ABSTRACT

NAMJOSHI, NIHAR. Web Information Retrieval using Web Document Structures. (Under the direction of Dr. Robert StAmant.)

Information domains such as the World Wide Web have enormous information content. The task of extracting information relevant to a particular topic, or trying to predict what sort of information a user is seeking is not a trivial task. For a user, finding information relevant to a particular area of interest can be inconvenient and sometimes frustrating as well. Studies have shown that when users are faced with such a task, they may get easily bored and thus leave a Web site.

Traditional Information Retrieval techniques rely on measures such as the frequency of a word in a given document, or the hyperlink connectivity of that particular web document. This approach may not necessarily bring out the important words or terms in a document and thus could be less effective while returning search results for queries. In our approach, we rely not only on the actual text in the document, but we also use the inherent formatting elements in Web pages, derived from the Hyper Text Markup Language (HTML) syntax to support our process of information extraction. We use rules to assign measures to important terms in a document in order to facilitate the relevant Information Extraction. We evaluated our system by asking users to test it and in addition, we compared our results with the results from a conventional search engine.

WEB INFORMATION RETRIEVAL USING WEB DOCUMENT STRUCTURES

by
NIHAR NAMJOSHI

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

COMPUTER SCIENCE

Raleigh

2004

APPROVED BY:

Christopher G. Healey
Dr. Christopher Healey

[illegible]
Dr. James Lester

[illegible]
Dr. Robert StAmant
Chair of Advisory Committee

DEDICATION

To,

Govind Namjoshi

&

Shakuntala Namjoshi

BIOGRAPHY

Nihar Namjoshi was born on November 27, 1979 in the city of Mumbai, India. He graduated from Fr. Conceicao Rodrigues College of Engineering (Mumbai University) in May 2001 with a Bachelors degree in Computer Engineering. He subsequently joined North Carolina State University to pursue a Masters Degree in Computer Science where he did research under the guidance of Dr. Robert StAmant.

ACKNOWLEDGEMENTS

First, I would like to thank Dr. Robert StAmant for his immense support, guidance and encouragement during the entire course of this work. I would also like to thank Dr. Christopher Healey and Dr. James Lester for their invaluable comments and suggestions.

There are simply no words fit enough to acknowledge the efforts and sacrifices taken by my family in order to let me be where I am today. Thank you all, Aai, Baba, Aaji and Mau. I just could not have done it without you.

Special thanks to all my friends, especially Nihar, Arpita, Shruti, Jitu, Ani, Rupali, Nagi, Lashmi, Sameer and Krish for always being there for me. Thanks to my roommates, Sushil, Amit and Sarang just for tolerating me the whole time. I would also like to thank all students from the Knowledge Discovery Laboratory for their support and for creating a competitive environment for research.

Table of contents

List of tables

List of figures

CHAPTER 1. Introduction

In this chapter, we discuss the unique nature of information on the World Wide Web. We outline why the process of Information Retrieval from Web documents is not a trivial process and then present an outline for the rest of the Thesis.

1.1 Information Structure on the World Wide Web

The World Wide Web has become an integral part almost everyone's lives. The number of Web pages on the Internet is extremely large[1] and this is an indicator of just how much information is present on the World Wide Web. The Web is an intricate nervous system of information and has had a steady and significant impact on our day-to-day lives.

The rapid and unregulated growth in the number of Web pages over the Internet has given it a haphazard structure, thus complicating the task of sharing and extracting information. Due to such rapid and unstructured increase in information, the question that needs to be addressed quickly and effectively is how to develop a way of representing information so that it presents an easier way of extracting relevant information to what the user is looking for.

In our work, we try to abate these inconveniences by intelligently extracting and structuring the information in a way so that makes it easier for a user to access it. It explores in a direction that is

[1] As of November 10, 2003, the popular search engine Google® was searching approximately 3.3 Billion web pages, which is just a subset of the total number documents on the World Wide Web [Web 1]

slightly different from contemporary methods. Our work involves determining the importance of specific information in a Web page, and organizing it so that we can understand in a better way what a particular Web page really represents. We monitor a user on a website and depending on the queries generated by the user, we try to extract information that is relevant to what the user wants. We have developed several heuristics based on the inherent structure of Web pages that is provided by the Hyper Text Markup Language (HTML) syntax. These Heuristics aid us in extracting vital information about the data residing on a given web page [18]. Although most websites are migrating to a content-based approach, there is still a huge amount of information present in existing Web pages that needs to be structured so that users can obtain the desired information. Our work looks into structuring information for effective extraction of data from the Web pages so that it can be interpreted for the information that it represents.

1.2 Challenges in Web Information Extraction

A large volume of information on the World Wide Web is present in the form of structured documents (HTML in most cases) and can be considered analogous to a huge text database (only in some respects) whose records are connected via hyperlinks; as is the case in a distributed database. However, extracting information from the Web is not the same as extracting information from a text database. Listed below are some of reasons why information extraction from the Web is not as trivial as mining regular text files or querying object-oriented databases.

(i) Amount of Information

There is no way of directly quantifying the amount of information present on the World Wide Web. Nevertheless, it is fair to assume that the Web contains information about almost any topic known to us. Depending on the type of a Web site, it could potentially hold a huge array of topics, each addressed in sufficient detail.

(ii) Information Structure of Web Documents

Web documents do not have a standard unifying structure that they rely upon. Web data has a scarce structure and its organization over the network makes information sharing and information mining a difficult task. Additionally, such a huge repository of information also contains a lot replicated and redundant information. There are no set styles or standard rules for creating Web documents. Web pages and Web sites are created by a set of very distinct people, thus resulting in the wide variety of Web documents.

(iii) Difference between Web Information Extraction and Text Database querying

The data on the Web has a non-uniform outer structure to it but at the same time it has uniformity in its internal representation. (By the term internal uniformity, we are referring to Web documents such as native Web pages that are bound by the HTML structure). The information present in the formatting structure can be used to an advantage while extracting information from the Web. However, in Web documents, there is no indexing by categories or by any other attribute as in conventional text databases.

(iv) Life Span of Information on the Web

The information on the web is highly dynamic. Some resources on the Web have extremely volatile content (e.g. news Web sites), while some resources have information that keeps appending (e.g. educational Web sites, or regular corporate Web sites).

Some information sites on the Web have remarkably static content; an example would be information websites such as online encyclopedias. Thus, the lifespan of information on the Web is myriad which makes it difficult to mine it for information.

(v) Relevant Information in Web Documents

In spite of its size, it is believed that only a small portion of information (roughly less than 1%) is relevant [8] to a particular user in a given session, while the rest seems unrelated. It is said that 99% of the information is useless to 99% of users.

(vi) Data Object representation in Web Documents

The property of a data object[2] represented in a Web page may not be dependent only on the context but also on the relationship between other objects that it may be hyper-linked with. These data objects can be quite dissimilar to each other at times even for a single Web site.

[2] Here we use the term object to indicate an entity that may be represented by a chunk of information on a particular Web page or a concept represented by the entire Web page.

1.3 Organization of the Thesis

The remainder of this Thesis is organized as follows. The next chapter discusses the related work in this field as well as the contributions that made this work possible. Chapter 3 discusses the various theoretical concepts, which were used extensively in our work. This chapter also gives an overview of how the documents on the Web are organized. Chapter 4 presents the system architecture in detail and how it was implemented while chapter 5 lays out the experimental results obtained during the course of this work. Chapter 6 concludes the Thesis while Chapter 7 gives directions for further work that can be done to append on this work. Appendix A provides a reference to the HTML tags that are used in our analysis, Appendix B enumerates the complete list of stop-words that we used in our system, while Appendix C describes in detail the Porter-Stemmer algorithm which we have used in our work.

CHAPTER 2.
Related Work

In this chapter we discuss the related work that has been done in this area. We also briefly look at contributions that run along a parallel vein to our work.

A large amount of information on the Web or in the form of regular documents exists in a Natural Language form. In order to use this information for analysis and further manipulation, it must be separated from the unnecessary information so that it has a more structure to it, thus facilitating extraction of facts. Search engines exploited the initial boom of information extraction. However, most search engines reply on caching the Web pages and it kind of defeats the bigger picture of extracting only the relevant information.

In the late 1950s and the early 1960s, research groups identified the importance of automatically analyzing and structuring natural language data

Ever since the boom of the World Wide Web, efforts have been made so that the users access the information presented on the Web easily. The earliest attempts to extract information from a document can be traces back to the 1950s when linguist Zellig Harris proposed the idea of reducing the information in a document to a tabulated information structure. Till today significant efforts have been made to try to predict user behavior on the Web. Though out work is a side-step from predicting web links that a user might intend to visit, we feel it is relevant to discuss some of the work that led to our idea or predicting the information needs of a user.

A large number of empirical studies have been conducted in detail as to how groups of selected users browse through Web data. Two of the most widely cited and the earliest studies on user browsing behavior are presented by Catledge and Pitkow [3] and Tauscher and Greenberg [15]. In both studies, data was collected on the client side for a specific set of users over a given time period. The Tauscher study found a string 'recency' effect viz. a page that is revisited is often a page that has been visited recently than in the distant past. This tends to co-relate with the empirical observation that users often tend to click on the 'Back' button of the browser to visit recently accessed pages fairly more than using History contents or Bookmarks. Greenberg [6] had also presented earlier that repetition of user actions is a strong characteristic of user behavior in several other domains.

Another important area that can be looked at is explicit querying done by the user in a question answer format. There has been work done at the School of Computer Science at the University of Waterloo. This work is along a similar vein to what we have looked into. This work relies on the notion that conventional search is keyword based and that the traditional search results involve sifting through a large number of returned hyperlinks. The WebQA project tries to provide a declarative query based approach to Web information retrieval by using question-answering technology. Their approach consists of first using a Meta search technique to candidate results from search engines and then using information extraction try to find the right answer to the query posted by the user.

CHAPTER 3. Theory

In this chapter, we lay the foundations of our work by introducing the theoretical concepts that are used in our study. We discuss the structure of Web pages and then review some fundamental elements in the process of Text Analysis. We define the various word measures that we have proposed as well as the rules that we have defined to classify the importance of various terms in a given document.

3.1 Structural Overview of Web Pages

HTML language is based entirely on formatting and presentation. The Web is evolving towards a knowledge-based system where data will be represented in a query-able knowledge base accessible to the users. However, the process of migrating from regular Hypertext documents such as HTML to such query able knowledge bases is not trivial. The process of converting a Web page into an appropriate format for entering into a knowledge base is known as wrapping. Since Web pages change often, it is necessary to have an automated system that performs such wrappings and extracts information in order to update the knowledge base. Data exchange standards like XML are rapidly gaining popularity, but it may be a while before the entire web is represented in that manner. In addition, organizations may prefer not to release the XML data to users but rather present the information using style-sheets.

Every Web document is a combination of textual information and tags into which text data has been embedded. This tag embedded text data is identified and processed by Web browsers to display the textual information in formats that have been defined by the corresponding tags in the original HTML source. Appendix A gives a brief reference to HTML tags. It also gives a list of certain HTML tags that were considered by our system while performing the analysis.

There are different kinds of HTML tags; they are used as references to other pages, as references to images, to signify textual attributes etc. These tags are thus used to assign special properties to a chunk of text in a Web page. The text embedded in a particular set of tags will inherit the property defined by that particular tag e.g. the text embedded within the <B> and </B> tags indicates that the particular text is to be represented in bold type face. Thus with the help of such tags, the Web page designers can indicate which text belongs to its corresponding attributes in the Web page.

The textual tags found in an HTML Web page are of profound importance in our work. Some of the important HTML tags considered in our work are explained in the next section.

The tags that indicate the title of the page i.e. the <TITLE> … </TITLE>, in addition to the <META> … </META> tags provide useful information as to what the essence of a particular Web page is. It seems obvious that if a particular word belongs in the page's <TITLE> tag, then that word should be considered more important when it comes to determining the relevance of that word on the page. However, on occasions, the information provided by the <TITLE> tag is far too generic and so in such cases the <META> tags come in handy. Another group of tags that are useful while determining the importance of certain words in a Web page are tags such as <B>…</B>, <U>…</U>, <EM>…</EM>, <I>…</I> and several others. Tags like </P> indicate a break in paragraphs and hence can be used to determine context of certain text within the scope of a paragraph. There is however, a marked difference between the emphases given to a word by the <TITLE> tag in comparison to other emphasis tags. This is because the emphasis to the body text is usually a result of a conscious effort by the person who lays out the content of that particular page whereas in some situations, the title of the page is generated automatically without explicit instructions from the Web page designer. In our work, we have an option of explicitly excluding or including the data present in the <TITLE> and the <META> tags.

For basic information extraction, any standard textual mining techniques can be used [7] [10]. However, the use of such techniques is not free from its share of limitations, primarily due to the variable nature as well as the fact that a large number of pages present. In addition, the nature of information that a user might need may be far from just the skeletal information extracted by a Web crawler or a Web bot deployed by a Web search or a Web directory service.

3.2 Introduction to Text Analysis

Text analysis is an integral part of any Information Extraction system. In the next few sections, we will discuss some techniques for the textual analysis of information on the Web pages. Some of these techniques have been developed over the years in the areas of Information Retrieval, Machine Learning, and they include methods such as indexing, categorization and scoring of textual documents. Although we will be using such techniques for working on only textual information, these techniques can also be applied to other variants of standard Web documents such as images, video and audio [2]. However, our focus will remain strictly only on the textual content in Web pages.

3.3 Tokenization

The process of tokenization involves the extraction of plain words and terms from a document by removing all the formatting elements as well as the structural data and metadata from Web documents. An example of this would be the removal of HTML tags from a Web page to get pure text. The process of extracting plain words and terms from a document and identifying each word as a separate entity is known as tokenization [1]. This information allows us to weigh a particular token (word) in term of its position, its emphasis and several other factors. However, the HTML tag information associated with a token is not relevant during the Lexical analysis of the text in a given document.

Since we have to keep track of the HTML tags associated with every token, the lexical analysis of a Web page is not trivial in our system. While parsing the document, we ignore all HTML syntaxes (that are not deemed important) i.e. abstract the Web page to a simple text document. Punctuation marks are removed during tokenization. The Entire text is converted to uppercase to reduce the overall index entries during the latter stages of processing.

3.4 Vocabulary Reduction

Some commonly occurring words in English language include prepositions, articles, pronouns etc. These words do not have any stand alone information content [5] in them apart from certain situations where a specific context makes them relevant. Such words are known as 'stop words' and they take up a large amount of real estate in a typical text fragment. In an average case, such stop words make up 20-30% of any given text document. Removing such stop-words from the document helps us in reducing the volume of overall text to be processed. Removing stop-words is also known to improve computational efficiency during the retrieval. Appendix B provides a list of stop words checked and removed by our system.

3.5 Text Conflation

It is often desirable to reduce all morphological variants of a given word to its single index term (stem). Stemming consists of reducing a word to its stem, which in turn could be used as an index for all its un-stemmed variants. A well-known stemming algorithm for English vocabulary [12] was developed by Porter as a simplification to some previous work by Lovin. It uses a predefined suffix list with a set of rules associated with each one of them. The Porter-Stemmer algorithm is a five step serial process. Appendix C explains the Porter-Stemmer algorithm in detail. Stemming does have some disadvantages. Unless there is a dictionary service or some other kind of look-up available for proper nouns, special terms or acronyms, the stemming algorithm could end up stemming proper nouns which is not a desirable situation.

3.6 Web Data

In order to study user information needs on a particular Website, we need to look at additional data apart from just the actual Web pages which the users browse. Web data is usually obtained through automated site management tools installed on the Web server or monitoring users directly while they are browsing. In the next few sections we look at methods to obtain the browsing data of various users during Web site visits.

3.7 Web Server Access Logs

Whenever any user visits a given site, all transactions that occur between the Web server software and the individual user are recorded in an ASCII format. These flat files are known as Web server log files. The primary log is known as the server transfer log or the access log. These log files typically contain fields such as the IP address from where a user request was instantiated, the time of the request, the method of request (e.g. GET, POST, HEAD etc.), a numerical code called the status code that indicates the server response, the size of the transfer in bytes and several other attributes. The access log is not the only log that records the transactional information of the user-server communication. The referrer log contains information about the Web page from which the request has originated, indicating the hyperlink from where the user came to this particular page (if it were the case). In addition to these standard logs, there are other logs such as error logs and agent logs which are not of any significance in our work, but could yield some interesting information.

These server side log files can provide a wealth of information about how users are behaving on a given Web site, thus presenting an indication of their information needs. However, interpretation of such information requires considerable care, as there are several privacy issues [4] [14] [9] that need to be addressed.

3.8 Identifying Individual Users

In the process of analyzing the logged data obtained from the Web servers, we would like to be able to associate specific page requests with individual users.

The most straightforward way to do this is to have users explicitly identify themselves to the Web server using an explicit login and password. This usually requires a registration process where a user profile is created. (e.g. personalized news Web sites). This method of user identification is the most reliable as well as accurate for associating users with their actions. There is a down side to this option though. There is that additional need of requirements on the Web Server to manage logins and sessions and additional effort from the user too, which in turn may serve as a discouragement to certain users.

Another way of identifying users is via the grouping of IP addresses obtained from the access logs. However, a given IP address may not have a one-to-one mapping to an individual user in circumstances where there are subnets created for an IP address or if proxies are being used. For subnets, one IP address may correspond to several users. In addition, there could also exist a situation where an IP address may belong to a shared computer and could possibly have different users at different times.

One approach to get over the above-mentioned problem is by using cookies. A cookie is a small text file that is temporarily created on a user machine by the Web server. A cookie allows the Web server to uniquely identify and track users' requests as individual entities on a given machine over a given period of time. When a user returns to a previously visited site, this cookie file can be accessed by the Web server and use it to identify the user as the same one that had visited the site earlier. Due to this explicit identification, the actions performed by the same user such as page requests etc, can be linked together using cookie information. One advantage of using cookies is that even if the IP address of a machine is dynamically altered over time, the cookie information remains the same.

Commercial Web sites use cookies extensively for improving quality of matched page requests to individual users. The use of cookies cannot be taken for granted. Although well over 90% of Web surfers have their cookie feature permanently enabled on their browsers, it is also possible to explicitly disable cookies, thus making it impossible for the Web server to store any information on the user's machine. Today, many websites require cookies to be enabled in order to visit their Web site. Cookies rely on an implicit co-operation from the user and hence raise a privacy concern.

3.9 User Side Data

Probably the best method of collecting information about a user is by monitoring the client side i.e. software present on the user machine that explicitly records all user actions. Such methods could easily abate certain problems associated with server side monitoring and logging. Such monitoring can be done through setting up an observation lab to study users or by having come sort of a real time monitoring system on the client side.

There are several advantages of obtaining data from the client side in comparison to obtaining server side data. These include,

- The most direct identification of users is obtained. A particular user may be logged on to a machine if it is a shared machine, thus users can be easily identified. In the case of a personal machine, the identification is straight forward.
- A record can be made of all the Websites that a user has visited instead of just the page visits of a particular Web site.

The client side data is extremely rich in content as compared to Server side data. Some important statistics such as “page view duration” and “Session Time” can only be obtained via client side monitoring.

3.10 Word Importance Measures

When we obtain a Web document with certain textual information in it, we need to have some measures, based on which all the words or tokens present in that document can be compared to each other on the basis of certain attributes. Such a comparison allows us to determine the importance of certain words over others. To carry out such a comparison, we have used certain word-measures. Similar criteria have been proposed and used in the past. However, in most cases only a single attribute is used. We have identified the need to make use of multiple attributes at the same time in order to compare words in a proper manner. For example, through our analysis, we have found that the text that represents the title of a page is not always adequate to sufficiently describe what kind of information that page contains. We can modify our claim by saying that if a certain word appears in the title of a page and it is also emphasized sufficiently in rest of the document body itself, or if it has a high frequency of occurrence in the document, then it can be said that the particular word is of considerable value on that Web page.

In order to calculate the word relevance in a Web page, we have devised certain attributes that allow us to obtain the measure of a page’s information content. For each of these measures we have discrete values associated with them. Some are bi-valued while others are tri-valued nominals that give us an idea of the strength of that particular attribute for any given word in the document. Based on the values taken by these attributes, we have derived rules that map these attributes with a scale of word relevance introduced later on in the literature. These attributes are defined as follows.

(i) Relative Text Frequency (RTF)

The Relative Text Frequency of a word is given by the ratio of the number of times a word appears in a Web page to the frequency of the most frequent word in that Web page. Mathematically,

$$T_f(i) = N_f(i) / N_{Page\ Max}$$

... $T_f(i)$ is the Relative Text Frequency of the word

... $N_f(i)$ is the frequency of the word.

... $N_{Page\ Max}$ is the frequency of the most frequently occurring word in the page.

The calculation of the relative text frequency does not seem intuitive enough because of the divisor value in the ratio, which is the frequency of the most commonly occurring word and not the total number of words in the document. This is done to reduce undesirable effects in the measure if there are too few or too large number of words[3] in the Web page. The bounds that decide whether the relative text frequency is high, low or medium are tunable, but for most test cases, we used an equally partitioned space for all three values. The values that the Relative text frequency can take are HIGH, LOW and MEDIUM.

[3] We have mentioned vocabulary reduction briefly. When the frequency measures were taken, we took the document in its reduced form. We also looked at cases where the original document is without any reduction to calculate the frequencies. Though, in our system the measures are all obtained after the stop words have been removed.

(ii) Paragraph Text Frequency (PTF)

This measure is similar to the Relative Text Frequency measure. In this case, the measure is restricted to a single paragraph only. As in the case with the previous attribute, this too is tri-valued. Mathematically,

$$T_f(i) = N_{fp}(i) / N_{Para\,Max}$$

... $T_f(i)$ is the Paragraph Text Frequency of the word.
... $N_{fp}(i)$ is the frequency of the word in the paragraph
... $N_{Para\,Max}$ is the frequency of the most frequently occurring word in the paragraph.

(iii) Word Emphasis Function (WEM)

This measure is not easy to calculate because of the various kinds of emphasis functions present i.e. Bold, Italics, Headings, Emphasized text etc. We studied several Web pages and created an ordering of these Word Emphasis elements. The maximum priority or weight is given to paragraph headings and the title in a Web page. In Appendix A, we see a list of all the HTML tags that are used in our work, which could constitute towards word emphasis in an HTML structured document. Paragraph headings are identified using the paragraph breaks in conjunction with certain other formatting elements such as Headings or bold text.

Moderate word emphasis is given to text that has formatting elements such as bold face, underlining etc. Lists (bullets) also come into this category. Currently we do not have a means to compare the font sizes of text in order to evaluate them, but it could be provide important cues towards word emphasis. Low word relevance is generally associated with regular unformatted text in the document.

We have devised rules for categorizing a word if it has a high, medium or low emphasis in the document. For example, occurrence of a word in the <TITLE> or the <META> tag of the document or as an emphasized heading in the document will return it a high degree of word relevance.

(iv) Title Text Frequency (TTF)

This measure is similar to the relative text frequency but considers the words that are present in the title and the Meta tags of the Web page only. Title text frequency is bi-valued. Therefore, a given word can have its Title text frequency attribute take on only one of two values, either TITLE or NOTITLE. Mathematically,

$$T_f(i) = N_{ft}(i) / N_{Title\ Max}$$

... $T_f(i)$ is Title Text Frequency of the word.
... $N_{ft}(i)$ is the frequency of the word in the Title and Meta tags combined
... $N_{Para\ Max}$ is the frequency of the most frequently occurring word in the Title and the Meta tags combined.

(v) Word Position (WOP)

It has been found that usually, the most important information in a text fragment is contained in the first third and the final third part. Thus, more weight has to be given to the initial third of the sentences that occur in a paragraph as well as concluding third of the sentences in a paragraph. This is again a bi-valued attribute, indicating whether the words belong to the important part of the text (first third or the final third) or not.

3.11 Document Word Relevance

Classical Information Retrieval techniques have traditionally represented text as a vector space model with multi dimensional attributes. The representation of text using the Vector Space [16] [11] model has a very high dimensionality. In our work, we try to circumvent this high dimensionality problem by trying to represent each unique word in a document with just one attribute. This attribute is responsible for representing the "relevance" of that particular word in the document. We name this attribute as the 'Degree of Relevance' of that word. Based on all the word importance parameters discussed earlier, each unique word in the document gets associated with a certain Degree of Relevance. The Degree of Relevance is a nominal valued attribute for a given word with four possible values that it can take. We have chosen four levels representing the Degree of Relevance of word that occurs in the document. These levels are described in table 3-1.

A relevance level of one indicates that a given word almost describes the page. A suitable example would be a word that occurs frequently in the document in regular text sections, title as well in certain paragraph headings. The second level Degree of Relevance usually means that the particular word strongly represents a part of the given document. This level of Degree of Relevance is the most frequently occurring and the most important one. The level 1 Degree of Relevance is rare because it is difficult to get a set of words to represent an entire document, while it is much easier to get certain words to strongly represent a part of the given document. In order to classify words into a Degree of Relevance we have developed a set of rules.

Table 3-1 The Four Degrees of Relevance

DEGREE OF RELEVANCE	DESCRIPTION
1	The word almost describes[4] the document by itself
2	The word strongly emphasizes sub-sections of the document
3	The word is moderately represents the document and sections
4	The word has negligible relevance in the document

3.12 Word Relevance Rules

Table 3-2 provides a complete list of rules that we have derived. A missing entry means that the particular attribute was not considered for that particular rule.

[4] We have found out that it is rare to find a single word with a Degree of Relevance of one. This is because a document can be described more effectively by a group of words rather than a single word.

Table 3-2 Rules to Assign the Degree of Relevance

ANTECEDENTS					DEGREE OF RELEVANCE
RTF	**PTF**	**WEM**	**TTF**	**WOP**	
HIGH	MEDIUM	HIGH	TITLE	-	**1**
HIGH	MEDIUM	HIGH	TITLE	YES	**1**
MEDIUM	MEDIUM	MEDIUM	TITLE	-	**1**
HIGH	HIGH	HIGH	TITLE	-	**1**
HIGH	HIGH	HIGH	NOTITLE	YES	**1**
HIGH	MEDIUM	MEDIUM	TITLE	-	**1**
HIGH	HIGH	MEDIUM	NOTITLE	YES	**2**
LOW	MEDIUM	LOW	NOTITLE	NO	**4**
MEDIUM	HIGH	MEDIUM	NOTITLE	YES	**2**
HIGH	HIGH	MEDIUM	NOTITLE	YES	**2**
MEDIUM	HIGH	HIGH	NOTITLE	NO	**2**
MEDIUM	MEDIUM	HIGH	NOTITLE	NO	**2**
LOW	LOW	LOW	NOTITLE	NO	**4**
MEDIUM	MEDIUM	MEDIUM	NOTITLE	YES	**2**
MEDIUM	HIGH	LOW	NOTITLE	YES	**2**
MEDIUM	MEDIUM	LOW	TITLE	NO	**1**
HIGH	MEDIUM	HIGH	NOTITLE	NO	**3**
MEDIUM	MEDIUM	HIGH	NOTITLE	YES	**3**
HIGH	HIGH	LOW	NOTITLE	YES	**2**

3.13 Assumptions and Limitations

Though we have developed certain measures of evaluating the importance and relevance of certain words in a given document, there are still cases where these measures end up giving abnormal results and we try to explain it in the following section.

One example is when a word is extremely abundant in the document but still does not accurately depict its importance in the document. Such a word with a very high frequency does not necessarily mean that it represents the document aptly. Such words could have multiple meanings depending up on the context in which they are used e.g. ‘class’. Such words are termed as ‘jokers’. Jokers are words that are frequent but have different meanings in different contexts and such words pose a problem to any Information Retrieval system. Though such occurrences are not that common, it is a situation where the frequency criterion does not give the correct indication. A limitation of the word position criterion is that it does not yield satisfactory results when the length of the paragraph is extremely long.

In our work, we try to offset all these limitations by using a number of attributes. It is our belief that using a large number of criteria will offset each other for certain limitations.

CHAPTER 4. The System

In this chapter we look at the architecture of our system as well as the various steps in the process and how different operations take place. We also present a sample test case with a step by step analysis as it propagates through various components of the System.

4.1 System Components

Figure 4-1 shows the overall architecture of our system. The main components in the system are the Web Server, the Data Repository, the Information Analyzer, the Information Repository and finally the Monitoring Engine. We look at each of these components in detail.

4.1.1 The Web Server

The Web Server holds the actual Web pages of the Web site that has to be analyzed by our system. For the Web Server, we used the i-planet™ Web server, running on a Windows 2000 Server™. This server provides support for Java Server Pages (JSP). This technology is used to generate HTML pages in real time with the query results that are deemed appropriate by the Monitoring Engine. The server access logs from this Web server are also necessary for the purpose of our analysis.

4.1.2 The Data Repository

The data repository is the place where all the raw information for the analysis is stored before being acted upon by the Information Analyzer. The data repository contains three types of data viz. the logs from the Web Server, the information obtained by performing some client side monitoring and the actual Web pages in consideration. The data from this repository is taken by the Information Analyzer to produce results, which are stored in the Information Repository. Our implementation of the Data Repository is a simple collection of flat text files (Web logs, Client side data text files[5]) and the HTML pages of the Web site.

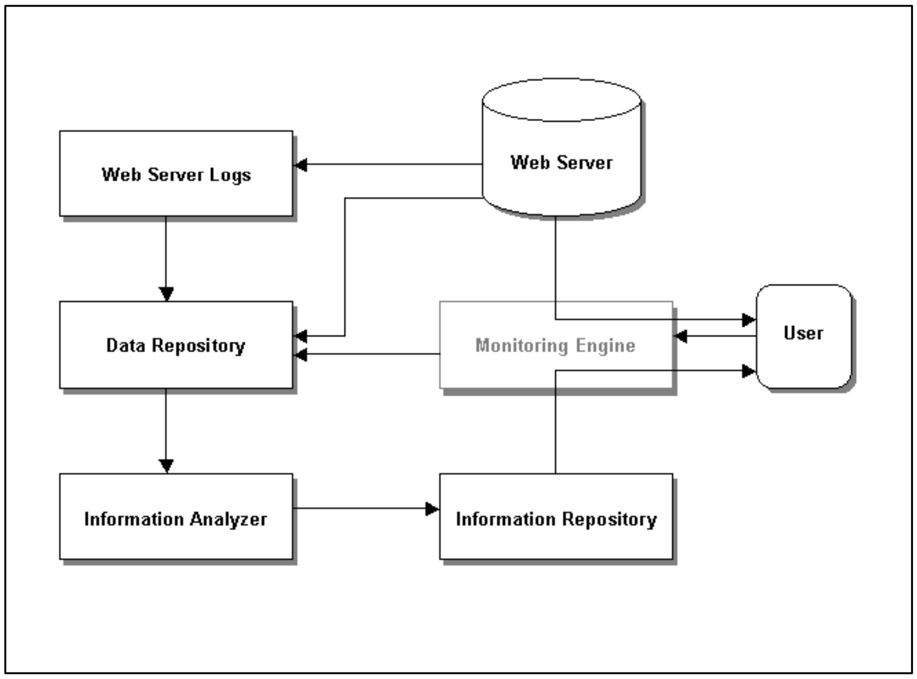


Figure 4-1. System Architecture

4.1.3 The Information Analyzer

This is the most important component of the system in the pre-processing stage. This component is responsible for extracting all the necessary information from the HTML pages that are stored in the Data Repository

The Information Analyzer is responsible for all the lexical analysis and the textual processing on the Web pages before they can be stored into the Information Repository. The first job of the Information

[5] The flat text file of client side data contains a tab-separated file that includes fields such as time_spent_on_a_page and the session_time.

Analyzer is to reduce the information content by filtering out the data that could possibly be useless in terms of information content. The Information Analyzer carries out the various text processing techniques that we discussed in discussed in chapter 3. They include vocabulary reduction by removing the punctuations, stop words etc. It is followed by the stemming of words to reduce complexity. Figure 4-2 shows the data flow and the various operations carried out by the Information Analyzer.

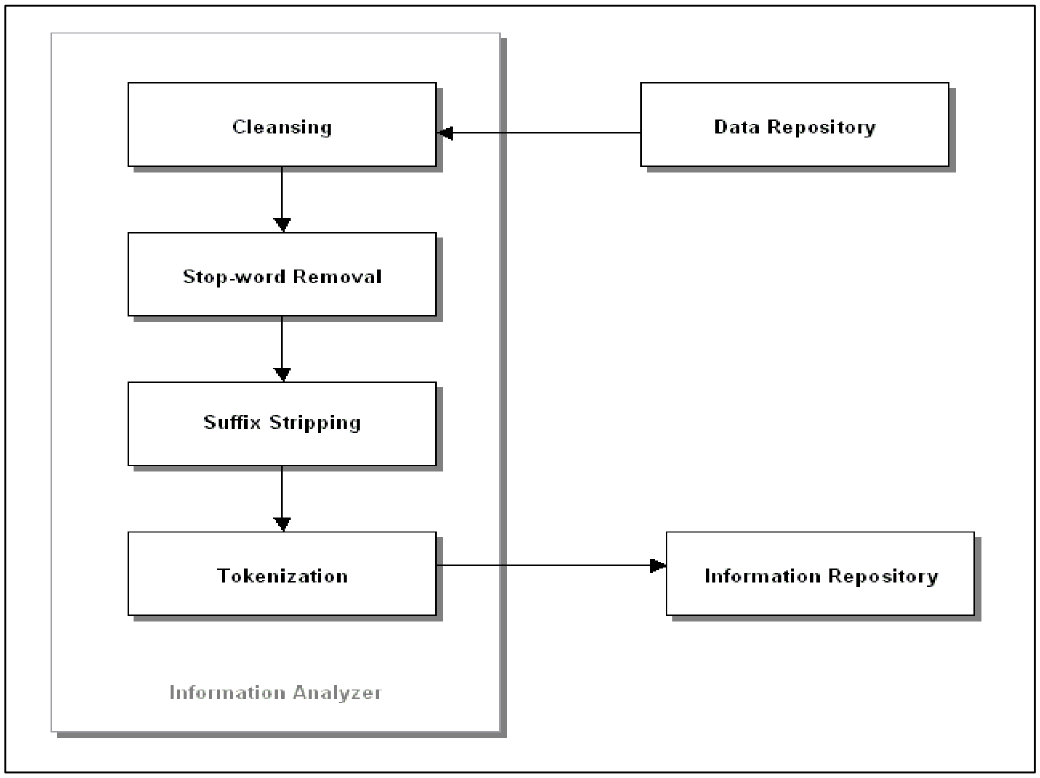


Figure 4-2. Data Flow in the Information Analyzer

The Information Analyzer fetches the Web pages from the Data Repository and cleanses it first. Cleansing includes removing information that has no relevance to our analysis. White spaces, comments, author information etc are examples of elements that are cleansed out by the Information Analyzer. For our analysis, we give no importance to the images that are present in the HTML document or scripts if they are present. The Information Analyzer strips the HTML document of all script code as well as image links. Since we may need information about links to other pages from the page currently being processed, images that are hyperlinks to other document are hence not removed.

The Information Analyzer parses the HTML document[6] for stop words and removes them. Stop-words are commonly occurring words in English language such as articles, prepositions and a large subset of adverbs that do not give any information about the semantic content of the text around it, or the document as a whole [6]. Removing these stop-words does not reduce the information provided by a document. Removing the stop-words from the HTML page reduces the number of tokens that will be generated, thus reducing the overall computational effort due to a substantially reduced document size. Some common stops words that are removed are 'it', 'the', 'of', etc. Appendix B provides a complete list of all stop-words used by the system for the cleaning up process.

The next step is the removal of suffixes from words to reduce word complexity. We used the Porter-Stemming suffix stripping algorithm to reduce the words to their basic stems. This reduces the overall word count that needs to be stacked away into the Information Repository. Refer Appendix C for a detailed explanation of the Porter-Stemming Algorithm.

The next operation handled by the Text-processor is tokenization. The process of tokenization involves parsing the Web page and identifying the unique elements in it and cleansing it at the same time. At this stage of processing, the Text-processor identifies each unique token along with its attributes such as frequency, word position and the tags in which it has been embedded into an index table. A sibling is created for every page processed from the Web server. A sibling of a given page summarizes the information content of the actual Web page from which it was derived. Thus parsing through a sibling will tell us what important terms are contained in its corresponding Web page. The Information Analyzer is responsible for creating every sibling.

[6] The Text-processor only deals with the text that is embedded within HTML tags. The actual tags are not processed. However, the Text-processor has to keep a track of tags that are currently embedding the text that it is dealing with. Monitoring this tags information allows Text-processor to store vital emphasis information, which is then used by the Information Manager.

4.1.4 The Information Repository

The Information Repository contains the sibling pages that correspond to the actual Web pages of the Web site. Each sibling contains information describing the actual Web page that it represents. Each sibling has two levels of Information Content. The top level information content in a sibling contains the tokens that have the maximal ratings in the analysis performed by the Information Analyzer. Therefore, it turns out that usually the Information that is entered in the first level is the summary of the most important terms in a document.

The second level of information in a sibling consists of the various paragraphs and the paragraph content. This level is important in order to pick out information from the corresponding Web page and present it to the user when required. We will see how these fields are created with the help of an example later on. The structure of a sibling is elementary. This is done to facilitate searching and modify siblings in real time if the need be. The header of the sibling indicates the Web page that it corresponds to.

A sibling contains a list of tokens (unique words found in the text) along with its Degree of Relevance associated with it. The sibling is thus a quick reference map of its corresponding document. With the help of siblings, content matching can be done via siblings themselves without actually comparing the pages directly to one another.

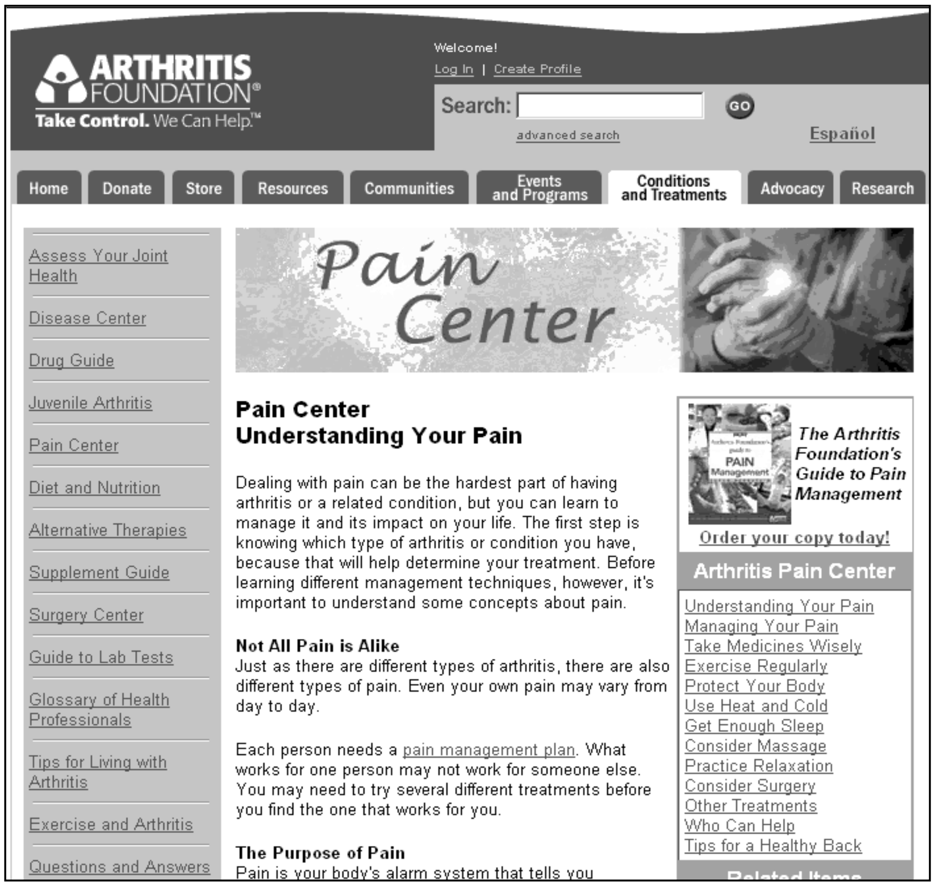
4.1.5 The Monitoring Engine

The Monitoring Engine is the place where the entire decision making of the system takes place. The Monitoring Engine has a two-fold purpose. First, it is responsible for collecting data while the user is browsing and second; it is responsible for giving the relevant information back to the user. The Monitoring Engine looks at the current document that is being viewed by the user. It then pulls out its corresponding sibling to load the information content of the current page. The Monitoring Engine then monitors the user while it navigates to another page. The user can also explicitly enter a search query in order to search for a key word.

4.2 An example

Having discussed the system components, we now give an example of how a sample Web page is operated upon by the Information Analyzer and how is gets archived into the Information Repository.

Figure 4.3 shows a randomly chosen Web page[7] from the Arthritis Foundation® Website[8]. We will use this particular page to illustrate step-by-step how each operation is carried out.



Pain Center
Understanding Your Pain

Dealing with pain can be the hardest part of having arthritis or a related condition, but you can learn to manage it and its impact on your life. The first step is knowing which type of arthritis or condition you have, because that will help determine your treatment. Before learning different management techniques, however, it's important to understand some concepts about pain.

Not All Pain is Alike
Just as there are different types of arthritis, there are also different types of pain. Even your own pain may vary from day to day.

Each person needs a pain management plan. What works for one person may not work for someone else. You may need to try several different treatments before you find the one that works for you.

The Purpose of Pain
Pain is your body's alarm system that tells you

Figure 4-3. A sample Web page

Figure 4.4 shows a part of the source code of the HTML document. Notice how the <META> tags in the source code are found wanting in terms of describing the content of the Web page. The title of the page provides a clearer indication of the information content on that page. Also, note just how much information is spent only on the formatting elements in the page e.g. font specifications, text color, and text alignment. The cleansing phase of the Information Analyzer removes all such text that is of

[7] http://www.arthritis.org
[8] http://www.arthritis.org/conditions/pain_center/default.asp

no use to us during the latter part of the processing. As mentioned earlier, it will also remove the image tags <IMG> that are not a part of a hyperlink. In our system, we cannot find associations amongst the various entries of tabulated data. Therefore, we consider any tabulated data as a regular text. In our case, the tabular structure will be ignored. We could have added a provision for ignoring the tabular formatting and only consider the text in the tables, but there are occasions where the tabular data is too vast to be considered.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 TRANSITIONAL//EN">
<!-- SAVED FROM URL=(0059)HTTP://WWW.ARTHRITIS.ORG/CONDITIONS/PAIN_CENTER/DEFAULT.ASP -->
<HTML><HEAD><TITLE>MANAGING YOUR PAIN</TITLE>
<META HTTP-EQUIV=CONTENT-TYPE CONTENT="TEXT/HTML; CHARSET=WINDOWS-1252">
<META CONTENT="ARTHRITIS FOUNDATION, ARTHRITIS" NAME=KEYWORDS>
<META CONTENT="ARTHRITIS FOUNDATION" NAME=DESCRIPTION>
<META CONTENT="MSHTML 6.00.2800.1264" NAME=GENERATOR></HEAD>
<BODY VLINK=#993399 ALINK=#FF6600 LINK=#0066CC BGCOLOR=#FFFFFF LEFTMARGIN=0
MARGINWIDTH="0" MARGINHEIGHT="15" OFFSET="0"><!-- BEGIN HEADER INCLUDE -->
<DIV ALIGN=CENTER>
<CENTER>
<TABLE CELLSPACING=0 CELLPADDING=0 WIDTH=700 BORDER=0>
  <TBODY>
  <TR>
    <TD WIDTH=700 COLSPAN=2><IMG HEIGHT=18
      SRC="MANAGING YOUR PAIN_FILES/WAVE.GIF" WIDTH=700></TD></TR>
  <TR>
                                                                                   …
<LI><B>
       DEPRESSION OR STRESS</B>, WHICH RESULTS FROM LIMITED MOVEMENT OR NO LONGER DOING
       ACTIVITIES YOU ENJOY. YOU CAN GET CAUGHT IN A CYCLE OF PAIN, LIMITED/LOST ABILITIES,
       STRESS AND DEPRESSION THAT MAKES MANAGING PAIN AND ARTHRITIS SEEM MORE DIFFICULT.
       </FONT></LI></UL>
     <P><FONT FACE=ARIAL,HELVETICA,SANS-SERIF COLOR=#000000 SIZE=2><B>DIFFERENT REACTIONS
     TO PAIN</B><BR>PEOPLE REACT DIFFERENTLY TO PAIN FOR SEVERAL REASONS. PHYSICAL
     FACTORS INCLUDE THE SENSITIVITY OF YOUR NERVOUS SYSTEM AND THE SEVERITY OF YOUR
     ARTHRITIS. EMOTIONAL AND SOCIAL FACTORS INCLUDE YOUR FEARS AND ANXIETIES ABOUT PAIN,
     PREVIOUS EXPERIENCES WITH PAIN, ENERGY LEVEL, ATTITUDE ABOUT YOUR CONDITION AND THE
     WAY PEOPLE AROUND YOU REACT TO PAIN.&NBSP;</FONT></P>
      <P><FONT FACE=ARIAL,HELVETICA, SANS-SERIF COLOR=#000000 SIZE=2>MANY          PEOPLE
     WITH ARTHRITIS HAVE FOUND THAT BY LEARNING AND PRACTICING          PAIN MANAGEMENT
```

Figure 4-4. A Partial sample of the HTML source

Figure 4-5 shows the result of cleansing our page. Notice how the cleansing phase has removed all unwanted formatting elements such as background color, text color, images, the table formatting elements and all the comments. In this particular example, it has removed all the information from the <META> tags as well. This is a default, but there is a provision where the <META> information can be considered with the <TITLE> information if explicitly specified. The cleansing phase has kept the tags that provide emphasis information untouched.

```
<HTML><HEAD><TITLE>MANAGING YOUR PAIN</TITLE>
<BODY >
<A
HREF="HTTP://WWW.ARTHRITIS.ORG/DEFAULT.ASP">
</A>
<B>WELCOME!<BR><A
HREF="HTTPS://WWW.ARTHRITIS.ORG/PROMO/SECURITY.ASP">
LOG IN</A><A
HREF="HTTPS://WWW.ARTHRITIS.ORG/PROMO/REGISTER.ASP">
CREATE PROFILE</A></B></FONT></FONT>
          …
          …
<LI><B>
DEPRESSION OR STRESS</B>, WHICH RESULTS FROM LIMITED MOVEMENT OR NO LONGER DOING
ACTIVITIES YOU ENJOY. YOU CAN GET CAUGHT IN A CYCLE OF PAIN, LIMITED/LOST ABILITIES, STRESS
AND DEPRESSION THAT MAKES MANAGING PAIN AND ARTHRITIS SEEM MORE
DIFFICULT.</LI></UL><P><B>DIFFERENT REACTIONS TO PAIN</B><BR>PEOPLE REACT DIFFERENTLY TO
PAIN FOR SEVERAL REASONS. PHYSICAL FACTORS INCLUDE THE SENSITIVITY OF YOUR NERVOUS
SYSTEM AND THE SEVERITY OF YOUR ARTHRITIS. EMOTIONAL AND SOCIAL FACTORS INCLUDE YOUR
FEARS AND ANXIETIES ABOUT PAIN, PREVIOUS EXPERIENCES WITH PAIN, ENERGY LEVEL,ATTITUDE
ABOUT YOUR CONDITION AND THE WAY PEOPLE AROUND YOU REACT TO PAIN.</P><P>
MANY PEOPLE WITH ARTHRITIS HAVE FOUND THAT BY LEARNING AND PRACTICING PAIN MANAGEMENT
SKILLS, THEY CAN REDUCE THEIR PAIN.</P> <P><B>PAIN FACTORS</B><BR><B>WHAT CAN MAKE YOUR
PAIN FEEL  WORSE?</B></P>…
…
```

Figure 4-5. Cleansing the Web page

Fig. 4-6 shows the same page after the stop words have been eliminated except for tokenization. The HTML formatting tags of importance are left behind along with just the stems of words present in the document. In our example, we observe that the text in the <TITLE> tags nicely brings out the purpose of the page; but this may not be true in a generic sense.

```
<HTML><HEAD><TITLE>MANAGE PAIN</TITLE>
<BODY >
<A
HREF="HTTP://WWW.ARTHRITIS.ORG/DEFAULT.ASP">
</A>
<B>WELCOME!<BR><A
HREF="HTTPS://WWW.ARTHRITIS.ORG/PROMO/SECURITY.ASP">
LOG IN</A><A
HREF="HTTPS://WWW.ARTHRITIS.ORG/PROMO/REGISTER.ASP">
CREATE PROFILE</A></B></FONT></FONT>
…
…
<LI><B>
DEPRESSION STRESS</B>  RESULTS  LIMITED MOVEMENT  NO LONGER DOING ACTIVITIES  ENJOY  GET
CAUGHT CYCLE PAIN LIMITED LOST ABILITIES STRESS DEPRESSION MAKES MANAGING PAIN ARTHRITIS
SEEM  MORE  DIFFICULT  </LI></UL><P><B>  DIFFERENT  REACTIONS  PAIN</B><BR>PEOPLE  REACT
DIFFERENTLY PAIN SEVERAL REASONS PHYSICAL FACTORS INCLUDE SENSITIVITY NERVOUS SYSTEM
SEVERITY ARTHRITIS EMOTIONAL SOCIAL FACTORS INCLUDE FEARS ANXIETIES  PAIN PREVIOUS
EXPERIENCES PAIN ENERGY LEVEL ATTITUDE CONDITION WAY PEOPLE AROUND REACT PAIN </P><P>
MANY PEOPLE ARTHRITIS FOUND LEARNING PRACTICING PAIN MANAGEMENT SKILLS REDUCE PAIN </P>
<P><B>PAIN FACTORS</B><BR><B> MAKE PAIN FEEL WORSE </B></P>…
…
```

Figure 4-6. Stop word Removal

Now that the Information Analyzer is left only with the raw text stems and elements, the tokenization can begin. Before tokenization, the Information Analyzer calculates the various word importance measures that we presented in chapter 3. While parsing, the Tags that embed words are monitored e.g. in Fig. 4-6 the text “Pain Factors” is bold as well falling into a separate paragraph.

Figure 4.7 shows an intermediate step before page siblings are created. It shows a list of all tokens that are extracted from the page. It stores a list of all tokens in the page being processed. Along with the actual tokens, the various attributes of the token are also stored. The token list separates the token through detected segments in the text. Segments are identified using information such as paragraph breaks and headers.

```
\T
MANAGE # <HTML> <TITLE>
PAIN # <HTML> <TITLE>
\S1
WELCOME # <B>
CREATE # <B>
PROFILE # <B>
\S2
DEPRESSION # <B>
STRESS # <B>
\S3
RESULTS # NULL
LIMITED # NULL
…
DIFFICULT # <U>
\S23
DIFFERENT # <B>
REACTIONS # <B>
PAIN # <B>
```

Figure 4-7. An Intermediate Step during Tokenization

The segmentation allows the Information Analyzer to calculate frequencies for each paragraph. The '\S' entries seen in Fig 4.7 show the logical segments that are identified and flagged. After the snapshot shown in Fig 4.7, the Information Analyzer calculates the various Word Importance measures for all the tokens in the document. After all the measures are calculated, we use the rules presented in section 3.10 to assign the Degrees of Relevance to all tokens in our file. The Information Analyzer then creates siblings for each page. Fig. 4-7 shows the sibling created for our example page. The first entry in the sibling file shows the page ID for which the sibling corresponds to. Each paragraph is represented distinctly in a sibling. This is done so that a single paragraph can be extracted from a given sibling. We also see that the \S separators are numbered. Since the paragraphs are sequentially numbered while parsing the original Web page, each paragraph separator number in a sibling directly maps to the same numbered paragraph in the original text.

```
!13

\T
MANAGE #1
PAIN #1
\S1
WELCOME #2
CREATE #2
PROFILE #3
\S2
DEPRESSION #2
STRESS #2
\S3
RESULTS #4
LIMITED #4
…
DIFFICULT #3
\S23
DIFFERENT #3
RACTIONS #2
PAIN #2
```

Figure 4-8. A Sibling page

In fig. 4-8, the sibling entries are followed by their corresponding relevance numbers. We maintain table of these Ids with the page titles to facilitate search. After each page has been converted into its corresponding sibling, what we have is a file repository that contains the information content of that Web site weighed according to relevance.

The Monitoring Engine is responsible for returning the results to the users. The results can be achieved by explicitly querying for a particular key word as in a search engine or by continuous monitoring of the pages viewed by the user. Getting results from an explicit query are straightforward. The system scans the siblings and returns finds the sections having the highest rating in terms of the word relevance parameters that we have specified.

The advantage of our system is the way we calculate these relevance parameters. Since they do not depend on only on statistical dependencies such as word frequencies or simply the link information between pages, the parameters are effective in zoning down on the necessary text content requested by the user.

The Monitoring Engine also keeps looking at the page currently being viewed by the user. When the user visits another page, the Monitoring Engine combines the information content of the previous sibling and appends to it the information from the current sibling (this happens only in the working memory of the Monitoring Engine, the actual siblings remain unchanged). Using the text fragments that are used being viewed by the user, the Monitoring Engine picks out the more significant tokens and makes a search on them thus returning a list of text chunks. We simulate a user browsing a website by using previously recorded web navigation data for the users. Time spent on a give web page is also an important parameter, but we have not used it in our analysis.

CHAPTER 5.

Experimental Results

In this chapter, we present the results obtained by running test with our system. We do a qualitative evaluation of the results that are generated by taking user opinions. We follow it up with a base line comparison to random extractions and then compare the system with the results obtained from a search engine.

5.1 Experimental Setup

For our experiments, we are assuming that only a single user is visiting the Web site. A sample Web site[9] was used on a locally installed Web server.

5.2 Preliminary Work

All the preprocessing was performed offline to create the Information Repository of that Web site. We asked users to browse the Website for a particular topic. The users were asked not just to locate the topic of interest but were also told to return some 'content' about that particular topic (A short paragraph of not more than two hundred words). The users were asked to start on the main page and were allowed to browse freely about site for the task. On the locally stored site, we had no support for facilitating search; in addition, we instructed users to avoid using the Web site search to find the given topic. To perform an evaluation of our system, we recorded the Web access logs of these users for the duration of their exploratory session. We set up our experiment in the following manner.

[9] http://www.arthritis.org [partially used]

We simulated a user on the Web site with the aim of obtaining information about a particular topic. Since we have the Web access logs of users, we ran a simulation to mimic a user browsing on the site. Then we then asked the system to extract the most "relevant paragraph" to the current working memory of information given the same query that was given to the original users. We then compared this extracted information with the results returned by the users.

5.3 Evaluation Measures

Given the information extracted by the system as well as the information obtained by running actual users with a specific goal, we need to lay down certain evaluation measures that could be used to compare these two sets of extracted text data.

The evaluation problem we faced was the comparison of two sets of text data to determine the degree of similarity between them. The results returned by our system call for a qualitative rather than a quantitative measure of comparison; thus, we rely more on user feedback rather than obtaining some computable number. For this reason we evaluated our results by taking user opinions on the results that were extracted by the system. Every user was asked to compare pairs of text segments (one extracted by the system and one by an actual user) and give a measure of similarity between them. The user was asked to classify the similarity of the two sets into one of four categories. These four categories were 'High Degree of Similarity', 'Moderate Degree of Similarity', 'Sparse Degree of Similarity' and finally 'No degree of Similarity'. Care was taken so that a user was asked to evaluate results for topics different from the one actually given to that particular user.

In order to validate the results with respect to other text segments, we also compared the results with other set of texts. We obtained these sets as follows. We took a randomly extracted paragraph with the search keyword in it and asked users to compare which of the two i.e. the randomly extracted paragraph or the paragraph returned by the system was closer to the results returned by the users. The idea behind picking up a random paragraph with the keyword in it was to give us a base line to ensure that our system performs better than just retrieving any random page. In another test, we used 'Google® Search' (site-specific) to search for the exact queries that we asked participants on our test

Web site[10]. We used it to compare with Google® as it gives the state of the art response to search queries.

Table 5-1. User evaluations on the results returned by the system.

TEST	SIMILARITY TO RESULTS RETURNED BY ACTUAL PARTICIPANTS			
	HIGH	**MODERATE**	**SPARSE**	**NONE**
1	3	9	1	0
2	11	2	0	0
3	0	0	10	3
4	1	12	0	0
5	10	2	0	0
6	12	1	0	0
7	0	0	8	5
8	0	8	4	1

5.4 Results

We ran eight different tests for finding out eight different query topics on the test Web site. We ran the test for eight users each being given a separate topic to explore. The results returned for a particular topic by a user as well as the system were evaluated by the remaining seven users as well as six other participants who did not participate in the main test. Here is a breakdown of the results of the evaluations given by the users comparing the text extracted by other users and the text extracted by the system. Table 5-1 shows how the users evaluated the results of each result. As we can see from the table that on the average, users found that the results obtained by the system had a moderate or better similarity to the results given by actual users who browsed the site. Fig. 5-1 shows the results graphically.

We also performed two other sets of comparisons. We asked six users (who did not participate in the browsing experiment) to compare the results generated by the system to results obtained by searching

[10] Google® Search can be used to search in a particular site by entering a search query as follows: "*search terms* site:(*site address*)"

for that particular topic on the Web site using Google® search. We must point out that while Google® search returns a link to the actual page from the Web site, our system is capable of returning results (text) that could possibly reside on physically distinct pages on the actual Web site.

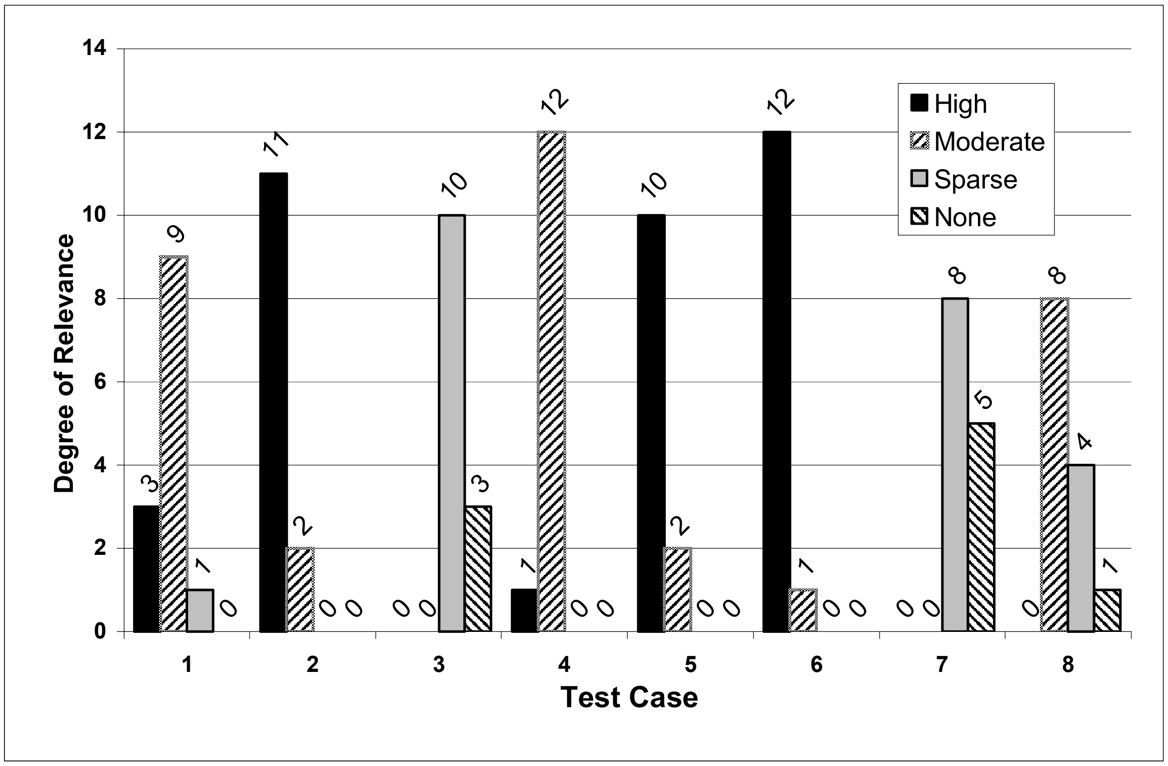


Figure 5-1. User Evaluations

5.5 A Test Case

In this section, we look at a sample test case that was given to a user. Specifically, it corresponds to Test no. 4 listed in Table 5-1. The user was asked to return a paragraph about the search query 'endorphins'. User 4 returned the following result. (The text returned is not a copy paste from the web site but an interpretation that the user entered)

> 'Endorphins are pain-relievers released naturally by the human body. The human body identifies pain in terms of nervous signals that travel from the source of pain to the brain. The body has its own defense for pain, the release of morphine like

> substances that serve to decrease the pain signals. The release of endorphins is controlled by emotions or by external factors. Emotions may trigger the release of endorphins so that pain is not noticed by the patient. On the physical front, external influences such as heating or icing an injured area may also release endorphins.'

When our system was asked to return results for the same query, the following text was extracted and returned[11].

> \S Pain signals travel through a system of nerves in your brain and spinal cord. At times, your body tries to stop these signals by creating chemicals that help block pain signals. These chemicals, called endorphins, are morphine-like painkilling substances that decrease the pain sensation. Different factors cause the body to produce endorphins. One example is your own thoughts and emotions.
>
> \S For example, a father who is driving his children is hurt in a car accident. He is so worried about his children that he doesn't feel the pain of his own broken arm. The concern for his children has caused the natural release of endorphins, which block the pain signal and prevent him from noticing the pain.
>
> \S The body also produces endorphins in response to external factors, such as medicine. Codeine is one example of a powerful pain-blocking medication. Other external pain control methods, such as heat and cold treatments, can stimulate the body to either release endorphins or block pain signals in other ways.'

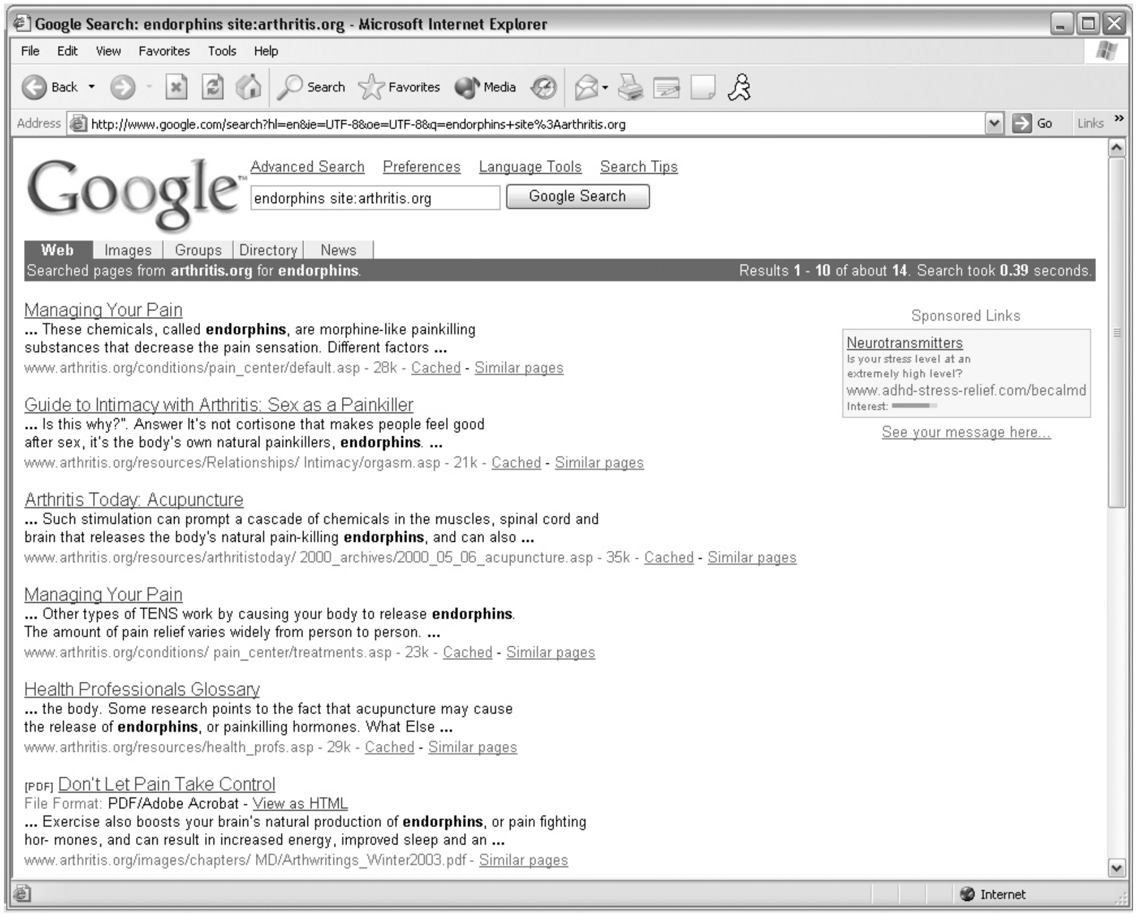
The results returned by the System are actual paragraphs from the website. Our system is capable of returning a paragraph even when the paragraph does not return a high degree of similarity but the paragraph heading associated with it does have a high degree of relevance. In this example out of the three paragraphs returned, only the first one had high relevance to the word 'endorphins', but succeeding segments had a moderate relevance, they were returned as well. When users were asked to

[11] The users were asked to ignore the Tags returned by the system e.g. \S. \P etc.

rate the similarity of these two texts, result was that a majority of the users felt that the two texts are moderately related. This may be attributed to the difference in the size of the two text segments.

5.5.1 Comparison to a Third Party Search

Fig. 5-2 shows the result of running the same search query using Google® search.


Google Search: endorphins site:arthritis.org - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Advanced Search Preferences Language Tools Search Tips
Google Search
Web Images Groups Directory News
Internet

http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=endorphins+site%3Aarthritis.org

endorphins site:arthritis.org

Searched pages from **arthritis.org** for **endorphins**. Results **1** - **10** of about **14**. Search took **0.39** seconds.

Managing Your Pain
... These chemicals, called **endorphins**, are morphine-like painkilling substances that decrease the pain sensation. Different factors **...**
www.arthritis.org/conditions/pain_center/default.asp - 28k - Cached - Similar pages

Guide to Intimacy with Arthritis: Sex as a Painkiller
... Is this why?". Answer It's not cortisone that makes people feel good after sex, it's the body's own natural painkillers, **endorphins**. **...**
www.arthritis.org/resources/Relationships/ Intimacy/orgasm.asp - 21k - Cached - Similar pages

Arthritis Today: Acupuncture
... Such stimulation can prompt a cascade of chemicals in the muscles, spinal cord and brain that releases the body's natural pain-killing **endorphins**, and can also **...**
www.arthritis.org/resources/arthritistoday/ 2000_archives/2000_05_06_acupuncture.asp - 35k - Cached - Similar pages

Managing Your Pain
... Other types of TENS work by causing your body to release **endorphins**. The amount of pain relief varies widely from person to person. **...**
www.arthritis.org/conditions/ pain_center/treatments.asp - 23k - Cached - Similar pages

Health Professionals Glossary
... the body. Some research points to the fact that acupuncture may cause the release of **endorphins**, or painkilling hormones. What Else **...**
www.arthritis.org/resources/health_profs.asp - 29k - Cached - Similar pages

[PDF] Don't Let Pain Take Control
File Format: PDF/Adobe Acrobat - View as HTML
... Exercise also boosts your brain's natural production of **endorphins**, or pain fighting hor- mones, and can result in increased energy, improved sleep and an **...**
www.arthritis.org/images/chapters/ MD/Arthwritings_Winter2003.pdf - Similar pages

Sponsored Links
Neurotransmitters
Is your stress level at an extremely high level?
www.adhd-stress-relief.com/becalmd
Interest:
See your message here...

Figure 5-2. Using Google® search to compare results

We searched our test Web site by querying it through Google®. Only the first two search results returned were offered to the user for evaluation. For this particular example, the result returned by the first search result was a link to the page from which all three of our paragraphs were extracted. . The second search result obtained by using this third party search pointed to a page containing the following paragraph,

> It’s not cortisone that makes people feel good after sex, it’s the body’s own natural painkillers, endorphins. At orgasm, says Seifer, the body experiences the same kind of endorphin surge that causes a “runner’s high,” and those endorphins could very well alleviate or at least distract you from your joint pain. The effect can last anywhere from 45 minutes to 3 hours

However, our system did not return this result because it did not assign a high degree of relevance to this particular segment. This may be attributed to the fact that no other paragraphs ‘near’ this one displayed even a sparse relevance to our search query.

For our next evaluation, we randomly chose a paragraph from the Web site that contained our search topic ‘endorphins’. We asked users to compare this randomly extracted paragraph with our result. For this test case, the randomly generated paragraph containing the word ‘endorphins’ was,

> In just a few months, you could be relaxing in your own spa and enjoying the many benefits of warm water therapy. Warm water therapy can help loosen tight muscles, which increases your range of motion. The warm water can also help increase your circulation, which relieves pressure on joints and muscles. And it helps stimulate the release of endorphins, the body's natural pain killers. Our resource center provides more information about the benefits of water exercise and specific exercises you can perform in water

As it turned out, this particular segment belonged to an advertisement on the Web site. Our system managed not to return its contents because it was associated with a very low degree of relevance.

In order to compare the system results to the results of randomly extracted results as well as the search results, we asked the user to tell whether the comparing results were less relevant, equally relevant or more relevant than the results generated by the system.

The results tabulated in table 5-2 show how the participants compared those obtained by explicit search and random extraction. It can be seen that the results obtained by the system were always better than randomly extracting a paragraph that contained the search topic. In most cases, the randomly extracted text segment was deemed less relevant to the user-extracted results in comparison to the system results.

The results obtained by using Google® search do not show any explicit trend. For some search queries, users found that the results returned by Google® were more relevant to the search topic than what the system returned. On the average, we can say that the results obtained by the system at par or a touch less relevant than the results obtained by using explicit Google® search. Since Google® return actual links to pages and not the actual text, we also asked users if they rather have results brought in on one page as a text or have explicit links generated like in Google®. Out of all the users questioned, about 72% felt that it was better that the system was able to collect text from physically distinct pages and combine them into a result, rather than returning links to all the pages where the search query could be found.

Table 5-2. Comparing System Results

TEST	**GOOGLE® SEARCH WITH KEYWORD**			**RANDOM EXTRACTION WITH KEYWORD**		
	LESS RELEVANT	**EQUALLY RELEVANT**	**MORE RELEVANT**	**LESS RELEVANT**	**EQUALLY RELEVANT**	**MORE RELEVANT**
1	0	2	4	6	0	0
2	1	5	0	6	0	0
3	0	1	5	5	1	0
4	2	4	0	5	1	0
5	6	0	0	6	0	0
6	1	5	0	4	2	0
7	0	0	6	3	3	0
8	0	4	2	2	3	1

CHAPTER 6

Conclusions and Future Work

We conclude our Thesis by presenting directions for further work that can be done to make our system more effective and we propose certain applications where this work could possibly be used.

6.1 Conclusion

We have presented a simple structure-based approach for relevant information extraction for a user while the user is foraging for information. The chief motivation for this work was the vast scope of extending this work to a variety of different areas.

Purely statistical Information extraction methods as well as structural oriented Information extraction methods have their own limitations e.g. structural methods may fail completely in certain Web pages that contain overly emphasized elements (advertisements etc) and statistical methods are already known to have failed in specific situations. We have struck a sort of equilibrium by using a combination of these techniques to efficiently retrieve information for a user.

Instead of a quantitative evaluation, we have corroborated the results generated by the system by users doing the same task by themselves. Thus, we have validated our results in the most user-centric way possible.

6.2 Application Areas

As we mentioned earlier, the motivation behind this work was the vast array of applications, which can be derived using this system as a base. The ultimate goal is to fabricate a personal recommender system for a user with specific information needs.

The system can be deployed in various application areas with little modifications. The system would primarily be used in information rich sites such as educational sites and NEWS sites. One constraint with news sites is the extremely volatile nature of information but our system does not have learning overhead.

6.3 Future Work

In this section, we look at certain enhancements that can be made to the current system to make it more effective in Information retrieval.

6.3.1 Additional Document Formats

The current system uses the inherent HTML formatting structure for most of its analysis. The Web however, has a collection of not just HTML documents but in many other forms too e.g. portable document format (.pdf), word documents (.doc), images, audio, video etc. Excluding the multimedia content, the system could be altered to cater to other document types. New heuristics would have to be developed to determine how the various formatting elements in those documents have to be weighed.

6.3.2 Dictionary Support

The addition of dictionary support to our system could make it very robust in terms of accuracy of results. Wordnet is a dictionary system with an associated lexical network. In Wordnet, unique concepts are represented as nodes, with directed edges to other words in the synonym set of the node. It even provides rich link structure with relations such as Hypernyms [12]and Meronyms[13] being offered. The addition of dictionary system will also be useful in reducing some errors occurring in our

[12] A Hypernym indicates a 'is a kind of' relation ship
[13] A Meronym indicates a 'is a part of' relation

system. In our work, care has to be taken as on occasions, the Porter Stemmer suffix stripper goes to work on a proper noun and may end up de-suffixing a proper noun. A dictionary look up before suffix stripping would allow the system to flag certain words, which in turn would be left alone by the Stemmer routine.

6.3.3 Natural Language Support

In addition to extracting information through statistical and structural methods, Natural language support could be added to obtain context of words with respect to their neighboring words; unlike the current system where the order of words has no importance and neither does contextual information for specific words. A good addition to the current system would be a Natural language engine to add additional word importance measures depending on discourse.

6.4 Enhancing other methods such as TFIDF

Our work of, assigning Importance measures to words in a document could be used in conjunction another technique known as the Text Frequency Inverse Document frequency (TFIDF). TFIDF is a standard weighing technique used in Information Retrieval.

We could use our word importance measures to create a weighted TFIDF measure that could possibly be more accurate than traditional TFIDF. A drawback of the traditional TFIDF method is that it does not distinguish between one occurrence of a certain term in a text and many. Since traditional TFIDF is concerned only with the fact that a term is contained in a given text or not (binary counting). TFIDF also believes that the importance of a term is inversely proportional to the number of texts that contain the term.

We could use our measures to obtain a weighted TFIDF that could correspond to a normalized term frequency over the entire collection, thus could be used to accurately identify important concepts in the documents.

APPENDIX A

HTML Tag Listing

A.1 A Brief Overview of HTML

HTML (Hyper Text Markup Language) is a non-propriety format that allows elementary tagging support for the structuring and layout of text and images, which are interpreted by a Web Browser to display the data in the way it is defined. The language allows one to specify a formatting structure to textual data in the form of headings, paragraphs, lists, hyperlinks etc.

A.2 HTML Formatting Elements

HTML 4.01 [12] is a language with a large set of elements. A valid HTML document consists of three parts viz. version information, a Header and a Body. The version information specifies which DTD (Document Type Definition) is used in the document; the head section contains metadata while the body contains the actual textual information. Here is a complete list of the Tags[14] supported by the HTML 4.01 specification. The tags marked are used by our system. The tags marked with the * symbol are currently check for in our system.

[14] http://www.w3schools.com/html/html_reference.asp

Table A-2. HTML Tag Reference

*	< !-- -->	*	<!DOCTYPE>	*	<A>		<ABBR>
	<ACRONYM>		<ADDRESS>	*	<AREA>	*	<B>
	<BASE>		<BDO>	*	<BIG>	*	<BLOCKQUOTE>
*	<BODY>	*	<BR>		<BUTTON>		<CAPTION>
	<CITE>		<CODE>		<COL>		<COLGROUP>
	<DD>		<DEL>		<DFN>		<DIV>
	<DL>		<DT>	*	<EM>		<FIELDSET>
	<FORM>		<FRAME>		<FRAMESET>	*	<H1> to <H6>
*	<HEAD>		<HR>	*	<HTML>	*	<I>
	<IFRAME>	*	<IMG>		<INPUT>		<INS>
	<KBD>		<LABEL>		<LEGEND>	*	<LI>
	<LINK>		<MAP>	*	<META>		<NOFRAMES>
	<NOSCRIPT>		<OBJECT>		<OL>		<OPTGROUP>
	<OPTION>	*	<P>		<PARAM>		<PRE>
	<Q>		<SAMP>		<SCRIPT>		<SELECT>
*	<SMALL>		<SPAN>	*	<STRONG>	*	<STYLE>
	<SUB>		<SUP>	*	<TABLE>	*	<TBODY>
*	<TD>	*	<TEXTAREA>	*	<TFOOT>	*	<TH>
*	<THEAD>	*	<TITLE>	*	<TR>		<TT>
	<UL>		<VAR>	*	<U>		

APPENDIX B

Stop Words

B.1 Stop Words

Stop words are words in text document that do not have any Information content of their own. Such stop words constitute about 20-30% of all words in an average text document. Removal of such stop words not only reduces the file sizes but it also eases the computational effort of the system. Examples of stop words are prepositions, punctuations, numbers, pronouns, articles and sometimes acronyms.

B.2 A List of Stop Words used by the system

This list of stop-words is not comprehensive, but only indicates the actual stop-words that were checked for in our analysis. More stop words can be added to the list depending on the nature of information being dealt with.

All pronouns	are	do
	as	does
All HTML tags that are listed in Appendix A	at	doing
	be	for
All punctuation marks in English language	been	from
	but	
	by	had
a		has
about	can	have
all	cannot	having
also	can’t	hence
am	could	however
an		
and	did	if

in	they	why
it	their	what
is	this	who
its	those	will
	these	within
not	to	won’t
	too	with
of		would
on	want	whenever
	wants	
so	was	
	what	
that	which	
the	where	

APPENDIX C
Porter Stemming Algorithm

C.1 Introduction

The Porter-Stemming algorithm is used for stripping suffixes on English language words. The Porter-Stemmer algorithm removes suffixes by automatic means, which is essential in the area of Information Retrieval. In a typical Information Retrieval environment, we have a large collection of text in the form of documents with each document described by special sections such as titles, headings, abstracts etc. Ignoring the origin of specific words, every document consists of a vector of terms i.e. a stem with a particular suffix. All the terms with a common stem usually have the same meaning e.g.

CONNECT, CONNECTED, CONNECTING, CONNECTION, CONNECTIONS

It has been found that the performance of any Information Retrieval system if the terms having the common stems are clubbed together. This may be achieved by removing various suffixes such as –ED, -ING, -ION, etc. for example in the above case stripping suffixes would leave us with the stem CONNECT. In addition to removing suffixes, it also reduces the total number of terms in an Information Retrieval system, thus reducing the total size and complexity and size of the system.

C.2 Definitions

Before presenting the Porter-Stemmer algorithm, we need to define some terms

Definition: A **consonant** in a word is a letter other than A, E, I, O or U and other than Y preceded by a consonant.

Definition: If a letter is NOT a consonant then it is said to be a **vowel**

A consonant will be denoted by ‘c’, a vowel by ‘v’. A list ccc… of length greater than zero will be denoted by C while a list vvv… of length greater than zero will be denoted by V. Any word, or a part of a word thus has to be one of the four forms,

CVCV … C
CVCV … V

VCVC … C
VCVC … V

These may all be represented by the single form:

[C] VCVC ... [V]
…where, the square brackets denote arbitrary presence of their contents.

Using (VC){m} to denote VC repeated m times, this may again be written as

[C](VC){m}[V].

Definition: m will be called the **measure** of any word or word part when represented in this form.

The case m = 0 covers the null word. Here are some examples:

m=0	TR, EE, TREE, Y, BY.
m=1	TROUBLE, OATS, TREES, IVY.
m=2	TROUBLES, PRIVATE, OATEN, ORRERY.

The *rules* for removing a suffix will be given in the form

(condition) S1 → S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of m, e.g.

(m > 1) EMENT ->

Here S1 is `EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2. The `condition' part may also contain the following:

*S - the stem ends with S (and similarly for the other letters).
v - the stem contains a vowel.
*d - the stem ends with a double consonant (e.g. -TT, -SS).
*o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

And the condition part may also contain expressions with AND, OR or NOT so that,

(m>1 and (*S or *T))
…tests for a stem with m>1 ending in S or T, while

(*d and not (*L or *S or *Z))

…tests for a stem ending with a double consonant other than L, S or Z.

Elaborate conditions like this are required only rarely.

In a set of rules written beneath each other, only one is obeyed, and this will be the one with the longest matching S1 for the given word. For example, with

SSES	→ SS
IES	→ I
SS	→ SS
S	→

(here the conditions are all null) CARESSES maps to CARESS since SSES is the longest match for S1. Equally, CARESS maps to CARESS (S1=`SS') and CARES to CARE (S1=`S').

In the rules below, examples of their application, successful or otherwise, are given on the right in lower case.

C.3 Algorithm

The algorithm consists of five steps. Below listed is each step in detail with the rules that it contains.

Step 1-A

SSES	→ SS	; caresses → caress
IES	→ I	; ponies → poni, ties → ti
SS	→ SS	; caress → caress
S	→	; cats → cat

Step 1-B

(m>0) EED	→ EE	; feed → feed, agreed → agree
(*v*) ED	→	; plastered → plaster, bled → bled
(*v*) ING	→	; motoring → motor, sing → sing

If the second or third of the rules in Step 1b is successful, the following is done:

AT → ATE	; conflat(ed) → conflate
BL → BLE	; troubl(ed) → trouble
IZ → IZE	; siz(ed) → size
(*d and not (*L or *S or *Z))→ single letter	; hopp(ing) → hop, tann(ed) → tan,
	; fall(ing) → fall, hiss(ing) → hiss
	; fizz(ed) → fizz,
(m=1 and *o) → E	; fail(ing) → fail, fil(ing) → file

The rule to map to a single letter causes the removal of one of the double letter pair. The -E is put back on -AT, -BL and -IZ, so that the suffixes -ATE, -BLE and -IZE can be recognized later. This E may be removed in step 4.

Step 1-C

(*v*) Y → I	; happy → happi, sky → sky

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

Step 2

(m>0) ATIONAL	→ ATE	; relational →relate
(m>0) TIONAL	→ TION	; conditional → condition
(m>0) ENCI	→ ENCE	; valenci → valence
(m>0) ANCI	→ ANCE	; hesitanci → hesitance
(m>0) IZER	→ IZE	; digitizer → digitize
(m>0) ABLI	→ ABLE	; conformabli → conformable
(m>0) ALLI	→ AL	; radically → radical
(m>0) ENTLI	→ ENT	; differentli → different
(m>0) ELI	→ E	; vileli → vile
(m>0) OUSLI	→ OUS	; analogousli → analogous
(m>0) IZATION	→ IZE	; vietnamization → vietnamize
(m>0) ATION	→ ATE	; predication → predicate
(m>0) ATOR	→ ATE	; operator → operate
(m>0) ALISM	→ AL	; feudalism → feudal
(m>0) IVENESS	→ IVE	; decisiveness → decisive
(m>0) FULNESS	→ FUL	; hopefulness → hopeful
(m>0) OUSNESS	→ OUS	; callousness → callous
(m>0) ALITI	→ AL	; formaliti → formal
(m>0) IVITI	→ IVE	; sensitiviti → sensitive
(m>0) BILITI	→ BLE	; sensibiliti → sensible

The test for the string S1 can be made fast by doing a program switch on the penultimate letter of the word being tested. This gives an even breakdown of the possible values of the string S1. It will be seen in fact that the S1-strings in step 2 are presented here in the alphabetical order of their penultimate letter. Similar techniques may be applied in the other steps.

Step 3

(m>0) ICATE	→ C	; triplicate → triplic
(m>0) ATIVE	→	; formative → form
(m>0) ALIZE	→ AL	; formalize → formal
(m>0) ICITI	→ IC	; electriciti → electric
(m>0) ICAL	→ IC	; electrical → electric
(m>0) FUL	→	; hopeful → hope
(m>0) NESS	→	; goodness → good

Step 4

(m>1) AL	→	; revival → reviv
(m>1) ANCE	→	; allowance → allow
(m>1) ENCE	→	; inference → infer
(m>1) ER	→	; airliner → airlin
(m>1) IC	→	; gyroscopic → gyroscop

(m>1) ABLE →	; adjustable → adjust
(m>1) IBLE →	; defensible → defens
(m>1) ANT →	; irritant → irrit
(m>1) EMENT →	; replacement → replac
(m>1) MENT →	; adjustment → adjust
(m>1) ENT →	; dependent → depend
(m>1 and (*S or *T)) ION →	; adoption → adopt
(m>1) OU →	; homologou → homolog
(m>1) ISM →	; communism → commun
(m>1) ATE →	; activate → activ
(m>1) ITI →	; angulariti → angular
(m>1) OUS →	; homologous → homolog
(m>1) IVE →	; effective → effect
(m>1) IZE →	; bowdlerize → bowdler

The suffixes are now removed. All that remains is a little tidying up.

Step 5-A

(m>1) E →	; probate → probat, rate → rate
(m=1 and not *o) E →	; cease → ceas

Step 5-B

(m > 1 and *d and *L) → single letter	; controll → control, roll → roll

The algorithm is careful not to remove a suffix when the stem is too short, the length of the stem being given by its measure, m. There is no linguistic basis for this approach. It was merely observed that m could be used quite effectively to help decide whether it was wise to take off a suffix. For example, in the following two lists:

list A	list B
RELATE	DERIVATE
PROBATE	ACTIVATE
CONFLATE	DEMONSTRATE
PIRATE	NECESSITATE
PRELATE	RENOVATE

-ATE is removed from the list B words, but not from the list A words. This means that the pairs DERIVATE/DERIVE, ACTIVATE/ACTIVE, DEMONSTRATE/DEMONS- TRABLE, NECESSITATE/NECESSITOUS, will conflate together. The fact that no attempt is made to identify prefixes can make the results look rather inconsistent. Thus, PRELATE does not lose the -ATE, but ARCHPRELATE becomes ARCHPREL. In practice, this does not matter too much, because the presence of the prefix decreases the probability of an erroneous conflation.

Complex suffixes are removed bit by bit in the different steps. Thus GENERALIZATIONS is stripped to GENERALIZATION (Step 1), then to GENERALIZE (Step 2), then to GENERAL (Step

3), and then to GENER (Step 4). OSCILLATORS is stripped to OSCILLATOR (Step 1), then to OSCILLATE (Step 2), then to OSCILL (Step 4), and then to OSCIL (Step 5).

Bibliography

1. Baldi P., and Smyth P. (2003). *Modeling the Internet and the Web, Probabilistic Methods and Algorithms*. West Sussex, England: John Wiley and Sons Ltd.
2. Del Bimbo A. (1999). *Visual Information Retrieval.* San Francisco, CA: Morgan Kaufmann.
3. Catledge L. D. and Pitkow J. (1995). "Characterizing Browsing Strategies on the in the World Wide Web," *Computer Networks IDSN Systems.* 27, 1065-1073.
4. Cooley R., Mobasher B. and Srivastava J. (1999). "Data Preparation for Mining the World Wide Web Browsing Patterns," *Knowledge Information Systems*, 1, 5-32
5. Fox C. (1992) "Lexical Analysis and Stoplists," In, *Information Retrieval: Data Structures and Algorithms.* Frakes W. B., Baeza-Yates R., (Eds.) ch 7. Englewood Cliffs, NJ: Prentice Hall.
6. Greenberg S. (1993). *The Computer User as a Toolsmith: The Use, Reuse and Organization of Computer-based tools*, Cambridge, MA: University Press.
7. Gudivada V. N., Raghavan V. V., Groksy W. I. and Kasanagottu R. (1997) "Information Retrieval on the World Wide Web". *IEEE Internet Computing*, 58-68.
8. Han J. and Kamber M. (2001). *Data Mining: Concepts and Techniques*, San Diego, CA: Morgan Kaufmann.
9. Mena J. (1999). *Data Mining your Website,*Boston, MA: Digital Press
10. Merk D. (1998) *Text Data Mining. A Handbook of Natural Language Processing Techniques and Applications for the processing of Languages as Text*. Dale R., Moisl H. y Somers H. (Eds.) New York, NY: Marcel Dekker.
11. Mladenic D. (1999) "Text Learning and related Intelligent Agents," *IEEE Expert Special issue on Applications of intelligent information retrieval*

12. Porter M. (1980) "An Algorithm for Suffix Stripping," *Program* 14, 130-137
13. Raggett D. and Hors A. L. and Jacobs I. (Eds.) (1999) HTML 4.01 Specification. *W3 Consortium Recommendation*. Also available online at http://www.w3.org/TR/html/4.)
14. Shahabi C., Banaei-Kashani F. and Faruque J. (2001) "A framework for efficient and anonymous Web usage mining based on client side tracking," *Proceedings of WEBKDD 2001. Lecture Notes in Artificial Intelligence*, vol. 2356, 113-144.
15. Tauscher L. and Greenberg S. (1997). "Revisitation Patterns in World Wide Web navigation," *Proceedings of the Conference on Human Factors in Computing Systems CHI'97*, 97-113. New York, NY: ACM Press.
16. Baeza-Yates R. and Riberio-Neto B. (1999). *Modern Information Retrieval*. ACM Press Books, Addison-Wesley.
17. http://www.google.com
18. http://www.w3.org/TR/2003/WD-xhtml2-20030506