UMEÅ UNIVERSITY
Department of Computing Science
Rickard Melkersson
dva99rmn@cs.umu.se

February 4, 2004

Intrasite key phrase generation and mapping

Abstract

This Master thesis presents an implementation of a system for summarizing a website into key phrases. These key phrases are then connected to individual pages within the site. The system could be divided into two parts - key phrase generation and mapping. The phrase generation process is based on automated text summarization by keyword extraction, in a distributed environment. The mapping process is based on internet search engines technologies, with focus on the external structure. An evaluation, based on the run time results, and ways to improve the system, are also proposed.

Contents

1 Introduction

Searching for information on the internet is more than just finding pages with matching phrases. The enormous amount of pages, which is continuously expanding, has to be sorted before any relevant information could be found. To get a page in the top of the result lists, it is not enough to contain several hundred words. It must also be classified as an important page by the search engines.

One of the largest [21], and well known, general search engine is Google. It is utilizing many solutions [3] to select the most appropriate pages in the result list. These methods are applied on several other search engines.

Despite the effectiveness and the fast developments of the search engines, there are pages that are not optimally indexed. This because of many reasons, e.g. the extreme rate of growing or the structure of some sites. The web consists of over 40 million sites [20] with a wide range of topics, which implies total lack of homogeneity.

There are many solutions to improve the performance, and the precision of the search engines. A transparent proxy to a website could be set up, and such a proxy requires correct summarized key phrases to the front end. These key phrases are usually generated manually. The system described in this paper is automatically doing this process.

1.1 Paper Overview

The work is based on two main areas. Automated text summarization by keyword extraction and intra site search engine techniques inherited from internet search engines. Information extraction, IE, aims to extract relevant facts from the document set, and the goal of information retrieval, IR, is to select relevant documents. These two definitions are blurred because both have a common goal; to find relevant documents, or parts of documents. IE are generally at a finer granularity level than IR, see [15].

These areas are briefly described in sections 2 and 3. The sections include methods which are relevant for the task, but also some methods that could not directly be applied on the system. Some methods are applicable on both areas.

The system, described in section 4, will focus on the design. The first part is keyword extraction, to summarize the websites into key phrases. Then internet search techniques, for the issues of distributed information when connecting the phrases are used. Finally the design of the system is evaluated.

2 Keyword Extraction

Keyword extraction is a simple application of automated text summarization. These systems' main purpose is to create a summary of the original document. This can be done in two ways; by extraction, or by abstraction. The extraction is made by parts, words or whole sentences, of the source documents. Known as data summarizing, DS, it produces a list of sentences that are present within the source document in an effort to summarize the document contents. Abstraction is a fully synthesized text from the original. Data mining, DM, of the page set is then performed. The process is extracting the key facts within the document and does not restrict itself to extracting only sentences already in the document. DM may be able to make use of DS techniques to extract facts. Dixon discusses the definitions in [8].

Hovy and Lin [12] split the main stages for text extraction in the following parts:

1. *Topic identification*
   First the topic has to be identified. This is done by removing all irrelevant parts of the document, and save the important parts. Amitay and Paris [1] describe three approaches: The first approach is paragraph based summarization which identifies a single paragraph or segment that addresses a single topic. Web pages may not be structured in a way of paragraphs. The second approach is a sentence based technique that analyzes terms and phrases in the text to get the most relevant key phrases. The last approach is based on natural language analysis of the text.

2. *Text interpretation*
   When summarizing, all duplicate segments, with the same meaning have to be eliminated. This can optimally be done with merging, by generalization.

3. *Text generation*
   The extracted material should be transformed into suitable form, depending on the application. For search queries the syntactic form is quite basic, consisting of just a few terms. Most of the queries are single term queries, and virtually no more than six words, see [13, 16].

Differences from non-hyperlinked document sets
Extracting information and summarizing a website is much harder than a non hyperlinked document set. Even within a single website the environment will be inhomogeneous.

The information required for a correct extraction may be distributed over a large set of pages. Then, linking pages has to be included, which makes the task more complex. Depending on the link pattern in a text, the links can have different meaning or information value. When predicting a topic by anchor text, Amitay and Paris [1] imply that the pattern with a paragraph beginning with an anchor is most useful. The surrounding text of an anchor is also an important part when summarizing, see [5].

2.1 Heuristics

The first and second approach for text summarization in [1] is used when analysing a text in a statistical view. This is done by applying different heuristic functions to find the relevancy of the sentences. The heuristic functions generate a score for each segment, which usually are a sentence or a paragraph.

Weekdays, numbers and quotations are important heuristic functions when summarizing documents but are excluded in this paper because of its irrelevancy for the task. Further explanations can be found in section 4.1.

The combination of scores is deciding the sentence's importance. Combining these is a hard task, just like the similar heuristics in internet search engines. The different combinations depend on what type of document set to analyze. The compression rate is also an important factor in how to weight the heuristics. If possible, user feedback can be used to train a decision tree on a small fraction of the document set, as in [18].

- *Baseline*
  The baseline method scores by a sentence position in text. This is based on the fact that sentences in the top of the text are more important than at the bottom of the page. Baseline can be used with greater weight on news articles where the most important are written first.

- *Title*
  If any word in the sentence also occurs in the title, it is credited with a higher score. When applying on a web environment, either the title or the header tag, or a weighted combination of both, can be used.

- *TF and TFIDF*
  A heuristic function based on term frequency TF. This is basically the same as the TFIDF-computation described in section 3.2.1.

- *Position score*
  The function credits a sentence by a higher score if it is in a predefined position in the text. This is similar to the internal structure described in section 3.2.3

- *Query signature*
  Scores by the number of query words in a sentence. It is much easier to find the relevant parts if you know what you are searching for. This makes it to one of the most important functions. An obvious disadvantage is that it only can be applied if the user supplies a query.

- *Average Lexical Connectivity*
  A sentence is more important if it is sharing several terms with other sentences in the document. Segments with many terms shared with other segments are good summarizing segments.

3 Internet Search Engines

Search engines are mostly associated to a user who types a query in a web form and receives a list of URLs. There are many other types, new paradigms or applications on current platforms, and it has recently been a growing research area. The most common categories, with a leading example, are the following:

A directory service, like *Yahoo*[1]. Yahoo is primary a directory service, with the pages organized by subject. The sites are mostly suggested by users, then the editors evaluate it and add it to the catalogue.

Contents search, like *Google*[2]. The database has a full text index and the search is performed on the database by ordinary queries. Their software [3] crawls the web and index the pages automatically.

Finding similar pages, like *Netscape's what's related*[3]. This is a totally different approach where the user does not enter a query string. Instead Netscape compares a page with other similar pages. This is done by analyzing the external structure, see [7].

Specialized search engines, like *CiteSeer*[4]. An application, described in [10], that is based on the underlying search engines.

This work will focus on query based search on full contents indexed pages. It is the most popular type, with Google in the overall leading position, see [21]. Google [3] is also a good example how to develop and take advantage of new technologies.

All types of search engines use ranking to determine which pages to display in the top of the result list. Different search engines use different techniques, but even if the same techniques are used the results may be totally different due to the combination of methods and weights to decide the importance of the page.

There are ongoing developments of current search engines, mainly globally over the web. One of the largest problems is the complexity and the rate of growth. A global search engine with 150 Gigabytes of data must be well structured to work well in a real time environment. Brin and Page [3] mean

[1] http://dir.yahoo.com, visited 2003-12-29
[2] http://www.google.com, visited 2003-12-29
[3] http://wp.netscape.com/escapes/related/, visited 2003-12-29
[4] http://citeseer.com, visited 2003-12-29

that the hardware development has made slow advances on disc seek times so they have to minimize the disc accesses and optimize the database for large volumes of data.

3.1 Locating Pages

Before ranking, the potential pages have to be selected. All pages must first be indexed, before full content database operations are possible. In a content based search, the intersection between the search phrase vector and the vector of words within a page can be used. Before that is applicable, the pages have to be fetched and the words, on the page and the search query, have to be normalized.

The focus in this paper will be on user searching by text queries. The search queries only contain just a few words, which is the most common form of searching, see [13, 16].

Crawlers
Crawling the web is one of the hardest phases for internet search engines. That is because you do not have any control of the web, like non-responding servers and other hazards described below. The crawling is a time consuming task, but it can be several crawling processes running in parallel, see [3] for the Google implementation. The definite sets of pages on the web are virtually infinite. Some page structures and hosting websites are not spidered at all because of the complicated task.

Heydon and Nojork [11] think that the crawling task is hard primarily because of the following:

1. *URL aliases*, including canonical DNS-entries, alternative paths to the same pages on the same host and mirroring of sites.
2. *Session IDs*, used to create dynamic pages for the visitor.
3. *Crawler traps*, are intentional or unintentional methods that create infinite number of web documents or symbolic links on the file system.

The constant adding and modifying of links and pages to the web makes it hard for the crawlers to track the changes in the same rate. Therefore it must be a non-trivial schedule which pages to crawl. To predict these changes different methods are applicable, e.g. in [2]. Methods for a non-periodically refresh of web page indexes can be found in [6]. For new pages, Google or AltaVista[5] want to discover the most important pages first, because it

[5]http://www.altavista.com, visited 2003-12-29

is impossible to index all pages on the web. Breadth first search order for downloading pages is the most efficient. Najork and Wiener [19] imply that it also is the most natural crawling strategy, though all search engines do not use it.

3.1.1 Normalization

When matching information, it is advantageous if they got a common format. Stemming of words may be done to find common form of words. Removal of stop words and lowercase conversion are normalization methods that usually are applied. Special characters may also be removed or substituted. CiteSeer [10] is using the following normalization algorithm to normalize citations:

- Lowercase conversion.
- Expansion of common abbreviations.
- Removing of tags, special characters & : [ ] ( ).

Stemming
Stemming is a technique to transform different derivations of a word to a common root. It can be used when comparing different words while searching in texts. The different algorithms are language dependent and a common, simple but relative efficient, for English is Porters algorithm, see [24]. The effecttiveness depends on how the languages are constructed, with number of prefix and suffixes. The length of the searchable text also makes difference. An investigation by Carlberger, Dalianis, Hassel and Knutsson [4] shows that the average stemming can improve the precision at approximate 15 percent, for Swedish. Stemming seems to be more efficient while applied on a single website rather on the whole web because you will always find enough pages anyway. On large sets of documents, such the whole web, there is not much advantage of truncating or stemming.

Truncating of words is a more simple method and can be enough effective in some cases. That is because the common stem does not have to be grammatically correct because you only use it when comparing. There is a risk of truncating too much, and get irrelevant hits, especially on the web environment with the huge document set with a great scope of subjects.

Word distances
When searching for words, it is an advantage if you can display matches which do not have the exactly the same spelling. This can be done by computing the distance to find similar spelled words. Even if the comparing words

have not the exact same spelling, they can be semantically identical. Some prefixes can be different or just misspelled. To include these matches when comparing, edit distance can used. Edit distance counts weighted number of deletions, insertions and replacements to get the words identical. The complexity is $O(n \cdot m)$, where m, n are string lengths, see [17]. The complexity is virtually too large when comparing all words in the document set.

A more complex alternative is *LikeIt*, where you can compare whole phrases, with words with different order. The complexity is $O(n + m)$, see [27]. With less time complexity and matching of whole phrases makes it more useful in many applications. Implementations have also been done in a distributed environment for improved performance.

Stoplists
Stoplists are list of words for a specific language which is used to remove common words, from the phrase, without any significance. Stoplists are language dependent and can therefore be used to detect the language of the document, by seeing if frequently used words exist.

3.2 Ranking Pages

The big issue for the internet search engines are to sort the pages and display the relevant ones in the result list. Because of the large amount of pages on the web, the information will probably exist but you have to find it.

Ranking the pages can be done with different techniques, and combinations of them. Because of the complexity of the web approximations of algorithms are often applied. The whole web is not accessible and if it was there is not enough time to do an exact computation.

One way of ranking pages is by using heuristic functions. Several heuristic functions are co-operating to get optimal precision. Brin and Page [3] mean that tuning the ranking modules to co-operate is one of the most difficult tasks. The different methods may be suitable on different types of sites. This makes it difficult because of the lack of homogeneity of the web. Almost all search engines use user feedbacks to automatically tune the rankings, depending on the human surfer selections.

There are different types of rankings, and the main parts can be slit up as:

- *Contents analysis*
  Analysis of the contents within a single page.
- *External page structure*
  Determine relevancy of a page using the relations between pages, often with links.
- *Internal page structure*
  Use text-attributes and word position to decide the importance.
- *User feedback*
  Analysis of the user behaviour, by monitoring user selections and search queries.

3.2.1 Contents analysis

The contents of a page are a primary part when ranking its relevancy. The pages are often analysed by its words, with attributes and position. There are several methods to compute relevancy by its contents. The most important, for this study, is described below.

Term Frequency
Checking the frequency of a term, Term Frequency (TF), in a document can determine its relevance. Salton [25] describes a method that is more useful when combing TF with a decreasing function of the number of documents it appears in (Inverted Document Frequency, IDF). Inverted Document Frequency can be hard to compute for the web because the number of documents containing the word is unknown, and an approximation has to be used. Terms with very little relevance can be classified as stop words.

TFIDF is computed as follows:

Term Frequency: $TF(d,t)$ = Frequency of the term t in document d.
Inverse Document Frequency: $IDF(t) = \log\left(\frac{N}{df(t)}\right)$
Where N is number of documents and $df(t)$ is number of documents containing the term t.

Term Frequency $\times$ Inverted Document Frequency: $TFIDF(t,d) = TF(d,t) * IDF(t)$

Large documents got a larger probability to contain a certain word. To give a fair ranking, you have to normalize it with the size of the document, with methods like cosine normalization. Normalizations of page lengths are applied on many methods.

Lawrence and Giles [16] describe another method which could be applied on queries with multiple words. It is to check the distance (number of characters) between the words, and weight it.

Document distance
Salton [26] describes a method to utilize the TF when comparing two documents, and compute a distance between them. This can be applied when selecting between several documents or check if its contents are identical. Finding identical pages is necessary to eliminate incorrect statistical computations.

3.2.2 External Page Structure

The relations between pages in a web environment are set up by hyperlinks. One way of ranking the importance of a page is to check the in-degree of the page, as a node in a directed graph. The popularity of a page can be expressed in how many links pointing to a page. It can be difficult to determine on the internet because the page has to be downloaded and examined before knowing if it has any links to the current page. The out-degree is much easier to determine, in the simplest case just counting the *href* tags. There are different methods to measure the importance of the page by links, PageRank [22] or HITS [14] are the most popular and successful ones.

There is a lot of research of the structure of the web, also used to find relevant information. Flake, Lawrence and Giless [9] describe a method where links can be used to identify web communities, topically related subsets of the web with more links to each other than external links. Identifying communities or analysing graphs is beyond the scope of this paper.

PageRank
PageRank is a successful method used by Google. The Google implementation [3] of PageRank is used together with other search engine methods, partially described in this paper.

The concept is to create a ranking method where the large fraction of low quality pages are not displayed in the top of the result lists. The intuition is that a page with many citations is quite important, and then come high up in the result lists. A study by Pandurangan, Raghavan and Upfal [23] shows that PageRank follows the power of law by the exponent 2.1. That means

that the fraction of nodes having PageRank r is proportional to $1/r^{2.1}$. It is the same as the in-degree exponent. This does not mean that high in-degree implies high PageRank and vice versa. To obtain the ranking, the algorithm approximates the relative importance of the pages.

The method also visualizes a random surfer who browses from one page to another. At each step it is some probability that the surfer goes to an independent page. The random surfer model is necessary because when pages cite each other the rank will sink. The independent jump can be set to a small set of pages, like the root page of a site to get a different and better score in some cases. The random surfer model simulates a real user which results in that PageRank is also applied in other subjects like traffic estimation, see [22].

The formula to compute a PageRank of page i:

$$r(i) = \frac{1-p}{m} + p \sum_{j \in In(i)} \frac{r(j)}{d_{out}(j)}$$

Where p is the probability that the user jumps to an independent page.
$1, 2, \ldots, m$ are pages on the web.
$d_{out}(i)$ is outgoing links from i.
$In(i)$ set of pages pointing to i.

The equation is recursive but it will converge. The leaf pages without any out-links can be removed because it does not affect the PageRank value.

3.2.3 Internal Page Structure

When analysing HTML documents, the internal page structure can easily be determined. Different parts are implicitly marked by common HTML tags, which could be transformed into a degree of importance. Google uses this method [3], together with virtually all other search engines.

Without any tags, or when trying to extract specific information, the task gets more difficult. Then, the position or sequence could be used when identifying the parts. This is done by CiteSeer [10] when extracting information from electronic research documents. The process includes extraction of title, author, abstract and citations.

3.2.4 User Feedback

Monitoring the user feedback can be applied when tuning the ranking system. Virtually all search engines use some kind of dynamic optimization. The part is not further described in this paper.

3.3 Intrasite Search Techniques

Searching in a limited domain rather than the whole web is different. Most of the differences are advantageous because the less complexity and more homogeneity. The web is expanding in a fast rate which makes it difficult to monitor, while within a single domain the pages could often be seen as a static set of pages. Crawling techniques to eliminate update issues are described in section 3.1. Searching within a single site is not necessarily easier because higher demands for precision etc.

Up to date and crawling
One of the hardest elements in web searching, crawling, is not necessary. All information is found within the site. If some parts have to be crawled, it would not be especially vulnerable.

Internet search engines have to determine if the found page already is in the database, from the current location or if it is from a mirrored site. This should not be indexed by normal routines because you only want to display one instance in the hit list of the search engines.

Another advantage is that a single site is trying to supply information with a set of pages, and does not "compete" to get as much hits as possible. This reduces the risk of spamming, irrelevant information to get the visitors to the page.

Homogeneity
The homogeneity on the web does not exist! A single website has hopefully more homogeneity, in both contents and structure. It results in less ambiguous query terms because a single website often has a common subject. If all pages on the site has similar structure it will be easier to identifying relevant information.

Accessibility
On the web, approximation of different values has to be done because of the huge amount of information. It is also a dynamic set of pages that change all the time, on purpose or unintentional through disconnected servers. Within a site, you can presume that everything is accessible, which facilitates exact

computations, and easier evaluations of matching pages. To compute exact values of *IDF* or *PageRank*, an accessible network is required.

4 Architecture of a System for Generating and Map Phrases to Internal Pages

The system should be a generalized tool for analyzing different types of websites, without modifying the structure or the rule set. It is a part of a larger system that connects a transparent interface to a specific website. A large amount of phrases, approximately 2000 to 3000, should be generated and connected to individual web pages, within the website. This makes it easier for the search engines to correctly index the pages.

Without any modification on the website the connections should be generated. The website is already indexed and stored. The goal is to get the connected phrase as close as possible to the correct page, on every generated phrase.

4.1 Analysis

The system should take as much advantage as possible of the advantages for the current environment, including homogeneity within a single website and accessibility.

An already fully spidered website is required before the analysis, which results in accessibility to all web pages. The system should analyse a single, specified, domain at once. Computations can be done on the complete set, including full graph structure of the links, word and document counting. This also eliminates the risk of up to date complications. No real time user feedback is possible. With user feedback, it would had been easier, but without it, it may be possible to approximate the certainly of relevance of different attributes. There are no real time requirements but the complexity should not be too high. Approximations should be applied if necessary.

No training with user feedback is possible. This makes it impossible of a potential analysis of user actions for incremental precision tuning. Any utilization of the user profiles, what he or she prefers, and are interested in, is not possible because the surfer does not have any direct connection to the application. One possible solution is that the user sets several parameters depending of the type of the website. This means that static rules will be applied, instead of dynamic tuning during runtime.

When possible, stemmed variants of words will be used on all computations. It should be advantageous when analysing a relative small amount of pages. The system is based on the Swedish language and sites with Swedish pages only.

The system is divided into two main parts, first generation of phrases then mapping each of the phrase to single web pages:

- *Phrase generation:*
  Data extracting by summarizing the most important parts. This is later used as a base for the key phrase generation.

- *Phrase mapping:*
  Map each generated phrase to a single page on the web site.

4.1.1 Key Phrase Generation

In this phase search queries should be generated, by analysing the contents of a website. It is a simple form of text summarization documents (Section 2), applied to a whole website. This includes extracting relevant information and transforms it into standard search queries. A single key phrase could, currently, be referred to several different pages.

The amount of phrases to generate is not a trivial issue. If it is too few, all segments may not be selected, and too many will result in a large amount of irrelevant phrases. Generally it is better to generate too many phrases because it is easier to manually remove irrelevant phrases than manually add missed.

First all segments will be created. Then the heuristic functions will be applied on each of the segments which results in individual scores. The scores are applied when selecting important information, as segments. The segments are then used as a base for query generation. The whole process can be split up in the following phases:

- *Indexing document contents:*
  Parse the documents into segments associated with several attributes.

- *Segment selection:*
  Several heuristic functions decide which parts that are important and not.

- *Phrase generation:*
  Transforms the relevant segments into key phrases.

Indexing document contents
The indexing process consists of three stages, document indexing, segment indexing and word indexing. Raw HTML documents are parsed and stored into a proper format.

The generated list of documents to be processed may be different from the website. Only the documents with an assigned unique document identification number will be included. The number of documents used for computations is based on this list.

A segment is a block of text, delimited by either a special character or an HTML tag. Segments are the smallest unit when computing scores and ranking information. Each segment are associated by several attributes including position.

Words are stored in two states. First the original state, picked from the document, but also a stemmed variant. Stemmed words are used on all statistical computations. When indexing, word and document counting are also performed and stored for later use.

Segment selection
This part picks out the most relevant segment for each document. It is using ranking modules for selections and rankings, similar to the distributed text summarization procedure, described in section 2.

The scores can then be used when selecting segments, either globally or relative within a single document. The segments should be distributed over a large fraction of the document set, but at the same time only include the most relevant parts.

Heuristic modules compute different scores. Each module submits a score which is individually weighted. The weight could be tuned manually, but also automatically depending on site and page properties. The combination of the modules weighted scores are transformed into a global score, for each of the segments.

The following segment selection heuristics are applied:

- *Average Lexical Connectivity:*
  This heuristic function credits segments with high connectivity within the document. *ALC* is an important heuristic function because it gets the overall context and not just focus on the details. General search phrases and terms are important to include in the final phrase list. This is similar to the function described in section 2.1.

- *Term Frequency* $\times$ *Inverted Document Frequency:*
  The function is applied on text summarization and on internet search engines, description in section 3.2.1. The idea is to pick the most relevant word from each page, and get some unique details to include as a base for the key phrase generation.

- *Numbers:*
  If a digit occurs in the segment, it should result in a large negative score because of digits do not have high information value for search queries. This is the opposite of general text summarizing where this is an important heuristic.

- *Title:*
  If a word in the segment also exists in the title it will be credited. This heuristic importance can vary depending on the relevancy of the title. If a website has the same title over all pages, it should be less weighted than a site with unique titles for every page. The title could be seen as a one liner description of a document, and therefore contains very important words.

- *Segment attributes:*
  Text with attributes may have more or less information relevance than others. This heuristic scores with respect to text attributes of the segments, e.g. list entry and anchor text. Depending on the attribute it will score differently, positive or negative. This function could be classified as a function that analyses internal structure, described in section 3.2.3.

Phrase generation
The selected segments above are the base for the generation part. The result should be simple key phrases containing one to six terms, booth singulars and plurals. Modifiers, often in form of an adjective, should be included in the results because it is the second largest category of terms in user queries, see [13]. The phrases should be syntactic simple sentences or a set of keywords. Figure 1 shows user query lengths of two studies.

The methods applied are independently developed and not based on any previous study.

4.1.2 Phrase Mapping

Mapping is applied because of several phrases may have references to multiple pages. The most relevant page should be selected and connected to the phrase. If no page or several similar pages match a mapping, the system should consider mapping to the parent page with more general information.

To decide which page to choose, variants of search engine heuristics, described in section 3.2, are used. If a phrase only has one connected page, it will be picked and no computation will be done. The heuristics could be divided into three categories: internal page structure like similarities of the

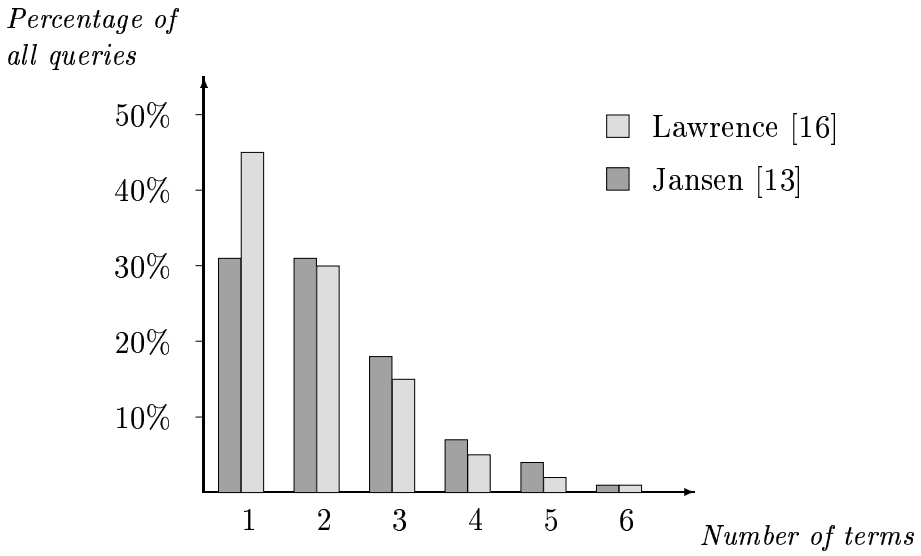

Figure 1: User query lenghts.

pages, external structure including link structure and pure contents analysis. Different types of ranking methods should be applied and possibility to add and remove these modules dynamically. The modules submit scores which are weighted. The heuristics should automatically be tuned for optimal performance depending on the characteristics of the website; like certainty in internal and external structure and external structure properties.

The focus will be on structure, and the contents in whole, because pure content based heuristics have already been applied in the previous phase. Comparison with selected segments and its contents will be done though.

The following heuristics are used:

- *Contents analysis:*
  Contents, either by words or its attributes, are used to determine its relevance. This is already, indirectly, computed in the segment selection process, described in section 4.1.1.

- *Anchor text:*
  Anchor text is one of the most descriptive texts for a document. The heuristic scores by finding similarities in the anchor text and the generated phrase. No normalization of number in-links will be used because many in-links may imply an important page (section 3.2.2).

- *Connectivity:*
  A heuristic function that scores depending on the similarities with other generated phrases on the current page. The score will be higher if several similar phrases are found. This is similar to the segment selection heuristic *Average Lexical Connectivity*.

- *External structure:*
  Compute a page importance by the link structure. A variant of the PageRank algorithm, described in section 3.2.2, will be used.

- *URL:*
  By checking path, filename and arguments of the documents, different scores will be credited.

4.2 Design

The system consists of four well separated phases with the database as a common base. The database should not be necessary at all parts but is time saving when developing different parts without a complete execution. The system could be split up in the following parts:

- *Initialization:*
  Transforms raw HTML files into more suitable data elements.

- *Segment selection:*
  Heuristic methods are applied to select relevant segments.

- *Key phrase generation:*
  Use patterns to generate key phrases and several post methods for increased connectivity.

- *Phrase mapping:*
  Map previously created phrases with search engine heuristics.

4.2.1 Initialization

The initialization is a preparing phase where everything is set up before the actual analysis. That includes parsing of HTML documents, segmenting and word indexing.

Before executing, following pre-conditions have to be set up:

- *Raw HTML-table:*
  A fully spidered web site must be located in the database, including contents of the pages with associating absolute URLs.

- *Dictionary:*
  A dictionary of the current language containing words with type and affix information.

Database

One of the weakest points in performance is the database. Analysing a site requires relative small databases, and no real time operations, but it will anyway be complex database operations. Intensive performance tunings are done on the larger databases storing global information for internet searches, see [3]. Reconstruction of indices depending on operation, removing unnecessary triggers and optimizing the physical storage are necessary when designing the database for this system. **Indexing documents**

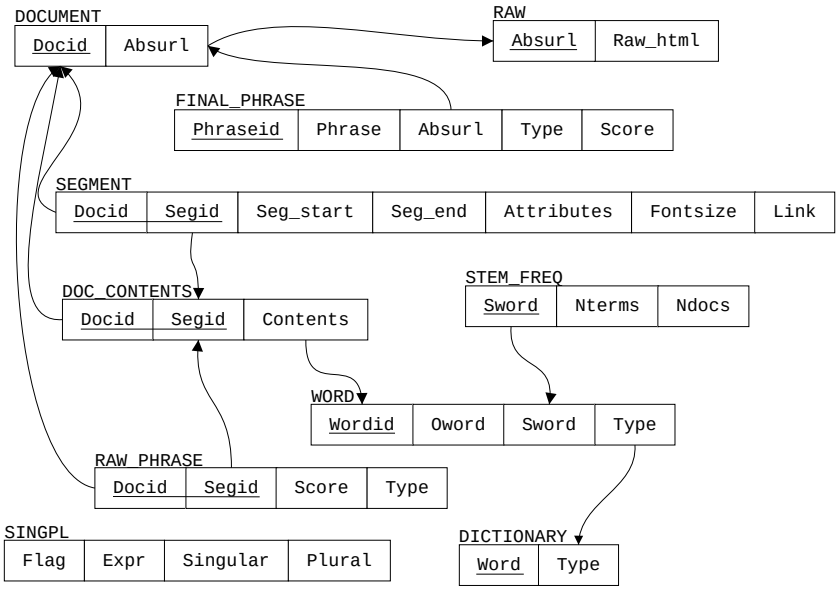

Figure 2: Relational database scheme.

The documents which should be used by the system are stored in the database table *Document*. For every HTML file which should be indexed; store the absolute URL with a unique document identifier, virtually an increasing integer sequence.

Indexing words and segments
Parsing is done at two parallel levels, for every page; tag parsing and charcter parsing. A segment can be delimited by either a tag (Table 1) or a character (Table 2). Some delimiters are marked to ignore text, like script tags.

<H1-H6>
<SPAN>
<DIV>
<HR>
<P>

Table 1: Segment tag delimiters.

: . ! ? @ ( ) " ; { }

Table 2: Segment character delimiters.

Documents are stored as segments with several word identifiers. All new words are assigned a unique word identifier, as an integer and stored in a word table.

Every segment is stored with a local segment sequence which is unique within each document, start and stop position, together with its attributes. Segments are marked with attributes, found in Table 3. If a FONT tag exists, additional information about font size is stored.

When all documents are parsed, stemming is performed on every word. The stemmed form of the words are stored together with statistical information, including number of unique occurrences and how many unique documents each word exists in.

<H1><H2><H3><H4>
<BIG><SMALL>
<UL><OL>
<STRONG>
<TITLE>
<FONT>
<EM>
<TT>
<I>
<B>
<A>

Table 3: Segment attributes.

4.2.2 Segment Selection

On each segment on every page $D = \{S_1, S_2, \ldots, S_n\}$, the score is computed by the heuristics in H, shown below. An individual predefined weight are assigned for each heuristic h in H, W_h.

$$Score_S = \sum_{h \in H} W_h \cdot h(S_n)$$

The segments are sorted by score; the highest score are picked first on a later selection process, described below.

Ranking heuristics
The ranking heuristics are applied on one segment at a time. Segments consist of one or several terms, $S = \{t_1, t_2, \ldots, t_n\}$.

- *Average Lexical Connectivity:*
  This heuristic function needs access to whole document $D = \{t_1, t_2, \ldots, t_n\}$, because of determine which terms that occurs elsewhere in the document. The check is preferably done once on each document.

$$Score_S = \sum_{S \cap D} \frac{1}{|D|}$$

- *Term Frequency × Inverted Document Frequency:*
  Term frequency and inverted frequency are computed on each term, t, in segment S. The whole document set, $W = \{D_1, D_2, \ldots, D_n\}$, has to be accessible. Removal of stopwords from S is done before computing the score.

$$Score_S = \sum_{t \in S} |\{t\} \cap D| \times \log \left(\frac{|W|}{\sum_{D \in W} \begin{cases} 1, & \text{if } |\{t\} \cap D| > 0; \\ 0, & \text{otherwise.} \end{cases} + |W|} \right)$$

- *Numbers:*
  A simple heuristic that makes it virtually impossible to give high credits to segments containing digits. It is of course depending of the weight, but it is expected to be large.

$$Score_S = \begin{cases} -1, & \text{if } \exists t \in S : t \text{ is a digit;} \\ 0, & \text{if } \forall t \in S : t \text{ is not a digit.} \end{cases}$$

- *Title:*
  Title terms set $T = \{t_1, t_2, \ldots, t_n\}$. T can be an empty set.

$$Score_S = |T \cap S|$$

- *Segment attributes:*
  The function scores depending on the attributes of the segment. The attributes, A, could be found in table 3. Several attributes can be combined at one time. W_a is the weight for attribute the a.

$$Score_S = \sum_{a \in A} \begin{cases} W_a, & \text{if } a \text{ is set;} \\ 0, & \text{if } a \text{ is not set.} \end{cases}$$

Segment distribution

When all segments got individual scores, the system has to decide which segments to be saved. If the highest ranked segments are picked, there may be an unfair distribution because of the characteristics of individual pages. For a better distribution, two thresholds are set up:

1. Minimum number of, locally best scored, segments per page. Currently set to 3.

2. Maximum number of segments per page. Currently set to 5.

When the segments are distributed, an additional set will be picked out, independent of prior number of selections. These will later be ranked in a second iteration.

The total number of segments selected, for standard and extra set, depends on the amount of documents in the set. Currently 1 $\times$ number of documents for standard set and 3 $\times$ number of documents for extra set.

4.2.3 Key Phrase Generation

Generation of key phrases is done by finding patterns in previous selected segments. Several different methods were evaluated and tested. Decision trees gave too high error rate because of approximation errors when minimizing the tree, combined with the indistinct input. By analysing the results from these tests, several general patterns could be found. They were later further developed and tuned with several sub conditions. The method also gave satisfied results with words that were not included in the dictionary. Other methods were also evaluated, but on a more shallow level. That included deeper semantic analysis of the segments. These methods where discarded because the heavy focus on the linguistic part, which is not a primary subject of this work. Section 6, Future Work, includes brief descriptions of branches, beyond the scope of this work that could be developed. The method could easily be substituted by a module implementation. A description of different modules could be found in section 4.3.

The pattern match method is used in the system. There are four different main patterns with several sub conditions with rules how to generate the phrases. Only one pattern is matched on each segment and this is done in a fixed order. Modifications of the sequence or which part to match may be variable depending on information values of the term.

The terms are categorized into six groups (Table 4), standard word classes, and special group Name which are words that could not be identified by the dictionary, and Fixed that are special words and often prepositions, i.e. "till" or "med". Adjective and adverb are being inherited by noun and verb, respectively.

N:	Name
S:	Noun
V:	Verb
H:	Fixed
O:	Conjuction
A:	Adjective/Adverb

Table 4: Word categories.

Mapping is done in following order and if a match is found, no other type will be matched. The symbols of describing patterns are "[ ]" for optimally words, "-" are alternatives and "^" is the start of the segments.

- *Type A:*
  The most complex pattern and therefore tested first. If the pattern is found it will generate queries with three to six terms.

$$[A]\,\frac{S}{N}\,[V]\,H\,[A]\,\frac{S}{N}$$

Sub conditions:

First A: Standard form of A.

Second, $\frac{S}{N}$: Transform to standard form, if first. / Straight mapping, if not first.

Third, V: Transform to standard form, if first. / Straight mapping, if not first.

Fourth, H: Straight mapping.

Fifth, A: Straight mapping.

Sixth, $\frac{S}{N}$: Transform to non-genitive form, if in genitive form. / Straight mapping, if not in genitive form.

- *Type B:*
  A simple conjunction rule which extracts two parts separated with a conjunction word.

$$\frac{S}{N} O \frac{S}{N}$$

Sub conditions:

First, $\frac{S}{N}$: $\begin{cases} \text{Transform to non-genitive form, if in genitive form.} \\ \text{Straight mapping, if not in genitive form.} \end{cases}$
Store in singular and plural form.

Third, $\frac{S}{N}$: $\begin{cases} \text{Transform to non-genitive form, if in genitive form.} \\ \text{Straight mapping, if not in genitive form.} \end{cases}$
Store in singular and plural form.

- *Type C:*
  This pattern is only applicable at the start of a segment. It will extract one to three terms, which can be nouns or names. The rule is useful because it usually include names that are not in the dictionary.

$$\hat{} \frac{S}{N} \left[\left[\frac{S}{N} \right] \frac{S}{N} \right]$$

Sub conditions:

First, $\frac{S}{N}$:	If only term and S:	Transform to non-genitive form, if in genitive form. Straight mapping, if not in genitive form. Store in singular and plural form.
	Else:	Straight maping.
Second, $\frac{S}{N}$:	Straight mapping.	
Third, $\frac{S}{N}$:	Straight mapping.	

- *Type D:*
  The last pattern extracts the whole segment if the sub conditions are satisfied. The rule is a final solution of adding important segments with uncommon or complex syntaxes. It is applied to minimize the number of segments to drop, and are usually applied to a relative small amount of segment.

  Sub conditions:

Ignore if:	Segment length > 6, or First or last term is a category H term, or Last term is a category V term.
Else:	Store the whole segment.

Singular and plural generation
If pattern type B, or a single word pattern type C, is found, two forms of the term is generated; plural and singular form. This is done by substituting prefixes of the base form. To find which prefix to substitute, a simple pattern match is performed by patterns in table 5. These are common patterns used in the Swedish Ispell dictionary[6]. Each word is a member in one or several word categories.

[6]http://www.sslug.dk/locale/ispell/, visited 2003-12-29

Word category	Trailing pattern	Singular substitution	Plural substitution
G	[^ae]	-	ar
G	e	e	ar
G	a	a	or
H	[^a]	-	er
H	a	a	er
I	en	en	nar
I	m	m	mar
I	[^m]el	el	lar
I	mel	mel	lar
I	[^m]er	er	rar
I	mer	mer	rar
I	mar	mar	rar
J	el	el	ler
J	ot	ot	ötter
J	and	and	änder
J	[^lmnr]	-	r

Table 5: Plural and singular generation pattern.

Part connection
The purpose of part connection is to connect overreaching terms to pages connected more detailed terms. It connects another candidate page to a single term phrase. An example[7] is where the term "Gasol" (liquefied petroleum gas) also connects to the pages with the phrase term "Gasolflaskor" (bottle of liquefied petroleum gas).

The following algorithm is used for connecting partial words and phrases:

1. For every single term query in set where its length are more than two characters:

   (a) For every multiple term query in set:

      i. If the single word is a beginning of any of the words in the

[7] A Swedish example is used because complex words are written connected, and makes the method more powerful.

multiple terms query: Add connection (Single term query to multiple term query URL).

Phrase removal
The bonus phrases selected (section 4.2.2) and generated (section 4.2.3), should be sorted out. The relevant phrases should be included into the standard set. This is done by checking the connectivity with already created phrases in the standard set. Similarity checks for the words are computed by edit distances, described in section 3.2.1. A check, similar to *Part connection*, described above, is used to add phrases that have terms with any common part with a word in the standard set.

The following algorithm is used to decide which terms to be included in the standard set:

1. For each phrase in bonus set:
   (a) For each phrase in standard set:
      i. Check the edit distance between every word with following conditions:
      Has a length of, currently, minimum 3 characters.
      Not a fixed word or a conjunction word.
      A. If the minimum of all distances are less than, currently 2, add it to the standard set.
      If the bonus set word are a part of the standard set word, and not a *H* word: Add it to the standard set.

Single term selection
All phrases generated are based on the segments. Another supplementary technique is to find single term that depends on the occurrences within the entire website. A keyword range has to be chosen depending of total number of words or documents. If a too wide range is specified, irrelevant words will be picked. If a word occurs more times than the upper limit, it is a too general word, maybe defined as a stop word, described in section 3.1. If the lower limit is set to low, many irrelevant words that just occur a few times will be picked.

4.2.4 Phrase Mapping

The phrase mapping is, just like the segment selection process, based on heuristic functions. It uses some of the results from segment scores. This

includes, besides of the generated phrases, the scores from the previously picked segments.

The heuristic functions will focus on the external structure because analysis of the contents has already been performed. Before the selection begins, identical mirror pages have to be eliminated.

Heuristics

The evaluation is applied on each phrase. A score is computed by the heuristics in H. An individual weight for each heuristic h in H, W_h, is individually set. On each phrase, $P = \{t_1, t_2, \ldots, t_n\}$, a single URL is picked from the URL-list $U_P = \{URL_1, URL_2, \ldots, URL_n\}$. If $|U_P| = 1$, direct mapping is applied. Otherwise the score is computed by following formula:

$$Score = \sum_{h \in H} W_h \cdot h(P_n)$$

The connection which got the highest score is picked, while all other connections are deleted.

The following heuristic functions are used:

- *Contents analysis:*
  Unlike all other mapping heuristic functions, this heuristic use the original segment information as the primary data. The score from each origin are compared and sorted with the highest score first. Credits are given depending on their position, by following formula:

$$Score_P = \frac{1}{position}$$

- *Connectivity:*
  This heuristic score depends on the connectivity by other phrases connected to the page. Edit distances or partial matches could be applied when matching. Normalization of the score is done by number of segments belonging to current document $D = \{S_1, S_2, \ldots, S_n\}$. Fixed words, H, could first be removed, $P \setminus H$.

$$Score_P = \sum_{S \in D} \frac{|P \cap S|}{|D|}$$

- *External structure:*
  The pages, i, got individual scores $Score(i)$. The computation is recursive and is done once. M is the document set and p is the probability to jump to an independent page.

$$Score(i) = \frac{1-p}{m} + p \sum_{j \in In(i)} \frac{Score(j)}{d_{out}(j)}$$

- *URL:*
  This heuristic function utilizes the URL to generate scores. It is divided into two parts; phrase connection, which scores positive, and file arguments, which scores negative. The two parts are added with individual weights, into a single score:

$$Score_P = W_{path} \times Score_{path} + W_{arg} \times Score_{arg}$$

  The phrase connection part first normalizes the URL, removes and transforms characters. Then, part connection scores are computed. This is similar to the algorithm described above. Length normalization of the URL is performed.

  The argument score is computed to eliminate the risk to connect irrelevant sub pages. The number of arguments is multiplied with d, a weight, currently, set to 0.1.

$$Score_{arg} = -\log(1 + d \times |arg|)$$

4.3 Implementation

An implementation of the system was done. It is written in Perl[8], primarily because of its regular expressions, easily accessible modules and intuitive database connections. PostgreSQL[9] is used as the relational database, because it is simple and powerful enough. It is the base for the whole system, and all modules communicate through it. The final output will also be presented as data in a table. Having the database as a hub will result in the bottleneck of the performance. An optimal database configuration could probably decrease execution times dramatically.
The system consists of the following modules (Figure 3):

[8] http://www.perl.org, visited 2004-01-15
[9] http://www.postgresql.org, visited 2004-01-15

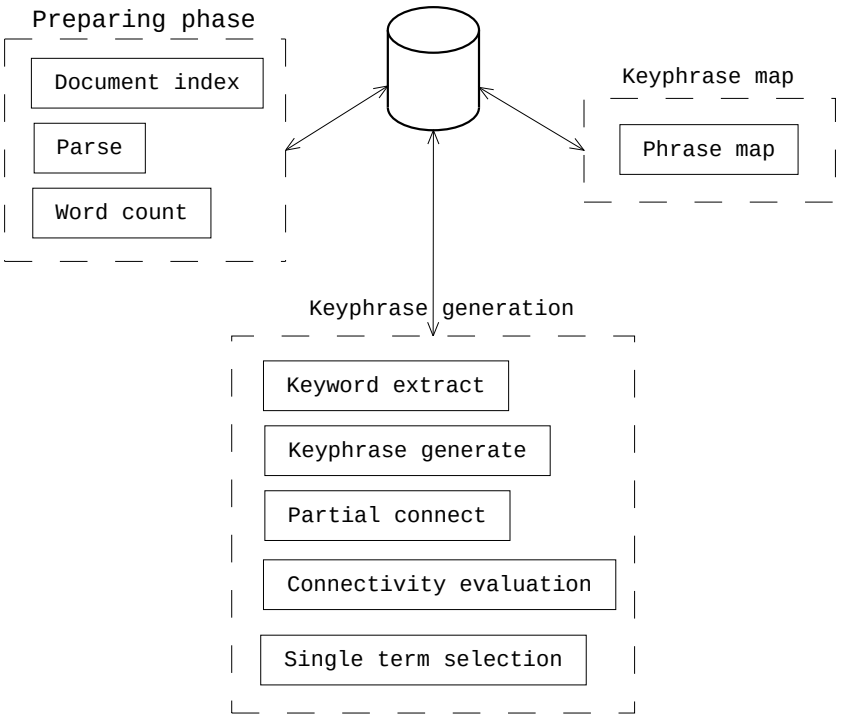

Figure 3: System overview.

Document index

The document indexer module selects which pages to include in the document set. In its simplest mode all pages will be included, but often, manually removal of irrelevant pages is necessary for optimal performance. Pages to remove may include non-Swedish documents or pages with pure data. A simple web tool for removal of files and directories was also developed.

Parse

This module parses the indexed HTML pages that have associated document identifiers. Words, documents and segments with attributes are saved in the database. In this phase, each new word is also assigned a unique word identifier, later used for statistical computations. It also stems the words, with utilizes Ispell, via Perl module *Lingua::Ispell*. The Perl module has to be modified to be able to produce stemmed words. The dictionary is a standard Ispell dictionary in Swedish. It is also stored in the database, because of later use by other modules. The data stored are words with associated affix information.

Word count
This module counts the words and segments where the words occur.

Keyword extract
This module is equivalent to the segment selection process described above. Its primary purpose is to select interesting text segments. This is done by heuristic functions. Each of these functions is implemented in a separate file, for easier modification. The files contain an interface for the preparing function and one for the scoring function. The preparing function is run once on each document or website, and the scoring function is called on each segment. The segment identifiers are put into a different category. The size of each category is set individually for each website.

Keyphrase generation
Finalizer is the second module of the key phrase generation. It generates simple key phrases, described in earlier sections, from previously selected segments. Pattern matching is used on the segments after the types of the words have been determined. This is done by Ispell affix categories, earlier saved in table *Dictionary*. Swedish Ispell dictionary is used in form of a database table, but also as a dictionary in Ispell. The resulting output of the generation is saved in *Final_phrase* and unknown patterns are ignored.

Partial connect
The part connector adds connections to single term phrases if they exist in other phrases, by part of, or complete terms. All single term phrases are compared with all other phrases, except itself. This may generate identical phrases and connections. After the connections have been made, a duplicate removal process starts. It will remove identical phrases on every page.

Connectivity evaluation
The final step in the key phrase generation phase is to include or drop phrases from the last category. The phrases that belong to the category are marked with a special category number in row *Type*. The analysis can be done with two methods, either by checking edit distances or just plain matched for faster executions. This phrase is irreversible because the primary input table are equal to the resulting table.

Single term selection
Instead of defining the range of keyword selection; the incompleteness of the dictionary is used. This decreases the risk to include stopwords and other irrelevant terms. Dictionary, $D = \{t_1, t_2, \ldots, t_n\}$. All terms within the web-

site, $W = \{t_1, t_2, \ldots, t_n\}$. $W \setminus D$ is sorted in number of documents where the terms exist. If t is a word shorter than three characters or includes a number, it will be removed. First n elements are picked, where n is a multiple of number of documents.

Phrase map
This module is, like the segment selection modules, divided into several heuristic functions. Each heuristic has a separate file, with interfaces to preparing and evaluation functions

4.3.1 Configuration

One of the goals was to minimize number of configuration options and let the system determine optimal settings. The following parameters has to be set before executing:

- *Database type:*
  Set the database type, which should be supported by Perls DBI package.
- *Database user:*
  Account that are authorized for all database operations on the tables.
- *Database password:*
  The password for the specific account.
- *Domain:*
  This option specifies the domain of the analysis. It can be a true DNS entry or a single directory.
- *Minimum phrases per page:*
  Minimum number of phrases to include from every page in the set.
- *Maximum phrases per page:*
  Maximum number of phases to include from every page in the set.
- *Size of bonus category:*
  Number of phrases to include in addition to the minimum set per page. The value specified is a multiple of the document set size.

- *Size of candidate category:*
  Number of phrases to include for connectivity evaluation. The value specified is a multiple of the document set size.
- *Single Keywords:*
  Number of additional single keywords to include in the phrase list. The value specified is a multiple of the document set size.

5 Empirical Results

Summarising texts is a difficult task, and even more difficult in a distributed environment. An error rate below 50% is the goal for this prototype. If the system is used in real life, manual tuning and traversing of the phrase lists would increase the correctness. In these tests, all lists will be in its original state.

Three different websites, or part of them, were tested during the evaluation. All were analysed by the same methods but with individually set thresholds. The results where then evaluated by methods depending on what to check. The sites varied in size; 1145, 908 and 300 pages. Several other sites were tested during development of the system, but not as a part in the final evaluation.

Generally, it is better to generate too many phrases, which will include a higher amount of irrelevant phrases. This was also tried to be reflected in these tests.

5.1 Evaluation

Each phase will be separately evaluated and then an evaluation of the system's overall result. Summarised tables with results from the different parts of the executions can be found in appendix B.

For relevancy and connection evaluation, the phrases were ranked by checking the originated page. On phrase generation, there was no relevancy check in respect to the original page.

5.1.1 Segment Selection

Table 6 shows the results of the segment selection process. The minimum, of currently three segments per page, is used as evaluation data. The most remarking is that no relevant segment was picked on only approximately 5% of the pages. Mostly, one segment was irrelevant.

If some segments are missed, it is not a disaster. Similar segments may be picked on other pages and result in same final phrases. The result could be modified by changing the minimum phrases per page threshold.

5.1.2 Phrase Generation

The results of the phrase generation process can be found in appendix B.

The high amount of short phrases generated is because the segment selection process credits short and concise segments higher.

The distribution of the terms is very good. This is shown in table 7 and figure 4. The number of phrases containing three terms is considerable higher than previous research of user query behaviour. This because of pattern type C, where up to three subsequent nouns or names could be matched, were used in over 21% of the cases (table 8). The third term included in the patterns is often relevant. With a more precise dictionary, the generated results from pattern type C may be a smaller part.

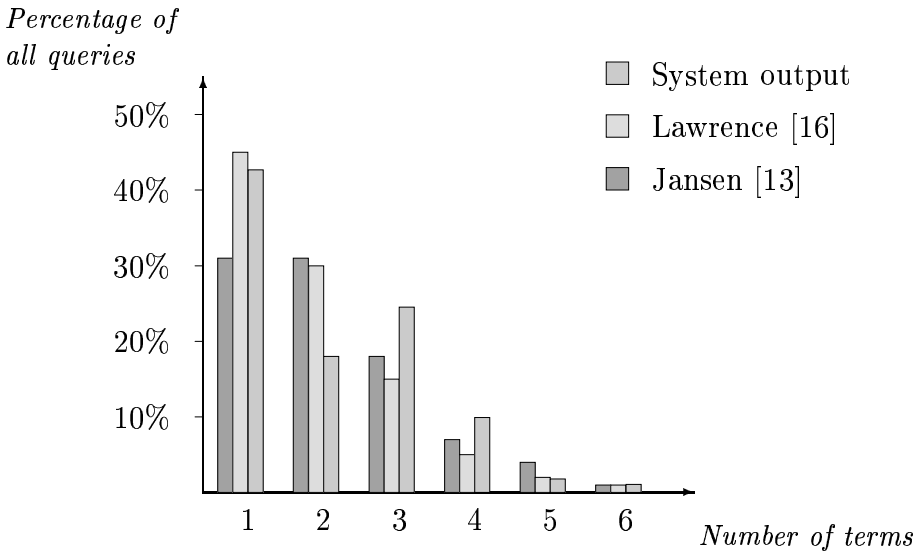

Figure 4: Lengths of generated queries.

The relevance of the generated phrases (table 9) is quite low. Approximately 70% of the phrases are relevant. Phrases over four terms are too few for a correct analysis. The irrelevant phrases generated could be phrases like "Mail us" or "Click here".

The incorrectness of the semantic (table 10) and the syntactic (table 11) are not big issues. The amount of correctly generated phrases are relative high though. This may be a result of the simple structure and few terms of the phrases.

Terms that are globally selected independent of any segment (table 12) are important. It is an essential that these words are included in the final phrase set. A correctness of approximate 61% makes it to an important function, applied when picking single terms. Some of the terms may have been included in other earlier phrases, generated by other methods.

5.1.3 Phrase Mapping

The mapping results are shown in table 14. The mapping is a quite important part, but it is not crucial if the precision is lower than expected. The results are satisfied with an error rate at 12.5%, and over half of the phrases got the most suitable page connected.

5.1.4 Overall

The overall result is the most important if it is considered to be used as a real product. An error rate below 50% is the goal, but the errors can easily be modified with different parameters. A high number of phrases and a high minimum on a single page may result in many irrelevant phrases. If the values are set lower, many phrases will be missed. By higher precision, the thresholds can be set low without missing any of the phases.

The overall result is a combination of previous phases, and would not be better than the individual phases. The phrases used on evaluation gave following results (Figure 5):

- Segment selection: 64.29%
- Phrase Generation: 68.75%
- Keyword selection: 61.02%

Note that phrase generation phase correctness is higher than the segment selection. Because the phrase evaluation was independent from relevant scores connected to a specific page, it will show these results. The reason was to do an independent and fair phrase generation evaluation. A single segment could also results in several resulting phrases. The single keyword phrase was separated because its independence of the selected segments.

Because of the error rate grows when combining the two phases, the results would be lower than any of them. The correctness could be computed as follows:

$$Correctness = m(0.64 \times 0.69) + n(0.61)$$

Where m is the part of segment dependent phrases and n is the part of independent phrases.

The values are set to $m = 0.7$ and $n = 0.3$. This will give 49.2% correct phrases. When counted on the phrase set, 45.56% of the phrases were OK,

or better. An extra ranking module that sorted the phrases was applied on the final results. That increased the result by approximately 5%, to 49.32%.

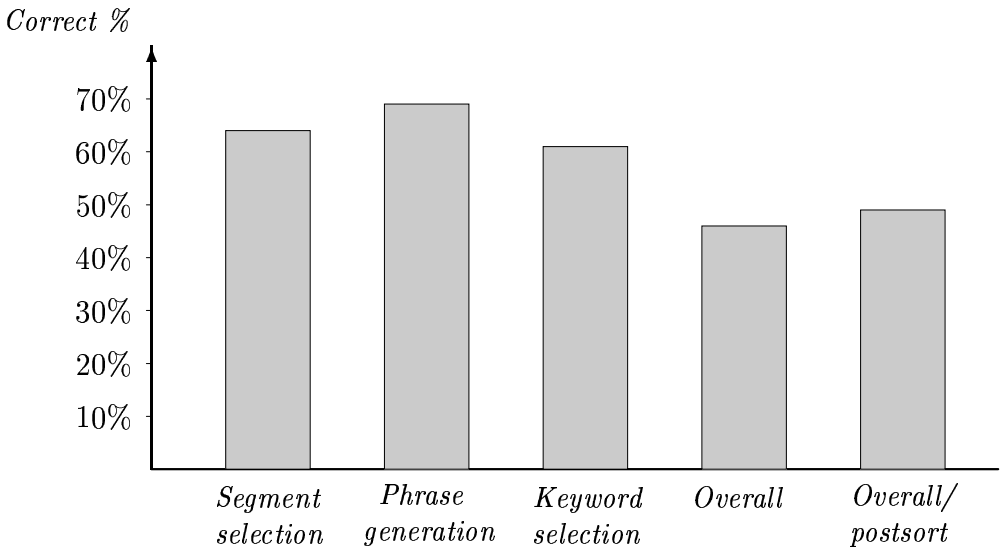

Figure 5: Overall results.

5.2 Conclusions

The error rate is generally above 50%. It is not too bad, but less than half of which was the goal. The error rate grows at a higher grade when utilizing two different modules. A single process with both extraction and generation may be one way to eliminate the error rate. Another solution may be an effective post sorter, which removes bad phrases. A simple post sorter, which was implemented, decreased the error rate by approximately 5%. If a larger phrase set is used with a more effective post sorter, it may be further decreased.

Each of the two modules, segment selection and key phrase generation, could be individually optimized, but they have to be at the same level of effectiveness for optimal performance. If an intense work of optimizing the modules to co-operate at a higher grade, the error rate would probably decrease. The connection of the phrases to individual pages is good, and does not need any major development. The mapping is quite an easy task because of the limited subject within a website.

6 Future Work

The system is not complete and could be further developed on several parts. The study was aimed to be a concept, to see if it worked, and what further developments to do. If it gave satisfied results, the work was intended to be used as a real life application. The future work could be divided into two categories; further development of current techniques and different paradigms.

The pattern matching technique for generating phrases from segments could be developed. If multiple patterns are found in a segment, only the first is used. All patterns may instead be extracted and then select one or several depending on the information value. An alternative solution for the pattern match could be applied. A deeper linguistically analysis for abstracting a segment may be suitable. Pattern matching generates syntactic simple phrases, which also should be the goal of any solution.

Other developed technologies of analysing different file formats than HTML may be applied. These could include images, PDF files, etc.

For better mappings, the topology of the website links may be computed, and then extract information from the pages' relatives. Better utilization of surrounding link texts could also be used then finding context of the current and related web pages. Like all text summarising, finding its context is useful, to minimize the ambiguity. Semantic world word trees[10] can also be applied. This can be used for abstracting the segment, but also to find the context of a segment.

Another point of improvement is the dictionary. The dictionary used in ths study was simple and incomplete, with approximately 24.200 Swedish words, in common form, and over 100.000 with all variants. The small amount of words was not the primary issue. Information of the types of words was often ambiguous, which resulted in wrong phrases.

The system does not have any user feedback during runtime, because the evaluation of full automatically executions. For better results, users may manually define the important parts by some markings, and then may use AI for training, see [18].

[10] http://www.cogsci.princeton.edu/~wn/, visited 2004-01-13

7 Issues during the Work

A major redesign of the database was necessary due to poor performance. Too many join operations were needed for several different operations. Because of the central role for the database, the redesign was a complex task. The new design includes rows with array elements, a specific feature in Postgres databases.

The execution times became large at an early stage. This problem got bigger than estimated, which resulted in extensive speed optimization when designing the succeeding parts. The execution times are not a primary problem on the final system, but it is unpractical when developing it. The solution was to create smaller modules which utilized more reversible database operations.

8 Acknowledgements

I would like to take the opportunity to thank Ulrich Wisser at Relevant Traffic. He made this study possible, by supporting me and by sharing some of his knowledge. The outline for this paper was discussed and read by my supervisor Per Lindström at the Department of Computing Science, Umeå University.

References

[1] Amitay, E., and Paris, C. Automatically summarising web sites - is there a way around it? In *CIKM* (2000), pp. 173–179.

[2] Brewington, B. E., and Cybenko, G. Keeping up with the changing Web. *IEEE Computer 33*, 5 (2000), 52–58.

[3] Brin, S., and Page, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117.

[4] Carlberger, J., Dalianis, H., Hassel, M., and Knutsson, O. Improving precision in information retrieval for swedish using stemming. Tech. rep., NADA-KTH, 2001.

[5] Chakrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Raghavan, P., and Rajagopalan, S. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference* (1998).

[6] Cho, J., and Garcia-Molina, H. The evolution of the web and implications for an incremental crawler. In *Proceedings of the Twenty-sixth International Conference on Very Large Databases* (2000).

[7] Dean, J., and Henzinger, M. R. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands: 1999) 31*, 11–16 (1999), 1467–1479.

[8] Dixon, M. An overview of document mining technology, 1997.

[9] Flake, G., Lawrence, S., and Giles, C. L. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Boston, MA, August 20–23 2000), pp. 150–160.

[10] Giles, C. L., Bollacker, K., and Lawrence, S. CiteSeer: An automatic citation indexing system. In *Digital Libraries 98 - The Third ACM Conference on Digital Libraries* (Pittsburgh, PA, June 23–26 1998), I. Witten, R. Akscyn, and F. M. Shipman III, Eds., ACM Press, pp. 89–98.

[11] Heydon, A., and Najork, M. Mercator: A scalable, extensible web crawler. *World Wide Web 2*, 4 (1999), 219–229.

[12] Hovy, E., and Lin, C. Automated text summarization in summarist, 1997.

[13] Jansen, B. J., Spink, A., and Saracevic, T. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management 36*, 2 (2000), 207–227.

[14] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*, 5 (1999), 604–632.

[15] Kosala, and Blockeel. Web mining research: A survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM 2* (2000).

[16] Lawrence, S., and Giles, C. L. Context and page analysis for improved web search. *IEEE Internet Computing 2*, 4 (1998), 38–46.

[17] Levenshtein, V. I. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission 1* (1965), 8–17.

[18] Lin, C.-Y. Training a selection function for extraction. In *CIKM* (1999), pp. 55–62.

[19] Najork, M., and Wiener, J. L. Breadth-First Crawling Yields High-Quality Pages. In *Proceedings of the 10th International World Wide Web Conference* (Hong Kong, May 2001), Elsevier Science, pp. 114–118.

[20] Netcraft. Web Server Survey Archives. Webpage, 29 Dec. 2003. `http://news.netcraft.com/archives/webserversurvey.html`.

[21] Notess, G. R. Search Engines Statistics: Database Relative Size. Webpage, 29 Dec. 2003. `http://www.searchengineshowdown.com/stats/size.shtml`.

[22] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford Digital Library Technologies Project, 1998.

[23] Pandurangan, G., Raghavan, P., and Upfal, E. Using PageRank to Characterize Web Structure. In *8th Annual International Computing and Combinatorics Conference (COCOON)* (2002).

[24] Porter, M. An algorithm for suffix stripping. In *Program* (1980), vol. 14, pp. 130–130.

[25] Salton, G. Developments in automatic text retrieval. *Science 253* (Aug. 1991), 974–980.

[26] Salton, G., and Buckley, C. Term weighting approaches in automatic text retrieval. Technical report, Nov. 1987.

[27] Yianilos, P. The LikeIt intelligent string comparison facility. Tech. Rep. 97-093, 1997.

Appendix A: Glossary

AI	Artificial Intelligence.
ALC	Average Lexical Connectivity.
Attributes	Textural attributes utilized when analysing internal structure.
Authority page	An important page because of its contents.
Crawl	Information retrieval process by traversing pages.
CSS	Cascading Style Sheet.
DM	Data Mining.
DS	Data Summarizing.
DT	Decision Tree.
External structure	Relation between individual pages, often represented by hyperlinks.
Hub page	An important page because of its outgoing links.
IDF	Inverted Document Frequency.
IE	Information Extraction.
Internal structure	The structure within a page.
IR	Information Retrieval.
Mirror	Same page of a different location - directory or server.
Phrase	Sentence or one to six keywords, sometimes referred as *key phrase*.
Query	User search string, often same format as key phrases.
Segment	The smallest unit analysed in the system.
Spam	Irrelevant words and phrases on a page to get more visits.
TF	Term Frequency.
TFIDF	Term Frequency $\times$ Inverted Document Frequency.

Appendix B: Runtime Results

Segment Selection

Relevant hits	Number of hits	Percentage of hits
3	16	30.19%
2	23	43.40%
1	14	26.42%
0	3	5.36%

Table 6: Selection of minimum set.

Phrase Generation

Number of terms	Number of phrases	Percentage of all
1	187	42.69%
2	79	18.04%
3	116	26.48%
4	43	9.82%
5	8	1.83%
6	5	1.14%
Total	428	100%

Table 7: Query lengths.

Pattern type	Number of patterns	Percentage of all
A	42	38.53%
B	6	5.50%
C	23	21.10%
D	17	15.60%
Dropped	21	19.27%

Table 8: Pattern distribution.

Number of terms	Relevant	Irrelevant	Relevant%	Irrelevant%
1	19	8	70.37%	29.63%
2	14	2	87.50%	12.50%
3	9	4	69.23%	30.77%
4	1	4	20.00%	80.00%
5	1	1	50.00%	50.00%
6	0	1	0.00%	100.00%
Total	44	20	68.75%	32.75%

Table 9: Relevance of generated queries.

Number of terms	Semantic correct	Semantic incorrect	Semantic correct%	Semantic incorrect%
1	24	3	88.89%	11.11%
2	15	1	93.75%	6.25%
3	9	4	69.23%	30.77%
4	3	2	60.00%	40.00%
5	1	1	50.00%	50.00%
6	1	0	100.00%	0.00%
Total	53	11	82.81%	17.19%

Table 10: Semantic correctness of generated queries.

Number of terms	Syntactic correct	Syntactic incorrect	Syntactic correct%	Syntactic incorrect%
1	24	3	88.89%	11.11%
2	14	2	87.50%	12.50%
3	12	1	92.31%	7.69%
4	3	2	60.00%	40.00%
5	1	1	50.00%	50.00%
6	1	0	100.00%	0.00%
Total	55	9	85.94%	14.06%

Table 11: Syntactic correctness of generated queries.

Relevant terms	Irrelevant terms	Relevant terms%	Irrelevant terms%
36	23	60.02%	38.98%

Table 12: Single terms selection.

Phrase Mapping

Page connections	Phrase percentage%
1	17.31%
2	51.75%
3	2.45%
4	17.83%
5	1.05%
6	3.15%
7	0.35%
8	1.75%
9	0.52%
10	0.70%
> 10	3.50%

Table 13: Phrase connections before mapping.

Category	Number	Percentage
Best	21	52.50%
Good	14	35.00%
Bad	5	12.50%

Table 14: Mapping evaluation.