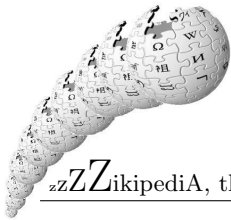


Internet Information
zzZikipediA, the zoomable WikipediA.

Ork de Rooij, 9901655
Xandra van de Putte, 0008516
Lior Kuijer, 0177938
Rogier Koppejan, 9912665
Julian Kooij, 0105864
Jelle Kastelein, 0026549

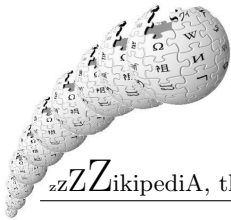
Universiteit van Amsterdam

2nd June 2006



Contents

1	Introduction	2
2	System Architecture	2
2.1	Mediawiki	2
2.2	Searching Wikipedia	3
3	Ins & Outs	3
3.1	Zooming In	3
3.1.1	Documents and Sections	3
3.1.2	Interpreting Results	3
3.1.3	Known Problems	4
3.2	Zooming In & Out	4
3.2.1	Keyword Based Clustering	4
3.2.2	Salient phrases	5
3.2.3	HITS: Authorities and Hubs	6
3.3	Zooming out	6
3.3.1	Co-citations	7
3.4	Alternative Techniques	8
4	Evaluation	8
4.1	Search results	8
4.2	Co-citations & Salient phrases	9
4.3	Keyword Based Clustering	9
5	Experiments & Results	10
5.1	Search results	10
5.2	Co-citations & Salient phrases	11
5.2.1	Co-citations	11
5.2.2	Salient phrases	11
5.3	Keyword Based Clustering	12
6	Discussion & Conclusions	13
7	Future Work	13



1 Introduction

What is Wikipedia¹? Well, let's ask them:

“Wikipedia (IPA: /,wɪkɪˈpiːdi.ə/, /,wɪki-/ , or /,wɪkə-/) is an international Web-based free-content encyclopedia. It exists as a wiki, a type of website that allows visitors to edit its content; the word Wikipedia itself is a portmanteau of wiki and encyclopedia and is often abbreviated to WP by its users. Wikipedia is written collaboratively by volunteers, allowing most articles to be changed by anyone with access to a computer, web browser and Internet connection.” (...)

Wikipedia on Wikipedia

Wikipedia is probably the most popular encyclopedia on the web. It is free, and it holds many articles on a wide variety of different subjects. It has active versions in over 132 languages. The English section alone comprises over 1.1 million articles, and this number is still growing. To search for a specific detail of a specific topic by hand would be like trying to find a needle in a haystack. On the bright side, there is a number of helpful search features available on the page itself. Unfortunately, the current best way to find articles on Wikipedia is still through an external search engine, such as Google. Obviously, this cries for an improved search strategy. The goal of this project was therefore to extend or enhance the basic search facilities in a significant way, providing more user friendly ways to navigate the Wikipedia domain. More specifically, we would like to provide two facilities that we deem important.

First of all, the search results returned by Wikipedia may contain rather large segments of text, only parts of which will be relevant to a specific query. This is fine if we are just looking for generic knowledge on a broad topic, but if we are looking for specific details, we may need to “zoom in” on certain subsections. This is the first of our goals: We wish to enhance a basic search engine by breaking the articles up into smaller subsections, which should be much more specific.

In contrast, if a user finds a page about something she's interested in, and aims to find out about

¹www.wikipedia.org

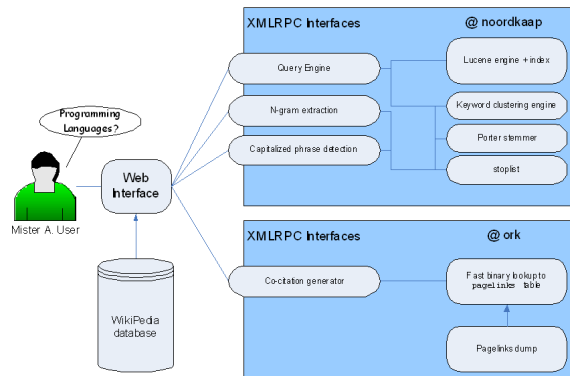


Figure 1: The Zikipedia system.

closely related topics, she may be more interested in the exact opposite. In this case, it would be useful to have a “find similar”, or “zoom out”, feature by which related content can be found. This information should be closely related to the current article, without producing a duplicate of the information that we already had.

We would like to stress in advance that our main goal is not *necessarily* to find an improved recall or precision measure over the original Wikipedia search engine. Rather, it is to test the viability of several approaches to making the search facilities more user friendly, which should be tested independently of the search engine itself.

2 System Architecture

In this section the framework architecture used for this project, affectionately duped “Zikipedia”, will be briefly described. Figure 1 shows the main components of our system.

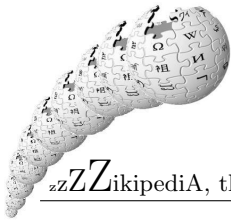
2.1 Mediawiki

In order to have full control over the Wikipedia database, and to be able to extend the current Wikipedia search facilities, a local Wikipedia mirror was created. Running on the system described in table 1, MediaWiki² (which is a wiki engine developed for and used by Wikipedia) was installed. MediaWiki requires an Apache³ web server, PHP⁴

²<http://www.mediawiki.org/wiki/MediaWiki>

³<http://www.apache.org>

⁴<http://www.php.net>



Processor	Intel Pentium 4 3.00GHz HT, 800 MHz FSB, 2 MB L2 Cache
Memory	2 x 1024 MB DDR 3200 (400 MHz) Dual Channel
OS	Fedora Core 3, Kernel 2.6.12, gcc 3.4.4

Table 1: Specifications of the server running the local wikipedia mirror.

and a MySQL⁵ database server, the latter of which was already present on our testing system. Once all prerequisites were met, a Wikipedia database dump was loaded into the MediaWiki engine and the project was ready for development.

2.2 Searching Wikipedia

A vector model search engine was used with a TF.IDF⁶ scoring scheme, and it has proven successful for our research needs. We did not attempt to write a new search engine from scratch, but instead used the popular Lucene⁷ JAVA API to implement the basic functionality needed. The API takes care of creating an index, maintaining it (on disk and in memory), and retrieving documents for a given query.

The implementation of the engine was straightforward, but effective, although several improvements can be made. Less straightforward, however, is the way Wikipedia articles are being treated by the engine.

3 Ins & Outs

This section will concentrate on the techniques we employed to serve two ends: First, several ways of zooming in on a specific domain for a target query will be proposed. Next, techniques will be introduced that may both narrow and broaden the

⁵<http://www.mysql.com>

⁶TF.IDF is a commonly used retrieval model. TF stands for term frequency, the frequency of a term in a document. IDF stands for inverted document frequency, the number of documents in which this term appears. The IDF is used to measure the importance (weight) of a term for retrieval purposes. Intuitively, the lower the IDF, the more specific a term will be, and the more likely it is that the term can be used to effectively distinguish two articles from one another.

⁷<http://lucene.apache.org>

search scope. Finally, a few techniques will be described that have mostly a zooming-out purpose.

3.1 Zooming In

The first of the extensions that we concentrated on was to make the search engine more specific. Particularly, rather than returning a full Wikipedia article, which might be very lengthy, we return the relevant subsections of such an article, given the query.

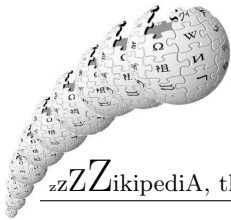
3.1.1 Documents and Sections

Most search engines treat each Wikipedia documents as the atomic unit to be stored in the index, where each full article is associated with the terms in its content. To meet the zooming in requirement adopted in the project, on the other hand, it is a necessity to subdivide Wikipedia documents in smaller units. This will enable the user to directly enter a document to the part that has the user's interest, even if the other sections of that document are less relevant. Luckily, the documents contain a tree structured section layout, much the same as how it would be represented in a table of contents. The sections provide a logical subdivision and, from the perspective of the search engine, each such *section* will represent a retrievable document.

Per section, several attributes are kept in the index, including its title and its depth (i.e. the depth within the tree structure). Also, a reference ID of the Wikipedia document is stored, as this enables the system to determine if retrieved sections come from the same source. The *root* section represents the content of the document that occur before any subsection is started, and takes the title of the document itself. It always has depth level zero. In some cases, the root is identified to be a *redirect* to a different document, i.e. its title is used as an alternative name for the referenced subject. Redirect roots have an added reference attribute in the index.

3.1.2 Interpreting Results

The retrieved documents for a query initially consist of sections from many different Wikipedia articles, as well as redirects, each with an associated TF.IDF score. The results are then restructured



into a new list, such that now a search hit actually represents a Wikipedia article, with relevant subsections indicated right below it, in order of relevance score.

First, each document associated with a section or referenced by a redirect was added to this final list. Then all sections and redirects were grouped over, and assigned to those documents. Finally, the relevance score for an article was computed as the maximum of the scores of its related subsections.

The following figure illustrates the way results are returned following the document-subsection search:

1. List of cheeses (1.0)
 - Cheese by place of origin (... etc ...)(more)*
 - Egypt (1.0)
 - South Africa (1.0)
 - Norway (0.92807776)
 - Belgium (0.9185587)
 - ... etc ...

The number of “...”s indicates the depth of the nesting of each section in the root document. The user can go straight to any subsection of choice, or to the main document.

3.1.3 Known Problems

Unfortunately, there are some problems with the implementation used. For one, only the terms in the content are searched, but not the title of the section or article (which should be very descriptive!). This has as a side effect that redirects aren't returned either, since they have no content. Another shortcoming is that terms are only used to index the section that directly contains them, but not sections higher up the layout hierarchy. Therefore, the root document generally will not score as well as it should on a *generic* subject, as its subsections may be too specialized. If we assume that the root contains the most general, diverse terms concerning a subject, and subsections the more specific terms, than we could combine the two, so that the retrieval results could better match the users requirement.

3.2 Zooming In & Out

The features discussed in this next section may be used both to broaden a search or to narrow it, depending on the user's preference.

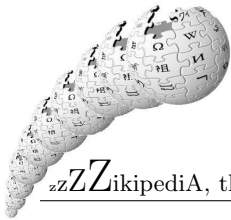
3.2.1 Keyword Based Clustering

When a list of articles is found for an ambiguous search term, or more generally, when a term relates to several different topics, one would like to group similar results together for visual clarity. Wikipedia has some predefined categories, but they are very broad, and not tailored to specific queries. During this project, we developed a simple clustering algorithm based on the titles of documents and their subsections as returned by the search engine. This algorithm finds common terms in the document titles, and suggests them as keywords relating to the current search query. We only use the titles, and not the body of text. This is more efficient, and it also allows for an assumption that we could not make when using the text body; we assume that all non-stopword terms in titles have a significant meaning which is somehow related to the current search term. In this, we disregard the IDF measure of the terms, and assume that all terms are equally relevant. If we were to use the text body, this assumption would presume far too much, and the data would be extremely noisy. But because the titles are meant to summarize a section in the shortest possible way, there is relatively little data, and the terms used can be assumed to be significant.

The algorithm starts out by removing the stopwords in the titles, and stemming all remaining words using Porter's stemming algorithm. Under the (sometimes incorrect⁸) assumption that all words with the same stem represent semantically related concepts, all articles which contain a certain stemmed term are assigned to the cluster identified by that stem (keeping track of the words that generated this stem, and the frequencies of both the stem and the individual words).

Let the number of elements in the set of stemmed keywords, the *identification set* S_C of C , be denoted by $|S_C|$, and similarly, let $|C|$ be the number

⁸A known problem with Porter's stemming algorithm is that it may assign words with a different semantic meaning the same stem, e.g.: “Universe” and “University” will both be stemmed as “Univers”.



of articles in C . The cluster relevance, $R(C)$, is calculated by

$$R(C) = \frac{1}{|S_C|} \sum_{s \in S_C} f(s) \times \frac{1}{|C|} \sum_{i=1}^{|C|} rel(C_i),$$

with $f(s)$ the frequency of stem s identifying the cluster C , measured over all titles of all search results, and $rel(C_i)$ the relevance score assigned to article i in C by the search engine. Assuming, as we did, that all the stems in S are related to the search term, this gives an ordering by cluster size (implicit in the average frequency of the elements of S), and query relevance (directly by averaging over the relevance scores of all articles in the cluster).

The algorithm then proceeds by looking for clusters that are similar according to a predefined measure, and merging them until no more cluster pairs meet the merging criterium. The difference between two clusters A and B is defined as

$$D(A, B) = D(A|B) + D(B|A),$$

where $D(A|B) = |A - (A \cap B)|$, the number of elements of A not found in B , and similarly $D(B|A) = |B - (B \cap A)|$. The criteria for the merging of the clusters then consists of the following conditions:

1. $D(A|B) \leq \theta$, and
2. $D(B|A) \leq \theta$, and
3. $D(A, B) < \frac{|A|+|B|}{2}$,

where θ is a predefined maximum merging distance threshold, which is decreased by 1 after every complete merging iteration, to avoid a hypothetical situation where clusters continue merging until we are left with only a single cluster. The score for a merged cluster is simply the sum of the scores of its parents⁹, A and B . After merging, the clusters are subsequently ordered on the basis of their score. The best m clusters and the keywords associated with them are returned (m was set to 20 in our experiments).

All clusters that contained only one article were discarded before the first merging iteration because they can never be merged with other clusters, due

⁹We use parent here in the literal sense; The parents of a merged cluster are the two clusters from which it was derived.

to the third merging constraint. This means that some articles that appear in the results set are not in any of the clusters. An advantage of this is that it functions as a noise filter, removing documents with weird titles, which are arguably suspicious. A disadvantage is that articles with non-descriptive titles will be lost.

3.2.2 Salient phrases

Another approach to zooming in on (and perhaps out from) a topic is by finding a number of suggestions for further, broader or perhaps more refined searches. The keywords from the above clustering algorithm can be used for such purposes, but an even better approach will be to find common phrases, or identify named entities. The latter are often very specific in nature. The former may either be very specific (e.g.: “Boltzmann Machine”), or relatively broad (e.g.: “Machine Learning”). Unfortunately, for this purpose, the text in the titles alone will contain insufficient data to base our estimates on. Instead, the full body of a section is parsed (on the fly) to find all prominent N -grams (not necessarily with a fixed N) and capitalized phrases. This is time and resource consuming, and so care should be taken to avoid long search delays.

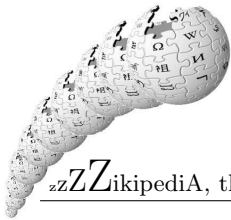
Currently, the N -gram extraction works as follows. First, all N -grams are extracted from the text, with N ranging from 2 to 5. Then, all N -grams that meet the following criteria will be removed:

- *Frequency* < 2 , or
- *Length* < 2 , after all stopwords are removed that occur before the first non-stopword, or after the last non-stopword, or
- N -Grams that are a substring of a different N -Gram phrase of a higher N , with the same or a higher frequency¹⁰.

Finally, the results are ordered by occurrence, and the first n returned (in our experiments, $n = 20$).

Similarly, the capitalized phrase finder searches for all (sequences of) words with the first character capitalized. Hopefully, several relevant named

¹⁰Obviously, an N -Gram of a higher N should never have a frequency higher than one of a lower N which is contained in it.



entities will be found in this manner. Sequences of words in which two capitalized words co-occur within two words of one another, with no delimiters in between, are also grouped together¹¹. Terms following a delimiter are ignored unless the length of the phrase in which they occur ≤ 3 . This is done because we regard a comma as a delimiter, which means that many named entities would be overlooked if the capitalized words after each comma in a comma separated list would be ignored. With these criteria, some noise should be expected (e.g.: when a summation of named entities ends with a sequence like “John and Mary”, this is extracted as one entity), as well as some missed hits (e.g.: when a named entity occurs at the start of the phrase, and the phrase has more than 3 terms, this entity is ignored completely), but the majority of named entities can be expected to be found in this way. As with the N -Gram finder, all capitalized phrases that consist only of stopwords are removed, and the results are ranked by their frequency. The first 8 phrases are output on the search results page (N -Gram and capitalized phrases together, ranked in order of occurrence), as links which can be clicked in order to perform a search on the terms in this phrase.

3.2.3 HITS: Authorities and Hubs

The idea behind the HITS¹² algorithm [4] was used with the goal of allowing the re-ordering of the returned document set for a given query in a general-to-specific or specific-to-general manner. Such an ordering of the results can assist the user in case he or she wishes for more general pages, or alternatively more specific pages to be shown first. Such a re-ordering can be realized by the computation of the *authority* and *hub* scores for the documents in the returned set for a given query.

A simplifying assumption was made. Namely, it is assumed that the documents in Wikipedia collection may all be considered high quality pages. If this is the case, it can be claimed that no recursive computation is needed in order to achieve a reliable score for page importance, or in this case, a reliable score which reflects the level of *author-*

¹¹To see why this is, think of names which include a prepositional phrase such as “Joan of Arc” or “Universiteit van Amsterdam”.

¹²HITS stands for “Hyperlink Induced Topic Search”.

ity and of *hub* of a document in the returned set for a specific query. The implementation performs a one-step computation of the scores. These can be computed in realtime for a given query. The *authority* score defines the specificity of a document, while its *hub* score defines its generality. A high *hub* score suggests the document is somewhat more general, and has a relatively large number of outgoing links to other documents that cover specific subjects, which are covered only briefly within the hub page. A high *authority* score suggests the document thoroughly covers a specific subject and therefore does not require outgoing links to other specific subjects, since the latter are not discussed in the document. The exact one-step computation is given below. The computation of in- and outgoing links is performed on the entire document set.

$$\text{authority}(d_i) = \frac{\text{inlinks}(d_i)}{\sum_{d_j \in D_q} \text{inlinks}(d_j)}$$
$$\text{hub}(d_i) = \frac{\text{outlinks}(d_i)}{\sum_{d_j \in D_q} \text{outlinks}(d_j)}$$

with

$$\text{inlinks}(d_i) = \sum_{j=1}^n A_{ji} \quad \text{where } i \neq j$$

and

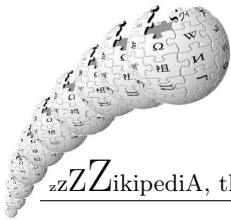
$$\text{outlinks}(d_i) = \sum_{j=1}^n A_{ij} \quad \text{where } i \neq j$$

$$A_{ij} = \begin{cases} 1 & \text{if } d_i \text{ links to } d_j \\ 0 & \text{otherwise} \end{cases}$$

At this point, only the HITS hubs and authority scores are output for each search result, but implementing a reordering by means of these scores should be relatively straightforward.

3.3 Zooming out

Rather than zooming in on specific subsections, the user may instead wish to broaden his search to venture into related and encompassing topics. Wikipedia is particularly suited for such an approach, as many of its users seem to surf from one page to another in search of information. But there will be little interest in finding two related pages if they carry exactly the same content of information. The challenge, then, is to find related pages that are not near duplicates of the current page.



3.3.1 Co-citations

One way to do this is to use a content related method, such as Latent Semantic Indexing (see below), to find pages that have a highly similar representation in term vector space. Unfortunately, using term vector space based similarity measures may lead to finding pages that have almost identical content to the current page, which is usually not what we want; we want to find pages that give different information about the same topic, so that our coverage increases.

Another, completely different approach to finding related content, which is related to the HITS algorithm described above, is to look at articles that are cited (linked to) on the current page, and expand a graph of outgoing (or incoming) links, which will be referred to here as a citation graph, to find locally related pages. But this way, only pages that have direct paths from one to the other can be found, and disconnected link subgraphs will be completely ignored, even if the pages in such subgraphs make for a better match than the ones that are linked to directly. Furthermore, if only outgoing links are considered, no new information is really added, since the links are already mentioned in the contents of the article. Instead, we may do better by expanding this approach by the notion of co-citations; that is, pages that are cited together by the same source.

In [1, ch. 7], Soumen Chakrabarti discusses the co-citation concept. This notion of relevance has its foundations in social sciences. Intuitively, it is based on the idea that those articles that are (more or less often) cited by the same source, may well be related in content. For example, an article discussing Darwin may cite an article on evolution and one on genetics. Although the genetics article may make no mention of evolution specifically, the fact that both are cited together by the Darwin article gives some support that the two topics are somehow implicitly linked. It seems likely that, the more often two articles are seen cited together, the stronger this support becomes. At the same time, articles with many citations will implicitly be rewarded by a prestige notion based on citation frequency, because articles that are cited often will be more likely to be co-cited with other articles.

To assist the user with navigation, the 5 highest ranked co-citations are output for each result on

the search page. An “easy access” list of the 10 highest ranked co-citations was also added to the top right of each article, so that the user can surf from one article to another without having to do any additional searching.

A note should be made at this point of a potential problem that deserves some attention. If a page is cited extremely often (say, on every page) it will, with high likelihood, always have a higher co-citation score than less popular pages. If, for instance, every article in Wikipedia contains a link to the front page, this page will probably have the highest co-citation score for any page in the database. Obviously, such a page will not be particularly relevant to the current page. Therefore, either these pages must be filtered out a priori, or the number of co-citations should be weighted somehow. So, if we are on page a , and wish to find the page b for which the normalized co-citation score between a and b is highest, the best match is found by:

$$\operatorname{argmax}_b C(b|a) = \operatorname{argmax}_b \frac{C(a,b)}{C(b)},$$

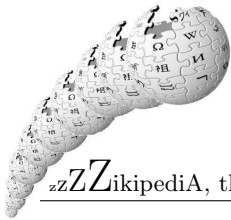
However, a downside of this weighting is that we lose the degree of authority that an often cited page has¹³.

Another downside is that, in order to calculate the normalization scores, the overhead will be much larger than the original cost of calculating the unweighted co-citation score. If m is taken to be the average number of outgoing links over n articles, then the number of co-citations that will be extracted will be:

$$N_C = n \times \binom{m}{2} = n \times \frac{m!}{2 \times (m-2)!} = \frac{1}{2} nm(m-1)$$

The English version of Wikipedia contains roughly 1.1 million articles, so obviously, this number will be quite large. We anticipated that we would find roughly 10 links per page on average, so that we would have $1.1M \times \binom{10}{2} = 49.5M$ co-citations, for each of which $C(b)$ would need to be calculated. We now estimate that the average number of outgoing links on a page is actually closer

¹³In fact, since many internal links in Wikipedia article contents will be likely to be highly specific, and contain much informational content, one could argue that the unweighted version may actually be more suited.



to 40, so that the article section alone contains about $1.1M \times \binom{40}{2} = 1.1M \times 780 = 858M$ co-citations¹⁴. This means that calculating the normalized co-citation measure between each pair of pages would require calculating the total number of co-citations for each of the (on average) 780 co-cited pages to retrieve the normalization factors. In contrast, when calculating the un-normalized co-citation score, we only need to retrieve the set of co-citations C with a and count the frequency of each co-citation $c \in C$ with a . Unfortunately, this means that, due to computational constraints, we were unable to test the effects of normalizing the co-citation score. Currently, the un-normalized co-citations scores are calculated directly from the Wikipedia citations database dump.

But the problem may not be very severe in a domain such as Wikipedia. We argue that, within a Wikipedia-like domain, since only links between articles are considered, and no external links, it is safe to assume that most co-citations will be with pages related to the current page, so that the amount of noise will be within tolerable limits. Our results support this assumption, as many of the co-citations found appear to be highly relevant, even without normalization. However, it does mean that more general pages will be preferred over more specific ones.

3.4 Alternative Techniques

During the viability assessments of various known techniques, some techniques were also encountered which proved unsuitable for the Wikipedia data set.

As an alternative method for indexing Wikipedia, Latent Semantic Indexing (or LSI [2]) was investigated. This technique reduces the original $Term \times Document$ matrix into lower dimensional $T \times k$ and $k \times D$ matrices, with $k \ll \min(T, D)$, in order to capture available latent semantics in the data set. In principle, this finds documents which discuss the same subjects, but using different words, and places them together in the lower dimensional space. When used within a search engine this works as follows. A search query would be transformed into a k dimensional vector, which is then compared

¹⁴Needless to say, when the 1.3GB 'pagelinks' database (holding the citation index) yielded a 120GB raw text file consisting only of co-citations, we were quite surprised.

with the k dimensional representations of all the documents. The nearest documents would be the articles which are closest to the original query, and therefore good candidates for the retrieved document set. This would include articles which contain none of the original query words, but do revolve around the requested subject. Also, clusters of related articles can easily be created from the $k \times D$ matrix by using similarity search.

However, LSI was not up to the task. First, we found that the creation of the lower dimensional matrices is not possible using regular Singular Value Decomposition [2] for a data set the size of Wikipedia, so alternative techniques were required. One such technique was Semi Discrete Decomposition [5] which resembles SVD, but in signed binary space. Using SDD, it was possible to construct lower dimensional matrices for a reasonable k . However, during various iterations with different parameters and settings for k , it was impossible to create matrices which gave good results when used for querying. Also, querying with LSI does not scale well. After transformation, every online query results in cosine distance calculations for 1.1 million articles times k dimensions, which is impractical to say the least.

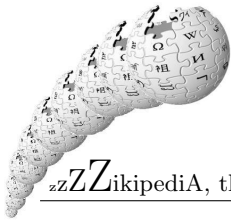
For these reasons, LSI was abandoned in favor of using Lucene as the document index, and we focuss on co-citation analysis for suggestions of related articles for each document.

4 Evaluation

This section describes the evaluation methods used in this project, starting with the evaluation of the search engine as such, followed by evaluation methods for the results of a number of the individual zooming components. At this point, a working version of the HITS reordering scheme is still lacking, and as such, we were unable to evaluate it.

4.1 Search results

Although the search engine in use was not our main focus, many of our system's components will be influenced by its performance. Therefore, before continuing on to the evaluation of its extensions, we provide an evaluation of the engine itself.



Two evaluation measures are used to evaluate the search results: average precision¹⁵ and precision at rank n ¹⁶, both for the first ten articles in the result list. This has been done for fifty queries chosen from a set of queries which has been logged by a search engine. To determine the relevant articles for each of these queries, Wikipedia was searched for relevant articles, mostly by using Google¹⁷, Wikiii¹⁸ and Wikipedia itself. Because it is impractical to find all relevant articles in Wikipedia, recall cannot be determined precisely. Therefore, this evaluation must be seen as a comparison between this search engine and the search engines named above. Also the search engine of the other Wikipedia team (group 1B) has been evaluated this way, so a good comparison with their search engine can be made.

4.2 Co-citations & Salient phrases

To evaluate the “see also” and “search also” suggestions that were returned for a search result, which correspond to co-cited articles and prominent phrase search suggestions respectively, a different evaluation metric is needed. We chose to evaluate these features independently from the search engine, since this evaluates each component in its own right. The evaluation method for each is described below.

Recall is not as important as precision here, since we do not need an exhaustive listing of all available alternative searches or all related articles. Such a list would be too large to be practical for most articles. Therefore, only a measure of precision is considered in the evaluation of these components.

Not all results will return suggestions for further searches or related articles. Here, for each of the three methods, the precision is only examined for articles that return at least one suggestion based on that method. Each method is evaluated independently of the two others, except that the same queries were used for all.

¹⁵The average precision is the sum over the precision at a given rank times the relevancy (0 if not relevant, 1 otherwise) of all the articles retrieved, divided by the number of relevant articles.

¹⁶Precision at rank n , or $p@n$, is the number of relevant documents with rank less than or equal to n divided by n .

¹⁷www.google.com

¹⁸<http://berk.science.uva.nl:8080/wikiii>

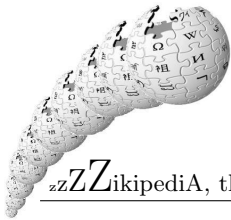
For a number of queries chosen from different domains (see table 2), the first 5 results that have co-citation-based suggestions are recorded. Over these results, the number of relevant co-citations is summed, as well as the total number of suggestions. The resulting sums are summed over all queries. Finally the total relevant suggestions are divided against the total number of suggestions to get the average precision over all queries.

The same measure is used for the number of relevant N -Grams, over the first 5 search results that contain N -Gram based suggestions, and for the number of relevant Capitalized Phrase based search suggestions, averaged over the first 5 hits that produced suggestions based on capitalized phrase extraction.

The relevance of each phrase or co-citation is determined by hand by a member of our team. All six team members evaluated a small number of results to rule out personal bias effects. The relevance is not measured with respect to the query that was entered, but to the article for which the co-citation or phrase was a suggested alternative. When in doubt, a result is duped irrelevant (as examining each suggestion would make the workload unrealistically high). This leaves us with a conservative estimate of the precision of each of the three tools.

4.3 Keyword Based Clustering

In order to evaluate our clustering scheme, the average purity of the first 5 clusters was measured (by hand), over the same queries that were used to evaluate the “see also” and “search also” results. The purity of a cluster is the number of relevant documents in a cluster, divided by the total number of articles in that cluster. A document is said to be relevant if it is part of the largest group of articles within the cluster under evaluation which share a common topic. The labels that were suggested for the clusters were disregarded in choosing this topic, but the keyword purity of each label was also measured after choosing the main topic. A keyword in a label of a cluster is said to be relevant if it corresponds to a term which is a part of the description of the chosen topic for a cluster whose purity was just examined. For instance, if we choose the topic “machine learning” for a cluster, keywords “machine” and “learning” are both said to be relevant. Obviously, keyword purity will be



Query	P_{Co-cit}	$P_{Cap.Phr.}$	$P_{N-Gr.}$
Cheese	0.8	0.625	0.72
Information Retrieval	0.76	0.714	0.955
Gentoo Linux	0.88	0.923	0.737
Big Bang	0.903	0.643	0.929
Horror Movies	0.76	0.846	0.72
Star Wars	0.88	0.667	0.75
Battlestar Galactica	0.72	0.391	0.647
Guitar	0.44	0.852	0.867
Heavy Metal Music	0.84	0.824	0.8
Average	0.779	0.7	0.786

Table 2: Suggestion tool Precision scores.

the matching sections, which are not indexed by the Google or Wikipedia engines to which we are making a comparison.

Given the poor performance of the engine (which was, after all, *not* the main focus of our efforts), we can expect any component of the system which makes use of these results to be affected. Particularly, the clustering algorithm based on titles will perform worse, as it bases its cluster scores on the average relevance scores of the documents in each cluster as assigned by the search engine. We ask that the reader bear this in mind when continuing reading.

5.2 Co-citations & Salient phrases

Table 2 summarizes the results for the three “find similar” tools; co-citation-based related article suggestions, and capitalized phrase- and N -Gram-based search suggestions. The average precision, P , measured as described above, is shown for each measure, for each of the 9 domains examined.

5.2.1 Co-citations

The conservative estimates of the precision of the algorithms so far are very encouraging. Particularly, the co-citation-based suggestions are potentially very useful because they provide direct links to another Wikipedia article. For example, take the article “Neural Networks”. The suggestions given as the top 5 co-citations are:

See also: *Artificial intelligence, Computer science, Genetic algorithm, Brain, Cognitive science*

For those of us familiar with (and interested in) neural nets, this should be a very interesting collection of pages.

As anticipated, most of the noise in these results is caused by the fact that no normalization was used. This is also the main reason for the instability of some of the results. For instance, most movies made in 2005 will be co-cited (if at all) with a page about the year 2005. This is not a very useful link, and therefore it was tagged as irrelevant. Applying normalization as described above should remove most of such articles. Note also that the results for “Neural Network”, though striking, are all quite general in nature. This seems to be a general trend in the co-citation results (which is why we have put it forward here as a means of zooming out). We suspect that, should normalization be applied, the results would also become much more specifically tailored to the article under investigation, as more specific relevant articles would be co-cited with *other* pages less often (or, in other words, relatively more often with the current page of interest), and would thus be likely to have a higher normalized score.

5.2.2 Salient phrases

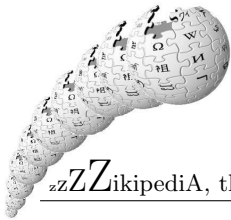
Even with minimal noise removal, the capitalized phrase and N -Gram search suggestions already provide some very useful results. To give another example, for the article “Artificial Neural Network”, we find the following suggestions:

Search also: *Optimization, pattern recognition, cost function, machine learning, unsupervised learning, gradient descent, reinforcement learning, Geoff Hinton*

Note that the words without capitalization were found using N -Gram search, whereas the words with the first character capitalized were found using our named capitalized phrase finder. Note how the first result here is actually a fluke, even though it can be argued to be relevant, as it is not a named entity. The N -Gram results in this example are very much related to neural networks.

Another example illustrates the point of capitalized phrase search, however. For the article “Metalucifer”, a Japanese heavy metal band, we find:

Search also: *Gore, Sabbath, heavy metal,*



metal inquisitor, Death, Bill Andrews, Samm

All capitalized phrases found here are either metal bands or band members, and all were extracted by our named entity recognizer. So it seems that each of these two approaches has strong points which can be useful in different domains. In our experience, however, the N -Gram suggestions are often more interesting, since capitalized phrases are often found to be countries or specific person names (e.g.: researchers), whereas the N -Grams comprise important terms related to the article in question. Furthermore, important person names may also be found using N -Grams, as long as they occur more than once and contain more than 2 terms¹⁹.

However, much improvement may still be gained by normalizing the scores of the phrases by an inverted document frequency measure. This may eliminate common N -Gram phrases such as “important events”, which are likely to be used multiple times in a single section, but are not useful as further search suggestions. Again, much of the instability seen in table 2 is the result of lack of normalization.

It should be stressed that the goal here was not to provide the highest possible levels of precision, but rather to give an indication of whether or not these tools could be successfully applied to a domain such as Wikipedia. With that in mind, the results certainly suggest that more investigation would be appropriate.

5.3 Keyword Based Clustering

As is seen in table 3, the purity of the clusters appears to be quite high. This comes as no surprise, since clustering on words in titles is bound to be quite accurate. Not all of the clusters are equally useable, but, taking into account the comparatively small amount of data, it works quite well. For instance, querying for “neural networks” gives us the cluster:

Cluster Keywords:

- **function, functions, sigmoid**

¹⁹This has as an additional advantage that first names like “John” are filtered, whereas full names like “John Doe” may be captured.

Query	$P_{Cluster}$	$P_{Keyword}$
Cheese	0.786	0.875
Information Retrieval	0.8	0.769
Gentoo Linux	0.857	0.8
Big Bang	0.784	0.818
Horror Movies	0.885	1.0
Star Wars	0.875	0.889
Battlestar Galactica	0.933	0.909
Guitar	0.967	0.6
Heavy Metal Music	0.92	0.889
Average	0.882	0.841

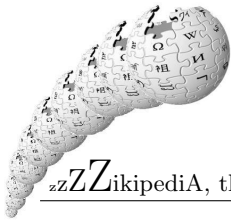
Table 3: Cluster Purity scores.

Articles in this Cluster:

- *Evolving classification function* → *See also*
- *Sigmoid function* → *See also*
- *Sigmoid function* → *Sigmoid functions in neural networks*
- *Radial basis function* → *Overview*
- *Radial basis function* → *Bibliography*

which is a cluster about activation functions used in neural networks of several types.

An interesting detail is again found in the effect of term weighting. Recall that it was noted earlier that one could argue that, since the keywords are not weighted by some IDF-like measure, a lot of information is lost about which words are important. As it turns out, however, after removing all obvious stopwords, most of the terms in the titles are useful for clustering purposes. In fact, we may go as far as to turn the tables and pose that, in order to get any kind of generalization, the more common terms are essential. For instance, when we are trying to find two clusters, one about the island of Java, and one about the JAVA programming language, we have more use for the much more common words “island” and “volcano”, than for a word such as “Pulosari” (a volcano situated in western Java). The latter term may be used to create a cluster separate from our JAVA programming language clusters, but it will be very specific. So, the more common terms are needed to generalize this type of clustering. When trying to implement an IDF weighted version of the same algorithm, it



was found that this produced only clusters of very small length, with barely any generalization. This is, in fact, still a major problem for the algorithm, as even with the common terms the titles already tend to be quite specific.

The biggest problem, at this point, is one of ambiguity. Consider again the case of searching for the term “Java”. We will get results for the island of Java, which has a history, as well as for the programming language JAVA, which has a version history. Obviously, these concepts will be clustered together on the basis of the word “history”.

A final result may be worth noting. Because the merged clusters always have higher scores than their smaller parents, the larger, more general clusters are usually ranked highest, and output at the top of a page, whereas the smaller, more specific clusters are reported near the bottom of the rank-ordered cluster list. Note how this conveniently relates to the idea of a zooming Wikipedia.

6 Discussion & Conclusions

During this project, we have thought up, implemented, explored and evaluated a large number of ideas, each with the intent of making the exploration of Wikipedia a more comfortable one. The most important ideas presented are (in no particular order):

- Indexing subsections, rather than documents to improve the ease of search for the user.
- On-the-fly clustering of results based on keywords in titles.
- Reordering of results using the HITS algorithm to achieve a general to specific ranking of pages.
- Finding related pages based on co-citation scores.
- Suggesting alternate searches using salient phrases based on N -Grams and capitalized phrases.

It is unfortunate that we have had to evaluate the components of our system using a search engine with relatively poor performance, but when evaluating each of the components in their own right,

we were still able to deliver fair results. Although some ideas certainly demand further *evaluation*, we feel that all demand further *exploration*. The results so far have been very encouraging, and we have no doubt that many (and perhaps all) of the ideas discussed here will be useful for the improvement of this very popular web based community.

See <http://noordkaap.science.uva.nl:8000/> for the online demo (active at the time of writing).

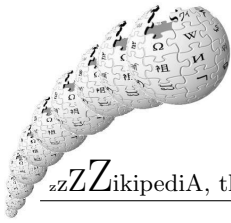
7 Future Work

Although the search possibilities have been improved a lot, there are still a lot more features which could enhance the ease of finding information within Wikipedia. There is a vast number of ideas to explore, and we will highlight some here.

One of those ideas would be to use an entirely different approach to clustering the results returned by Lucene for a given query. Similar to the previously described HITS implementation, in- and outgoing links could be tracked for each entry in the result list and compared to those of other entries. It is very likely that, given a specific query, several of the returned topics share common features, and fall within the same cluster on the basis of their citation graphs. Note that external pages, i.e. pages which have not been returned by Lucene, should not be taken into consideration for clustering. By identifying the clusters based on the results, a more intuitive graph-like or textual representation of the relations between the results could be displayed. Such a feature could help the user in grasping the underlying structure of the results based on his or her query.

Other future work should include the effect of weighting the co-citation scores, as well as the weighting of extracted salient phrases by IDF scores.

Finally, a route that was initially conceived for this project, but which subsequently fell victim to the pressure of time, was to try to investigate the effects of authorship on ranking (for an analysis of authors and their semantic coverage, see [3]). More active authors in a specific field may be more knowledgeable, and we may be able to take advantage of this in our ranking scheme.



References

- [1] Soumen Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kaufman, 2002.
- [2] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
- [3] Todd Holloway, Miran Bozicevic, and Katy Borner. Analyzing and visualizing the semantic coverage of wikipedia and its authors, 2005.
- [4] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [5] Tamara G. Kolda and Dianne P. O’Leary. Computation and uses of the semidiscrete matrix decomposition. Technical Report CS-TR-4012, 1999.