- artificially increase the weight *(idf)* of the predicate term by 10% * *(maxIDF – minIDF).*

Method 1 is a proposal and its efficiency will be evaluated as part of future work.

B. Method 2: Reducing the Size of the Inverted Index

This method differs from the other ones in the way the inverted index is constructed. If we start with the assumption that the terms which appear as subjects or predicates inside phrases from documents are more relevant to the information need, then we can consider the option of indexing only those terms. In this way, if the terms which have a small grammatical relevance are not saved to the index, its size is significantly reduced, permitting us to index a bigger collection of web pages with the same storage capacity. This technique can be especially useful for small documents collections like blogs and forums which are very subject-specific and thus a text/blog/forum can be identified by a small number of keywords.

To evaluate the efficiency of this method, we measured the size of a normal inverted index and of an index containing only subjects and predicates (called hereafter syntactic reduced index) for the same collection of documents. We constructed both indexes from Wikipedia web pages, starting with the *URL: http://en.wikipedia.org/wiki/The_Hunger_Games*. Using different depth levels for crawling, we measured the following dimensions: the size of the relational database representing the index (in KB), the number of indexed documents or the number of terms in the vocabulary. In Figure 3, we illustrate the dimensions of the relational database (in KB) for both types of indexes which were constructed with depth levels 2, 4 and 5. Figure 4 shows a similar graph presenting the number of terms which are stored in both indexes. We can see in both figures, that indexing only terms with a more important syntactical function (e.g. subject and predicates). Also, the bigger the depth level, the bigger the difference between the sizes of a normal index and a syntactic reduced index.

C. Method 3: Full-Fledged Syntactic Index

The last approach uses the syntactical metadata memorized by the index in the ranking phase of the search process, i.e. building a full-fledged syntactic index. The data structure of the index as constructed by SynCrawler is:

- Document: *documentID, URL*
- Term: *termID, value, idf*
- Posting [for each (term, doc) pair]: *termID, docID, tf, subjectFrequency, predicateFrequency*

We used this syntactic information (subjectFrequency and predicateFrequency – representing the number of times a term appears as subject/predicate inside a document) to improve the traditional *tf-idf* weighting scheme as follows. The score of document *d* for a query *Q* is computed the classic way using the same (1) above. The weight of a document *d* with respect to a query term *q* is:

$$weight(q,d) = [tf(q,d)] * [idf(q) + w_s(q,d)] \tag{2}$$

where $w_s(q,d)$ is the syntactic weight and is defined by:

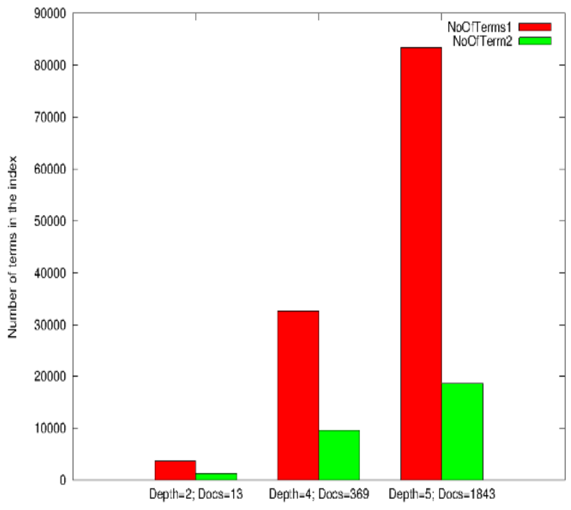


Fig. 3. Number of saved terms in a normal index (labeled 'NoOfTerms1') and in a syntactic reduced index (labeled 'NoOfTerms2').

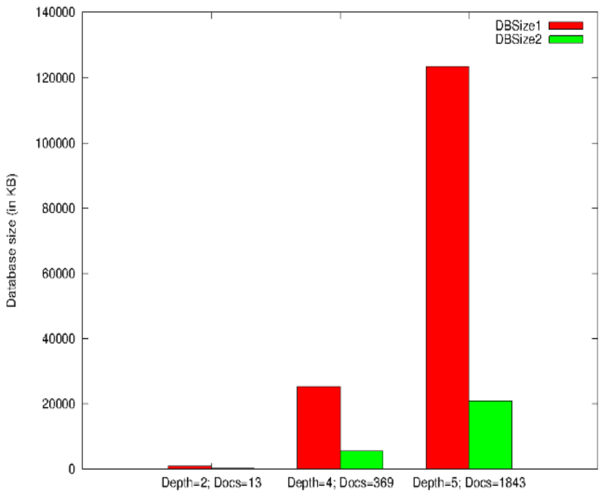
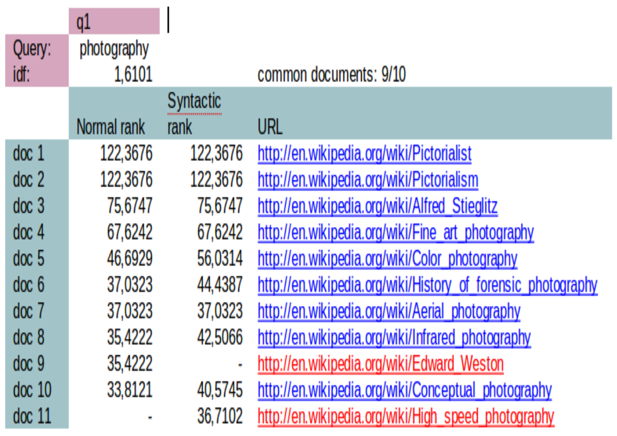


Fig. 4. Relational database size measured for a normal index (labeled 'DBSize1') and for a syntactic reduced index (labeled 'DBSize2').



	q1		
Query:	photography		
idf:	1,6101		common documents: 9/10
	Normal rank	Syntactic rank	URL
doc 1	122,3676	122,3676	http://en.wikipedia.org/wiki/Pictorialist
doc 2	122,3676	122,3676	http://en.wikipedia.org/wiki/Pictorialism
doc 3	75,6747	75,6747	http://en.wikipedia.org/wiki/Alfred_Stieglitz
doc 4	67,6242	67,6242	http://en.wikipedia.org/wiki/Fine_art_photography
doc 5	46,6929	56,0314	http://en.wikipedia.org/wiki/Color_photography
doc 6	37,0323	44,4387	http://en.wikipedia.org/wiki/History_of_forensic_photography
doc 7	37,0323	37,0323	http://en.wikipedia.org/wiki/Aerial_photography
doc 8	35,4222	42,5066	http://en.wikipedia.org/wiki/Infrared_photography
doc 9	35,4222	-	http://en.wikipedia.org/wiki/Edward_Weston
doc 10	33,8121	40,5745	http://en.wikipedia.org/wiki/Conceptual_photography
doc 11	-	36,7102	http://en.wikipedia.org/wiki/High_speed_photography

Fig. 5. Rank measurements for one term query: '*photography*'.

$$w_s(q,d) = \begin{cases} 20\% \times idf(q), \text{if at least 25\% of the occurrences of the term } q \text{ inside document } d \text{ are as subject or predicate;} \\ 0, \text{otherwise;} \end{cases} \quad (3)$$

To evaluate this method, we used a syntactic index constructed from Wikipedia web pages, starting with the same initial URL as used for method 2: *http://en.wikipedia.org/wiki/The_Hunger_Games* with a depth level of 5. We considered 3 types of queries: one-word queries, two-words queries, and three-words queries. We retrieved the top 10 documents resulted for each query, by using both *tf-idf* weighting scheme and the syntactic heuristic formula described above. The results are shown below.

In Fig. 5 and 6, we show the normal rank and syntactic rank of the retrieved documents for the query term *"photography"*. Fig. 7 and 8 depict the normal rank and syntactic rank for a two-words query: *"director role"* and Fig. 9 and 10 depict the same measurement for the three-words query: *"drama fight aspiration"*. We can see in these figures that the syntactic ranking algorithm alters the ranking of some retrieved documents, especially the rankings of the last documents from the top 10 retrieved documents and this effect is more pronounced when the query phrase contains more than one word.

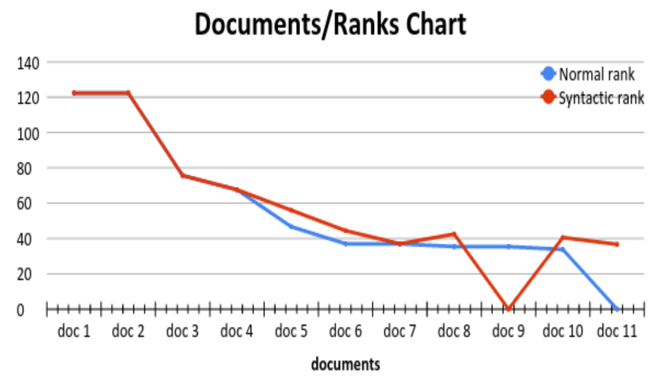
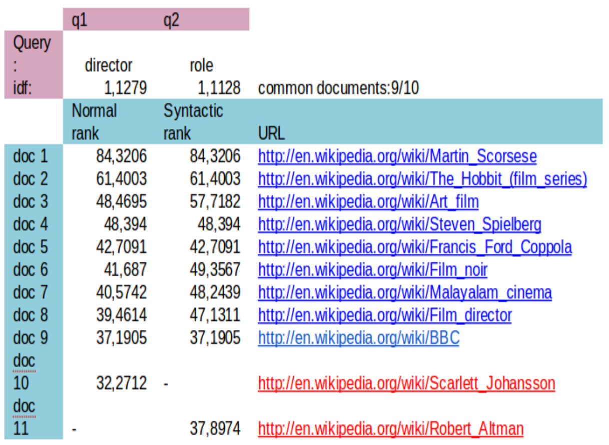


Fig. 6. Documents/ranks chart for one term query: *'photography'*.



	q1	q2	
Query:	director	role	
idf:	1,1279	1,1128	common documents:9/10
	Normal rank	Syntactic rank	URL
doc 1	84,3206	84,3206	http://en.wikipedia.org/wiki/Martin_Scorsese
doc 2	61,4003	61,4003	http://en.wikipedia.org/wiki/The_Hobbit_(film_series)
doc 3	48,4695	57,7182	http://en.wikipedia.org/wiki/Art_film
doc 4	48,394	48,394	http://en.wikipedia.org/wiki/Steven_Spielberg
doc 5	42,7091	42,7091	http://en.wikipedia.org/wiki/Francis_Ford_Coppola
doc 6	41,687	49,3567	http://en.wikipedia.org/wiki/Film_noir
doc 7	40,5742	48,2439	http://en.wikipedia.org/wiki/Malayalam_cinema
doc 8	39,4614	47,1311	http://en.wikipedia.org/wiki/Film_director
doc 9	37,1905	37,1905	http://en.wikipedia.org/wiki/BBC
doc 10	32,2712	-	http://en.wikipedia.org/wiki/Scarlett_Johansson
doc 11	-	37,8974	http://en.wikipedia.org/wiki/Robert_Altman

Fig. 7. Rank measurements for query: *'director role'*.

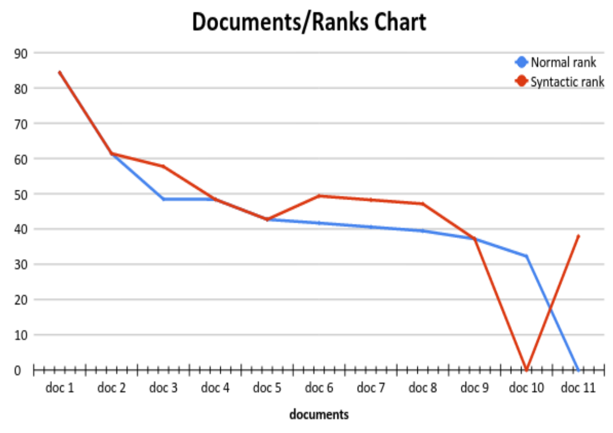
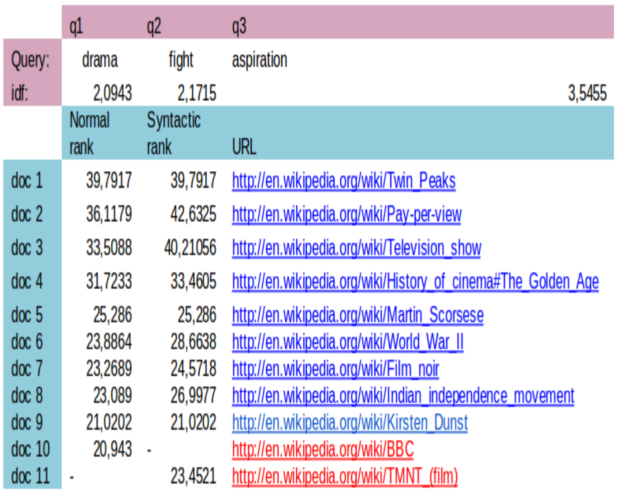


Fig. 8. Documents/ranks chart for query: *‘director role’*.



	q1	q2	q3	
Query:	drama	fight	aspiration	
idf:	2,0943	2,1715		3,5455
	Normal rank	Syntactic rank	URL	
doc 1	39,7917	39,7917	http://en.wikipedia.org/wiki/Twin_Peaks	
doc 2	36,1179	42,6325	http://en.wikipedia.org/wiki/Pay-per-view	
doc 3	33,5088	40,21056	http://en.wikipedia.org/wiki/Television_show	
doc 4	31,7233	33,4605	http://en.wikipedia.org/wiki/History_of_cinema#The_Golden_Age	
doc 5	25,286	25,286	http://en.wikipedia.org/wiki/Martin_Scorsese	
doc 6	23,8864	28,6638	http://en.wikipedia.org/wiki/World_War_II	
doc 7	23,2689	24,5718	http://en.wikipedia.org/wiki/Film_noir	
doc 8	23,089	26,9977	http://en.wikipedia.org/wiki/Indian_independence_movement	
doc 9	21,0202	21,0202	http://en.wikipedia.org/wiki/Kirsten_Dunst	
doc 10	20,943	-	http://en.wikipedia.org/wiki/BBC	
doc 11	-	23,4521	http://en.wikipedia.org/wiki/TMNT_(film)	

Fig. 9. Rank measurements for query: *‘drama fight aspiration’*.

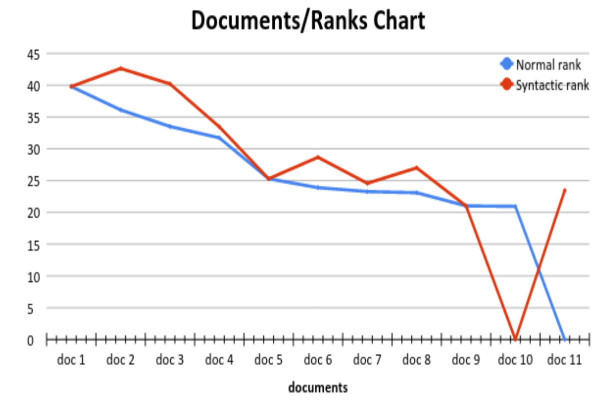


Fig. 10. Documents/ranks chart for query: *‘drama fight aspiration’*.

We can see in these figures that the syntactic index does not return totally different results than a normal *tf-idf* index, so the returned documents are relevant with respect to the query (considering that a normal *tf-idf* index returns relevant results), but it also alters the ranking of the returned results and sometimes it adds new documents to the list of the top 10 documents, depending on the syntactic category of the query terms in that specific document.

IV. Conclusions And Future Work

In this paper, we presented three methods of using syntactic categorization to get more relevant results for a web search. The first method does a syntactic analysis of the query phrase and increases the weight of the query terms which have a more important grammatical category inside the phrase. The second one aims to reduce the dimension of the inverted index by indexing not all the words which appear on a web page, but only the ones with the most important syntactic categories (subjects and predicates). And the last method illustrates a way of using the extra information offered by the syntactic index in a ranking/scoring scheme. We also presented experimental evaluations of the last two methods that show the expected benefits of using them.

As future work, we intend to test the first method, syntactic analysis of the query phrase, using both heuristic rules in order to assess its practical utility. We also intend to evaluate our methods on larger collections of crawled documents.

References

[1] D.C. Manning, P. Raghavan, and H. Schütze, An Introduction to Information Retrieval. Cambridge, England, Cambridge University Press, 2009.

[2] C. Stefano, Web Information Retrieval. Berlin, Springer, 2013.

[3] T. Lahtinen. Automatic Indexing: An Approach Using an Index Term Corpus and Combining Linguistic and Statistical Methods, PhD thesis, University of Helsinki, 2000.

[4] D.A. Evans and C. Zhai, “Noun-phrase analysis in unrestricted text for information retrieval,” in Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, June 1996, pp. 17–24.

[5] C.A. Bechikh and H. Haddad, “A quality study of noun phrases as document keywords for information retrieval,” International Conference on Control, Engineering and Information Technology, 2013.

[6] F. Jelinek. Statistical Methods for Speech Recognition. The MIT Press, Cambridge, Massachusetts, 1998.

[7] P. F. Brown, J. Cocke, S. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, “A statistical approach to machine translation,” Computational Linguistics, vol. 16, no. 2, pp. 79–85, 1990.

[8] A. Berger and J. D. Lafferty, “Information retrieval as statistical translation,” in Proceedings of SIGIR’99, 1999, pp. 222–229.

[9] D. Hiemstra, “A linguistically motivated probabilistic model of information retrieval,” in European Conference on Digital Libraries, 1988, pp. 569–584.

[10] J. Lafferty and C. Zhai, “Document language models, query models, and risk minimization for information retrieval,” in Proceedings of SIGIR’01, 2001, pp. 111–119.

[11] The Stanford Parser, http://nlp.stanford.edu/software/lex-parser.shtml.

[12] A. D. Grossman and O. Frieder. Information Retrieval: Algorithms and Heuristics, 2nd ed., Chicago: Springer, 2004.