On-line Resource Discovery Using Natural Language

Osmar R. Zaïane Andrew Fall Stephen Rochefort
Veronica Dahl Paul Tarau
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6
{zaiane, fall, srochefo, veronica, tarau}@cs.sfu.ca

Abstract

With huge amounts of information connected to the global information network (Internet), efficient and effective discovery of resources from the "global information base" has become an imminent research issue, especially with the advent of the Information Highway. This article proposes the use of novel Artificial Intelligence and Database techniques (Assumption Grammars, Concept Hierarchies, Multi-Layered Databases, Intelligent Agents) for intelligently searching information pertaining to a specific industry on the web.

Keywords

Information Retrieval, Concept Classification, Concept Hierarchy, Concept-Based Indexing, Natural Language Processing, Internet, Search Engine

1 Introduction

Global information systems, such as the Internet, are quickly turning into major information resources. Anyone who wants to make information available can do so by simply placing a file in a designated location on their Internet-connected computer. They can then announce that the file exists, and other users can quickly and easily retrieve it for their own use. New web sites spawn every day. It has been reported that the World-Wide Web (WWW) contains more than 54 million documents[1]. There are already electronically-available academic journals, stock market quotations and weather reports; the future may see virtual libraries, universities and co-operatives, especially as electronic highways are opened to ordinary people and business activities. It is foreseeable that in the near future, industry, as well as government and academia, will also become more dependent upon such electronic resources.

However, one useful thing that this "electronic library" lacks is a good card catalogue system. There are software tools, such as the *World Wide Web Worm* [MCBR94], *Lycos*[2], *Excite*[3] and others, which can find articles containing user-supplied strings of characters; it is usually possible to locate the e-mail address of any author. However, a user attempting to do a subject search of the available documents will find it very difficult. Most search engines use spider-based indexing techniques like RBSE [EICH94] which systematically crawl the web to access as many on-line documents as possible. These techniques, in general, because they emphasize on speed, not only flood the network and servers but also lose the structure and context of documents. AltaVista, which claims to have the fastest indexing scheme, spends six weeks to go over all accessible documents on the WWW. Other indexing solutions, like ALIWEB [KOST94] or Harvest [BOWM94], behave well on the network but still struggle with the difficulty to isolate information with relevant context.

In physical libraries, skilled cataloguers handle the problem of trying to classify the subjects of the books they receive from publishers, working from standard lists of subject headings. On the Internet however, users supply documents themselves. The rapidity with which the available information grows, changes and is made obsolete, makes professional cataloguing of documents infeasible. The document providers are also notoriously lax at taking the time and effort to supply subject classifications for their information, let alone complete and standardized ones.

[1]HotBot search engine (http://www.hotbot.com) claims to have indexed more than 54 million documents.
[2]Lycos is available at http://www.lycos.com
[3]Excite is available at http://www.excite.com

It seems that what is needed is a computational agent which will at least approximately classify documents according to their subjects, as a first step toward a subject index of the library that is the Web. Such an agent, an "automated librarian", would read and assign documents to appropiate categories.

Many search engines (programs and services designed to help users search for documents of interest to them) have been implemented for the WWW. Netcreations[4], which has a URL announcement service, lists 402[5] different search engines and directories on the Web. We know of none that does anything but textual pattern-matching on the titles and headers of documents or the entire text of documents, except Excite which claims to use some concept classifications.

The approach we propose in this paper uses novel Artificial Intelligence and Database techniques to extract relevant topics from a text document and to categorize the document in concise indices. We show in this paper how we can use this technique to index documents in a global information network (internet/intranet), and how the same approach can be used to extract the pertinent interest subjects from the user's query before searching for the relevant documents.

The remainder of this paper is organized as follows: in section 2, we review some problems that face internet users, and discuss how search engines index on-line documents; our approach to solve the problem of poor indices is presented in section 3; in section 4, we describe our implementation effort; finally, section 5 summarizes our conclusions.

2 Today's Search Engines Will Kick the Bucket

How often have you been unsuccessful in finding what you wanted in an on-line search because the words you used failed to match words in the material that you needed? The frustration is greater when an existing document, directly relevant to your quest, does not appear in the list presented by the search engine as a response to your query. When presenting idiomatic phrases, like "kick the bucket", search engines usually come-up with hilarious answers. The causes are many. The important factors that make a search engine effective are the way documents are analyzed and indexed, the techniques provided to the user to submit queries, and the ranking method used to sort the list of documents in the response list.

[4]http://www.netcreations.com
[5]The list is accessible at http://www.netcreations.com/postmaster/thelist.html

The techniques used to index on-line documents are simple and crude. Because of the dynamic nature and the size of the WWW, speed and efficiency of these techniques are primordial. Some search engines index documents based on the words in their titles or headers (i.e. section titles). This method is fast, generates small indexes, but is inaccurate since it may miss pertinent topics from the main body of the document. Other techniques simply take all the words from a document and index the document based on all these words, creating an inverted index. These techniques produce very big indices, from 10 to 40% of the document original size, and are more accurate, but still generate a lot of *noise*[6] in query answers. Some variations of the "index all words" technique tend to reduce the number of words by eliminating "empty" words like articles (i.e. the, a, etc.) or common verbs (i.e. is, do, have, etc.), or aggregating words from the same canonical form (e.g.: clearing, cleared, clears, clear). This reduces the size of the inverted index and thus accelerates the search. The aggregation of words from the same canonical form reduces *silence*[7]. Some search engines attempt to reduce the silence effect (without adding noise) by aggregating some common synonyms. Noise and Silence are also known as *recall*[8] and *precision*[9].

The means provided to the user to submit queries is usually standard keyword based. Some advanced interfaces allow conjunctions of keywords, combinations of disjunctions of keywords, and even negations. The boolean combination of keywords is evaluated and matched with the words in the inverted index, built in advance, to retrieve the identification codes of the documents containing these keywords. Some search engines, like Alta Vista[10], allow entering exact phrases instead of simple keywords. This generally makes the inverted index larger and more complex.

Finally, the ranking methods applied to sort the document list presented to the user, are used to order the documents by relevance to the query. The rank of a document is usually based on the conformity of the keywords of the query and the document's keywords, as well as the occurrence of the keywords in the document. Some ranking formulas may also include in the ranking the relative position of keywords in the document.

Today's search engines use brute force to index on-line documents for the sake of simplicity.

[6]Noise is an irrelevant document which appears in the answer list.

[7]Silence happens when an existing document, relevant to the user's subject of interest, does not show up in the response list given by the search engine.

[8]Recall= the percentage of relevant documents in the answer list among all possible relevant documents.

[9]Precision= the ratio of relevant documents to the total number of documents retrieved.

[10]Alta Vista is available at: http://www.altavista.com/

Changing the ranking formulas will not be sufficient to balance the reduction of noise and silence. Some search engines added new features, like date of document, internet domain of origin (location), and even format of the document, to narrow the search. These are intermediate solutions that temporarily ease the user's frustration.

With most current popular search engines like AltaVista, all words are indexed. The inverted file index can quickly identify the documents that contain a given word. This was acceptable for a relatively small global information network or a local document database, however, because of the dynamic nature of the WWW and its continuous fast growth, the unique huge inverted file index approach is not viable. Indeed, as pointed out in [GRAV94], a major problem with this approach is the scalability of the index. The bigger the WWW becomes, the less satisfactory the answers from today's search engines will become. Our proposal focuses on qualitative search; while we do not ignore efficiency issues, we feel it is more important to focus on user efficiency and to produce accurate search results. However, the techniques we propose may actually lead to increased search efficiency, since the use of domain restriction, concepts and relevance may decrease the size of the search space and index structure.

An ideal document retrieval system should allow natural language (or pseudo natural language) querying, and should index documents based on the concepts present in them. Documents containing idioms like "kick the bucket", "bite the dust" or "meet its maker", should be indexed by the concept "death". The use of concept classification could allow the application of relationships like *parent-child* or *sibling* to link the concept "death" from a query to concepts like "funeral", "obituary" or "suicide" present in documents. The use of concept hierarchies could also allow the introduction of qualifiers like "like", "close-to", "related-to", etc., in the query language to help the user refine the request.

The other obvious advantage of the concept-based indexing is the reduced size of the index. Rather than indexing all words in this paper for example (there are 1300 different words), we could extract 10 to 20 major concepts and index the paper on them. The index is reduced this way by two orders of magnitude.

Concept-based retrieval systems attempt to reach beyond the standard keyword approach of simply counting the words from the request that occur in a document. The conceptual indexing system we present in the next section attempts to extract pertinent subjects from documents to categorize these documents by concepts, and uses knowledge of concepts and their interrelationships to find correspondences between the concepts in the request and those that occur in the documents.

By using the concept classification, we could also divide the index in a hierarchy of indices, each index specialized in a given domain. The information retrieval process would be split into two steps: one to select the appropriate indices, and a second to use these selected indices to find the documents. This approach is more scalable than the brute force indexing approach used by most of today's search engines, which have a precision that tends to degrade with the increase in size of the Internet. Each specialized index could be independently constructed and updated, and could be queried or mirrored separately.

3 Methods and Approach

As a first step toward a concept index of the library that is the Web, a computational agent which, at least, approximately classifies documents according to their subjects, is needed. We call such agents *Subject Extractors*. A Subject Extractor extracts relevant topics from a given document.

It is unrealistic, in the first stage, to expect the development of such an agent for an arbitrary range of applications. However, by focusing on a restricted domain, such as medicine or forestry industrial problems, etc., we can provide domain-oriented search and concept-extraction tools. Moreover, each specialized index could be constructed by another specialized Subject Extractor.

As an example of how domain-specific search tools will help with queries, let us consider what happens when we ask two of the existing popular search tools for "clear cuts near water", a forestry related query: *Lycos* [MAUL94] responds with "Complete poetical works from William Wordsworth", among a list of other equally wrong associations. *Excite*, which succeeds with more specific queries such as "Clear cuts near river", also produces nonsense replies to the query "clear cuts near water", such as "Plumbing frequently asked questions". It is important to understand why this happens. The "Plumbing frequently asked questions", for example, simply contains occurrences of "water", "near", "cuts" and "clear" and thus, is indexed under these words in the inverted index. Another document about cleaning toys might have contained the same words with high frequency and would have appeared in the response list.

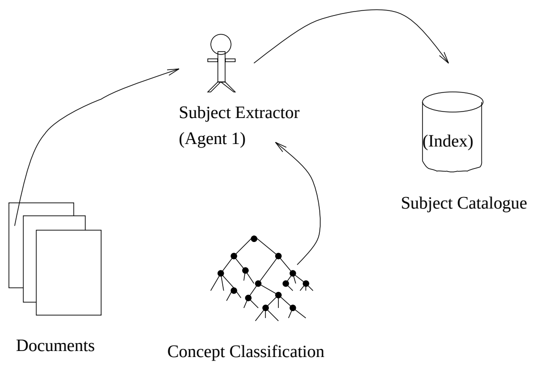

Figure 1: *Subject Extractor Expert indexing selected resources.*

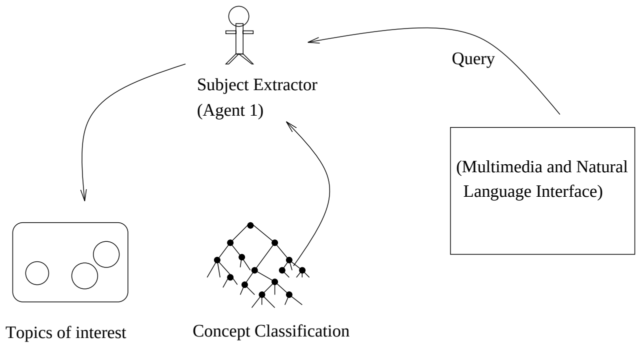

Figure 2: *Subject Extractor Expert extracts topics from the pseudo natural language query.*

Our approach, in contrast, can use a taxonomy of forestry-related concepts which allows it to specialize or generalize given concepts (e.g. going from "water" to "lakes" or vice versa), and thus is able to use the contextual information provided by forestry domains in order to avoid nonsensical answers. Therefore, a document like "Plumbing frequently asked questions" would not be indexed with the words it contains, but with its semantic content.

Using Assumption Grammars and Concept Classifications, the *Subject Extractor*, as shown in Figure 1, is used to educe relevant topics from given documents. The topics are then stored with the document identification in a subject catalogue for future matches with document retrieval requests (Figure 3). The subject catalogue uses the same concept classification to order the index in a multi-layered database. Figure 2 shows how the same *Subject Extractor* can be used to educe topics of interest from a user query written in natural language or controlled English.

With the Subject Extractor and the Subject catalogue using the A.I. tools described in this article, we propose a WWW agent for searching, indexing and organizing information relative to a given industrial context (Figure 4). We call this second agent *Network Crawler*, which is

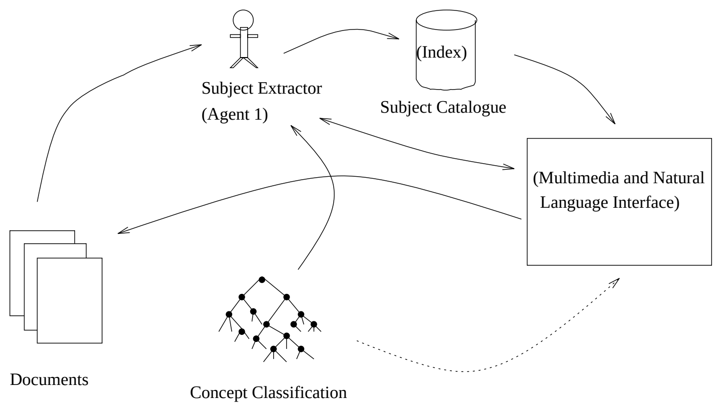

Figure 3: *Pseudo Natural Language Interface using the subject catalogue and concept classification to retrieve documents.*

a soft robot (also called wanderer or spider). The Network Crawler works in conjunction with the Subject Extractor to index and organize documents. It starts from a given web page, gives its content to the subject extractor for processing, then, like any web spider and following the standard for robot exclusion[11], recursively follows all the hyper-links in the document to load other web pages. The search of documents can be restricted to a given directory, a given web site, or a network domain.

In Figure 4, the *Load Monitor* is an agent which observes the usage of the Concept Classification and keeps a statistical record of the employment of paths in the concept hierarchy in order to optimize the use of the concept classification by eliminating paths that have been used less often (i.e. by generalizing sibling concepts) or by reorganizing the overused paths in more a efficient way (i.e. by specializing concepts and adding new subsumed concepts). Figure 4 represents the complete concept-based retrieval system for a global information network.

The following subsections detail the concept classification, the assumption grammars used, and the multi-layered database.

3.1 Concept Classifications

For the concept classification component in the computer generated phase, we use methodologies for the efficient management of hierarchies. The fundamental basis for the concept classification in our system is a *partial order* of concepts: a pair $(\Sigma,\leq)$, where Σ is a set of concepts and $\leq$ is a reflexive, anti-symmetric and transitive relation, called *subsumption*,

[11]Non enforced set of rules that if followed would protect web servers from unwanted accesses by robots. Accessible at http://info.webcrawler.com.mak/projects/robots/norobots.html

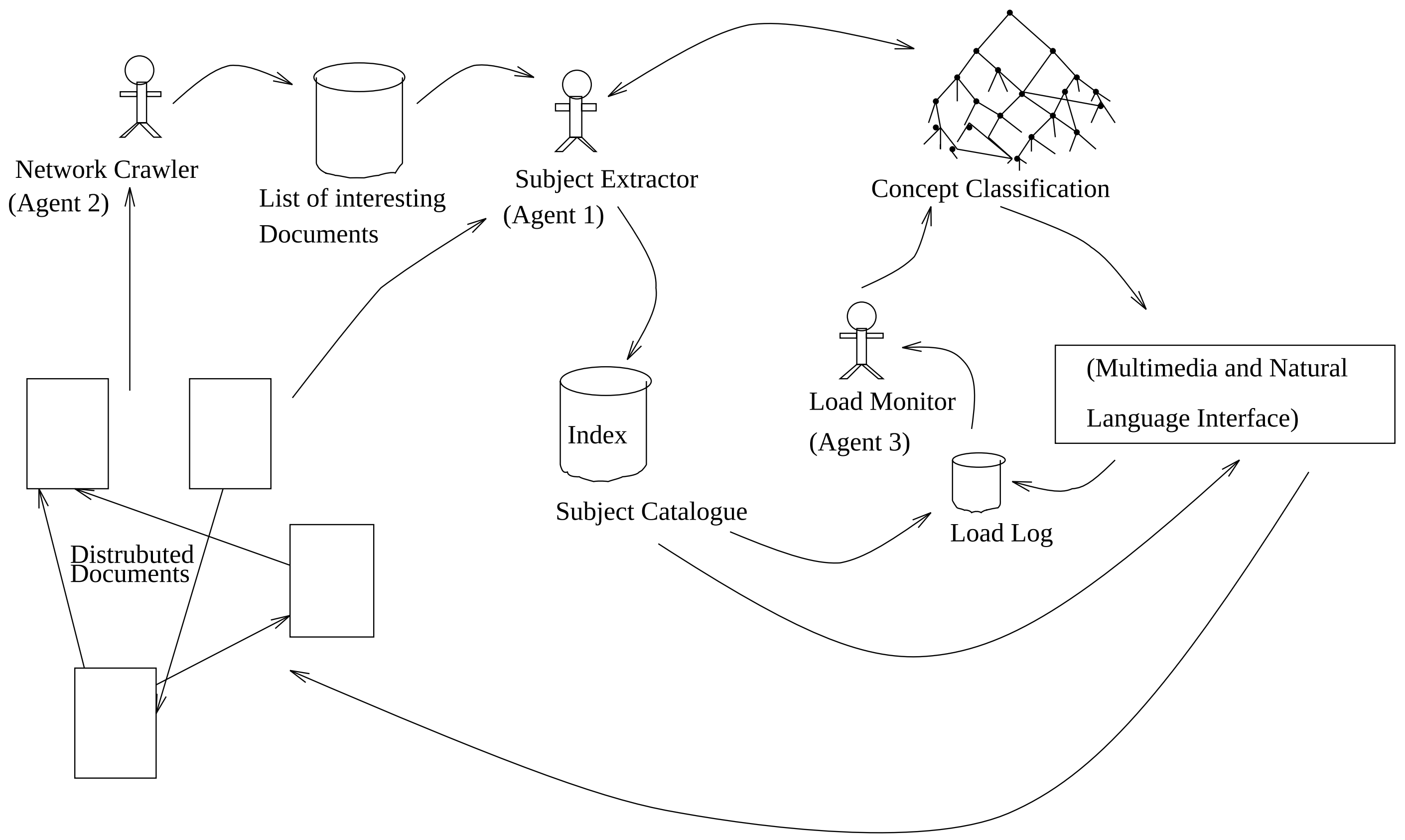

Figure 4: *The Concept-Based Retrieval System applied to the global information Network with an intelligent concept classification updating agent.*

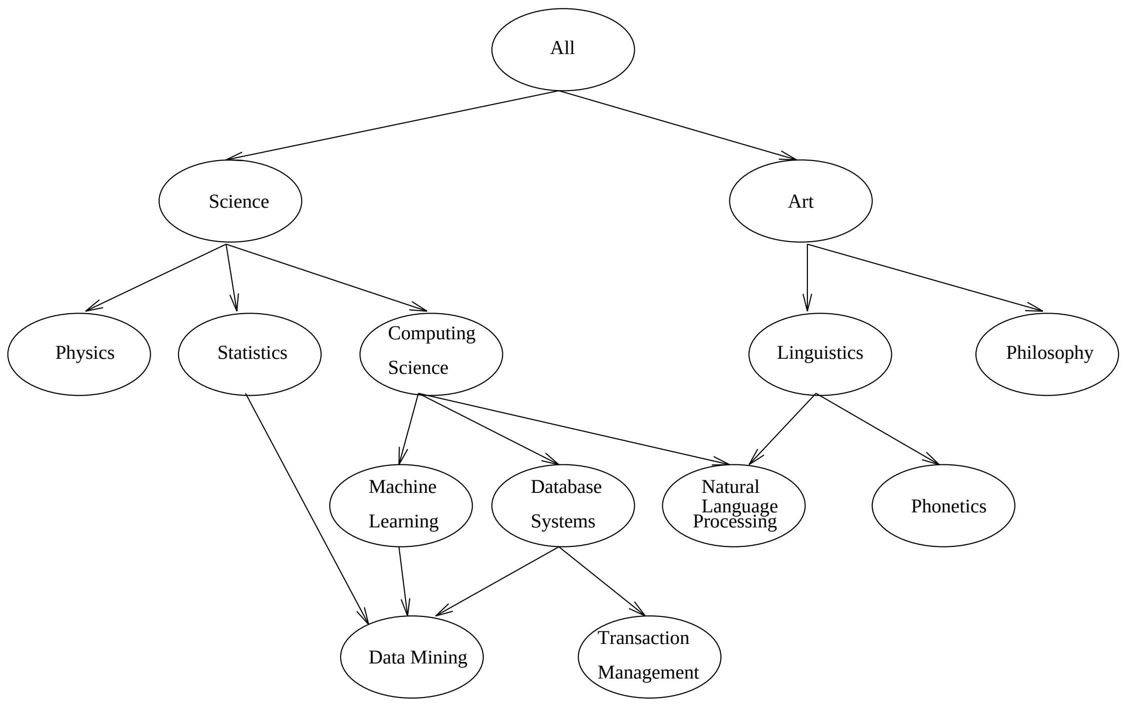

Figure 5: *An example of a Concept Classification.*

among those concepts[DAVE90]. In a graphical viewpoint this is equivalent to a directed acyclic graph, where more general concepts appear above more specialized concepts.

An example of concept classification is shown in Figure 5. Note that the graph is not limited to a tree form - some nodes may have multiple parents. For example, the *Data Mining* concept is subsumed by the concepts for *Statistics*, *Machine Learning* and *Database Systems*.

The primary operations on concept hierarchies include testing the subsumption relation, and computing *greatest lower bounds* and *least upper bounds*. For example, in Figure 5, the relation *Data Mining* $\leq$ *Science* holds, but not *Phonetics* $\leq$ *Science*.

The greatest lower bound, $\sqcap$, of a set of concepts A is the most general concept (if one exists) that is subsumed by all the concepts in A. For example, *Computing Science* $\sqcap$ *Linguistics* = *Natural Language Processing*. A greatest lower bound may not exist for a set of concepts, or there may be more than one maximal lower bound. Our system for managing hierarchies deals with all these situations. The least upper bound, $\sqcup$, of a set of concepts A is the dual of a greatest lower bound. It is the most specific concept (if one exists) that subsumes all the concepts in A. For example *Data Mining* $\sqcup$ *Physics* = *Science*.

If greatest lower bounds and least upper bounds exists for all pairs of concepts, then a partial order is called a *lattice* [DAVE90]. Although some systems require hierarchies to be lattices, we feel that the more general structure of a partial order permits more intuitive and less restrictive specification of the concept classification.

There are a number of important issues regarding the construction, representation and use of such hierarchies. Obtaining relevant concept classifications is a non-trivial task. In the current system, we hand-generate the concept classification using domain specific knowledge of the target application area. This construction occurs in consultation with domain experts, and can be partially automated and directed by using some techniques from formal concept analysis [GANT96, STUM97, WILL89]. Given a set of concepts, where the subsumption relation has been partially specified, these tools guide experts through the completion of the hierarchical structure. This is done using efficient algorithms that fill out the hierarchy with a minimum number of user queries.

As concept classifications grow in size, the need for efficient representations or *encodings* of hierarchies becomes important [FALL96a]. Hierarchical encoding techniques have been developed for a variety of tasks, from knowledge representation [ELLI93], natural language processing [MELL88] and logic programming [AITK89], to use in databases [AGRA89] and operating systems [MATT89]. These methodologies encode a hierarchy already given (either hand-generated or automatically generated), by associating with each element in the hierarchy a carefully constructed code which allows for efficient subsequent consultation. This is obtained by taxonomic encoding techniques which reduce expensive hierarchical operations to inexpensive set operations [FALL95]. We have developed algorithms for the efficient encoding and management of concept classifications in dynamic environments [FALL96b, FALL96c].

3.2 Natural Language Interface

Assumption Grammars[DAHL96, TARA96a] can be used for representing natural language so that it becomes executable (i.e., so that automatic inferences can be made from our description which result in automatic translations between natural language and meaning representation). Such a meaning representation is then used to consult our system and extract answers to our queries. We have shown how to solve several crucial language processing problems (i.e. coordination, anaphora, free word order) through assumption grammars [DAHL97]. A first assumption grammar prototype for English has already been successfully produced for another web application [ROCH97]. This prototype is easily adaptable to other domains by adjusting the lexicon corresponding to nouns, verbs and adjectives. We maintain this approach in the present work so that adapting the natural language processor to different domains will not be too complex a task.

Assumption Grammars are basically logic grammars, like Definite Clause Grammars, except that they are augmented with:

1. linear and intuitionistic implications scoped over the current continuation;
2. hidden multiple accumulators, useful in particular to make the input and output strings invisible.

Hidden multiple accumulators are important for efficiency and for other uses than the one we exploit in this paper. Since for our purposes here, however, we can disregard them, here we shall only describe our hypothetical reasoning tools. We refer to a companion paper [DAHL95] for a description of multiple accumulators and for all details of design and implementation in BinProlog [TARA95].

Intuitionistic implications temporarily assume clauses usable in later proofs. Such clauses can be used an indefinite number of times. Linear implications temporarily assume clauses usable *at most once* in later proofs. Both of these assumptions vanish on backtracking. Linear implications can be seen as implicitly existentially quantified.

To give an intuitive idea of how assumptions can serve natural language processing, consider a sentence that refers to an individual through an anaphora (pronoun, definite article) which is not yet defined in the discourse. It is reasonable to delay resolving the referent and abduce (make the hypothesis) that such an individual exists. Information from the current sentence will be used to generate constraints on the referent of the anaphora. When the anaphora is tentatively solved, abduced constraints are used to eliminate inconsistent interpretations. Likewise, long-distance dependency problems (such as finding an antecedent for a relative clause), and other important natural language processing problems (such as free word order), can be solved through hypothetical reasoning. The reader is referred to [DAHL97] for the details.

More discussion on how Assumption Grammars are used in our natural language interface can be found in [ZAIA97].

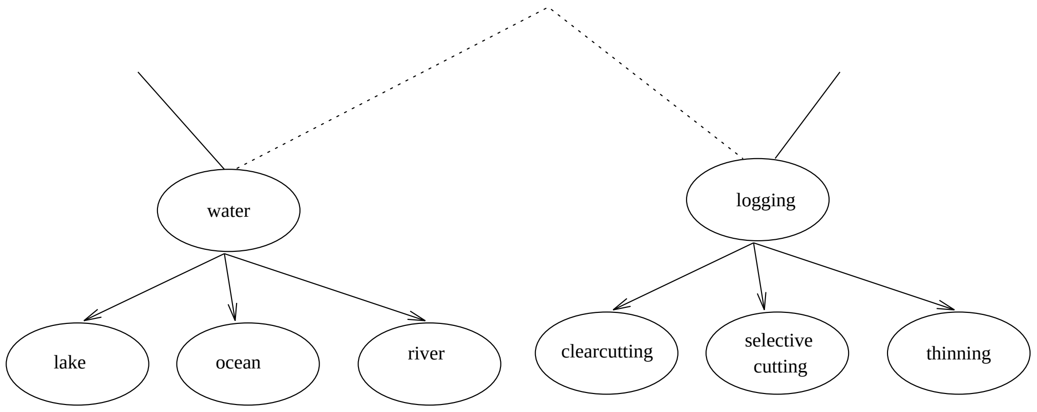

Figure 6: *2 sub-trees from a concept classification.*

Table 1: Layer 0

Clearcutting	D1, D2, D3, D11, D15
Lake	D2, D6, D8
Ocean	D5, D14
River	D7, D9, D10, D11
Selective cutting	D5, D9, D13, D14
Thinning	D6, D7, D12

Table 2: Layer 1

Logging	D1, D2, D3, D4, D5, D6, D7, D9, D11, D12, D13, D14, D15
Water	D2, D5, D6, D7, D8, D9, D10, D11, D14

Figure 7: *Example of inverted index in a MLDB structure.*

3.3 Multi-Layered Databases and Concept Hierarchies

We use a different approach, called a Multiple Layered DataBase (MLDB) to represent our indices. This approach uses our concept hierarchies to classify the inverted indices, which in turn would facilitate information discovery in global information systems [ZAIA95]. A multiple layered database (MLDB) is a database composed of several layers of information, with the lowest layer (i.e., *layer-0*) corresponding to the primitive information stored in the global information base, and the higher ones (i.e., *layer-1* and above) storing generalized information extracted from the lower layers. Every *Layer* i generalizes *Layer* $i-1$. In other words, concepts present in *Layer* $i-1$ are subsumed by concepts present in *Layer* i. The proposal is based on the previous studies on *multiple layered databases* [READ92, HAN94] and *data mining* [PIAT91, HAN93, HAN95].

Building our indices in a multiple-layered database is simple. The inverted index constitutes Layer 0. In our case, the entries in the inverted index are not words from the documents, but are the subjects extracted. Each entry in the Layer 0 is a subject (i.e. concept) and the set of documents related to it. The subsequent layers are built by going up in the concept hierarchy for each concept, and uniting the sets of documents related to all subsumed concepts. In other words, the entry in *Layer* i is associated with all its descendents in *Layer* $i-1$. Notice that there may be additional documents at the higher level. In the example using the concept hierarchy shown in Figure 5, some documents can be indexed under “Linguistics” but not under “Phonetics” or “Natural Language Processing”. This can continue to Layer n (where n is the depth of the concept hierarchy), or can be stopped at an arbitrary lower level of the concept hierarchy. The set of documents related to a concept is a subset of the documents related to a subsuming concept. Note that any sub-order of the concept hierarchy can be used to create a specialized index.

Let’s take an example: assuming we have 15 documents about logging practices and locations, and the two sub-trees from the concept hierarchy presented in Figure 6, table 1 in Figure 7 can be the inverted index containing the concepts extracted and the identification of the documents from which they were extracted. The concept “thinning”, for instance, is in documents D6, D7, and D12. Table 1 in Figure 7 is considered Layer 0. “Water” subsumes “lake”, “ocean” and “river”, therefore, when building Layer 1, the set of documents associated with “water” is the union of all sets of documents containing concepts from Layer 0 subsumed by “water”. Table 2 in Figure 7 shows Layer 1. Note that although D4 does not appear in table 1, it is indexed under “logging”. D4 contains the concept “logging” but none of “clearcutting”, “selective cutting” or “thinning” concepts. D8 is indexed under “water” but not “logging”. Apparently, the topic logging or logging practices were not present in the document.

Inverted indices are used by search engines to immediately find documents that contain a given keyword. The list of keywords is sorted and, by using a B-tree structure, the number of I/O accesses is reduced significantly. With the multi-layered database approach, this still holds true. Each layer is an inverted index generalizing the previous layer, and specializing the next one. The main advantage of the multi-layered database architecture is its progressive search capability. A user can start by using high level concepts, then progressively narrow the search by drilling down in the concept hierarchy. For example, using Layer 1 and 0 in tables from Figure 7, and submitting the query *“logging near water”*, we would get the answer D2, D5, D6, D7, D9, D11, and D14, which is the intersection between the “logging” and the “water” entries in Layer 1. By specializing the concept “water” to “river”, the query becomes *“logging near river”*, and the answer D7, D9, D11, which is the intersection between the entry "logging" in Layer 1 and "river" in Layer 0. Finally, by specializing "logging" to "clearcuts", the query becomes *"clearcuts near river"*, and the answer is reduced to D11. Note that the same can be applied for generalizing concepts and enlarging the potential document set.

4 Implementation

We have used the described methods and approachs to implement a user interface with controlled English for LogiMOO [DEBO96, TARA96c, TARA96b] with an interesting use of Assumption Grammars. LogiMOO is a BINProlog-based Virtual World running under Netscape for distributed group-work over the Internet and user-crafted virtual places, virtual objects and agents. We developed a Controlled English parser allowing people unfamiliar with Prolog to get along with the basic activities of the system. The current implementation of the Concept Extractor does not take advantage of such a parser. However, we plan to adapt LogiMOO's controlled English parser we wrote, to include it into the Concept Extractor.

We have completed a preliminary implementation of a Concept Extractor specializing in computer science. The documents we used contain abstracts of technical reports published by the School of Computing Science at Simon Fraser University.

We have also implemented a network crawler that can automatically and freely crawl all accessible pages on the web starting from a given set of pages, or be restricted to an Internet domain, a site, a directory structure, or just a list of URLs. All documents are fed to the concept extractor as a list of documents. Note that a list of document URLs received from a common search engine like AltaVista or OpenText can also be fed to our network crawler and the concept extractor in order to automatically filter out irrelevant documents.

The current implementation has focussed on the development of the Concept Extractor, for determining relevance of concepts related to a document, and the Natural Language Interface, which is used to match requested concepts to those found in the documents. The following sections highlight each of these components of the prototype.

4.1 A Natural Language Interface

The interface, provided through a Netscape environment, allows the user to enter words related to a topic(s) that they wish to retrieve documents on. From this input, a list is generated containing concepts to which the input is related. It is these concepts that are used to match concepts of documents. The translation of input words to concepts is accomplished through the same mechanism documents are categorized into concepts. This will be discussed in 4.2.

As an example, we can image a set of keyword inputs such as: *speech, parsing, translating* being translated into a concept list of: *Natural Language Processing.*

By being able to get a list of concepts from the users's input, we can advance the interface to include mechanisms for generalizing and specializing the concepts we are interested in. Through the use of concept hierarchies, as outlined in Section 3.1, we could possibly generalize "Database Systems" to "Computing Science" or we could specialize it to "Transaction Management".

This ability to use concepts to focus in (i.e. drill down) and out (i.e. roll up) of topics of interest to the user is advantageous. One advantage is that we indirectly guide the user to use concepts that are available through the system rather than having to worry about missing concepts or words in the lookup database. Secondly, and maybe more importantly, we are able to use advanced natural language processing techniques to convert user input to a usable form for concept matching. An example here would be to use these advanced techniques to convert idioms such as "kick the bucket" to the concept of "death".

4.2 A Concept Extractor

Basically, the Concept Extractor takes an input such as keywords from the user interface or free text from the documents and extracts relevant words. These words are then used to determine the original text's relevance to concepts stored in the concept database. This database maps words in the input to associated concepts. Note that words may have multiple meanings and hence may map to more than one concept. Domain restrictions may reduce this set.

The input can be parsed to remove unnecessary words. For example, we may not need to deal with common words such as "the", "a", and "and". Once common words are removed, we are left with a set of distinguishing words that can be used to determine concept relevance and form a list of concepts-relevance pairs about the input.

The first step in being able to accomplish this is to develop a concept "dictionary" or database. This mapping can be extracted from the on-line lexical reference system WordNet[12]

[12]http://www.cogsci.princeton.edu/~wn/

that relates words with their underlying lexical concepts [MILL95]. Each word may map to more than one concept, and each concept may have more than one synonymous word associated with it. These synonym sets are linked together through a variety of relations to form a large semantic network. By restricting attention to the concepts in our classification hierarchy and words in our lexicon, we can retrieve the required information from WordNet to create our concept database.

The entries in this database associate a concept with a list of words. It is these words that can be matched with the words extracted from the parsed input. For example, we could have the following entries in the dictionary:

- entry((“Natural Language Processing”)-(parsing, generating, translating, speech)).
- entry((“Phonetics”)-(speech, pronunciation, gaps))

Then we could take an input from either the user interface or a document and determine what concept the input is related to. The following could be an example input: *I need a document about translating speech to text.*

Now, let us assume that when this input is parsed, we have in our list of common words the following: (i, a, “,”, “.”, need, some, about). The result of the parse will be the relevant words from the input: (document, translating, speech, text)

We can then match these words to words in the database entries to determine concepts. In this case, we can determine that the example input is related to “Natural Language Processing” and “Phonetics”. In the case that the relevant words from input match in two or more concept entries, we can return a list of concepts that the input is related to. In the case that the query has qualifiers, such as “like” for synonyms, “close-to” for siblings, and “related-to” for ancestors, this list of concepts can be enriched with the concepts’ siblings or direct ancestors in the concept hierarchy.

We need to ask how relevant the input is to each of the possible concepts. Relevance of input to concepts is a difficult task to deal with. As an example, imagine a document detailing the plumbing repairs in a house. We run into the difficulty of determining if the document is more about plumbing, or home repairs.

As an attempt to deal with this difficult task, we associate the use of word count with amount of relevance. By doing this, we can use a function over word count and the words associated with individual concepts to evaluate the relevance of an input to the concepts. Furthermore, we can introduce a logarithmic function so as to address the concern of documents or input containing numerous occurrences of particular words. This is a technique used by web designers to fool common search engines into listing the designer's web page as a highly relevant document by numerously adding a given word in a hidden META html tag.

Assumption Grammars can be used at this point to assist in creating a dynamic method for revising the concept hierarchies. As we can see in the above plumbing example, plumbing has a relation to home repairs. At this point, an assumption of the relation can be made. These relation assumptions can then be consumed later in order to revise the concept hierarchies.

5 Conclusion

We have described how recently developed AI techniques can be put to use to produce a prototype system for intelligently retrieving and classifying documents found in the Web. We are currently developing such a prototype. This tool will prove useful for decision makers, as already stated, but also to all people with an interest in getting information from the Web.

It is worthwhile mentioning that the use of the techniques here proposed will allow for relatively straightforward adaptations of our prototype from one domain into another, by replacing the concept hierarchy used for a given application domain by that required by another application domain. In this sense we expect the finalized system to be flexible enough for easy portability to different specific areas of expertise, and construction of diverse specialized indices.

References

[AGRA89] R. Agrawal, A. Borgida, and H. Jagadish. Efficient management of transitive relationships in large data bases, including is-a hierarchies. In *Proceedings of ACM SIGMOD*, 1989.

[AITK89] H. Aït-Kaci, R. Boyer, P. Lincoln, and R. Nasr. Efficient implementation of lattice operations. *ACM Transactions on Programming Languages*, 11(1):115–146, 1989.

[BOWM94] M. Bowman, P. Danzig, D. Hardy, U. Manber, and M. Schwartz. *Harvest, A scalable, Customizable Discovery and Access System.* Technical Report CU-CS-732-94 Department of CS, University of Colorado, Boulder, July 1994. Available from *ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.FullTR.ps.Z.*

[DAHL95] V. Dahl, A. Fall, and P. Tarau. Resolving co-specification in contexts. In *In Proc. IJCAI'95 Workshop on Context in Language*, Montreal, July 1995.

[DAHL96] V. Dahl, A. Fall, S. Rochefort, and P. Tarau. A hypothetical reasoning based framework for NL processing. In *In Proc. ICTAI'96*, 1996.

[DAHL97] V. Dahl, P. Tarau, and R. Li. Assumption grammars for processing natural language. In *ICLP'97*, 1997. (Submitted for publication).

[DAVE90] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, England, 1990.

[DEBO96] K. De Bosschere, D. Perron, and P. Tarau. Logimoo: Prolog technology for virtual works. In *In Proc. PAP'96*, pages 51–64, London, April 1996. ISBN 0 9525554 1 7.

[EICH94] D. Eichmann. The RBSE spider - balancing effective search against web load. In *Proc. 1st Int. Conf. on the World-Wide Web*, pages 113–120, Geneva, Switzerland, May 1994.

[ELLI93] G. Ellis. Efficient retrieval from hierarchies of objects using lattice operations. In *Conceptual Graphs for Knowledge Representation. Proc. First International Conference on Conceptual Structures*, Quebec, Canada, 1993. Springer-Verlag.

[FALL95] A. Fall. An abstract framework for taxonomic encoding. In *Proc. First International Symposium on Knowledge Retrieval, Use and Storage for Efficiency*, Santa Cruz, CA, 1995.

[FALL96a] A. Fall. The evolution of taxonomic encoding techniques. In *Estudios Sobre Programación Lógica y Sus Aplicaciones*, pages 201–231. Servicio Publicacions da Universadade de Santiago de Compostela, 1996.

[FALL96b] A. Fall. Sparse term encoding for dynamic taxonomies. In *Fourth International Conference on Conceptual Structures*, Sydney, Australia, 1996. Springer-Verlag.

[FALL96c] Andrew Fall. *Reasoning with taxonomies.* PhD thesis, Simon Fraser University, July 1996.

[GANT96] B. Ganter. Attribute exploration with background knowledge. In *Proceedings of the Conference on Order and Decision Making*, Ottawa, Canada, August 1996.

[GRAV94] L. Gravano, H. Garcia-Molina, and A. Tomasi. The effectiveness of gioss for the text-datase discovery problem. In *ACM SIGMOD*, 1994.

[HAN93] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5:29–40, 1993.

[HAN94] J. Han, Y. Fu, and R. Ng. Cooperative query answering using multiple-layered databases. In *Proc. 2nd Int. Conf. Cooperative Information Systems*, pages 47–58, Toronto, Canada, May 1994.

[HAN95] J. Han, O. R. Zaïane, and Y. Fu. Resource and knowledge discovery in global information systems: A scalable multiple layered database approach. In *In Proc. Conference on Advances in Digital Libraries*, Washington, DC, May 1995.

[KOST94] M. Koster. ALIWEB – archie-like indexing in the web. In *Proc. 1st Int. Conf. on the World-Wide Web*, pages 91–100, Geneva, Switzerland, May 1994.

[MATT89] F. Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. Elsevier/North-Holland, 1989.

[MAUL94] M. L. Mauldin. Lycos: Hunting www information. CMU, 1994.

[MCBR94] O. McBryan. Genvl and wwww: Tools for taming the web. In *Proc. 1st Int. Conf. on the World-Wide Web*, pages 79–90, May 1994.

[MELL88] C. Mellish. Implementing systemic classification by unification. *Computational Linguistics*, 14(1):40–51, 1988.

[MILL95] G. A. Miller. WordNet: A lexical database of english. *Communications of the ACM*, 38(11):39–41, 1995.

[PIAT91] G. Piatetsky-Shapiro and W. J. Frawley. *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

[READ92] R.L. Read, D.S. Fussell, and A. Silberschatz. A multi-resolution relational data model. In *Proc. 18th Int. Conf. Very Large Data Bases*, pages 139–150, Vancouver, Canada, Aug. 1992.

[ROCH97] S. Rochefort, V. Dahl, and P. Tarau. Controlling virtual words through extensible natural language. In *In Proc. AAAI Symposium on Natural Language Processing for the World-Wide Web*, Palo Alto, California, March 1997. (to appear).

[STUM97] G. Stumme. Concept exploration - a tool for creating and exploring conceptual hierarchies. In *Proceedings of the 5th International Conference on Conceptual Structures (to appear)*, Seattle, USA, August 1997.

[TARA95] P. Tarau. Binprolog 4.00 user guide. In *Technical Report 95-1*, Departement d'Informatique, Universite de Moncton, 1995.

[TARA96a] P. Tarau, V. Dahl, and A. Fall. Assumption grammars. In *Proc. Asian 96 Conference*, 1996.

[TARA96b] P. Tarau. Logic programming and virtual worlds. In *In Proc. INAP'96*, Tokyo, November 1996. keynote address.

[TARA96c] P. Tarau and K. De Bosschere. Virtual works brokerage with bin-prolog and netscape. In *In Proc. 1st Workshop on Logic Programming Tools for Internet Applications*, Bonn, September 1996.

[WILL89] R. Wille. Knowledge acquisition by methods of formal concept analysis. In E. Diday, editor, *Data Analysis: Learning Symbolic and Numeric Knowledge*, pages 365–380. Nova Science Publisher, New York, 1989.

[ZAIA95] O. R. Zaïane and J. Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *Proceedings 1st International Conference on Knowledge Discovery in Databases*, Montreal, Canada, August 1995.

[ZAIA97] O. R. Zaïane, A. Fall, S. Rochefort, V. Dahl, and P. Tarau. Concept-based retrieval using controlled natural language. In *NLDB: Workshop on Natural Language and Databases*, Vancouver, June 1997.