Wikipedia Search: Combining Language Modeling and Link Analysis

Jacek Wolkowicz
Michael Shepherd
Vlado Keselj

Technical Report CS-2009-01

January 29, 2009

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

Wikipedia Search: Combining Language Modeling and Link Analysis

Jacek Wołkowicz
Faculty of Computer Science,
Dalhousie University
Halifax, NS. Canada
jacek@cs.dal.ca

Michael Shepherd
Faculty of Computer Science,
Dalhousie University
Halifax, NS. Canada
shepherd@cs.dal.ca

Vlado Keselj
Faculty of Computer Science,
Dalhousie University
Halifax, NS. Canada
vlado@cs.dal.ca

ABSTRACT

As a result of its recent rapid development, Wikipedia is likely the largest resource of organized content, manually edited through a collaborative effort by millions of "wikipedians". The complex structure of Wikipedia with many features that normal corpora do not have, such as semantically rich inter-document links, allows for successful application of new approaches to information retrieval. The contribution of this paper is twofold. First we analyze Wikipedia's usefulness as a research corpus for modern IR techniques. The second contribution is that we present a novel Wikipedia retrieval method which combines link analysis and language modeling with different weights for better ranking of the result set. The experimental results demonstrate statistically significant improvements in both precision (5%) and recall (8%) when compared to language modeling or link analysis based techniques separately.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms

Algorithms, Experimentation.

Keywords

Wikipedia, language modeling, link analysis, search engine development, information retrieval system.

1. INTRODUCTION

Information Retrieval (IR) is a well developed domain of research where even a small performance improvement seems to be an important milestone. It is true if one considers plain text as a source of data. However, many repositories, such as Wikipedia, provide more sophisticated tools, incorporating many different techniques at the same time in order to obtain more precise and complete retrieval. Most of these repositories are web based, created by real users who spend their time and efforts to create valuable, rich content such as blogs, forums, and wikis. These repositories contain more than pure text and even more than basic html, i.e., content + structure + navigational links. These kinds of repositories usually have a structure that looks like a semantic web, with named relations and objects with content. Information retrieval from these repositories should be more effective if one incorporates this information. The contributions of this paper include showing the features of Wikipedia as an interesting object for IR research and demonstrating a method for incorporating these structure-based features of the corpus to improve information retrieval performance.

A very important factor is that Wikipedia is not a static corpus. Since 2002, when it was introduced to the broad audience, it is expanding exponentially, doubling its size in each year [15] with a little slow-down in the last years (Figure 1). This makes an analysis of this corpus an ongoing research, checking if some regularities that were described previously still hold.

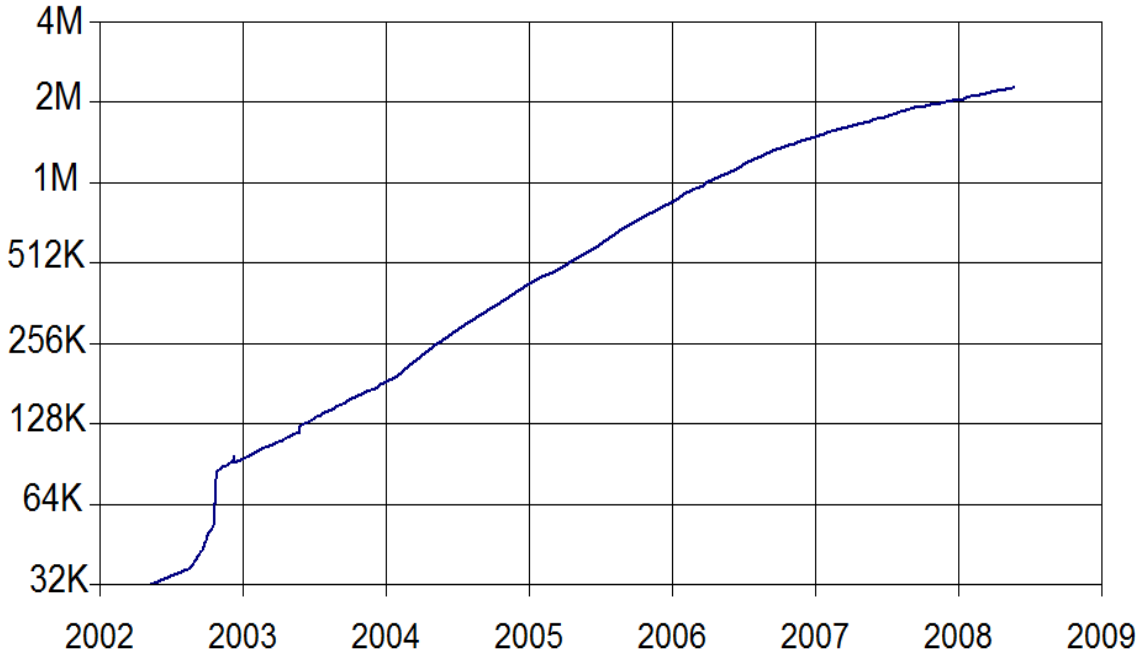

Figure 1. Articles count

The other very important problem in developing an IR system that operates on large databases is the system response time. In smaller corpora that are used as benchmark sets in IR research, some computationally expensive search methods can be successfully applied, but when the number of indexed documents exceeds millions, the computational complexity of the algorithms plays an important role. A user is not a researcher who can wait hours for results, so the algorithms have to be either computationally inexpensive or the computationally expensive parts need to be done in advance. The issue of response time is addressed in our method.

Related work is described in section 2. The methodology for the retrieval task is introduced in section 3. Sections 4, 5 and 6 contain various aspects of Wikipedia analysis. Evaluation of the retrieval system is presented in sections 7 and 8 followed by conclusions.

2. PREVIOUS WORK

Due to the advantages of Wikipedia mentioned in the previous section, Wikipedia IR becomes an interesting area of research. Lots of work has been done, that utilizes very specific features of Wikipedia. Some part of this research addresses the problem of link analysis and language modeling, but approaches combining both factors are very rare.

2.1 Wikipedia as an IR corpus

Research on search in the Wikipedia dataset seems to be quite new, with most of the research papers appearing in the last two years, so the domain of Wikipedia search seems to be an unexplored task in current IR research. Voss [15] presented various Wikipedia projects as dynamically changing text repositories with many interesting and new features compared to other corpora. His research was based on quite old Wikipedia releases and mostly on a German one, which is a few times smaller than the English one. At the time of this publication, English Wikipedia reached 500,000 articles while the corpus used in this paper was the first Wikipedia image that quadruples this number. The name Wikipedia comes from Hawaiian 'wiki' for 'quick' intentionally referring to the quick and simple access to information in wiki resources. However now this term has the new meaning for quick and rapid Wikipedia projects development and the pace of development constitutes the need of examining Wikipedia corpus periodically.

There are various approaches to search in Wikipedia. Fissaha and Rijke [5] explained why research in this area is an important task in IR development. The main hypothesis is that people tend to ask different queries to encyclopaedia-like systems than to web-based search engines like Google. The other reason is the 'semantics' of Wikipedia, which makes Wikipedia the first large text collection with explicitly encoded semantic dependencies between concepts. The need to deepen those relations such that Wikipedia would become a real Semantic Network was argued in [14].

Wikipedia has been already proposed as a test corpus for some specific IR tasks. One of them was the INEX initiative for structural XML retrieval. The brief overview of this project was given by Denoyer and Gallinari [4]. For this purpose corpora of several Wikipedia's were converted to unified XML representation allowing researching more sophisticated structural

retrieval e.g., Sigurbjornsson et al. [12]. If the aim of our paper was to compete in terms of performance with other Wikipedia search engines, this would be a good field to reliably compare different solutions. However the purpose of this corpus was different (i.e. structural retrieval) and the development of Wikipedia has made this corpus "outdated" so the most recent for the beginning of this research, available Wikipedia corpus was used instead of the INEX collection.

Part of this work is to investigate the use of link analysis for retrieval in Wikipedia. Brief analysis of document linkage can be found in Voss [15]. Detailed analysis of Wikipedia using the most important link analysis methods was performed by Bellomi and Bonato [2]. They applied both the PageRank and HITS algorithms to determine the highest ranked documents. The outcome of their work consisted of lists of page authorities with the majority being geographical and political names or historical events. The interesting point was that the PageRank algorithm, unlike HITS, ranked documents dealing with religion very high and these have taken about 60% of places in the ranking lists. It might be a mistake since the authors did not present any reason for this surprising result. Moreover, our research for PageRank authorities shows this does not hold and our results i.e., lists of PageRank authorities, are more similar to the lists from HITS algorithm than the ones from PageRank presented in that paper which may be an indicator for some misstatements.

2.2 Language modeling with link analysis

Link analysis can be combined with language modeling in any kind of retrieval tasks that deal with a corpus of linked documents. Link analysis can be used as an indicator for a prior document probability in language modeling and this approach dominates in the literature.

Prior knowledge about document relevancy may not only come from document linkage. Kraaij, et al., [8] evaluated different methods of combining document prior probabilities with language modeling among HTML web pages. They indicate that prior document probabilities resulting from various page features such as document length, number of in-links or URL depth may be helpful. This research encourages building models that take prior knowledge about the page into consideration.

There are other approaches to combing link analysis and language modeling. Richardson and Domingos [10] proposed a method which incorporates language modeling in PageRank calculations by introducing a paradigm of an "intelligent surfer" who jumps not only randomly, but according to page content. This paper introduced a different approach to the problem of combining web structure with web content by building one concise model where link analysis plays major role and language modeling is just a part of the modified PageRank algorithm. Their approach does not increase computational complexity of pure PageRank algorithm and the authors report that results significantly outperform pure PageRank.

One of the approaches that attempts to combine content information with prior information about the structure of the corpus is presented by Kimelfeld, et al. [7]. The research is done on a Wikipedia corpus, so this work can be compared to the results presented in this paper. It is shown that combining different approaches to search may help with improving the system, but actually it turned out that the HITS algorithm which they used for determining prior page probabilities decreases performance of the system. The authors deduced that content information is the most important factor in these cases, however they incorporated language modeling and links analysis with the same weights without investigating their distributions. The authors suggested experiments with the Pagerank algorithm, which is actually less sophisticated and computationally expensive than HITS. In this paper we have looked deeper in this problem, aligning both components of the documents ranking algorithm and modeling the importance of those components. We have shown that incorporating even a simpler links analysis algorithm improves the results if it is properly combined with the language modeling part. It is even expected that the importance of links analysis should be diminished since it measures only the overall importance of documents while the main language modelling objective is to determine the relevancy of a given document to a query which directly affects performance measures such as precision and recall. Performing this analysis is the main contribution of the research presented in this paper.

3. METHODOLOGY

There are many approaches to document retrieval but among them one can distinguish two basic ones:

1. Intersection based retrieval – deals with and retrieves only the documents that report relevancy to ALL the terms from the query.
2. Union based retrieval– takes into consideration documents that report relevancy to ANY term from the query.

In most commercial search engines, focused on usability, the first approach dominates. Queries given to a system are usually short (1-2 related words) and most users only review the first page or first two pages of retrieved documents [6]. Those systems should also be fast. On the other hand, while taking union, usually a large proportion of the documents in the collection show some relevancy to a query, especially if the query is long and has terms commonly used in the corpus. In this case, ranking calculations for the documents are applied to a large proportion of the collection increasing computational

complexity, especially if the system copes with many millions and billions of documents. This may not be essential in research experiments, but a prompt search engine response is a hard requirement for online users.

From the reasons presented above, the following model is proposed:

1. Retrieve documents that contain all terms from the query.
2. Rank retrieved documents according to the given model.

The main research question in this paper is to determine how to combine Language Modeling (LM) and Link Analysis (LA) for the best retrieval performance. Wikipedia has specific features that favour LM. There is almost no information noise in the documents, content is rich and valuable and words define meaning of a document. Another important fact that differentiates Wikipedia from the web is that, as in encyclopaedias – every document may be relevant to some query i.e., there are no unimportant pages. However, some documents (like "List of…") contain lots of keywords but carry very little semantic relevancy. On the other hand, we have lots of links between documents (statistically one link in eight terms) which were manually implemented by the users. Although, the specificity of web links makes them more useful compared to Wikipedia links. A web link defines a webpage identified by an URL which is not designed to be a part of the text. That is why those links have to be given a caption that describes the target and provides something to be clicked. Those link captions sometimes better inform about the content of the targeting page than a page itself. On the other hand documents are identified by a title in Wikipedia and the method of creating links makes them poor. In order to make a link a user has just to take a term in braces and the link to the document with the same name as the marked text is created. This method enables but does not require creating a special caption describing the content of a targeting page which leads to the situation that links do not help in determining the semantics of the content of the target page. These facts lead to the conclusion that the influence of Link Analysis should be minor.

To start with a basic Bayesian rule, the probability of a document being relevant to a given query can be expressed as:

$$P(d \mid Q) = \frac{P(Q \mid d) \cdot P(d)}{P(Q)} \tag{1}$$

where $P(Q)$ states for the probability of a query which remains constant for every document and thus can be omitted or set to *1*. After this omission the resulting value $P(d|Q)$ no longer represents a probability. One can observe, that $P(Q|d)$, the probability of a query with respect to the given document, expresses the language modeling part of the equation while prior document probability, $P(d)$, may be expressed as PageRank of a page [3]. The aim of the following reasoning is to enhance this equation so one can change the formula to obtain desired influence of LM and LA. One of possible approaches presents the following formula:

$$R'(d \mid Q) = P(d)^{\beta} \cdot P(Q \mid d)^{(1-\beta)} \tag{2}$$

where $R'(d|Q)$ is the rank of a document d for a given query, Q and □ is the LA importance factor. The same can be expressed in more convenient, logarithmic form, with pr_d standing for PageRank factor of a document:

$$R(d \mid Q) = \beta \log(pr_d) + (1-\beta)\log(p(Q \mid d)) \tag{3}$$

We have used the standard PageRank equation presented in [3]. More details about calculating pr_d will be shown in a further section and now let us concentrate on the LM part. Ponte and Croft [9] proposed the following model, that incorporates various aspects of given documents contents. It starts with the assumption that terms are independent so:

$$p(Q \mid d) = \prod_{q \in Q} p(q \mid d) \tag{4}$$

Each $p(q|d)$ is a combination of the likelihood of a query term occurring in a document and for a corpus overall according to the formula:

$$p(q \mid d) = p_{ml}(q,d)^{(1-R(q,d))} \cdot p_{avg}(q)^{R(q,d)} \tag{5}$$

This formula will be clarified in the following paragraphs.

p_{ml} is the likelihood that a document is relevant to a term, given as follows:

$$p_{ml}(q,d) = \frac{tf_{q,d}}{dl_d} \tag{6}$$

where $tf_{q,d}$ is term frequency in a certain document and dl_d is length of a document (number of word occurrences). However, if the $tf_{q,d}$ (term frequency of a document) is close to the average in the documents containing given term, which is expressed by the following formula (df_t stands for the number of documents a term occurs in):

$$p_{avg}(q) = \frac{\sum_{d \in D: q \in D} p_{ml}(q,d)}{df_q} \tag{7}$$

then *p(q|d)* is calculated in equation 5 to take the advantage of the average corpus distribution according to this factor:

$$R(q,d) = \left(\frac{1}{1+f_q}\right) \cdot \left(\frac{f_q}{1+f_q}\right)^{tf_{q,d}} \tag{8}$$

where f_q is an average frequency of terms in all documents containing this term.

In this paper the model is combined with a very successful, Okapi BM25 ranking procedure presented by Robertson and Walker [11], so at first the formula for p_{ml} changes to the one proposed in equation:

$$p_{ml}(q,d) = \frac{tf_{q,d}(k+1)}{tf_{q,d} + k\left(1 - b + b\frac{dl_d}{\sum_{i \in D: q \in D} dl_i \,/\, \sum_{i \in D: q \in D} tf_{q,i}}\right)} \tag{9}$$

where k may be taken as 1.2 and b is 0.75 according to the values proposed in [13]. BM25 incorporates also idf_q (inverse document frequency) to give proper weights to terms according to their distribution in the corpus according to the given formula (N stands for the total number of documents in the collection):

$$idf_q = \log \frac{N - df_q + 0.5}{df_q + 0.5} \tag{10}$$

and then use sum instead of multiplication to obtain the probability for a given query (which is more like a likelihood, not a "real" (normalized) probability in this case):

$$p(Q \mid d) = \sum_{q \in Q} idf_q \cdot p(q \mid d) \tag{11}$$

4. WIKIPEDIA CONTENT ANALYSIS

Wikipedia maintains regular backups. They are all freely available online for everybody on http://download.wikimedia.org. Each Wikipedia database structure consists of 36 tables carrying information about edited documents, all historic revisions, user activity, document classification structure, links between documents, logging, media and much more. Most of these snapshots of Wikipedia are publicly available, especially the text which is the most interesting for IR researchers,. There were a number of attempts of building an IR corpus based on Wikipedia [4]; however, Wikipedia changes so dynamically, that past versions of the corpus are usually very small compared to the latest Wikipedia release. All information was manually edited by millions of Wikipedia editors and it has become one of the largest resources of unified human effort. Various language versions of Wikipedia and the strong interconnection of all language versions enable multilingual search. Wikipedia allows users to assign categories to documents. There are currently more than 300,000 categories organized manually in a graph structure. Incorporating categories and categories structure to the search could enable various types of category searching. Utilizing document abstracts may help in developing better retrieval systems. PageLinks table contains links between documents which may be helpful in documents link analysis. The latter feature is utilized in this approach.

The Wikipedia snapshot from 18 October 2007 was used for this research, which was the current snapshot at the time of the research. It consists of 10,171,410 total pages which gives over 12 GB of textual data; however most of them are not valid articles. The first operation done on the data was to remove pages that are not regular documents, such as categories, talks (discussions), and redirections between documents. Redirections were kept for further processing to preserve future data consistency. The next step consisted of removing all the information that is not printable and does not appear in a document preview, such us markup, tables' layout, special Wikipedia keywords and commands were removed from valid pages, see Figure 2.

The main problem in this area is that users do not follow exact Wikimedia[1] specification. The second important problem in processing Wikipedia is that the corpus is very large and it does not fit in memory, so it must be processed in many steps.

[1] Wikimedia is an organization that hosts wiki projects, not to be confused with Wikipedia which is just one enterprise of Wikimedia foundation. To add spice, Wikipedia engine that drives all Wikimedia projects is called Mediawiki.

```
{{pp-semi-protected|small=yes}}
{{redirect4|John Kennedy|JFK}}
{{Infobox_President
| name                = John Fitzgerald Kennedy
| signature           = John F. Kennedy signature.gif
}}
'''John Fitzgerald Kennedy ''' ([[May 29]],
[[1917]]&ndash;[[November 22]],
[[1963]]), was the thirty-fifth [[President of the United
States]],
serving from 1961 until his [[John F. Kennedy
assassination|assassination]] in 1963.
```

```
John Fitzgerald Kennedy (May 29, 1917–
November 22, 1963), was the thirty-fifth
President of the United States, serving from
1961 until his assassination in 1963.
```

Figure 2. Removing Wikipedia Markup

As a result of those efforts, a file named 'documents.xml' containing only the necessary information was created. It is radically smaller, than the original Wikipedia snapshot (4.88GB vs. 12.1GB). Each document of this file consists of:

1. Document ID according to original Wikipedia pageID.
2. Document title.
3. Categories that a document belongs to.
4. Timestamp of the last edition.
5. PageRank of the document.
6. Cleaned document content.

The pageID and PageRank features are used in the system to identify and rank documents; title and text are used in indexing while the timestamp is only presented to the user during presentation of the results. Category information is not utilized by this system, but is left for future work.

Various statistics describing this corpus are presented in the following paragraphs.

2,080,837 – the number of legitimate and clean articles.

814,015,293 – the number of words occurrences.

4,098,366 – the number of different, case insensitive words, including:

1. 2,120,405 distinct words that occur only once (52% total)
2. 549,625 distinct words that occur twice (13.5% total)
3. 259,620 distinct words that occur three times (6.4% total)
4. 162,702 distinct words that occur four times (4.0% total)

3,020,526 – the number of different stems using Porter's stemmer and the collection of words presented above including:

5. 1,476,822 distinct stems that occur only once (49% total)
6. 407,298 distinct stems that occur twice (13.5% total)
7. 196,640 distinct stems that occur three times (6.5% total)
8. 125,197 distinct stems that occur four times (4.2% total)

Since using stemming does not always lead to significantly better solutions [1] and may be a source of certain errors and ambiguities, stemming was not used in the results presented in this paper. Both stems and words statistics satisfy Zipf's distribution.

814,015,293 – the number of words occurrences where:

1. 'the' occurs 50,279,037 times
2. 'of' occurs 25,999,577 times
3. 'and' occurs 20,415,660 times
4. 'in' occurs 19,089,648 times
5. 'a' occurs 15,886,812 times
6. 'to' occurs 14,493,166 times

The other words that occur more then 2,000,000 times are: ‘is’, ‘was’, ‘for’, ‘s’, ‘as’, ‘on’, ‘by’, ‘with’, ‘he’, ‘that’, ‘from’, ‘it’, ‘at’, ‘his’, ‘an’, ‘are’, ‘this’, ‘or’, ‘also’, ‘where’, ‘which’, ‘be’.

Figure 3 represents Zipf’s distribution of terms in the corpus. Both words and stems statistics are very similar.

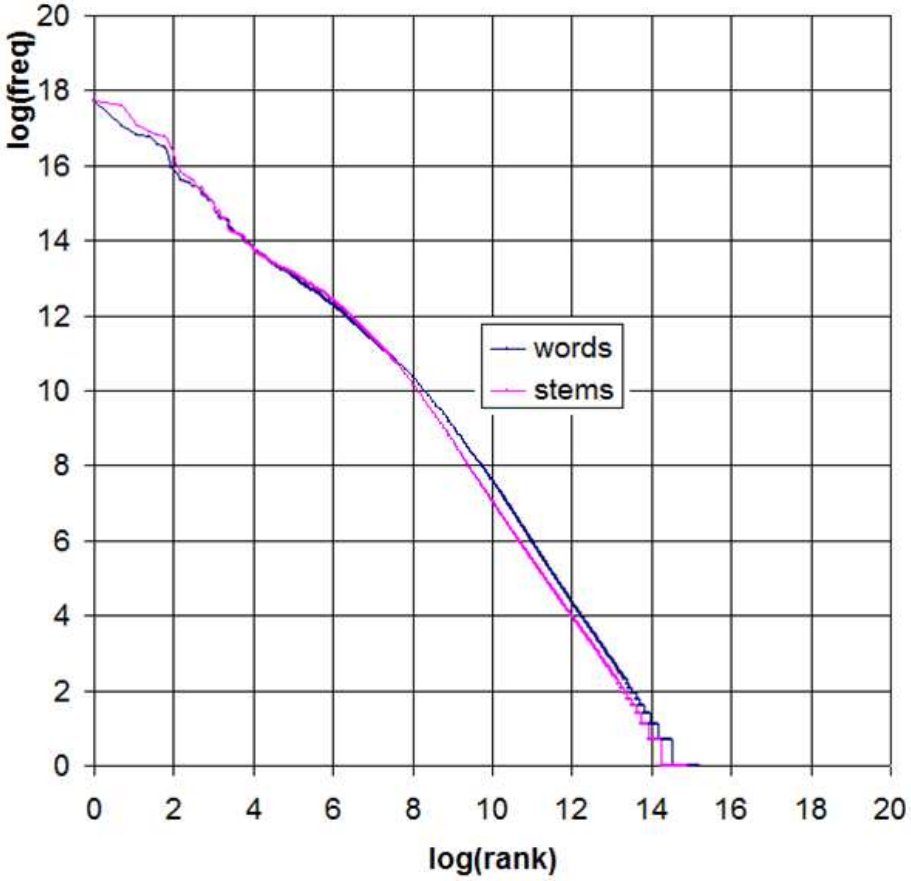

Figure 3. Zipf’s distribution of words and stems in Wikipedia

5. PAGERANK CALCULATION

Links information was extracted from the PageLinks table that contains information linkage between documents. The links consist of source page represented by pageID and target page represented by title. The target is represented by the title rather than by pageID as many of the target pages have not yet been created and are place holders for the pages to be added. Since the order of page IDs preserves the order of documents creation time, one can track the links that refer to non-existing pages. It turns out that most of the links on old pages refer to existing documents and broken links are found mostly on newly added documents (Figure 4). Each point on the plot represents 100,000 links ordered in ascending order according to source page ID. What is interesting, J. Voss came to the same conclusion using different reasoning [15].

This proves a well known Wikipedia’s rule that if you want to have a document – just leave an empty link and wait a while for somebody to write the article for you.

In this version of Wikipedia, there are 186,887,004 links. This includes:

1. 84,736,287 – the number of existing links between documents.
2. 89,024,498 – the number of broken links between documents.
3. 13,126,219 – others, pointing to categories, talks and other types of pages that are not regular documents.

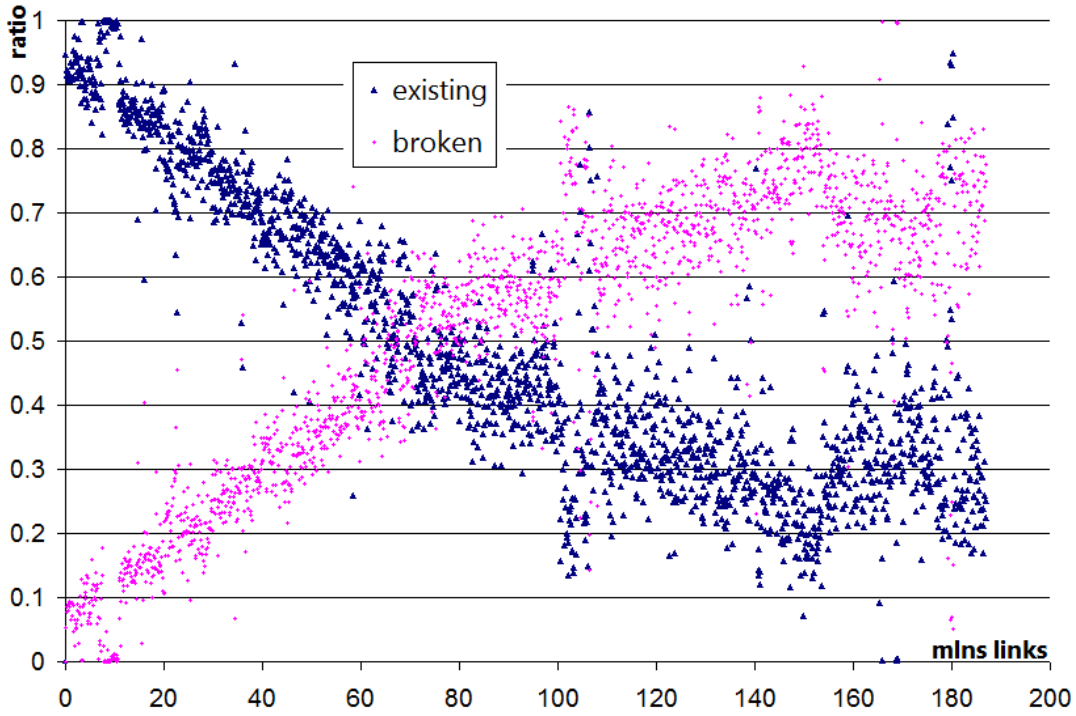

Figure 4. Existing and broken links in Wikipedia

The document graph was then extracted using existing and correct links. The original PageRank algorithm was implemented by Page and Brin [3]. PageRank is based on the idea of a random surfer, who starts at a random page and with some probability (*1-d*) jumps to another random page or clicks a random link on an existing page (probability *d*). The main idea of this system is that if a page has a lot of incoming links – it is important and their outgoing links are also important. The simple illustration of Figure 5 shows how it works:

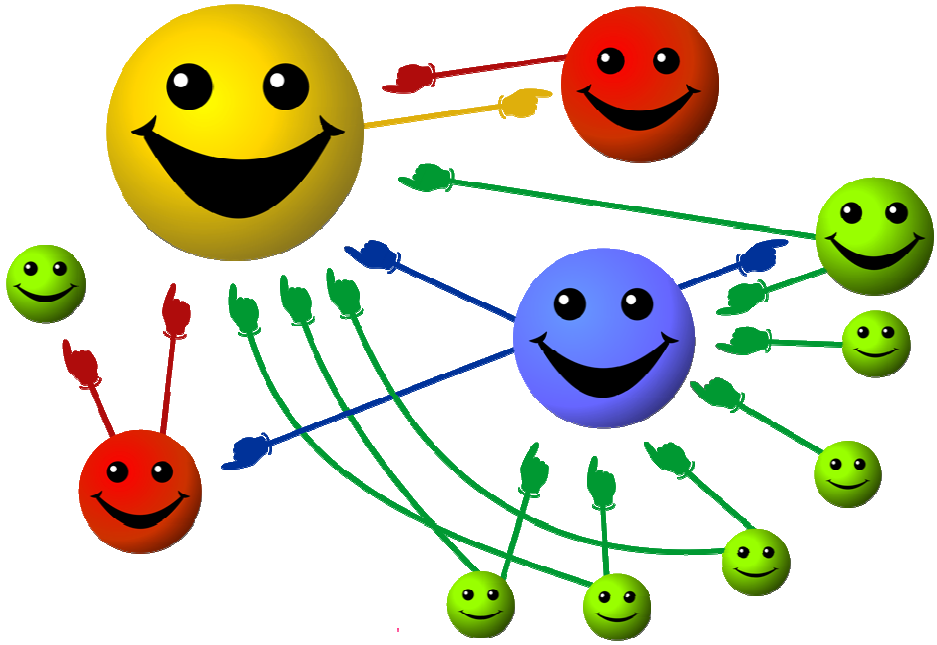

Figure 5. Pagerank page importance illustration

The original Brin's and Page's iterative formula was used in calculating rank of documents:

$$pr_{d,i} = \frac{(1-d)}{N} + d \sum_{\delta : e(\delta,d) \in E(G)} pr_{\delta,i-1} \Big/ \sum_{\Delta : e(\Delta,\delta) \in E(G)} 1 \qquad (12)$$

Where $pr_{d,i}$ is the rank of document d in *i'th* iteration, $e(d_1,d_2)$ is a link between two documents and $E(G)$ is a set all links in Wikipedia and N is the number of documents in the collection. In this solution d is fixed and equals 0.85, according to [3].

This algorithm is unlikely to converge quickly for large graphs. However, the aggregated difference of all factors in each two consecutive steps is decreasing exponentially. In this approach 50 iterations of the PageRank algorithm were calculated. Figure 6 shows the convergence of PageRank calculations in logarithmic scale (the difference is multiplied by the number of documents). The total error after calculating these steps is smaller than an average PageRank of a single document so this might be a good indicator to finish calculations. Terminating calculation is an important fact since each step has taken about 6 minutes to be done on a 3GHz machine and the very exact precision in determining PageRank values is not needed.

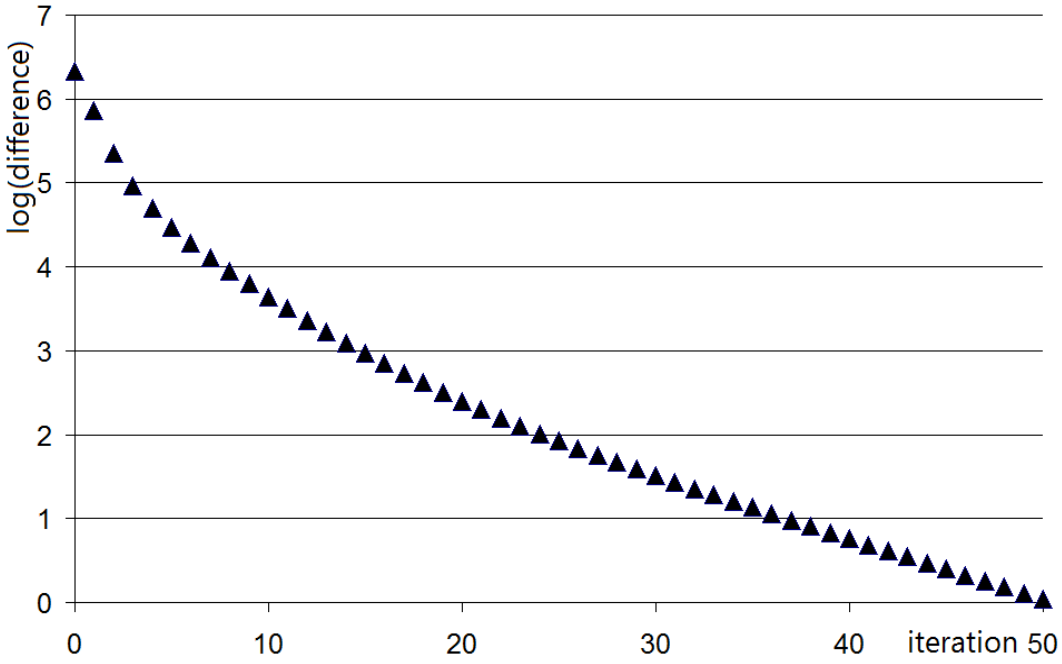

Figure 6. Convergence of PageRank calculations

Since each PageRank value approximates the probability of entering a page in an infinite number of steps, the value of RageRank is usually very low if the number of documents in the collection is high. In order to cope with the problem, each probability was multiplied by the number of documents. This new value shows for how many pages this page can 'count'. The actual values of PageRank in this approach vary from 0.15 to over 7000. However, in order not to store floating point values of PageRank, each value was then multiplied by 100 and represented as an integer value in the document.xml corpus. This step was made for storing purposes only. The list below shows PageRank authorities – the pages that carry the highest PageRank. One can observe that all of them represent a country or a year, which shows what the most important concepts in Wikipedia are: places and events.

1. United States 7133
2. 2007 3818
3. United Kingdom 3258
4. 2006 2999
5. Geographic coordinate system 2974
6. Canada 2089
7. 2005 2045
8. France 2013

5.1 Combining Pagerank with LM

In order to combine LA and LM, one has to be sure that they operate in the same feature space, i.e., they have comparable distributions in the results of a query. In order to observe this, one can plot the distribution of PageRank values (Figure 8) and compare it to the distribution of values that result from the LM only (Figure 7). One can see that relevancy values i.e., LM likelihood values derived according to the model, for the results for the query "information retrieval" vary from 2.5 to over 20 and the distribution of values is more or less homogeneous. Since LM likelihood values depend on queries and documents it is hard to determine their ranges but usually they have a similar range as for the sample query.

On the other hand one can observe that the PageRank for the same query contains the values from 0.15 to 300 spread in a much diversified way. Just few documents have high rank and most of the values are closer to the lowest 0.15. However, after applying a logarithm function to the PageRank results, the values vary from 2.7 to 10 and present more uniform distribution, that the original PageRank values (Figure 9).

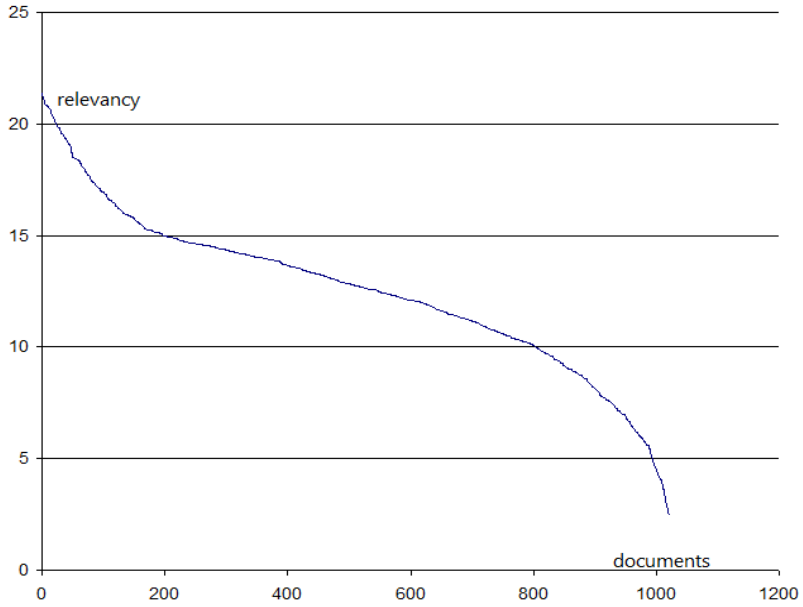

Figure 7. LM ranks distribution, query "Information retrieval"

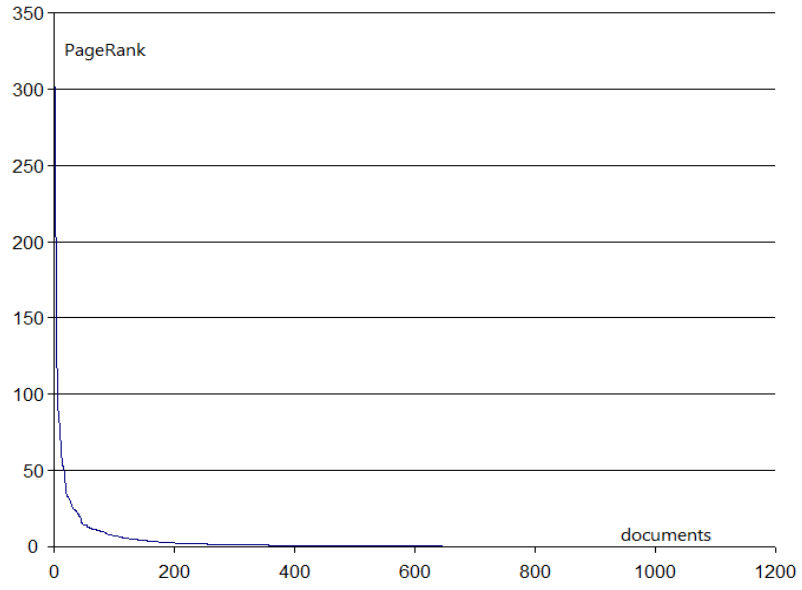

Figure 8. PageRank's distribution, query "Information retrieval"

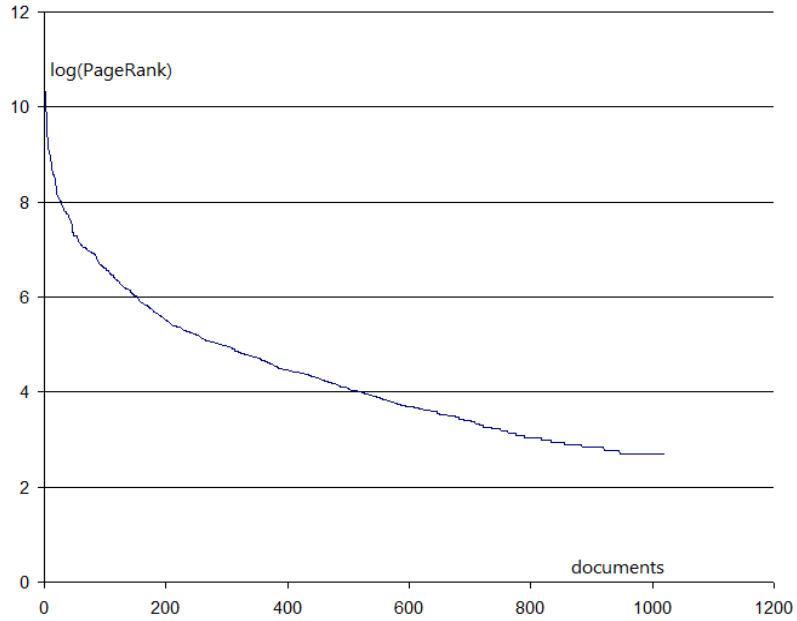

Figure 9. Logarithm of PageRank distribution

Since the distribution of the logarithm of the PageRank (Figure 9) values is comparable to LM values distribution, the formula for pr_d in Equation 3 can be evaluated as follows:

$$pr_d = \log(Pagerank_d) \tag{13}$$

6. INDEXING WIKIPEDIA

While building an index on small datasets, like TREC corpora, one does not have bother memory limitations and performs the whole index construction in the memory and then dump it into a hash file, like gdbm file. In this situation, the whole index will not fit in the memory and the simplest approach of storing the index in a hash file will be to slow for this amount of data.

I have proposed my own structure of the index. It was implemented as a PERL module with simple client object-oriented interface allowing the following operations to be performed:

9. `create` – creates an index with desired parameters.
10. `existing` – opens existing index.
11. `expand` – rebuilds the index by increasing the depth of the index by 1.
12. `sync` – dumps the cache of the index and synchronizes modified contents.
13. `get` – retrieves a value connected with a specific key passed as a parameter.
14. `add` – add elements to the index.

The simplest use cases of the indexer are as follows:

15. Building index

```
index = create
while there are items to be indexed
    index.add(items)
index.sync
index.expand
```

16. Using index

```
index = existing
value = index.get(key)
```

The given functionalities will be explained in the following paragraphs.

This approach relies on system directory structure. Many widely used file systems, like NTFS or EXT3 supports storing large amount of data in large number of files enabling fast file access based on B+ trees directory index structures. Storing index in many small files allows simple index updates without expensive index rewriting and allows simple access to a certain file knowing its path and name. However, most of file systems do not like storing too many files at one level. In this

approach each tree level consists of at most 16 nodes representing one hex character. According to these rules, the obtainable numbers of items placed in this structure (as leafs) are all powers of 16, i.e. 16, 4k, 64k, 1M etc. depending on the depth of the structure. On the other hand, each leaf file should not contain only one term since most of them occur only few times but the minimal file size is fixed by the file system allocation unit size (i.e. 4kb).

In order to find (or store) a value in this structure one has to calculate hash function of a key to find a corresponding path and the scan through a leaf-node file looking for a specific key and value or store a value in that file. MD4 hash function was used in this approach as a very fast algorithm that uniformly distributes hash values in the domain. Then the beginning of that hash determines the path of the element. For instance, a word 'the' has MD4 sum like 'e3c78ad5a802ba92d0093daca19d5a5e', so with 4-level structure the path to the file that should contain a value for 'the' is index/e/3/c/7. 4-level structure gives 64k flat files containing information about terms which gives approximately 60 terms per flat file, assuming incorporating all terms. This seems to be a good trade-off between number of files in index (which has an influence on the speed of creating an index and file system exhaustion) and the number of different term in a flat file (smaller number – faster searching). The sample structure of the index with the method of retrieving information about terms is shown in Figure 10.

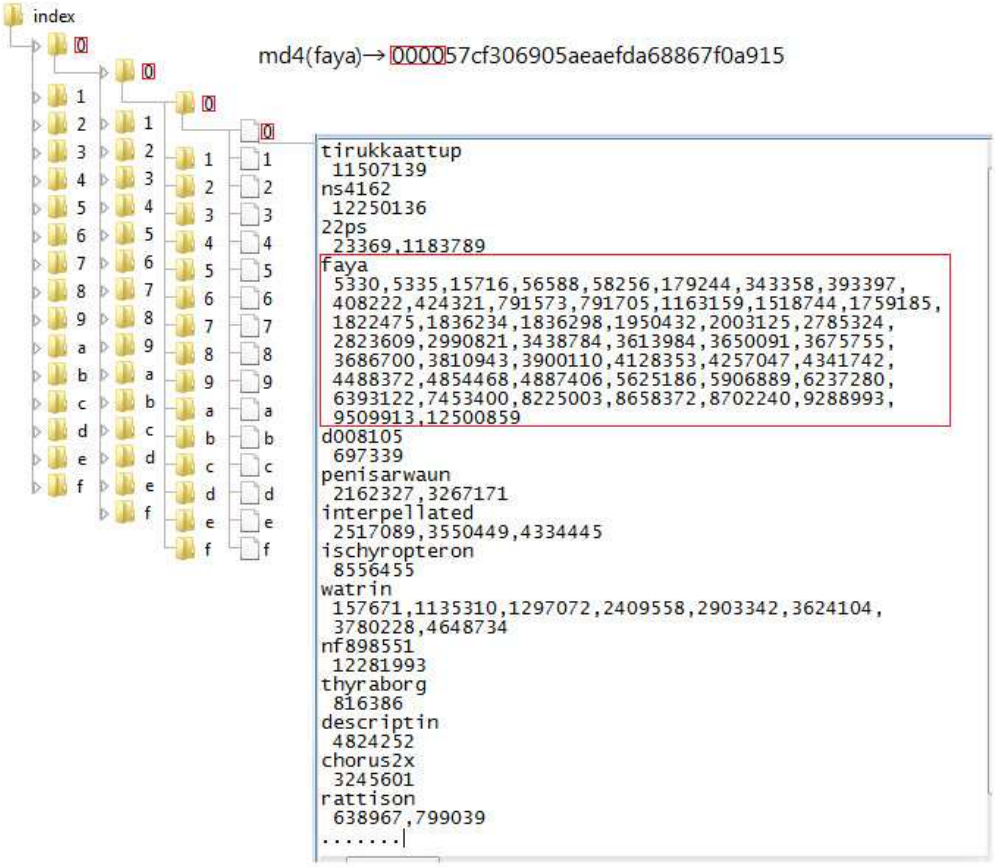

Figure 10. Simple index structure for word 'faya'

This approach enables index caching while it is being created. Since the most expensive is closing and opening the file (which should be performed about 1,000,000,000 times), the information about word occurrences is cached in the hash structure in the memory until the number of cached elements exceeds a certain limit (here, 5,000,000 word occurrences limit gave 500MB of memory usage). In the cache dumping stage, all cached index information is stored in temporary files (usually appended to the existing temporary file). After all the information is captured the index is compacted (`sync` operation), i.e. index cache is flushed and all the information about each term in a file is merged and the actual index file is created. In this approach an index with level 3 (3 levels of the tree) is created and then, after indexing is done, index is expanded to 4 levels (operation `expand`). This operation spares much time because 3-level index contains 4k leaf files but 4-level index – 64k leaf files and cache dumping is performed significantly faster. Expanding operation takes about an hour

comparing to the decline from 7 hour index building time (for 4-level index) to 3 hours (for 3-levels index) on a laptop (Turion X2. 2GB RAM, Windows Vista).

For this case indexing has taken:

3 hrs – index building
1 hr – index merging
1 hr – index expanding

The disk space used by the basic index build upon Wikipedia takes 2.53GB of disk space; however, the index is fully textual and can be compressed to save some disk space (to about a fifth of its original size). The basic index contains information about documents IDs only, while in order to retrieve the content of the document one needs the offset in documents files. It is done by the mapping file, which maps document IDs to the offset in the documents.xml file. The access to the mapping file consists on binary search so the access to the content of a document is logarithmic.

The basic index does not contain additional information about term occurrences, document features like Pagerank value, so the actual indexer can store multi-feature values and can also combine keys with additional information about keys (terms). The structure of the index does not change, but the structure of flat files differs. The actual index contains information about the following collection/document/term features:

1. word name (as a key)
2. number of documents in the collection
3. number of all word occurrences in the entire collection
4. number of a certain word occurrences
5. number of documents containing given word (document frequency)
6. document IDs containing given word (as a primary value)
7. word count for a given word and a given document (term frequency)
8. length of a current document
9. pagerank of a current document

The sample index page is shown in the Figure 11.

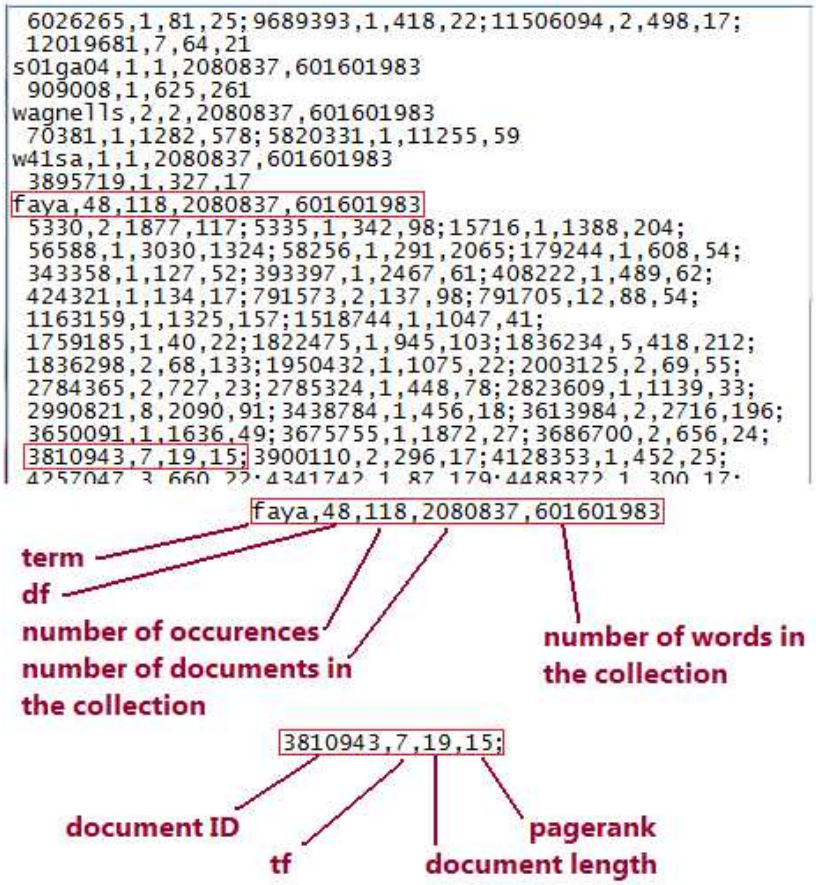

Figure 11. Enhanced index content

For the purpose of Wikipedia Search engine, 40 terms with the frequencies above a million, that do not carry any semantic meaning (stop words) were omitted. However, all the remaining words were indexed even if they occur only once. The words that occur once make a half of the different words amount but it is not a significant amount in terms of data storage (about 100MB, 50 bytes per term). The whole, enhanced, pure textual index requires about 5.9 GB of disk space. Compression would also allow sparing more than 70% of original space.

7. EVALUATION

Since that there are no relevance judgments done for Wikipedia queries, a user study was performed in order to evaluate the system. The retrieval program was given a Web-based interface. The interface was similar to Google's interface, so it will be easy for any user to use this system. Nine users, students and faculty members, were asked to take part in the study. In the first step, each user submitted a set of queries, resulting in 120 user queries being generated. The queries may refer to any topic. Sample queries are given as follows:

Asian Food
Clifford the big red dog
Generalized Vector Space Model
Hilary Clinton
…

In the next step, the system was run for every query in order to obtain all the documents that occur in the top 10 positions for any values of □(Equation 3). In this case, 'any' means the step of 0.001 from the range of 0 to 1. As a result of that process, the sizes of the document sets vary from 3 to over 50 depending on the dynamic of a certain query results set.

In the next step, every query was given to 3 randomly chosen participants and they were asked to mark the retrieved and ranked documents as relevant or not relevant to the given query. Thus every document was judged 3 times in terms of its relevancy and there was a total of 3410 threefold judgments. 63% of the judgments were assigned unanimously, 21% (703) relevant, 42% (1436) not relevant. 37% were judged by the majority (2:1) where 18% (612) likely relevant, 19% (659) likely not relevant.

8. RESULTS

Knowing all judgments about documents that appear in the first 10 positions for any settings of the system enables us to make relevancy graphs which visualize the position of relevant and non relevant documents. Sample pictures are shown in the Figure 12, Figure 13 and Figure 14. Rank (places from 1 to 10) represents a position of a document in the list of 10 most relevant documents. Colors represent relevancy of a document in a particular place for a particular value of □and span from green color that represents 3:0 agreement that the document is relevant to red that represents 3:0 for non relevancy. For instance, if the color for a particular place and for a particular value of □is white then it means that there was a relevant document at this place on the list of 10 most relevant documents. On the first picture one can observe the migration of not relevant documents for small values of □; however it does not happen in Figure 14. In all pictures one can observe, that for □=1 (only LA), the non-relevant documents are a majority.

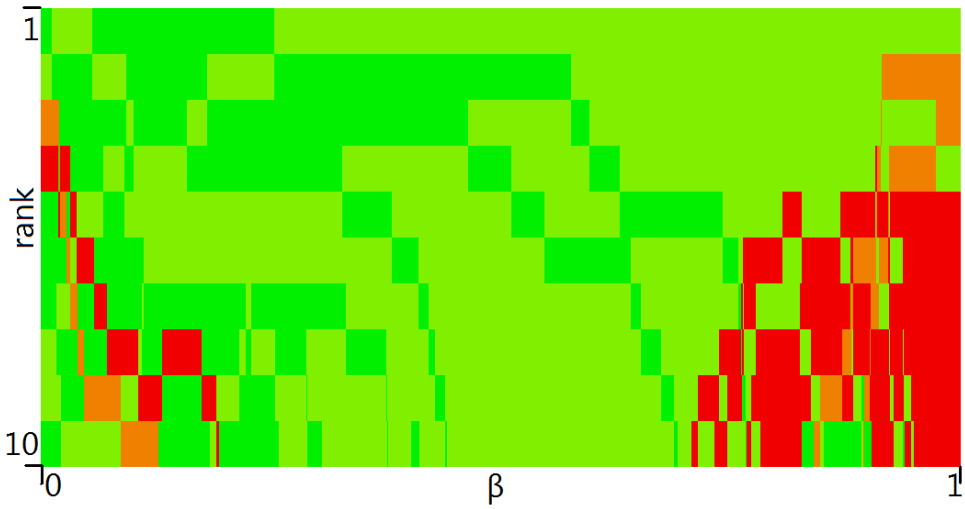

Figure 12. Position of relevant documents for various □ query "Canadian Economy"

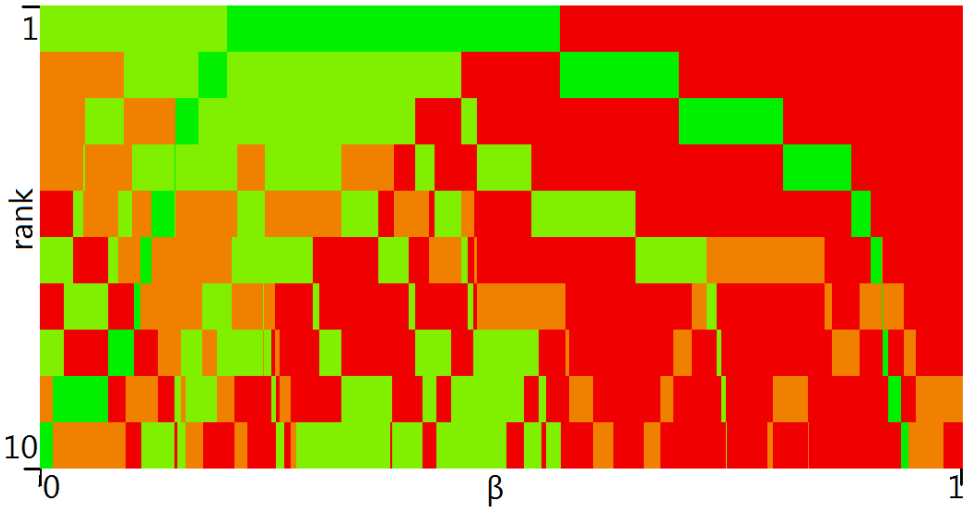

Figure 13. Position of relevant documents for various □ query "Web Graph"

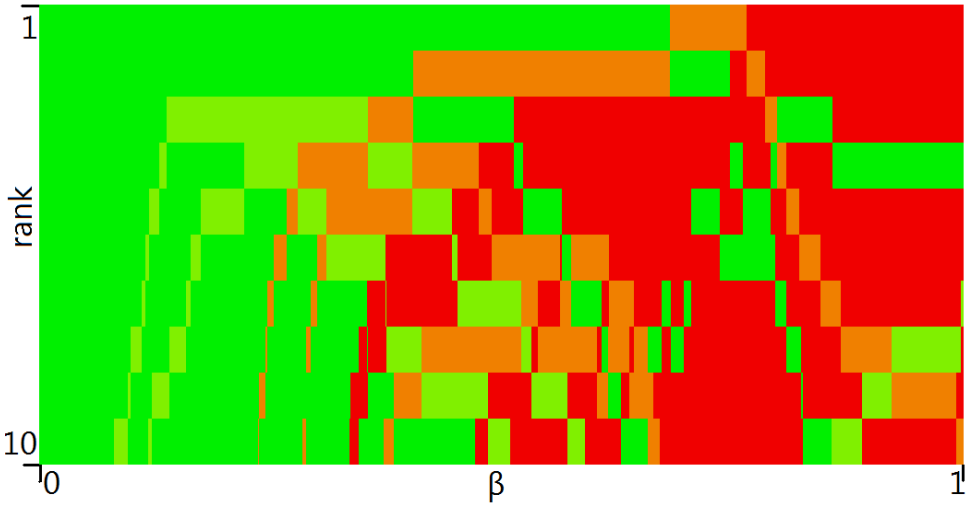

Figure 14. Position of relevant documents for various □ query "Cowboy Bebop"

In order to calculate the best value of □, average Precision at 10 (P10) was calculated for all collected queries. These results are shown in Figure 15, parts a and b. In both parts, the x-axis represents the □ values from 0 to 1. In part a, the y-axis represents the precision at 10. In part b, the y-axis represents the statistical confidence level.

In Figure 15a, one can observe a slight improvement in precision for □ values in the range 0-0.3 compared to the average precision for □=0. The biggest improvement reaches 4.7% for □=0.08 (Figure 15a).

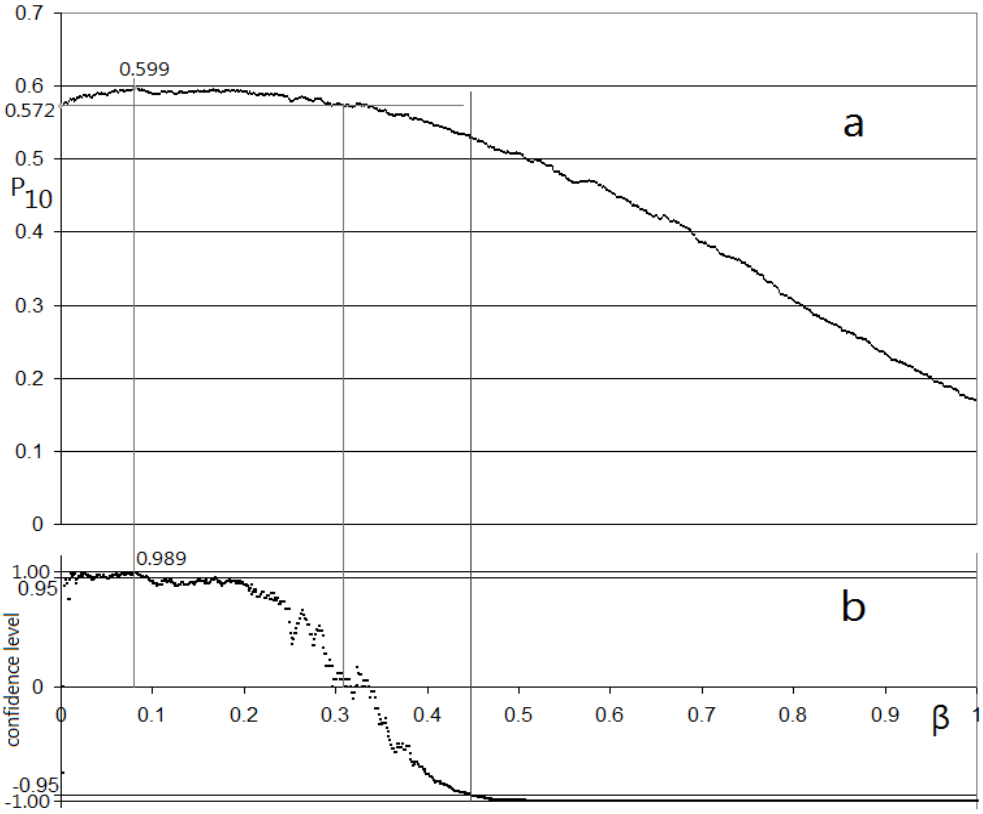

Figure 15. Average precision at 10 for different values of β

In order to determine if the improvement is significant, the Z-test was done for normalized P10 results for the queries at all values of β compared to their performance for the pure LM model (β=0). The use of the Z-test is justified since there are more than 100 test points per this sample.

Figure 15b shows that the confidence levels for the corresponding P10 value is significantly greater (lower) than for the P10 at β=0. The 95% confidence threshold was marked to indicate for which β the results are statistically significantly different comparing to the initial point. All the values with absolute confidence above this level are considered as significantly different from the starting point. When P10 reaches its highest point, the confidence level that the improvement is significant reaches 98.9%. Precision improvement is statistically significant for various β values in the range from 0 (excluding 0 of course) to 0.1. The system performs statistically significantly worse for βvalues larger than 0.45.

The next value that was measured for this system is normalized recall. Since the global number of relevant and not relevant documents of each query is unknown, normalized recall measured at the level of the 10 first documents is a feasible evaluation measure in this situation. It tells how high relevant documents are placed on the first page of 10 documents presented to the user.

In Figure 16a, the y-axis represents the normalized recall. In this case (Figure 16a), one can also observe an improvement but it is greater (8.3% for β=0.04) than the best P10 improvement. The same statistical analysis that was performed to determine statistical significance for P10 was also performed for normalized recall. Figure 16b shows that significance level for the highest value of recall is 99.7% and it remains significant greater for βin the range from 0 to 0.065 (excluding some values very close to 0). The recall performance decreases significantly for βvalues larger than 0.67.

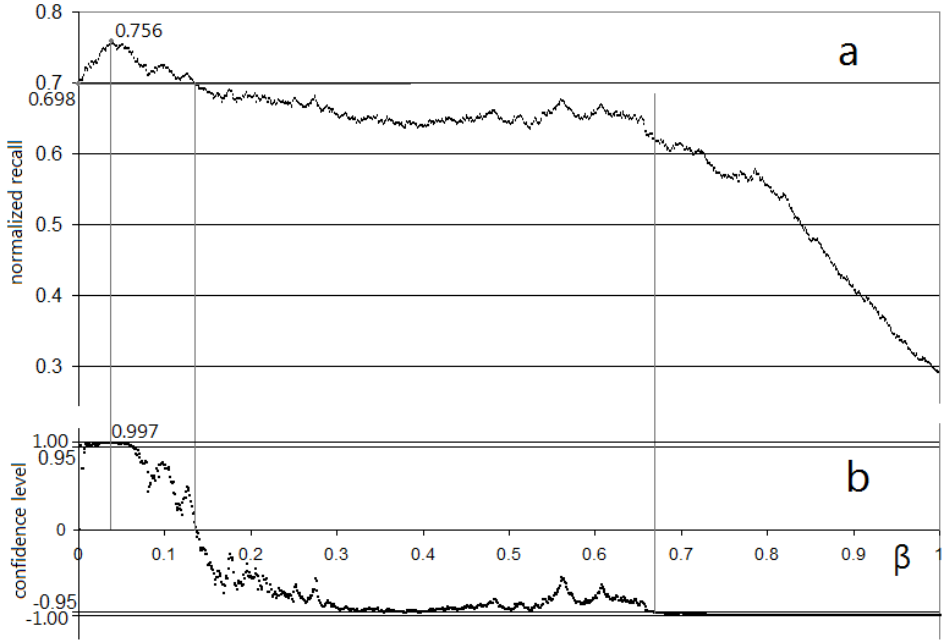

Figure 16. Average normalized recall for various □

The other view for the problem of choosing the best value for □can be shown on the recall/precision graph for different values of □(Figure 17). One can observe the walk of the system on the recall/precision plot. It shows that choosing □value one can obtain better system in terms of recall or precision. The performance of the system decreases for large values of □ (high influence of Link Analysis part) which shows, that Language Modeling is more important for Wikipedia retrieval.

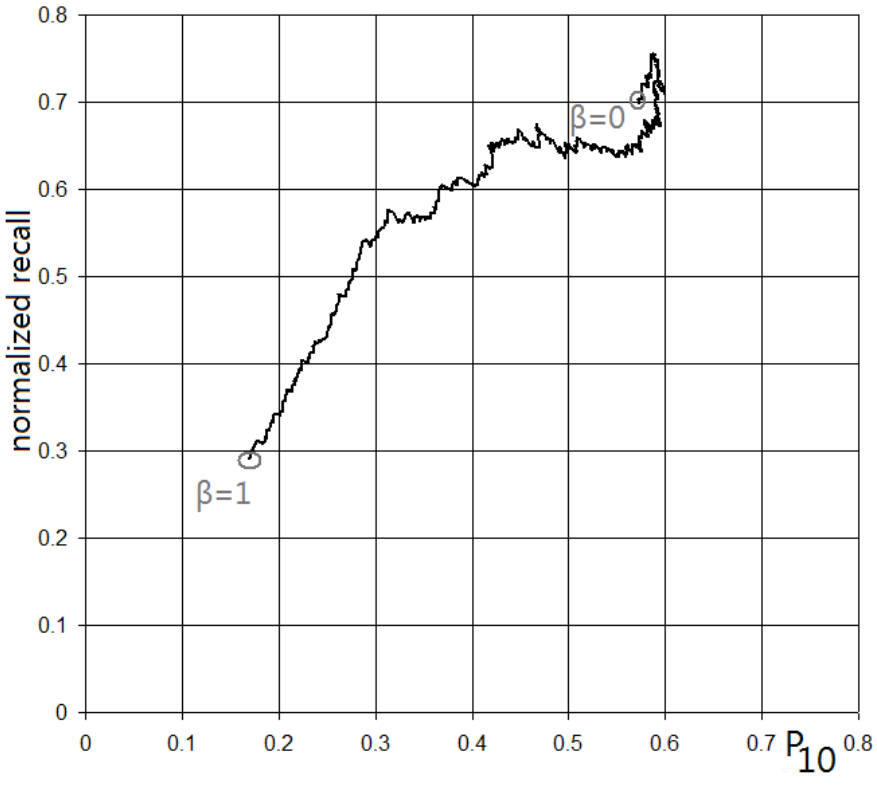

Figure 17. Recall for precision graph

The magnified section for best system settings is presented in Figure 18. According to these results, the best values for □lie in the range from 0.04 to 0.08 depending how precise or complete system we need.

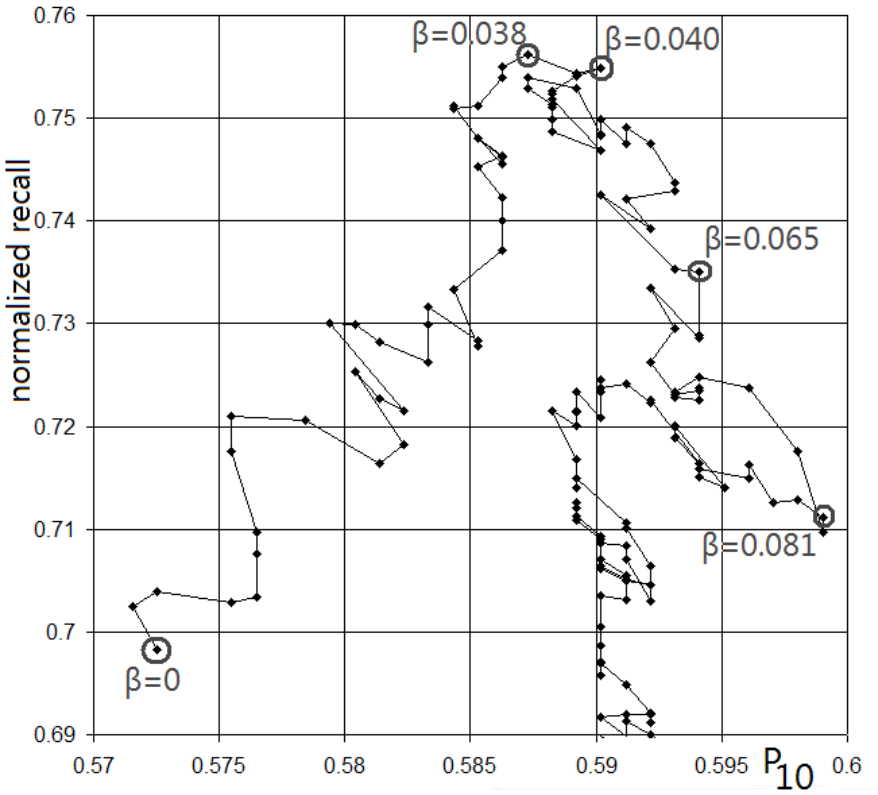

Figure 18. Choosing best □factor on recall/precision graph

9. CONCLUSIONS

In this paper, we analyzed Wikipedia in terms of using its resources in Information Retrieval. Various statistical and structural features were discussed. Those features enable applying various types of search, e.g. category search, multilingual search. The link structure of Wikipedia was also investigated.

The main contribution made by this paper is that we have shown that combining language modeling techniques and link analysis significantly improves both precision and recall relative to either language modeling or link analysis alone. We performed a user study based on real user queries. The improvement on both, precision and recall, was shown to be statistically significant for various system settings at the same time. Usually an increase of one measure causes a decrease of the second one which did not take place in this situation. The proposed method does not need complex calculations to be done during retrieval; all complex calculations may be done in advance and all information may be stored in the index file.

The methods used in this paper may be improved. The PageRank algorithm presents a very simple analysis of links structure. One can suppose that more detailed methods would improve future outcomes and then allow comparing other results to the ones presented in this paper. This includes utilizing category information, language dependencies and more sophisticated features which result from "semantic-web"-like Wikipedia structure.

10. REFERENCES

[1] Abdou, S., Savoy, J., Ruck, P. 2005. Evaluation of Stemming, Query Expansion and Manual Indexing Approaches for the Genomic Task. In *The Fourtheenth Text REtrieval Conference (TREC 2005) Proceedings*, Gaithersburg, Maryland, USA, November 2005

[2] Bellomi F., Bonato R. 2005. Network Analysis for Wikipedia. *In Proc. of Wikimania 2005*. Frankfurt, Germany. Wikimedia Foundation.

[3] Brin S., Page L. 1998. The anatomy of a large-scale hypertextual Web search engine. *In Proc. of the 7th International WWW Conference*. Computer Networks and ISDN Systems vol. 30, iss. 1-7, pp. 107-117. Brisbane.

[4] Denoyer, L. and Gallinari, P. 2006. The Wikipedia XML corpus. *SIGIR Forum* 40, 1 (Jun. 2006), 64-69.

[5] Fissaha S., Rijke M. de. 2006. Exploratory Search in Wikipedia. *In Proc. of the ACM SIGIR 2006 Workshop on Evaluating Exploratory Search Systems.* Seattle, pp. 35-38.

[6] Jansen, B. J., Spink, A., and Saracevic, T. 2000. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manage.* 36, 2, pp 207-227.
[7] Kimelfeld B., Kovacs E., Sagiv Y., Yahav D. 2006. Using Language Models and the HITS Algorithm for XML Retrieval. *In Proceedings of the Initiative for the Evaluation of XML Retrieval.* Wadern, Germany. pp. 253-260.
[8] Kraaij W., Westerveld T., Hiemstra D. 2002. The Importance of Prior Probabilities for Entry Page Search. *In Proc. of the 25th International ACM SIGIR Conference on Research and Development in Information Retrieval.* Tampere, Finland.
[9] Ponte J. M., Croft W. B. 1998. A Language Modeling Approach to Information Retrieval. *In Proc. of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne. 275-281.
[10] Richardson M., Domingos P. 2002. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in Pagerank. *Advances in Neural Information Processing Systems 14*. pp.1441-1448. Cambridge, MA: MIT Press.
[11] Robertson S. E., Walker S. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *In Proc. of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 345–354. Springer-Verlag.
[12] Sigurbjornsson B., Kamps J., Rijke M. de. 2006. Focused Access to Wikipedia. *In Proc. of the Dutch-Belgian Information Retrieval Workshop.* Delft. pp. 73-79.
[13] Sparck Jones K., Walker S., Robertson S.E. 2000. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management 36*. pp. 809-840.
[14] Voelkel M., Kroetzsch M., Vrandecic D., Haller H., Studer R. 2006. Semantic Wikipedia. *In Proc.of the 15th International WWW Conference.* pp. 585-594. Edinburgh.
[15] Voss, J. 2005. Measuring Wikipedia. In *Proc. International Conference of the International Society for Scientometrics and Informetrics : 10th*. Stockholm, Sweden.