MSEEC – A Multi Search Engine with Multiple Clustering*

Peter Hannappel, Reinhold Klapsing, Gustaf Neumann
Information Systems and Software Techniques
University of Essen, Universitätsstraße 9, D–45141 Essen, Germany
{Peter.Hannappel, Reinhold.Klapsing, Gustaf.Neumann}@uni-essen.de

Adrian Krug
Support Delivery Engineering, Hewlett Packard,
Berliner Straße 111, D–40880 Ratingen, Germany
Adrian_Krug@hp.com

Abstract

This paper presents a scalable architecture for a multi search engine for web documents with multiple cluster algorithms (MSEEC[12]). Querying search engines in the web may result in an overwhelming amount of matching documents. Clustering techniques are used to find a set of similar documents which are presented using a suitable cluster title. The scalable and modular architecture of MSEEC allows to process information with multiple cluster algorithms to present alternative clusters and the related cluster title to the user. This paper presents as well a novel clustering technique that is based on the LZW compression method.

1 Introduction

A prerequisite for the reuse of knowledge is accessibility. In order to profit from previous work it is necessary to access and contribute documents containing the knowledge. Open protocols such as used in the web can provide an infrastructure for information exchange over heterogenous systems. The web technologies are suitable to build distributed information and authoring systems where people are able to search and retrieve digital documents. There is already a huge amount of information contained in digital libraries and the web and thus no database can store all information. Searching in multiple databases can increase the information coverage. However, finding the right information in the sometimes overwhelming amount of search results is a challenge.

This paper presents MSEEC, a novel a multi search engine for web documents which is able to provide the user a synopsis of the matching documents by applying cluster algorithms on the search results. In Section 2 we describe approaches for finding information in the web including multi search engines, catalogues and clustering techniques. We give in this section a short overview of related work on clustering of web documents as well. Section 3 describes the architecture of MSEEC and introduces the required terminology. The cluster processing environment of MSEEC used to condense information is described in Section 3.2. Optionally

*Published in: Proceedings of the '99 Information Resources Management Association International Conference, Hearshey, PA, May, 1999.

linguistic processing (Section 3.2.1) can be used to filter and transform the input data for the cluster algorithms. Section 3.2.2 introduces a modified version of the *theme detection method* [4], Section 3.2.3 presents the novel *phrase detection method*. The generation of the cluster trees is described in Section 3.2.4. Section 4 gives a summary and presents some ideas for future developments.

2 Web based Search and Retrieval of Digital Documents

Generally the following approaches can be used to find information in the web:

(a) *Brute force search (no guidance):* Primary search engines can be used to search for web documents containing a certain query term. A web robot automatically collects (new or modified) web documents which are used to build an index. The search engine provides an interface to accept queries and uses the index to generate a list of references to the documents containing the search term. However, Lawrence and Giles [8] argue that currently no primary search engine indexes more than about one third of the indexable web.

(b) *Use of additional information to guide search: Web catalogues* are manually generated hierarchies of web information. A web catalogue provides access to documents that are categorised by humans (see e.g. [17]) such that similar documents can be found at a common place. Brin and Page [3] argue that human maintained web catalogues can cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics.

    Meta information assigned to web documents can be used to guide the search by properly classifying and structuring the information (e.g. RDF [7] or PICS [13]). Meta information based on RDF can be used [7] in resource discovery to provide better search engine capabilities, in cataloguing for describing the content and content relationships available at a particular web site, page, or digital library, by intelligent software agents to facilitate knowledge sharing and exchange, in content rating etc. However, Marchiori [11] argues that it will take some time before a reasonable number of people start using meta information to provide a better web classification and that currently no one can guarantee that a majority of the web documents will be ever properly classified via meta information.

(c) *On the fly structuring of results of (a) or (b) to improve accessibility:* The results from (a) or (b) can be a huge amount of matching documents that can overwhelm the user´s cognitive abilities. Grouping documents based on similarity measures could help to condense information by presenting only a symbol for a set of similar documents. A user can select a set of similar documents by means of the symbols for further processing. Clustering (see 2.2) can be used to find a set of similar documents and a suitable symbol (cluster title) for the document set.

2.1 Multi Search Engines

A *secondary search engine* uses the results of a primary search engine. A secondary search engine uses much less resources by using the already existing index of the primary search engine (e.g. bandwidth, disk storage and computational power to build an index is not needed).

Lawrence and Giles [8] further argue that combining the results of multiple primary search engines can significantly increase coverage. A secondary search engine using multiple primary search engines (called *multi search engine*) warrants the utilisation of the disjunct information covered by primary search engines. Furthermore it is possible to use multi search engines in such a manner that one primary search engine's advantage can overcome a disadvantage of another primary search engine (e.g. the primary search engine with the most recent pages may not be the most comprehensive). Moreover a multi search engine can provide an unique interface to different databases [14]. Thus it is possible to query primary search engines and arbitrary databases (e.g. special purpose databases) with the same query term (Note: from a point of view of an enterprise the combination of internal and external information can be provided).

A multi search engine should support following requirements. A user should not need to know specific properties (such as query syntax) of the primary search engines. The query term formulated by a user automatically has to be modified to be suitable for each related primary search engine. The basic logical operators (e.g. `AND`, `OR`) should be supported by the query interface. The elapsed time for querying primary search engines should be minimised. The results returned by the primary search engines should be merged and presented in a unique format and duplicate entities should be removed. The secondary search engine should be able to detect modifications of the query or result syntax of a primary search engine and should either adapt itself or provide an easy interface for the adoption to reduce the cost of maintenance. The architecture of a multi search engine should scale to provide adding or deleting of primary search engines.

2.2 Clustering of Search Results

Clustering is the process of grouping distinct entities based on similarities. Similar entities are joined to clusters which are joined with other entities or clusters. The result of the cluster processing are cluster trees. There are variuous criteria and measures that can be used to compute the similarities between entities. This section discusses cluster criteria which can be used to cluster web documents.

- *URL as cluster criterion:* An URL (Unique Resource Locator) [1] is used to reference web objects. Using an URL as a cluster criterion is based on the idea that URLs which partially match against each other refer to similar content. It is a spatial criterion specifying the locality of a web server and further a locality of a resource in a web server's information space. Resources with spatial closeness can be considered containing similar information. The domain name (a part of the URL) can be considered as a natural cluster. The top-level domain name (e.g. *.de*) can be used sometimes as an indicator for the language of the document.

- *Link structure as cluster criterion:* Web documents contain references (links) to other (parts of) web documents. These links point to related information that the author considered as relevant for the document. Using the link structure as a cluster criterion is based on the idea that documents referring each other (even over a link chain) contain similar information (see for example [2, 15]). The quality of this cluster criterion is potentially high if the links are assigned by a human. Another use for exploiting link information is described in [10] where the match of a document is determined by the content of the document itself but also by the contents of the documents that can be reached via links (which obtain lower weight).

- *Recurring text patterns as cluster criterion:* Using text patterns as cluster criterion is based on the idea that a set of web documents containing similar information also contain recurring text patterns (see for example [4]) or phrases.

In this paper we are investigating on the search of web documents and we use the output of the primary search engines for clustering. This output contains typically a few pieces of the text from the original document that we use for clustering. Therefore we concentrate on the algorithms based on text patterns and present a modified version of a theme detection method (Section 3.2.2) and a novel method for phrase detection (Section 3.2.3). In general other cluster algorithms could be added to our system as well that provide different measures of similarity.

3 A Multi Search Engine with Multiple Cluster Algorithms

MSEEC [12] is a novel multi search engine for web documents with post processing capabilities to merge and condense information in the form of cluster trees. The current implementation contains two cluster algorithms, but can be extended with other cluster algorithms in a flexible way. Most processing steps of MSEEC can be controlled through parameters. A user can switch on or off the use of a certain primary search engine or a cluster algorithm dynamically.

3.1 Architecture of MSEEC

The query interface of MSEEC (see Figure 1) accepts a query consisting of a query term (a search string with boolean operators), parameters for the multi search engine (selection of primary search engines, number of solutions), and various parameters controlling the cluster generation. These parameters are used for example in the linguistic processing of the results, for selecting and controlling of the cluster algorithms.

To simplify the query formulation we provide a simple user interface for the casual user or an advanced interface where all parameters are available to a sophisticated user. Figure 2 shows in the top section a user interface for the casual user.

Before the transmission of the query it is modified to be suitable for the specific primary search engine. The primary search engines are queried in parallel. The results of the primary search engines are collected and normalised to generate a *raw document descriptor* for each web document. A raw document descriptor is a tuple containing the obtained information for each document. It contains the reference to the full document (URL [1]), the title and a text description of the document (leading text fragment) and some other contextual information such as the name of the primary search engine etc. It is important to note that all attributes of the document descriptor must be returned by every selected search engine.

The raw document descriptors are passed to the cluster processing environment which is described below in more detail. In short it performs some linguistic processing on the raw document descriptors and invokes the selected cluster algorithms that generate as a result the cluster trees.

Figure 2 shows an example of a generated cluster tree. Each node of the tree represents the subsumed documents. The nodes are labelled with the cluster titles generated by the cluster algorithms and are implemented as a hypertext link to the contained documents. When such a link is activated the cluster contents are presented in the right half of the display (see Figure 2). Every documented is presented based on information of the raw document descriptor.

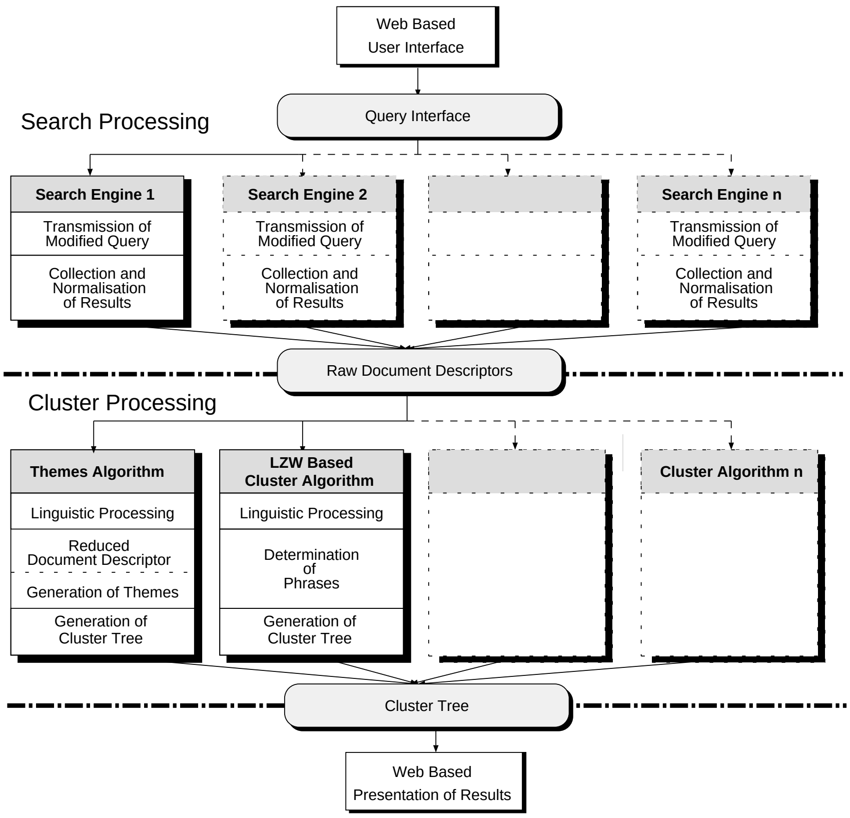

Figure 1: Architecture of MSEEC

3.2 The Cluster Processing Environment

The cluster processing environment of MSEEC consists of two major components, (a) the linguistic processing component and (b) the cluster algorithms. Both of these components can be controlled by the parameters supplied through the query, such that – for example – the linguistic processing can be performed in a different way (e.g. for a different language) or turned off.

The input of the cluster processing component of MSEEC are the normalised raw document descriptors returned from the multi search engine. These document descriptors are passed to the linguistic processing component that returns the processed document descriptors. These are again the input of the cluster algorithms. The clustering environment of MSEEC allows several cluster algorithms to be used. In our current implementation we use a modified version of the theme detection method (see Section 3.2.2) and a newly developed method for phrase detection (see Section 3.2.3).

3.2.1 Linguistic Processing of Document Descriptors

For information retrieval linguistic processing of the query or of text descriptors is an important step to improve the query results. The most important forms of linguistic processing are stemming, the elimination of non-words and explicit treatment of word senses.

Stemming [5] is used to reduce variant word forms to common roots which represent a much smaller vocabulary. The reduction to the reduced vocabulary improves the ability of the system to match the query and the document vocabulary. A stemming algorithm

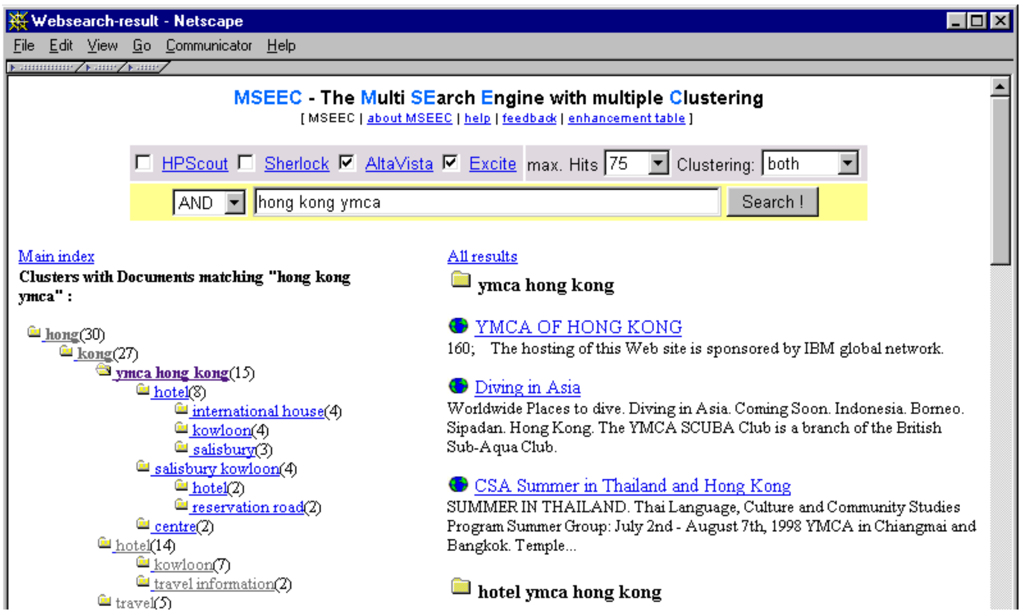

Figure 2: Web based Presentation of Condensed Information

processes a word and computes from the word the common root typically by removing word affixes (pre- or suffixes) using tables of common particles and some heuristics. Examples for stemming transformations are words→word, policies→policy, or presented→present. Note that the stemming algorithms and heuristics are language dependent. The number of possible word forms for English are for example much less than these for German or even Hungarian.

Another simple approach to reduce the vocabulary and to detect meaningful words is the elimination of non-words and words with very low semantic content. These words (such as articles or pronouns) occur very likely in about every document, which make it bad candidates for search terms and as words characterising the documents' content (which is important for clustering). A list of non-words for every supported languages can be easily kept in a table, which can be used to eliminate these words from the document descriptor. However if the search term contains solely non-words (as for example in the search for the pop group "the who") this elimination must be omitted.

Another important area of linguistic processing in information retrieval is the multiple word senses and the treatment of synonyms. The disambiguation of word senses can be used to differentiate between the various meanings of a word and can reduce the number of relevant matches [6] but requires a significant degree of text understanding. The treatment of synonyms and homonyms can be used to increase the number of matches.

The current implementation of MSEEC supports word stemming and non-word elimination. As noted above linguistic processing is language dependent. Therefore the linguistic processing component has to obtain a parameter to select the language. MSEEC uses linguistic processing only on the document descriptors and not on the query term.

3.2.2 Clustering based on the Modified Theme Detection Method

The *theme detection method* [4] is based on the idea to find a recurring text pattern (called a *theme*) in a collection of texts. A theme consists of one or more significant words describing the content of a text. We implemented a modified version of the theme detection method that differs from the original by (a) separating the linguistic processing from the theme detection method and by (b) generating multiple themes used for cluster titles rather than producing a single theme that characterises a document (or a set of documents) best.

For each processed document descriptor a *reduced document descriptors* is generated. In a first step, for each word contained in a document descriptor the ranking is determined. The ranking is the frequency of its occurrence in all document descriptors optionally weighted by the origin of the word. A word from the title is assumed to be more important than a word from the document body and is more likely to be kept in the reduced document descriptor. From this word list all words contained less frequently than a threshold are removed, since they do not contribute to clustering.

In the next step, the text in the document descriptor is reduced by keeping only the words with the highest rankings from the word list (bounded to a maximum number per document descriptor). The result is called the reduced document descriptor. Figure 3 shows a simplified example of a generation of a reduced document descriptor with 5 as the ranking threshold.

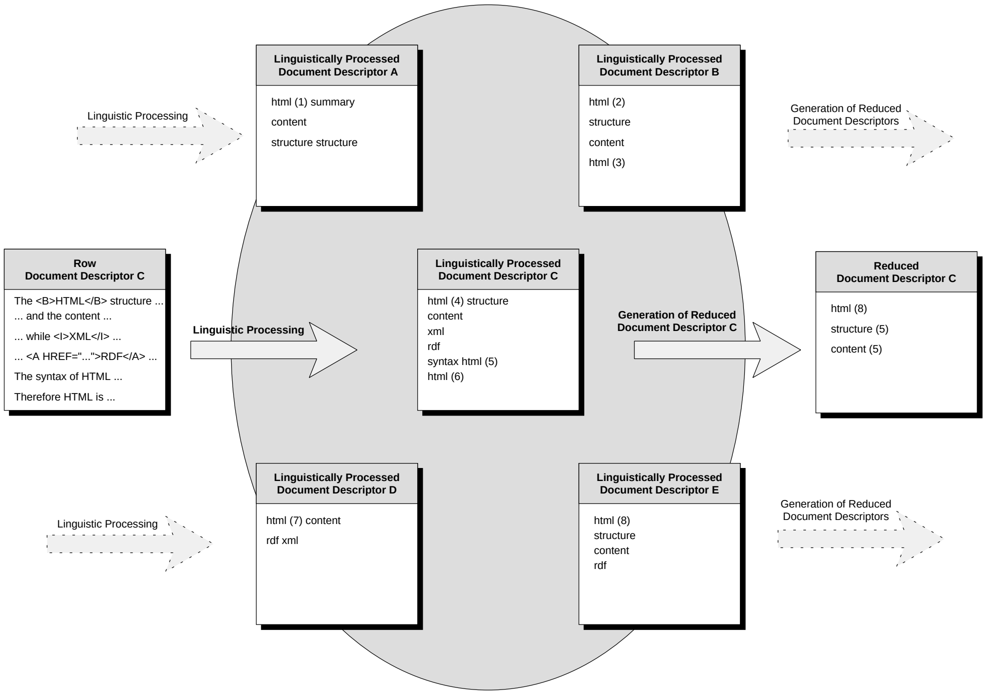

Figure 3: Example for the Generation of a Reduced Document Descriptor

The reduced document descriptors are the source for the theme detecting. From every reduced document descriptor the word with the highest ranking is selected and regarded as a theme candidate. Then, it is checked whether the theme candidate is contained in more than one reduced document descriptor (*valid theme candidate*). If yes, the word with the second highest ranking is added to the theme and builds a new theme candidate. If the test fails the theme candidate is dropped and the theme candidate without the last added word is kept.

This algorithm iterates through all words of the reduced document descriptor and builds a set of themes. For every theme the list of document descriptors that contains the theme are kept. The computed themes can be used as a cluster titles for the final step of the clustering algorithm where the cluster tree is built (see Section 3.2.4).

The theme detection module can be controlled by various parameters by the user to control the threshold, weight factors and an upper bound for number of computed themes.

3.2.3 Clustering based on the LZW-Method for Phrase Detection

Whereas the last section described theme detection, we will describe here a novel approach for phrase detection. Like a *theme*, a *phrase* consists of terms that occur in a text. The identified phrases can be used as cluster titles as well. Contrary to a theme the words in a phrase must occur in the same order in proximity in the document.

The purpose of the phrase detection algorithm is to identify similar or common used word phrases in different documents that can be used for its characterisation. We propose here a new method for phrase detection that is based on the widely used LZW compression algorithm [9, 16]. This compression algorithm identifies often occurring patterns in the data to encode these patterns in another, more compressed form to save storage. We used this method in MSEEC not for compression but for the determination of often recurring text pattern in the document descriptors.

The LZW compression algorithm as suggested by Lempel, Ziv, Welch [9, 16] is used for data compression. The LZW algorithm adapts dynamically to different data by building an encoding table during a single pass through the data (instead of using a fixed predefined encoding table). The LZW compression algorithm takes as input a steam of data (e.g. 8-bit characters). The output of the algorithm is a stream of encoded and compressed values (x^2 tokens, depending on size of encoding table). To proceed all needed steps an internal storage buffer (ISB) is needed to keep track of a list on input tokens, as well as a fixed size encoding table (ET) to encode the token lists.

```
Program Initialization
  Setup a fixed size encoding table (ET)
     (size is usually a x^2 size table with x = 10 or 16)
     and predefine the first 255 values with the default ASCII char set
  clear internal storage buffer (ISB)
  while input stream not empty
  do
    read token (T) from input stream
    if value of ISB appended by T not in ET then
        encode ISB  by using ET, and put this value to the output stream
        store ISB appended by T in ET in the next empty place
        ISB = T;
    else
        ISB  = ISB appended by T;
  done
  encode ISB by using ET, and put this value to the out stream
```

The LZW compression algorithm has several attributes that make it very valuable for the phrase detection: It detects often used token phrases automatically, the phrases are extended on the fly, the algorithm needs only a single pass which has the advantages of runtime efficiency and incrementality.

For *phrase detection* in documents we modified the LZW algorithm in various respects:

- *Words as basic units:* While the original algorithm works on characters the modified version works on words as input tokens. Therefore the coding table and the coding sequence are words and recurring text phrases.

- *Phrase rating;* In order to identify the most characterising phrases of a document we introduced a rating which is based on the frequency of the multiplied by an additional weight factor that is influence by the location of the word phrase. For example, phrases in the title of a document receive higher weights than phrases from other parts.

- *Enhanced encoding table:* It was necessary to extend the encoding table (ET) that needs additional fields to keep track of the rating of each word phrase, as well as the document descriptor of the detected word phrase.

- *Unbounded coding table size:* For the LZW compression method the first data has higher impact on the structure of the coding table than later data since the coding table has bounded size. We lifted this restriction to achieve more symmetry by allowing the creation of phrases from later documents as well.

In order to create the phrases for the dissemination of the document the phrase detection algorithm is applied on all document descriptors. For every phrase the a list of document descriptors is maintained pointing to the document descriptors containing the phrase. As mentioned above the phrase rating is updated after each detected phrase. After the processing of all document descriptors we select the phrases with the highest rankings as cluster title.

From our experience with users the phrase detection method returns more intuitive results for the cluster titles than the theme detection method. One reason is the more natural word order. The phrase detection method is also faster than the theme detection method. On the other hand the theme detection method returns better results for less related documents where the phrase detection method fails to detect interesting common phrases. For example for the texts “The Salisbury House Hong Kong” and “Hong Kong, The Salisbury House” the phrase detection method is able to detect the phrases “Hong Kong” and “Salisbury House” but is not able to connect these phrases which is no problem for the theme detection method.

3.2.4 Generation of Cluster Trees

The output of the theme detection and the phrase detection methods are passed to the same cluster tree generation module. The themes and phrases are used as cluster titles and determine solely the structuring of the cluster trees. Each node in a cluster tree represents the documents contained in the whole subtree. In order to keep the presentation of the clusters tree small it is possible to limit the number of nodes in a tree (by merging clusters with a high similarity) and to limit the depth of the tree (by merging clusters deeper than a certain depth in the tree). These parameters can be controlled by the user though query parameters.

The tree generation tries to find for every group of document descriptors (denoted by the cluster title) a parent (ancestor) node that is less specific, which means that it contains a subset of the words of the cluster title. The algorithm starts with an arbitrary (e.g. the longest) cluster title T_0 and eliminates in step 1 these cluster titles that contain words not in T_0 (see Figure 4). In step 2 the longest titles are kept as candidates for the parent cluster, in step 3 the cluster title with the most document descriptors is chosen as the parent cluster.

These three steps are performed for every cluster title. If no parent cluster could be determined the cluster title is regarded as a root of a tree. Therefore this algorithm results not in a single but multiple cluster trees (cluster forest).

The maximum depth of a cluster tree can be determined by a query parameter. If the specified depth is exceeded nodes from deeper subtrees are merged with the node at the maximum depth (see Figure 5).

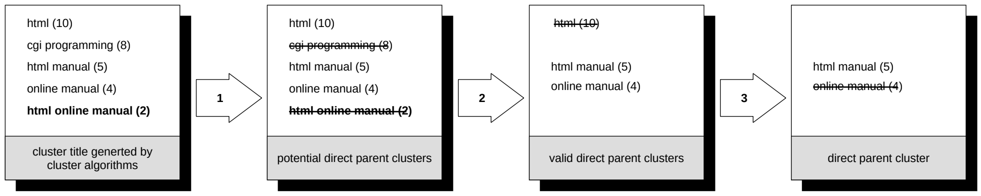

Figure 4: Example of finding a *direct parent cluster* of `html online manual`.

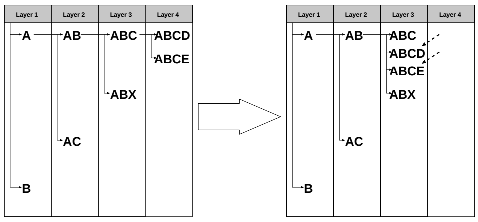

Figure 5: Limiting the Depth of the Cluster Tree.

4 Conclusion and Future Work

This paper presented a scalable architecture using a secondary search engine to query multiple information servers. The additional layer introduced by a secondary search engine can be used to provide a unique query and result interface to multiple primary search engines. The design of the system as a the secondary search engine reduces consumption of resources by using the index of primary search engine. The use of multiple primary search engines has the advantage that it can result in a higher coverage of indexed web documents and allows to select for certain queries different sources. This enables us for example to query databases in an intranet and the Internet with a single query.

Queries may return excessive amounts of matching documents that can overwhelm a user. We use cluster algorithms to give the user a better synopsis and some guidance to differntiate between the matches. We introduced a modified version of the theme detection method and a novel approach for phrase detection which are suitable to cluster the results returned by a secondary search engine. The cluster algorithms also can be combined in such a manner that one algorithm's advantages can overcome a disadvantage of another cluster algorithm.

Of course there are various areas where we think we can improve the system. One interesting area is to invest more on the users perception of the cluster algorithms, to allow the user to navigate through the information space based on cluster algorithms with different similarity measures. It should be possible to reach different types of clusters through links from the user interface.

Another area with high potential for improvements is the linguistic processing, which is currently based on simple heuristics. It appears to be useful to provide the user with mechanisms for specifying additional non-words that should be ignored for a query.

From our experiences we would expect even better results from the phrase detection algorithm by enhancing it to allow skipping of words. This could be implemented by changing the internal storage buffer to a stack of buffers. The depth of the stack would directly determine the maximum word skip distance.

To reduce the maintenance work for the multi search engine a more sophisticated pattern recognition would be of advantage to parse the results of the primary search engines.

References

[1] Tim Berners-Lee, Larry Masinter, and Mark McCahill. Uniform resource locators (URL). *RFC 1738*, December 1994. Network Working Group, Category: Standards Track, ftp://ftp.nic.de/pub/doc/rfc/rfc-1700-1799/rfc1738.txt.

[2] Rodrigo A. Botafogo. Cluster analysis for hypertext systems. In *ACM-SIGIR´93*, pages 116–125. ACM, 1993.

[3] Sergey Brin and Lawrence Page. The anatomy of large-scale hypertextual web search engine. In *Proceedings of the Seventh International World Wide Web Conference*, volume 30 of *Computer Networks and ISDN Systems*, pages 107–117, April 1998.

[4] David K.Y. Chiu and David R. Brooks. Detecting themes in web document descriptors. In *Proceedings of WebNet97 - World Conference of the WWW, Internet & Intranet*, pages 123–128. AACE, 1997.

[5] Robert Krovetz. Viewing morphology as a inference process. In *16th Annual ACM SIGIR Conference of Research and Development in Information Retrieval*, pages 191–202, 1993.

[6] Robert Krovetz and Bruce Croft. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems*, 10(2):115–141, April 1992.

[7] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax. Technical report, W3C, February 1998. W3C Working Draft, http://www.w3.org/TR/1998/WD-rdf-syntax-19980216.

[8] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98, April 1998.

[9] Abraham Lempel and Jacob Ziv. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, May 1977.

[10] Massimo Marchiori. The quest for correct information on the Web: Hyper search engines. In *WWW6 - The sixth international World Wide Web conference*, pages 265–276, 1997. http://www6.nttlabs.com/HyperNews/get/PAPER222.html.

[11] Massimo Marchiori. The limits of web metadata, and beyond. In *Proceedings of the Seventh International World Wide Web Conference*, volume 30 of *Computer Networks and ISDN Systems*, pages 1–9, April 1998.

[12] Mseec (current) home page. http://nestroy.wi-inf.uni-essen.de/MSEEC.html, July 1998.

[13] Martin Presler-Marshall, Christopher Evans, Clive D.W. Feather, Alex Hopmann, Martin Presler-Marshall, and Paul Resnick. PICSrules 1.1. Technical report, W3C, December 1997. W3C Recommendation, http://www.w3.org/TR/REC-PICSRules-971229.

[14] Erik Selberg and Oren Etzioni. Multi-service search and comparison using the MetaCrawler. In *Proceedings ot the 1995 World Wide Web Conference*, 1995.

[15] Ellen Spertus. Parasite: Mining structural information on the web. In *WWW6 - The sixth international world wide web conference*, pages 201–214, 1997. http://www6.nttlabs.com/HyperNews/get/PAPER206.html.

[16] Terry A. Welch. A technique for high performance data compression. *IEEE Computer*, 17(6):8–19, June 1984.

[17] Yahoo home page. http://www.yahoo.com/, July 1998.