Netoracle, an intelligent information agent

by Joris Van den Bogaert

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science
Montana State University

Abstract:
Searching the World Wide Web today is comparable to giving a librarian a list of words and expecting him to find a set of documents that exactly answer your request. Because so little information is supplied, the librarian can only return partially satisfactory results. For example, some returned documents might be irrelevant or not enough relevant documents are found. There is no easy solution to producing a concise, valuable response to a query due to the enormous amount of information on the Web combined with the lack of homogeneity among Web pages.

This thesis addresses this problem and provides a prototype of a search tool that exhibits some intelligent behavior. A number of improvements to the current state of the art in search engines are suggested.

NETORACLE,

AN INTELLIGENT INFORMATION AGENT

by

Joris Van den Bogaert

A thesis submitted in partial fulfillment

of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY – BOZEMAN

Bozeman, Montana

April 1998

APPROVAL

of a thesis submitted by

Joris Van den Bogaert

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

John Paxton _____ 4/15/98
(Signature) (Date)

Approved for the Department of Computer Science

Denbigh Starkey _____ 4/15/98
(Signature) (Date)

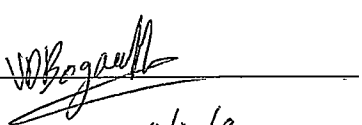
Approved for the College of Graduate Studies

Joseph Fedock _____ 4/27/98
(Signature) (Date)

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master of Science degree at Montana State University – Bozeman, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature 

Date 4/15/98

ACKNOWLEDGEMENTS

It is my pleasure to acknowledge the great help of Dr. John Paxton, who introduced me to the fascinating field of Artificial Intelligence and who spent countless hours advising, counseling, and encouraging me. I also owe him special thanks for his meticulous reviews of my drafts, and for his lucid discourses on writing style.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Searching the World Wide Web today is comparable to giving a librarian a list of words and expecting him to find a set of documents that exactly answer your request. Because so little information is supplied, the librarian can only return partially satisfactory results. For example, some returned documents might be irrelevant or not enough relevant documents are found. There is no easy solution to producing a concise, valuable response to a query due to the enormous amount of information on the Web combined with the lack of homogeneity among Web pages.

This thesis addresses this problem and provides a prototype of a search tool that exhibits some intelligent behavior. A number of improvements to the current state of the art in search engines are suggested.

CHAPTER 1

INTRODUCTION

In recent years, the number of World Wide Web sites has grown tremendously. Experts predict that, at least for the next decade, the number of sites will keep increasing exponentially. Due to the chaotic structure of the Web, quickly locating relevant or useful documents in these (approximately) 200 gigabytes of information is a real challenge [2]. Current search engines don't seem to bring order to the chaos and the Net is outgrowing them.

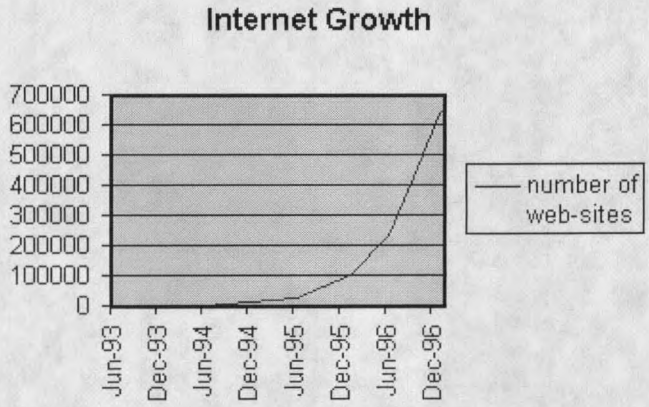


Figure 1.1: Internet growth

The main objective of this thesis is to show that techniques can be employed that help the user to more efficiently locate relevant information on the Web. It is not uncommon that the user finds himself "drowned in information" when too many links match the query. Typically, a user will go back and refine the search by tweaking the original query: adding synonyms, using quotes around adjacent words, applying boolean and advanced operators, etc. This will improve search performance, but it requires a knowledge of the special operators inherent to the search engine. Also, imagine how much time is wasted checking out nonexistent or outdated links!

The quality of the search results is very much dependent on the quality of the query. The average user is not specific enough and leaves too much room for ambiguity.

Systems that provide a natural language interface partially solve this problem: natural language forces the user to create a minimal context which helps the engine determine the semantics of the query terms. The more information available about what the user is looking for, the more accurate the engine will be when retrieving and ranking documents. Although some engines claim to accept natural language queries, they are not using this extra information in a way that would add a significant amount of power to the retrieving process.

The Next Generation

The ideal search engine would have to understand not only the natural language query, but also the contents of the documents. That would allow it to return documents that contain the information sought. Or it may even return the most relevant information within a document, or ultimately summarize the Web page, giving an impeccable answer to the question. However, according to Stuart and Norvig, "It is possible to tune natural language processing (NLP) techniques to a particular subject domain, but nobody has been able to successfully apply NLP to an unrestricted range of texts."

NetOracle is a prototype developed that approaches a text structurally as opposed to linguistically. While it does not try to *understand* the contents, it uses statistical word information (frequency, position, vicinity) to extract information from a Web page, and to use this information in evaluating it.

CHAPTER 2

CURRENT "STATE-OF-THE-ART"

(or: the need for better search tools)

2.1 Search Tools

There are numerous search tools to locate information on the Web. The most commonly used are the search engines that use databases automatically created by Web robots (eg. Alta Vista, Hotbot, etc.). Another category are subject directories (eg. Yahoo). These are hierarchically organized indexes that allow one to browse through Web sites by subject or topic. The latter ones are generally smaller in size, but they reduce the retrieval of irrelevant documents. In addition to these commonly-used types of largely automated search engines, there are other tools or services available that ensure good search results by allowing human intervention in the search process.
An example of each type of system is given, along with a brief description.

2.1.1 Keyword searching: AltaVista

AltaVista is one of the most commonly used search engines. Its huge database is created by Scooter, a "Web spider" that roams the Web collecting Web pages. Every word that is not a stopword [1]is indexed, along with some information about its location on the Web page. AltaVista allows the user to have control over the search by offering a variety of search operators. Another one of its strengths is the fast search speed.

---

[1] stopwords are common "noise" words with insignificant retrieval power, eg. a, the, between, etc.

AltaVista does not stem words in the indexing process, so an exact match is required. The ranking system of AltaVista is based on measurements described in section 2.2.

2.1.2 Subject searching: Yahoo!

Yahoo! is a "directory of the Internet". A user chooses a general topic and branches out to many sub-topics eventually reaching specific sources of information. Every site added to Yahoo!'s database is manually reviewed, making quality more important than quantity. Its ranking system is very simple: the frequency of a keyword in a document determines the ranking. Also, a better ranking is assigned to matching documents that are located higher in the Yahoo! tree hierarchy. One drawback of these subject hierarchies is that the categories get too specialized and ambiguous. Users may find it difficult to know exactly where to go and look.

2.1.3 Expert-aided searching: AskJeeves

People tend to look for similar things. AskJeeves contains a database of typical questions and the links containing the answers. It accepts plain English questions and matches it to one of thousands of "question templates". The knowledge base of AskJeeves is manually created and updated, making it a highly accurate search tool to quickly find answers to common questions. If the question is not contained in the database, AskJeeves will simultaneously contact a number of popular search engines. AskJeeves also presents a special add-on component that allows users to build custom knowledge bases.

2.2 How do search engines rank documents?

Every search engine has its own way of classifying documents. In most cases, the user can control this ranking using symbolic operators. For example, AltaVista recognizes the '–' symbol to prohibit the inclusion of a term in a document. There are a few other relevancy measurements:

- *Location of terms*: rate documents higher when the keywords appear in the title.
- *Frequency*: documents containing query terms with a higher frequency are likely to be more relevant than others.
- *Proximity*: documents containing query terms that are in close proximity to one another could be ranked higher.
- *Search engine spamming*: with the intention of appearing to be more relevant than it actually is, page designers sometimes include a number of hidden relevant words. A search engine may penalize documents when a word is repeated a number of times to "boost" its relevancy.

2.3 Measuring the overall performance of a search engine

Several criteria can be used in measuring the overall performance of a search tool:

- relevancy (how related are the retrieved documents to the query)
- number of hits returned: precision (the number of relevant documents vs. the total number of documents returned) and recall (does the search engine omit important, relevant documents?)
- available operators (what advanced operators are available to refine the search?)
- currency of hits
- retrieval speed

CHAPTER 3

NETORACLE, AN INTELLIGENT INFORMATION AGENT

3.1 Basic idea

If the user is looking for a short answer, there is no need to look around and try to find as much information as possible on the subject. An alternative is to just display or highlight the most relevant paragraph in a document.

NetOracle is designed around the idea that the type of question determines what searching methods are to be used. In a natural language environment, the system can infer some information from looking at the interrogative word. A "who"-type question as in "Who invented the light bulb?" tells the system that the user is probably looking for a short answer, in this case a name! A "when"-type question ("When did Napoleon die?") is an indication that dates or timelines should be looked for. There are also numerous complex queries, where a ranking of documents is preferred ("Tell me about religion in Thailand").

3.2 NetOracle overview

The following picture depicts a high-level overview of the system. At the end of the chapter, a complete example is elaborated.

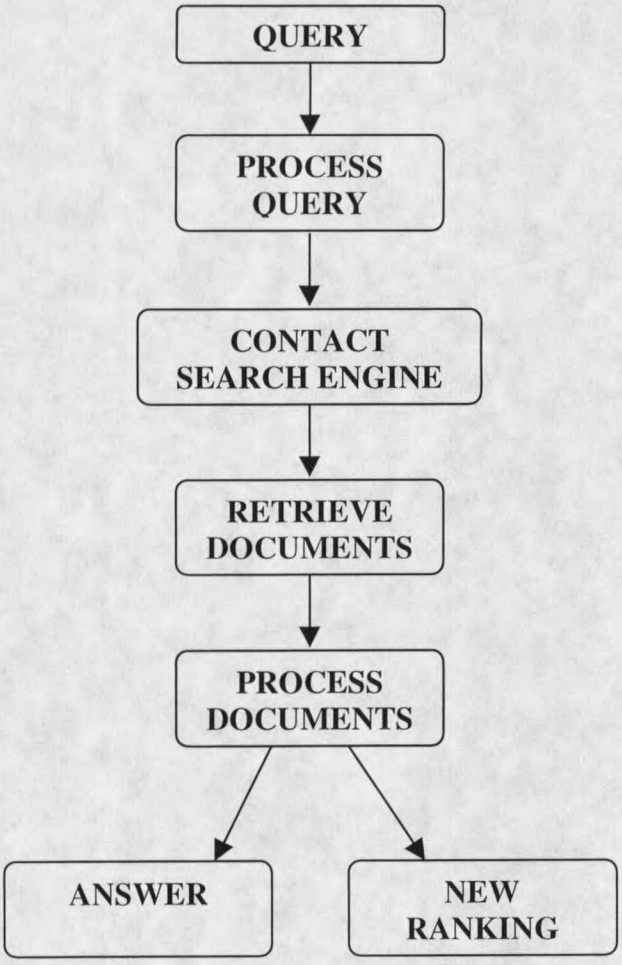


Figure 3.1: NetOracle overview

3.2.1 Process query

The system does not use any computational linguistics to extract information from the query. However, it aims to recognize certain types of queries and uses this information to lead the search. To determine what query words are important, certain operations will be performed.

Removing stop words

Stop words are high-frequency words that comprises 40 to 50 percent[2] of the text words [11] (eg. a, in, below, became) They do not add any discriminative power to the document and are removed from the query. About 560 common English words make up the stop list[3].

Stemming words

Terms with a common stem will usually have similar meaning, for example:

CONSTRUCT
CONSTRUCTED
CONSTRUCTION
CONSTRUCTIONS
CONSTRUCTING

In general, the performance of an information retrieval system can be improved if words are stemmed before indexing them. M.F. Porter presented an algorithm that strips suffixes from words, using a suffix list with various rules. There are a couple of flaws in the algorithm (eg. 'baby' stems to 'babi', words ending in –ly are not stemmed properly: eg. 'quickly' stems to 'quickli'). No attempts have been made to fix these flaws, due to the findings of a Chinese researcher who showed that Porter's algorithm is fragile. Fragility means that a slight change to the rule set has a large impact on the quality of the results.

Prefix stripping might also be useful, but a study has shown that this would take up 70% of the total manipulation time. Prefix stripping may also alter the meaning of the root word too much and thus hide the meaning of the word in the sentence.

---

[2] Salton/McGill Modern Information Retrieval

[3] This stop list is used by Britannica Online

Expanding the query

Even for experienced users, searching the World Wide Web can be frustrating. Typically, many irrelevant documents are retrieved and some relevant ones are missed. Some advanced search engines allow the user to refine a query giving a list of words that are related to the query terms. Doing this query expansion automatically, versus manual selection, is not a trivial process and can be both a blessing and a curse. The problem is the lexical ambiguity, where a word can have more than one meaning. A number of studies in the area of word sense disambiguation empirically show that it is better not to do this automatically. Sanderson's investigation concludes that the performance of Information Retrieval (IR) is "insensitive to ambiguity but very sensitive to erroneous disambiguation".

Expanding the query using WordNet and Fuzzy Logic

WordNet, a lexical database for English developed at Princeton University, is built around the notion of a synonym set or synset. A synset represents a concept and consists of all the terms that can be used to represent that concept. The idea is to assign fuzzy membership values to each of the synsets of a word and to augment the query with those synsets exceeding a threshold value (the α-cut). To do this, one can take the topic words of the query and look for relationships between them to determine the semantics of the words in that specific context. For simplicity, only nouns are considered. WordNet provides several types of relationships: synonym (equivalent word), hypernym (more generic word), hyponym (more specific word), holonym (a word that names the whole of which a given word is part), meronym (a word that names parts of a given word) and coordinate terms (words with the same hypernym).

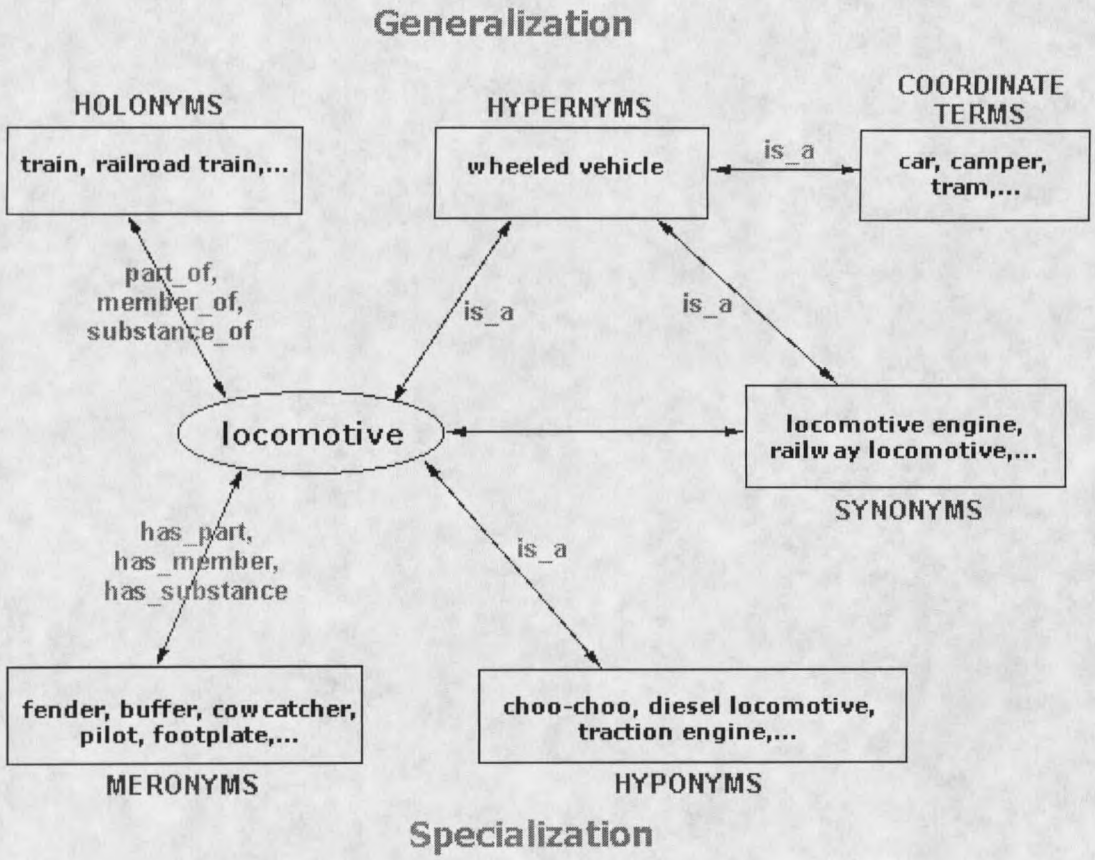


Figure 3.2: Word relationships

Weights in the range [0,1] are assigned to each type of relation. A first order synonym should be assigned a higher weight than a hypernym, which is just a generalization of the concept. Likewise, synonymous terms should not be given the same importance as the original query terms. After constructing a synset vector that consists of a collection of terms and weights, we can measure its relative importance by comparing it with the synset vectors of other query terms. This results in a vector table Q that contains the original query terms with a weight of 1.0 and a collection of related words with the assigned fuzzy value.

Assigning fuzzy values to the different kinds of relations, is based on intuition. For example, consider the word *neuron*:

Word	Type	Fuzzy value
neuron	query word	1.0
nerve cell	synonym	0.9
brain cell	hyponym	0.8
axon, dendrite	meronym	0.7
nervous system	holonym	0.5
cell	hypernym	0.2
zygote, bone cell, ...	coordinate term	0.2

Table 3.1: Fuzzy relationships

3.2.2 Contact search engines

The synonym set that is determined in the previous step, can be used to narrow down the search. To get a statistically significant set of links, the engine is contacted more than once each time with different features. For example, when the question is: "*Who is the inventor of the light bulb*", the engine is queried with the following syntax:

"inventor of the light bulb"

+inventor, +"light bulb"

(inventor NEAR "light bulb")

+inventor, +"light bulb", "electric lamp", "incandescent lamp", ... (+ synonyms)

The engine will return a raw HTML-page, containing the links. They are filtered out and duplicate links are removed.

3.2.3 Contact URLs

As many links as possible are downloaded within a reasonable user-defined period of time. At this time, dead sites are discovered and documents that are not found or inaccessible are discarded.

3.2.4 Process documents

With the retrieved documents all in raw HTML format, several pre-processing steps are necessary: HTML tags (eg. <H1>) and HTML codes (eg.) are removed and JavaScript code is ignored. Then the document is subdivided into paragraphs and sentences are identified using a number of syntactical rules. Given the heterogeneity of Web pages, this is a non-trivial algorithm. Finally, stopwords are thrown away and the remainder of the words are stemmed and stored in a list.. This list will be used for further processing.

Ranking documents

To determine the relevancy of a document, there are a number of weighting schemes that use word frequency as a measure. A popular one is the so-called TDIDF[4] weight.

For every word d_i in document T, a weight v_i is calculated, based on the following parameters:

- *tf(i)* - the term frequency, the frequency of d_i in T;
- *df(i)* - the document frequency, the number of documents in the collection that contain d_i;
- *n* - the number of documents in the collection;
- tf_{max} - the maximum term frequency over all words in T

---

[4] Term Frequency x Inverse Document Frequency

A detailed discussion of this scheme can be found in [11].

The weight v_i is then given by:

$$Vi = \frac{\left(0.5+0.5\frac{tf(i)}{tf(\max)}\right)\left(\log_2 \frac{n}{df(i)}\right)}{\sqrt{\sum_{dj \in T}\left(0.5+0.5\frac{tf(j)}{tf(\max)}\right)^2\left(\log_2 \frac{n}{df(j)}\right)^2}}$$

For example, suppose that the query "religion in Thailand" returned a collection of 10 documents. Suppose furthermore that "religion" appears in every one of them. When calculating the weight, the second term in the numerator

$$\log_2 \frac{10}{10}$$

equals zero implying that "religion" is not a useful word for discriminating relevant from non-relevant documents. Words that occur in very few documents are also not very useful.

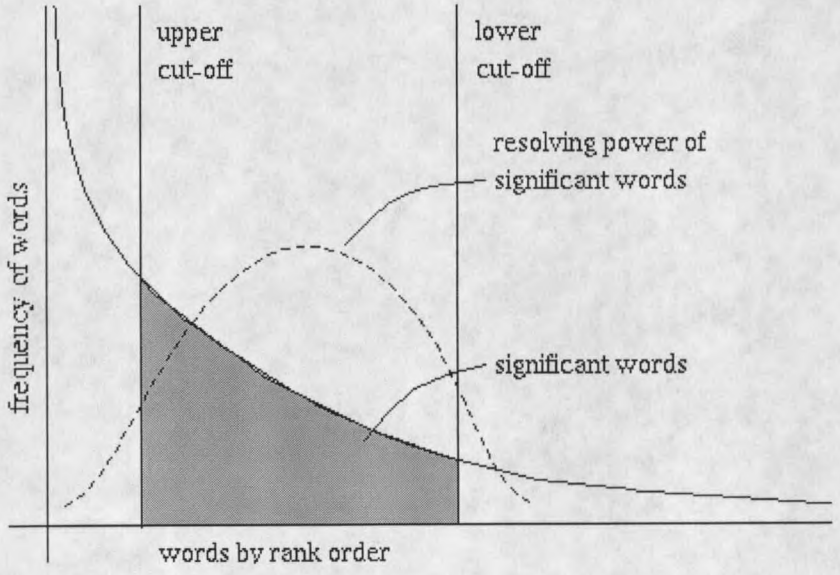


Figure 3.3: Discriminating power of words

After all weights are calculated, a vector table V is created. V contains a weight for each term in the document. The relevancy of the document can then be calculated by simply taking the dot product with the query vector Q:

$$R(T) = \vec{V} \cdot \vec{Q}$$

Notice that only words in the query vector are considered.

Paragraph Relevancy

In the case of a query where a short answer is preferred, it would be time-saving to the user to have the most relevant paragraph in the set of documents displayed, which might reveal the answer to the query. A relevant paragraph can be defined as a paragraph that contains all or most of the term words in combination with words that appear to be highly correlated with the term words or its synonyms. A correlation formula between document words and term words could be based on both the vicinity and the term frequency. The vicinity of a word is simply the sum of the nearest distances to the term words. For simplicity, all information about the relevancy of a word is encoded in an integer. For example, suppose the word "*inventor*" appears in 3 documents of the collection, that the total vicinity is 76 words, and furthermore that the total number of term words considered in calculating this total vicinity is 6. The weight of a word is then calculated as follows:

$$Weight_{word} = \frac{total\_vicinity}{total\_termwords\_considered}$$

In the example, the weight turns out to be 76 / 6 = 12.6. Only words with weights within a user-defined *vicinity bound (vb)* are considered. Finally, the weighted words are sorted, given the following sorting comparison function:

Condition	Winner
$Weight_{word1} < vb$ AND $Weight_{word2} < vb$	word with highest frequency
$Weight_{word1} < vb$ AND $Weight_{word2} > vb$	word 1
$Weight_{word1} > vb$ AND $Weight_{word2} < vb$	word 2
$Weight_{word1} > vb$ AND $Weight_{word2} > vb$	word with lowest weight

Table 3.2: Word comparisons

For example, the query "Who was the inventor of the light bulb?" may yield the following words after analyzing the collection of documents with a vicinity bound of 15:

rank	word	# of documents	# of keywords considered	total vicinity	weight
#1	*Edison*	11	22	63	2.86
#2	*Thomas*	7	14	76	5.42
#3	*year*	6	12	135	11.25
#4	*electric*	5	10	17	1.7
#5	*people*	5	10	66	6.6
#6	*power*	4	8	19	2.38
#7	*vacuum*	4	8	11	1.38
#8	*work*	4	8	47	5.88
#9	*history*	4	8	61	7.63
#10	*system*	4	8	24	3

Table 3.3: Highly rated related words

The same method can be applied to the synonyms of the term words, taking into account their importance (fuzzy values). The most relevant paragraph is the one containing the highest number of term words as well as the words obtained from these correlation calculations.

3.3 Elaborated example

This section will show a working example of a query by going through the different steps.

Query input

A very simple interface accepts natural language queries from a user:

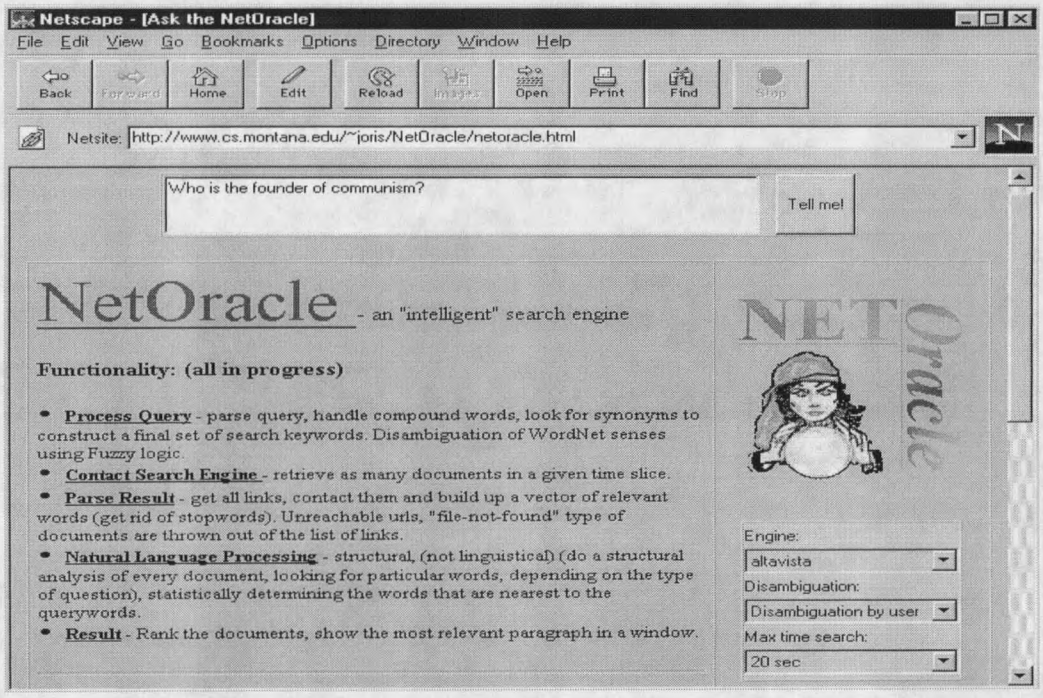


Figure 3.4: NetOracle's interface

AltaVista is selected as a back-end search engine, disambiguation is non-automatic, and the maximum download time allowed is 20 seconds.

Process query

Because the user preferred to expand the query manually, NetOracle will contact WordNet and will on-the-fly create an HTML page containing the senses and synonyms

of all the significant query terms. The user can select any one or more relevant synonym sets requiring NetOracle to include them in the retrieval process:

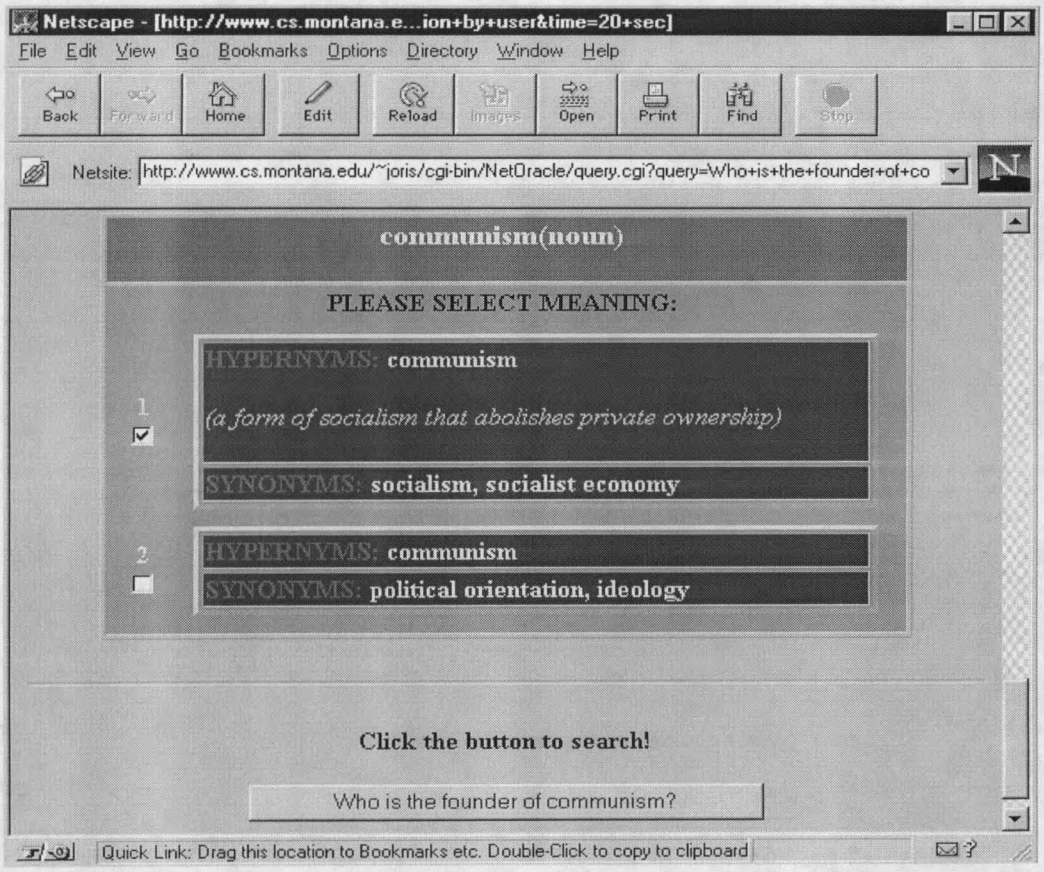


Figure 3.5: Disambiguation by the user

Contact search engine

Several http requests are made to AltaVista with the intention of keeping some diversity in the pages that are retrieved:

"founder of communism"

+founder, +communism

(founder near communism)

+founder, +communism, socialism, "socialist economy"

Contact URLs

As NetOracle is limited by a certain period of time (20 seconds in this example) to retrieve documents, its performance depends on current network conditions. To analyze a document, it must first be downloaded, so there does not seem to be a way around this bottleneck. A possible improvement is proposed in Chapter 5.

At this time, NetOracle will peek at the contents of the documents, retrieving the outdated ones (eg. "404 errors - File Not Found") from being processed further.

Process Documents

The document is subdivided into paragraphs. Stopwords are removed, the remainder of the words are stemmed and word weights are assigned using the formula discussed in Chapter 3.

Paragraph relevancy

A score is assigned to each paragraph of each document. The most relevant sentence(s) of the best paragraph are determined and shown in a separate window. Other links are ranked and displayed along with their best sentences.

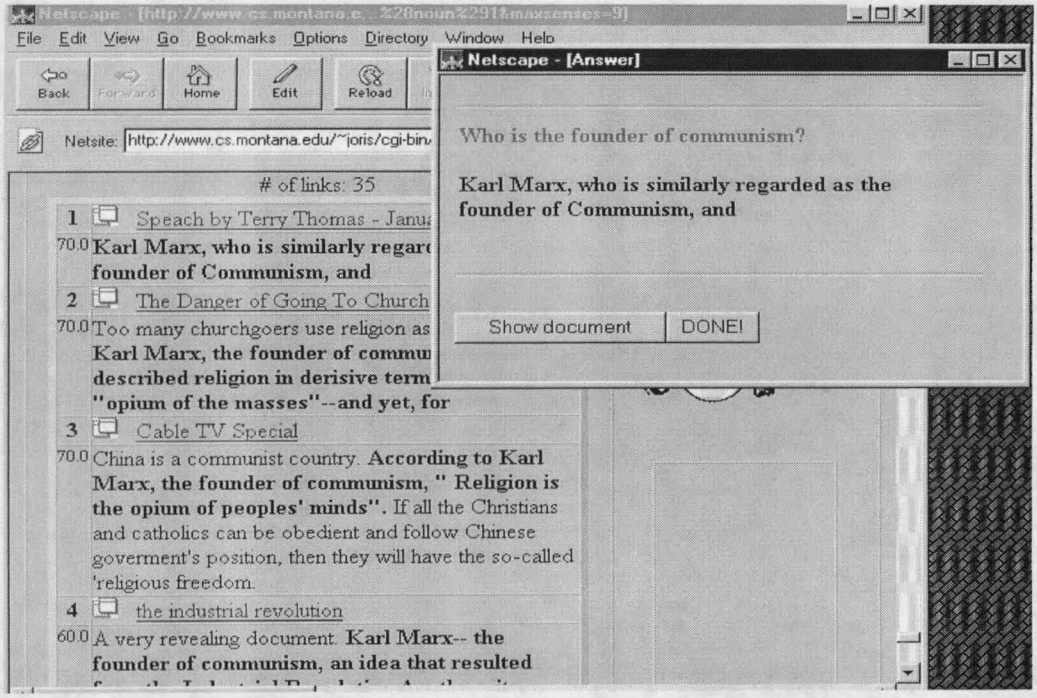


Figure 3.6: Query response

CHAPTER 4

EMPIRICAL STUDY

A number of tests were performed to evaluate whether the overhead of re-ranking is justified. The rankings are examined, as well as the paragraph that purports to answer the question.

4.1 Comparing NetOracle's ranking with Alta Vista's ranking

To evaluate the ranking performance, both systems were subjected to a number of diverse queries. Only the first 10 documents found by each system were considered. The rating is based on a very simple criterion: did the search engine find what was intended to be found. The resulting links were manually inspected and each URL was given a relevance score in the range [0..3].

3	Document is highly relevant, it contains all the information necessary to satisfy the query
2	Part of the document is relevant, but the information is intermingled with other topics
1	Document contains the keywords but is not sufficiently related to the topic
0	Document is not relevant / File not found / Server not responding

Table 4.1: Relevancy scores

The user values the first listed document the most. It should be given a higher importance than, say, the fifth document. Therefore, the relevance score is multiplied by a weight, which is simply an inverse function of the rank of the document. The first document is given weight 1.0. The tenth document is given weight 0.1.

Given the term words and the synonyms, AltaVista could be contacted in a number of ways:

1. **No expansions**: use the query as-is, without any special operators;
2. **NEAR operator:** use the NEAR operator between query terms;
3. **User selected synonyms:** contact AltaVista using the query terms and augment it with the synonyms;
4. **Require query terms:** contact AltaVista using the query terms and require them to appear in every document with the inclusion operator (+). Augment the query with the synonyms;
5. **Automatic selected synonyms:** Same setup as 4, yet without user interaction: automatically select the synonyms sets.

Comparisons were run among the following setups:

Setup	**Method(s) used in contacting AltaVista**
AltaVista	1
NetOracle Setup 1 *no query expansion*	1, 2
NetOracle Setup 2 *user selected synonyms*	1, 2, 3, 4
NetOracle Setup 3 *automatic selected synonyms*	1, 2, 3, 5

Table 4.2: Initial configurations

A sample query is elaborated: **hypnosis history**

To illustrate the diversity of the sites found, a short keyword was chosen to uniquely identify a site. For example, the label [medweb] refers to a site where all documents contain both the keywords but are poorly related to the query. The documents were manually reviewed and given a score. The top left number in the table is the total relevancy weight.

Setup			Total Weight
AltaVista			8
#	label	score	weight
1	[cet]	3	3.0
2	[drmike]	0	
3	[bloomu]	1	0.8
4	[wayneperk]	3	2.1
5	[medweb]	1	0.6
6	[medweb]	1	0.5
7	[medweb]	1	0.4
8	[medweb]	1	0.3
9	[medweb]	1	0.2
10	[medweb]	1	0.1

Setup			Total Weight
NetOracle setup 1			6.0
#	label	score	weight
1	[medweb]	1	1.0
2	[medweb]	1	0.9
3	[drmike]	0	
4	[medweb]	1	0.7
5	[medweb]	1	0.6
6	[cet]	3	1.5
7	[medweb]	1	0.4
8	[bloomu]	1	0.3
9	[wayneperk]	3	0.6
10	[monmouth]	0	

Setup			Total Weight
NetOracle setup 2			8.6
#	label	score	weight
1	[gainside]	2	2.0
2	[taiwan]	3	2.7
3	[bloomu]	1	0.8
4	[pasttimes]	0	
5	[reading]	3	1.8
6	[drmike]	0	
7	[westsussex]	0	0.9
8	[set.net]	3	0.2
9	[gen.emory]	1	0.2
10	[ibase]	2	

Setup			Total Weight
NetOracle setup 3			6.0
#	label	score	weight
1	[isadore]	0	
2	[drmike]	0	
3	[wayneperk]	3	2.4
4	[medweb]	1	0.7
5	[cet]	3	1.8
6	[medweb]	1	0.5
7	[monmouth]	0	
8	[medweb]	1	0.3
9	[medweb]	1	0.2
10	[bloomu]	1	0.1

Table 4.3: Comparison result

The best ranking was given by *NetOracle setup 2*, the configuration where the user has total control over the synonym sets that are to be used for evaluation. All four systems seem to have problems throwing out unrelated documents. Take for example the document on the site [drmike]. One may wonder why it is highly ranked by all 4 systems. The phrase that is deceiving is "Hypnosis Institute – Company history". Although these words are in close vicinity, it is totally unrelated to the history of hypnosis.

The systems were tested on numerous other queries:

Q1: premier Bulgaria

Q2: coke drug

Q3: diamond Antwerp

Q4: plants radioactive

Q5: religion in Thailand

System	*Q1*	*Q2*	*Q3*	*Q4*	*Q5*	*Normalized Total*
Alta Vista	**4.1**	**0.9**	**10.5**	**8.7**	**8.9**	**1.0**
NetOracle Setup 1	**2.3**	**0.8**	**12**	**6.4**	**6.6**	**0.78**
NetOracle Setup 2	**2.3**	**12.9**	**11.9**	**6.4**	**7.3**	**1.2**
NetOracle Setup 3	**5.1**	**11.2**	**12.3**	**8.8**	**8.5**	**1.4**

Table 4.4: Overall results

The most striking problem that arose with performing these queries is that AltaVista didn't do a very good job of detecting pages that came from the same site, resulting in poor diversity.

A number of observations were made:

Q1: premier Bulgaria

This query suffered heavily from the diversity problem: only pages from three average-quality sites were retrieved, which makes it hard to do a good comparison.

Q2: coke drug

The expansion of this query is critical in retrieval performance. "coke" is a non-descriptive and ambiguous word. But "cocaine" is a helpful synonym and "drug" allows the system to pick the correct sense. Note that although AltaVista did obtain a higher

score than *NetOracle setup 1*, the latter threw out all irrelevant and "File Not Found"-types of links.

Q3: diamond Antwerp

The results of this query showed the effects of incorrect automatic disambiguation. Although it came up with the best overall ranking, four of the links were related to baseball. In the disambiguation process, "Antwerp" and "diamond" are correlated through words like "geographical region" and "area". WordNet does not find a correlation between "Antwerp" and "gem, jewelry". This is where an extra lexical database may be beneficial.

Q4: plants radioactive

No good set of synonyms was found for any of the query terms. The user didn't augment the query in this case, so the score turns out to be the same as "no expansion".

Q5: religion in Thailand

This query results in one dominant site, which has, for each html page on that site, an index on the bottom of the page. This index happens to have religion and Thailand in close vicinity, even though they are unrelated. The relevance scores don't reveal any information.

CHAPTER 5

CONCLUSIONS AND FUTURE DIRECTIONS

With the dramatic expansion of the World Wide Web, finding the page you're looking for is sometimes like finding a needle in a haystack. The more information, the more chaos, ambiguity and redundancy. Paradoxically, although search engines get faster and faster, a user may end up spending more time searching for relevant information. Yet, this will change in the near future: the search engine market has become saturated and with the user as a supercritical evaluator, only the services that can keep up with the increasing expectations will survive. Thus, search engines are under high pressure to be more creative and more accurate in answering user queries. This goal is certainly achievable with the currently available IR technology.

NetOracle was designed as a basic example to show that more accurate responses can be obtained by better exploiting the information available. In this chapter, a number of improvements are listed that can be employed to increase the overall precision and recall of the system.

5.1 Some basic enhancements

- Increase the diversity of Web pages by contacting several search engines and combining the results.
- Filter out documents that come from the same site, eg. show only the best-ranked document of a certain site, for it is likely that the user will manually explore the site.

- Exploit HTML information from the document to come up with a better evaluation, eg. words in title or in the headers are weighted higher, follow and evaluate links, etc.

5.2 Improvements in automatically disambiguating words

Extra knowledge base

Consider the query: "Where do I find information on diamonds that are manufactured in Antwerp". This query contains enough information for humans to deduce the correct sense of diamond: "jewel, gem". Yet, NetOracle suggests "baseball diamond" as the sense to be used to augment the query. The word that triggers this wrong correlation in WordNet is "area":

Antwerp → city → geographical area → area

Diamond → baseball diamond → piece of land → area

In this example, the system would benefit from having access to an extra knowledge base that contains relationships between pairs of words that frequently co-occur.

Part-of-speech tagger

NetOracle only uses the noun-senses of a word for its automatic disambiguation process, which might lead to incorrect disambiguation when, for example, that word is grammatically used as a verb; e.g., consider the following queries all containing the word "lead" but with different meanings:

Where does the Amazon river lead to?

What is the atomic mass of lead?

Who played the lead female role in the movie "African Queen"?

The solution would be to determine the interpretation of the word (verb, noun, adverb, etc.) using a program that is commonly called a part-of-speech tagger.

5.3 Batch retrieval

A valid critique on NetOracle is that all of this downloading and analyzing of documents takes time. One may cache frequently retrieved data or store answers to frequently asked queries. Another improvement is based on the fact that users often retrieve the next set of documents using the previous query. The engine could perform a quick-search and return a set of "preliminary" links when the query is first performed. While the user is evaluating this first set of documents, the engine can use this idle time to do a complete textual analysis on the next set of documents, which will then qualitatively outperform the first set. This method minimizes delay.

BIBLIOGRAPHY AND INTERNET REFERENCES

[1] J. Allen. Natural Language Understanding, Benjamin/Cummings, 1987, ISBN 0 8053-0330-8

[2] AltaVista Search Engine: http://www.altavista.digital.com

[3] AskJeeves Search Engine: http://www.askjeeves.com

[4] M. Balabanovic, Y. Shoham, Y. Yun. "An Adaptive Agent for Automated Web Browsing", CS-TN-97-52, February 1997

[5] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery. "Learning to Extract Symbolic Knowledge from the World Wide Web", January 1998

[6] E. Foxley, G. Gwei. "Morpheme Extraction using Prefixes and Suffixes", (http://ftp.cs.nott.ac.uk/~ceilidh/papers/Morpheme.html)

[7] Carnegie Mellon Univeristy, "Lexical FreeNet: a program that finds relationships between words": http://www.link.cs.cmu.edu/lexfn/

[8] D. Jakob. "Finding Information on the Web", ISSN 1201-4338, 1995, (http://www.nlc-bnc.ca/publications/netnotes/notes15.htm)

[9] M. Porter. "An Algorithm for Suffix Stripping". Program (Journal), pages 130-138, 1980.

[10] S. Russell, P. Norvig. Artificial Intelligence, A Modern Approach, Prentice Hall, 1995, ISBN 0-13-103805-2

[11] G. Salton, M.J. McGill. Introduction to Modern Information Retrieval, McGraw-Hill, 1983, ISBN 0-07-054484-0

[12] M. Sanderson. "Word Sense Disambiguation and Information Retrieval", In Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval. Dublin, Ireland. pages 142-151, June 1994

[13] Guides, tutorials and statistics about search engines: http://www.searchenginewatch.com

[14] M. Stairmand. "Textual Context Analysis for Information Retrieval", In Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval. Philadelphia, US, pages 140-147, 1997

[15] E. Voorhees. "Using WordNet to Disambiguate Word Senses for Text Retrieval", In Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval. Pittsburgh, US, pages 171-180, 1993

[16] Yahoo! Search Engine: http://www.yahoo.com