Exploiting Structure for Intelligent Web Search

Udo Kruschwitz
Department of Computer Science
University of Essex
Wivenhoe Park, Colchester
CO4 3SQ
United Kingdom
udo@essex.ac.uk

Abstract

Together with the rapidly growing amount of online data we register an immense need for intelligent search engines that access a restricted amount of data as found in intranets or other limited domains. This sort of search engines must go beyond simple keyword indexing/matching, but they also have to be easily adaptable to new domains without huge costs. This paper presents a mechanism that addresses both of these points: first of all, the internal document structure is being used to extract concepts which impose a directory-like structure on the documents similar to those found in classified directories. Furthermore, this is done in an efficient way which is largely language independent and does not make assumptions about the document structure.

1 Overview

> *"The issue of extracting the structure of some text [...] is a challenging issue."* S. Abiteboul [1]

Finding information on the Web is normally a straightforward task. For most user requests the information can be located by applying a standard search engine using simple pattern matching techniques. However, by restricting the search to a well defined subdomain, like corporate intranets or university Web pages, this can become a tedious task. If no matching document can be found, the user is normally either left alone with a great number of partially matching documents or with no result at all. This is a well known problem and approaches exist to overcome that. But those approaches tend to rely very much on a given document structure or expensively created concept hierarchies. While this is appropriate for product catalogues and other applications where the information is stored in database formats, it is no help if the document collection is of heterogenous nature.

We present a mechanism that imposes a classification hierarchy on a set of Web pages by extracting *concept words* based on the structure detected in documents and relations between documents. The basic idea is to select keywords for documents which are found in more than one of the following contexts:

- *meta* information
- document *headings*
- document *title*
- *emphasised* parts of the document.

This allows us to separate important terms from less important ones. It works largely language independent because it exploits layout more than anything else. Furthermore it is cost effective. The extracted concepts function as classifications not just for individual documents but directories and linked documents as well, similar to those classifications in manually maintained Web directories like *Yahoo!*[1] or *LookSmart*[2]. With the presented work we revise and go beyond the preliminary thoughts described in [18].

The structure imposed on the data collection is employed in a dialogue system which can cope with user queries that do not retrieve documents or result in too many matches.

The remainder of this paper is structured as follows. First we discuss related work in some detail (section 2). The next section is about the *offline* processes of indexing the documents together with an example from our sample domain and explains how the classification is being built up using the results of the indexing process (section 3). Then we will look at how an *online* system uses these index tables

[1] `http://www.yahoo.com`
[2] `http://www.looksmart.com`

(section 4). Finally we report some implementational issues (section 5), recent experiences (section 6), and give a conclusion (section 7).

2 Related Work

Roughly speaking, any approach to solve the problem of finding the right information for a user's need from a set of semistructured documents either relies on enormous amounts of data or on structural information associated with the documents, neither of which we can take for granted in the context of this work. In the following we look at some related work in more detail.

Indexing documents has been a research topic in the information retrieval (*IR*) community for many years [28]. But documents are normally very long, contain little internal structure and collections are "typically measured in gigabytes" [38]. A Web track was introduced in the TREC series. One question for the *Small Web Task* is to find out whether link information in Web data can be used to obtain more effective search rankings than can be obtained using page content alone. It was found that hyperlinks did not result in any significant improvement [29]. And the overview of all results summarizes that no measurable benefit was gained on standard TREC retrieval measures through use of links [16].

There has also been work on layout analysis. One motivation is to convert printed collections of documents into their corresponding tagged electronic version. Autotag [32] is a system that aims at processing many types of documents based mainly on heuristic rules. It performs a physical followed by a logical analysis. The intention is "to capture every conceivable object that may add to a document's presentation". However, it does not go as far as to capture the semantic content of the documents.

Finding the logical structure of a document without knowing the actual documents or styles used in them is the focus of [31]. This is a two-step process, *segmentation* and *classification*. The segmentation process is primarily based on layout, contour and font shape information. The classification step is based on the comparison of the text segments to predefined structure prototypes, mainly by means of geometric cues. The only linguistic knowledge used in the described prototypes is the identification of typically appearing or non-appearing symbols. If some information about the *style* of the documents is known, then this can be incorporated in the processing steps as well.

Another relevant field of related work is document clustering. Like information retrieval approaches, clustering relies very much on large amounts of data. IBM's *TaxGen* text mining project aims at the automatic generation of a taxonomy for a large corpus of unstructured news wire documents [25]. A hierarchical clustering algorithm first builds the bottom clusters and then works its way upward forming higher-level clusters by grouping together related clusters. The clustering is based purely on linguistic elements in the documents, e.g. co-occurrences of words (*lexical affinities*) or names of people, organizations, locations, domain terms and other significant words and phrase from the text (*linguistic features*).

A recent example of conceptually indexing a document collection is *Keyphind* which is described in [15]. Machine learning techniques are applied in order to extract *keyphrases* from documents in the context of browsing digital libraries. This comes close to our idea of imposing a structure on the collection by extracting "important" phrases from each document, but in *Keyphind* the documents are much longer and furthermore a manually tagged training corpus is needed to build the classifier. *Extractor* is a similar system for extracting keyphrases using supervised learning [34, 35].

Clustering is also being used for *concept-based* relevance feedback for Web information retrieval [6]. Following a user query the retrieved documents are organised into conceptual groups. Unlike in our approach this structure is not extracted for the indexed domain but for the search results. Another example is *Grouper*, an interface to a metasearch engine which dynamically groups the search results into clusters labeled by phrases extracted from the snippets that the search engines returned with the retrieved documents [37]. These applications are quite different to traditional *offline* clustering approaches where the whole document collection is clustered, rather than a number of retrieved documents. The motivation for Zamir *et al.* is that post-retrieval document clustering has been shown to produce superior results. Moreover, clustering a few thousand documents is much quicker than doing this for millions of documents. However, for our problem post-retrieval clustering would miss the point since a classification structure for the whole data set will have to be built in advance.

Document classification is closely related to clustering, the difference to clustering is that predefined categories exist, normally manually constructed, and after a training phase new documents can be classified on-the-fly like in [7]. Northernlight[3] is an example of a search engine where the results are grouped in dynamically created categories, so called *Custom Search Folders* which help a user to narrow down the search in case of too many matches. The search covers the whole Internet and thus has access to much more data than what we assume. Moreover, *Custom Search Folders* are not offered if the query did not retrieve answers and needs to be relaxed, e.g. *Minox dealer in Colchester* or *Minox Colchester*.

Ontologies and customised versions of existing language resources like *WordNet* [24] are being successfully em-

[3] `http://www.northernlight.com`

ployed to search product catalogues and other document collections held in relational databases [14, 12]. Part of that research is the actual construction of ontologies [9]. The cost to create the resources can be enormous and it is difficult to apply these solutions to other domains where the document structure is not known in advance.

Quite a different way of dealing with the *semistructured* information on the Web is to retain the structure and store the data in graph-structured data models by means of *wrappers* [20, 22, 27, 33]. It is not just about retaining but also capturing the structure, for example to transform HTML documents into XML or other formats. Databases and query languages have been developed for this, the most prominent database system is Stanford's Lore [23]. But the drawback is that the indexing depends very much on a formally defined structure of the expected input. There has so far also been little experience using *semistructured* databases for substantial applications [30].

The *Clever Project* [4] looks at topic related search. The domain is the complete Internet and the problem to be solved is filtering out those pages which are truly relevant for a specific topic, i.e. the problem of too many matches. *Authorities* and *hubs* are distinguished, places that are either relevant or are collections of links to those pages, respectively. *Authorities* and *hubs* are found by purely analysing the connections between Web pages. This is based on the *HITS* algorithm developed by Kleinberg [17]. A modification of that algorithm was presented in [5], again as part of the *Clever Project*. The extraction of *hubs* and *authorities* is performed by a combination of text and link analysis. The text in a window around the hyperlinks is evaluated and the initial weight of a hyperlink which is set before the iterative calculation starts is increased with the amount of topic-related text.

Hyperlink Vector Voting is introduced in [21]. Rather than depending on the words appearing in the documents themselves it uses the content of hyperlinks to a document to rank its relevance to the query terms. That overcomes the problem of spamming within Web pages and seems appropriate for Internet wide search but would cause problems in subdomains where the number of links between documents is much smaller and certainly problems will occur for those pages which are referred to by a small number of documents only or no documents at all. One can go further by not just using the anchor text but also additional structural information found in the context of the hyperlink as Fürnkranz does [13]. For the task of classifying pages using a given set of classes he reports that it is possible to classify documents more reliably with information originating from pages that point to the document than with features that are derived from the document text itself.

The *Cha-Cha* system has been developed for *intranets*. It imposes an organisation on search results by recording the shortest paths to the root node in terms of hyperlinks [8]. But this is only applied once results could be found by using the search engine. It exploits hyperlinks but ignores the internal structure of the indexed Web pages.

However, internal structure is being incorporated more and more in standard Web search engines. A prominent example is *Google*[4] whose development was inspired by the idea of improving the quality of search as opposed to efficiency [3]. It makes use of both link structure and anchor text. Each page gets a ranking value depending on how many other pages reference to it. Moreover the system associates Web pages not just with the text in it but also with the text found in anchors linking to this page from elsewhere. This makes it possible to return pages that have not been crawled. Furthermore *Google* keeps track of some visual presentation details like font size to give for example heavier weight to words which appear in a larger font than the rest of the document. However, *Google* is not of any help if the query cannot be matched against at least one document.

The YPA is a system that addresses a similar problem as described here where a user is looking for advertisers that could provide certain goods or services [10, 11]. The documents are not Web pages but advertisements from BT's *Yellow Pages* and *Talking Pages*[5]. But rather than having to build classifications and cross-references these were already an implicit part of the input data sources.

3 Indexing Web Pages

As we have seen, classifications have been built, some of them being impressively large, and they do work fine in either domain independent context with lots of data to index like the Web or in specialized applications with manually tailored classifications. But the semantic content of a document collection can vary, it may be very domain dependent with no classifications ready to hand. It may change over time, and the amount of data may not allow a reliable classification. In these cases a structure constructed from the actual data seems more desirable.

The aim is to extract a small number of important keywords from the documents which can then function as classifications, because a relation between two documents (i.e. a hyperlink) can as well be interpreted as a relation between the extracted concepts.

The important assumption we make here is that not so much the frequency of a term is of interest, but where it is found in the document. More specifically we assume, if a term can be found in two or more different contexts (e.g. different types of tags) in the same document, then we call it a *concept*. This is quite different to standard *IR* paradigms.

[4] `http://www.google.com`

[5] Yellow Pages® and Talking Pages® are registered trade marks of British Telecommunications plc in the United Kingdom.

For Web pages we distinguish five types of contexts where candidate keywords can occur:

- *meta* tags
- *heading* tags
- *document title* tags
- all sorts of *emphasis* (like bold font, italics etc.)
- any free text in the document.

If a keyword or a phrase is found in at least two of the first four contexts, it is selected as a *concept* for that document, given it is not filtered as a stopword. This means that we ignore free text completely and do not rely on meta tags which are used infrequently and can be a source of spamming rather than conceptual information. It also reduces the number of keywords selected for a document to a minimum, possibly no keyword is selected at all. However, a fall-back strategy is always to go back to the most frequent keywords found in the document or title information etc. We assume that nouns are the most content bearing words, which is why we ignore all other keywords altogether, unless they can be extracted as part of a phrase which would contain at least one noun.

To illustrate the mechanisms, we will pick some actual examples using documents found on the Web server of the University of Essex.

3.1 Example - Essex University Web Pages

First of all, for this domain the list of stopwords was expanded to include appropriate terms like *university*, *colchester*, *essex* etc. Furthermore, we use unstemmed word forms in the examples, but in the indexing process we do use a stemmer.

A page entitled *About the University of Essex*[6] gives an overview of the University and has hyperlinks to two other files in the same directory: *Travel information*[7] and *Campus guide: finding your way around*[8]. Figure 1 shows what we get if we select the four most frequent keywords in the documents. The keywords selected for the whole directory are the most frequent keywords in all three documents.

Compared to that, figure 2 shows the indexes that were selected applying our *concept approach*. The terms associated with the directory `/about` are the most frequent concepts selected for all the files listed in the directory.

Most strikingly, this demonstrates how frequent words are not necessarily the most desirable choices for a query system: the words *square* and *kilometre* are two examples

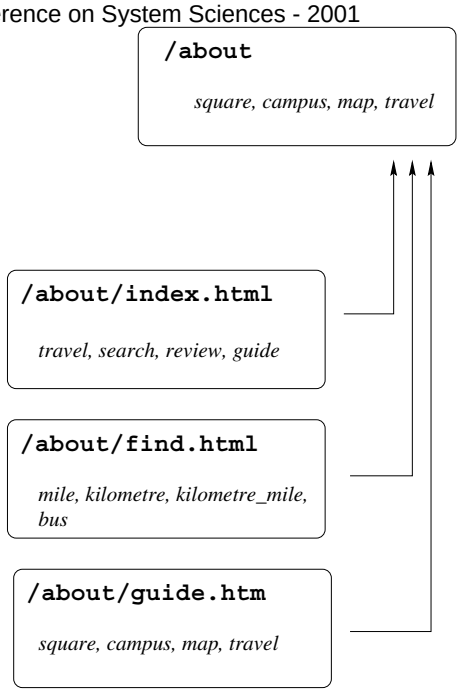

Figure 1. Most frequent keywords

which we rather do not want to see as classifications but they cannot just be added to the stopword list to work around it.

In addition we also investigated which keywords would be selected for a document if we ignored the actual text and structure of that document and only looked at the content of all pages referring to it by means of a hyperlink. The results differ quite significantly from case to case, which is why this approach was considered less appropriate.

3.2 From Classifications to Cross-References

The extraction of concept words leads to two types of classification, first of all each document is classified under all its selected concepts. Furthermore, concepts can be associated with (the starting page of) directories by abstracting from single documents and interpreting them as a number of items grouped together. The most frequent concepts in all those documents are selected to represent the content of the directory. Hence, directories are now treated like normal documents.

But the structure in classified directories like *Yellow Pages* is richer than that. There are links from one heading to others, i.e. *cross-references*. Typically, two types of links exist, the so called *see* and *see-also* links. The first one is used for classifications under which no addresses are actually listed, e.g. for the Colchester area the local directory lists nothing under *Slaughterhouses* but contains a *see* reference to *Abattoirs* and *Horse Slaughterers*. The second

[6] `http://www.essex.ac.uk/about/index.html`
[7] `http://www.essex.ac.uk/about/find.html`
[8] `http://www.essex.ac.uk/about/guide.htm`

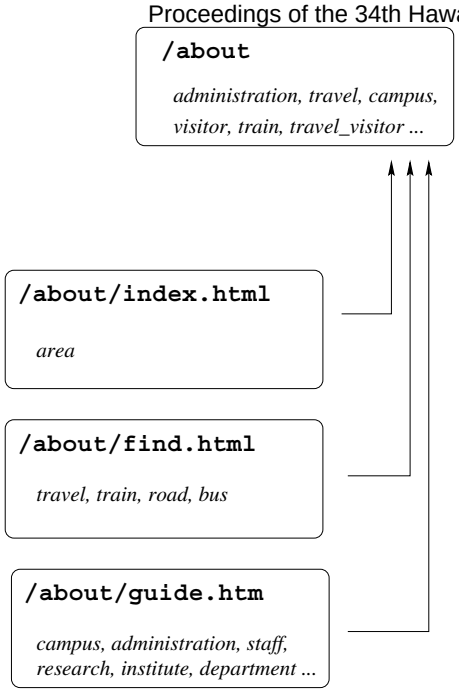

Figure 2. Selected concepts

type of link is used to refer the user to other related businesses, for example *Roofing Services* has a *see-also* link to *Builders*.

In this context we will only need an equivalent to the *see-also* link because the indexing process is data oriented and we build up a structure bottom-up as opposed to classifying documents in an already existing classification scheme. That means there are no empty classifications.

We define *cross-references* on the basis of *related concepts* which are pairs of concepts that were extracted for the same document or directory. For example, the two terms *administration* and *visitor* (see figure 2) are *related concepts*, so are *road* and *bus*. We do not intend to capture the semantic content of this relation, but merely the fact that there is some correlation. In the next section it will be shown how the user actually decides how this information has to be interpreted by either choosing or ignoring certain related words.

However, there is a problem of overgeneration. Consequently, *cross-references* are only a subset all *related concepts*, those whose frequency falls between certain thresholds (which depend on the domain and the size of data to be indexed). That means, in the above example the pair *road* and *bus* is excluded. Alternatively, one could define *cross-references* based on related concepts extracted for directories as opposed to single documents.

4 A Dialogue System for Web Search

So far we discussed how an index database is being constructed. For most *online* queries this process might look like a bit over the top. And that is correct as long as a simple keyword lookup returns an acceptable number of documents. Yet, experience has shown that problems do turn up when too many or no documents at all are found for a user query [19]. General queries tend to retrieve a large number of matches even in small domains. At the same time it is obvious that the smaller the domain, the more likely it is that the user query cannot be matched precisely. Therefore we use the index database as a backend in a dialogue system. Note, that for successful queries it is not necessary to initiate a dialogue step and the concept hierarchy needs to be applied only for ranking purposes.

But how can the classification structure or a dialogue system help to solve the observed problems? Let us start with an example for too many matches. Querying the University's Web pages[9] with the simple request *visitor information* leads, not surprisingly, to more than 50 matching documents. After reading through ten each time one realises that only the very last screen contains a link that refers to an appropriate page. With the presented conceptual approach we reduce the number of matches by finding the best matching documents, i.e. those that list the term *visitor* as a concept (the word *information* is a stopword). This is not a frequent concept, but as described in the earlier example (see figure 2) it is asociated with the directory whose starting page is *About the University of Essex*[10], a perfect match, despite the fact that the word *visitor* cannot even be found in this page.

Alternatively, we could exploit the *cross-references*. This would involve a dialogue step by presenting the user with some options of how to constrain the search if the query was indeed too general. The user could choose one or more terms which would be added to the query for resubmission. In the above example the user is offered to continue the search by selecting *cross-references* for *visitor* which include *administration*, *travel* and *campus*. Compared to the search in a classified directory this means that advertisements are retrieved that are listed under various classifications at the same time. Of course, the user can also decide that the query is specific enough and force the system to display the matches, which are ranked based on the concept hierarchy.

For too few matches or no matches at all one could try to relax automatically, but this is not always helpful. One thing that can happen is that automatic query term expansion can worsen precision quite dramatically [19]. And it is not always obvious why certain documents were retrieved

[9] `http://www2.essex.ac.uk/search/`
[10] `http://www.essex.ac.uk/about/index.html`

when automatic relaxation was applied. What we do is to incorporate a dialogue step here which lets the user decide what *sort of relaxation* should be performed. Not all of the following options are always applicable, but in general the user can choose among as many ways to relax as appropriate in the actual dialogue step:

- Search for partial matches. The user decides which parts of the original query should be used to continue.
- Expand or modify the query by exploiting *cross-references*. Let the user decide which *bag-of-words* should be chosen to replace a part in the original query. E.g. in a search for *Wivenhoe House hotel travel information* the most appropriate relaxation could be to choose *bus* and *train* from the list of *cross-references* for the term *travel* and continue the search by looking for *bus OR train* instead of *travel*.
- Find documents that match part of the query and the other part is matched by documents that refer to it by a *hyperlink*.
- Display the starting pages of directories which contain matches for the query (but not in a single document).

This is an extendable list of relaxations. It also depends on the context whether one or the other option will be displayed, for example some of the relaxations can be checked in advance to see whether they help at all. Additional domain knowledge can enhance the relaxation in specific applications.

Here we come back to a claim we made earlier, that it is not so interesting to find out *why* two concepts are related but *that* they are. The fact that they have some semantic relation might be domain dependent rather than language specific. Moreover, this link can be useful for one query and irrelevant for a second query involving the same terms. The dialogue system allows the user to decide.

5 Implementational Issues

The gathering robot starts at some root point(s) and collects in a breadth-first strategy all relevant Web pages that can be found. It currently ignores all documents that are not in HTML format or located in a different domain. The gathered documents are passed to the indexing component.

The robot as well as most of the index construction programming is done in Perl making use of existing modules (LWP, HTML, URI, Text etc.). For the indexing process we also use the Brill tagger [2] and *WordNet*. The Porter stemmer [26] is used to reduce words to their stems. Other resources would have to be applied for languages other than English. An anecdotal note about the stemmer: while there are a number of implementations of the Porter stemmer available, they all seem to interpret the algorithm somehow differently. Just one example is the word *artificial* which is supposed to be stemmed to *artifici*. The Perl modules Text::English or Lingua::Stem return *artificy* and *artificial*, respectively. We decided to apply the C and Perl implementations provided by Porter himself.[11]

The Web based *online* dialogue system runs as a *Sicstus* Prolog executable accessed via sockets. It was initially in parts based on the YPA and subsequently reimplemented. The external databases we are using are *mSQL* and *Oracle*.

Our platform is a *Sparc*station Ultra 10 with 128 MB working memory running Solaris 2.6.

We focus on indexing a sample subdomain of the Web, the Web pages of the University of Essex. This is an arbitrary choice and once the framework has been fully implemented we plan to validate the approach using different domains.

Figure 3 shows the starting page for the online search system.

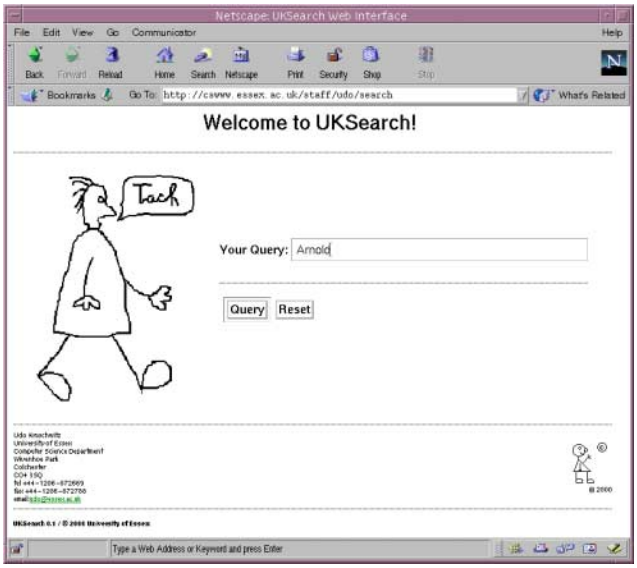

Figure 3. User interface

6 Recent Experiences and Results

Since the fine tuning of the algorithm is not yet complete we will not present the results of an evaluation. However, initial tests showed some significant results. First of all, a classification hierarchy as described in this paper can be built with very little effort. This can usefully be applied in cases where no pages or too many pages would be retrieved

[11] `http://www.muscat.com/~martin/stem.html`

following a user query. Another result concerns the retrieval results for successful queries. Initially we displayed only the most significant matches, i.e. those pages for which the query terms are listed as concepts. But that made some pages come up again and again. If there is a large number of matching Web pages, it seems more useful to offer the user related concepts to constrain the query or ask whether he or she wants to see the most significant matches only. Figure 4 is a screenshot of a sample query. If the user chooses to see the *most significant matches*, then twelve Web pages relating to three different academics working in different departments are displayed, while each of the other options leads to a larger number of pages referring only to a single person.

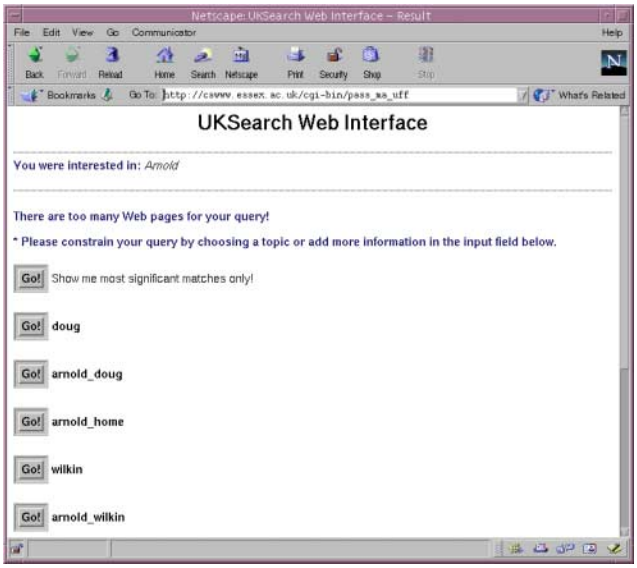

Figure 4. Selected concepts

We also want to present some statistics to demonstrate the amount of data extracted and indexed. In the University of Essex domain we currently index about 23,000 Web pages, which are all HTML or ASCII pages accessible from the starting page[12] presuming the robot exclusion files do not prevent robots to crawl them. The following table contains statistics for our domain. In the table we list *main concepts* alongside *concepts* which are those terms that were found in three rather than two contexts in a single document. We do this to highlight that this is quite a strong restriction and we do not use *main concepts* in our prototype.

[12] `http://www.essex.ac.uk`

Number of indexed pages	23,243
Number of free text keyword indexes	4,777,444
Number of selected concepts	11,096
Number of selected main concepts	2,395
Number of pages with concepts	17,646
Number of hyperlinks	266,090

That gives the following breakdown:

Average number of keywords per page	205.54
Average number of concepts per page	0.48
Percentage of pages with no concepts	24%

We also want to give a list of the most frequently selected concepts (stemmed forms) that were found in the Web pages of our sample domain:

Concept	Number of Occurrances
work	504
paper	479
paper_work	471
research	182
linux	128
histori	124
model	96
kde	92
studi	76
new	67

These are the most frequent main concepts selected:

Main Concept	Number of Occurrances
studi	1,555
cours	1,408
research	750
paper	696
work	605
vision	554
paper_work	523
unit	495
histori	491
project_vision	448

But note that the most frequent concepts are not always the most appropriate to constrain or relax a given user query. Constraining a query by adding very general terms will often result in too many matching documents again. An upper threshold is therefore used in the current prototype which prevents too frequent concepts to be applied at all.

A major task remains the comprehensive evaluation of the developed system against standard techniques. For that we will be using the logfiles that contain all queries submitted to the University search engine in a period of 24 hours. 42,270 queries were collected on the 15 August 2000. A

sample of those queries will be chosen and tested against the system accessing two different index databases, one with indices and concepts as described in this paper, the other one containing only the flat keyword indices (very much like a standard search engine). The users will have to fill in a user satisfaction form similar to the one used in [36].

In order to improve the system it will also be interesting to see how the automatically extracted keywords can be combined with some pre-existing classification hierarchies.

7 Conclusions

We presented a method that allows more sophisticated indexing of a Web site or other subdomains of the Web which captures the structure and relations between documents. By doing this we retain and uncover as much information as possible without having to make assumptions about the document structure. The resulting index database can be applied in a dialogue system that navigates a user through various options if a query could not be answered otherwise. In any other case it provides a way of judging the relevance of documents based on the concepts they relate to, i.e. the classifications that were automatically created for them.

We also see a possible application in the semi-automatic creation of classification hierarchies where the documents are processed as described and the resulting classification structures are then manually being refined using a toolbench.

Our experiments have so far only involved HTML documents, but for non-HTML documents it should be possible to use similar approaches as long as some layout analysis is applied which preprocesses the documents in order to mark the various sorts of layout being found.

Acknowledgements

Thanks to the anonymous reviewers whose comments are hopefully addressed by this final version of the paper. Thanks also to Sam Steel who spent time discussing the issues of this paper while he should have been preparing his slides for the next term.

References

[1] S. Abiteboul. Querying Semi-Structured Data (invited talk). In *Proceedings of the 6^{th} International Conference on Database Theory (ICDT)*, pages 1–18, Delphi, Greece, 1997.

[2] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy, 1992.

[3] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, Brisbane, 1998.

[4] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Hypersearching the Web. *Scientific American*, June, 1999.

[5] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, Brisbane, 1998.

[6] C. H. Chang and C. C. Hsu. Enabling Concept-Based Relevance Feedback for Information Retrieval on the WWW. *IEEE Transactions on Knowledge and Data Engineering*, July/August, 1999.

[7] H. Chen and S. T. Dumais. Bringing order to the web: Automatically categorizing search results. In *Proceedings of CHI'00, Human Factors in Computing Systems*, Den Haag, 2000.

[8] M. Chen, M. Hearst, J. Hong, and J. Lin. Cha-Cha: A System for Organizing Intranet Search Results. In *Proceedings of the 2^{nd} USENIX Symposium on Internet Technologies and Systems (USITS)*, Boulder, CO, 1999.

[9] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98) and the Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 509–516, Madison, Wisconsin, 1998.

[10] A. De Roeck, U. Kruschwitz, P. Neal, P. Scott, S. Steel, R. Turner, and N. Webb. YPA - an intelligent directory enquiry assistant. *BT Technology Journal*, 16(3):145–155, 1998.

[11] A. De Roeck, U. Kruschwitz, P. Scott, S. Steel, R. Turner, and N. Webb. The YPA - An Assistant for Classified Directory Enquiries. In B. Azvine, N. Azarmi, and D. Nauck, editors, *Intelligent Systems and Soft Computing: Prospects, Tools and Applications*, Lecture Notes in Artificial Intelligence 1804, pages 239–258. Springer Verlag, 2000.

[12] S. Flank. A layered approach to NLP-based Information Retrieval. In *Proceedings of the 36^{th} ACL and the 17^{th} COLING Conferences*, pages 397–403, Montreal, 1998.

[13] J. Fürnkranz. Exploiting Structural Information for Text Classification on the WWW. In *Proceedings of the 3^{rd} Symposium on Intelligent Data Analysis (IDA-99)*, Amsterdam, 1999. Springer Verlag.

[14] N. Guarino, C. Masolo, and G. Vetere. OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems*, May/June:70–80, 1999.

[15] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27:81–104, 1999.

[16] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the TREC-8 Web Track. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, Gaithersburg, Maryland, 1999.

[17] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*. American Association for Artificial Intelligence, 1998.

[18] U. Kruschwitz. UKSearch - Web Search with Knowledge-Rich Indices. In *Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search*, Technical Report WS-00-01, pages 41–45, Austin, TX, 2000. AAAI Press.

[19] U. Kruschwitz, A. De Roeck, P. Scott, S. Steel, R. Turner, and N. Webb. Extracting Semistructured Data - Lessons Learnt. In *Natural Language Processing - NLP2000: Second International Conference*, Lecture Notes in Artificial Intelligence 1835, pages 406–417. Springer Verlag, 2000.

[20] N. Kushmerick, D. Weld, and B. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of IJCAI-97*, Nagoya, 1997.

[21] Y. Li. Towards a Qualitative Search Engine. *IEEE Internet Computing*, July/August:24–29, 1998.

[22] D. Mattox, L. Seligman, and K. Smith. Rapper: a Wrapper Generator with Linguistic Knowledge. In *Proceedings of the 2nd International Workshop on Web Information and Data Management*, Kansas City, MO, 1999.

[23] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3):50–66, 1997.

[24] G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 1990. (Special Issue).

[25] A. Müller, J. Dörre, P. Gerstl, and R. Seiffert. The TaxGen Framework: Automating the Generation of a Taxonomy for a Large Document Collection. In *Proceedings of the 32nd Hawaii International Conference on System Sciences (HICSS)*, Maui, Hawaii, 1999.

[26] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[27] A. Sahuguet and F. Aznavant. Looking at the Web through XML glasses. In *Proceedings of the 4th International Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, 1999.

[28] G. Salton and M. J. McGill, editors. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York, 1983.

[29] J. Savoy and J. Picard. Report on the TREC-8 Experiment: Searching on the Web and in Distributed Collections. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, Gaithersburg, Maryland, 1999.

[30] L. Seligman, K. Smith, I. Mani, and B. Gates. Databases for Semistructured Data: How Useful are They? (position paper). In *Proceedings of the International Workshop on Knowledge Representation and Databases (KRDB-98) at ACM-SIGMOD*, Seattle, WA, 1998.

[31] K. Summers. *Automatic Discovery of Logical Document Structure*. PhD thesis, Cornell University, 1998.

[32] K. Taghva, A. Condit, and J. Borsack. Autotag: A Tool for Creating Structured Document Collections from Printed Material. Technical Report TR 94-11, Information Science Research Institute, University of Nevada, 1994.

[33] B. Thomas. Logic Programs for Intelligent Web Search. In *Proceedings of the 11th International Symposium on Methodologies for Intelligent Systems (ISMIS'99)*, Warsaw, Poland, 1999.

[34] P. D. Turney. Extraction of Keyphrases from Text. Technical Report ERB-1057, National Research Council of Canada, Institute for Information Technology, 1999.

[35] P. D. Turney. Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, 2(4):303–336, 2000.

[36] N. Webb, A. De Roeck, U. Kruschwitz, P. Scott, S. Steel, and R. Turner. Evaluating a Natural Language Dialogue System: Results and Experiences. In *Proceedings of the Workshop "From Spoken Dialogue to Full Natural Interactive Dialogue - Theory, Empirical Analysis and Evaluation" (at the 2nd International Conference on Language Resources and Evaluation LREC2000)*, Athens, Greece, 2000.

[37] O. Zamir and O. Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. In *Proceedings of the Eighth International World Wide Web Conference (WWW8)*, Toronto, 1999.

[38] C. Zhai. Fast Statistical Parsing of Noun Phrases for Document Indexing. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington DC, 1997.