Acerca de agentes que personalizan el trabajo con Internet, clustering de documentos y el uso de la estructura de HTML en medidas de similitud para documentos

Autor
Joaquín Rapela
Lu: 652/91
rj2a@dc.uba.ar

Director
Dr. Pablo M. Jacovkis
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Co-director
Dr. Paul Maglio
IBM Almaden
Research Center

Abstract

Utilizando Web Broswer Intelligence (WBI), una arquitectura de agentes que trabajan en pos del usuario para personalizar su trabajo con la Web, desarrollo "La Gran Memoria", una aplicación que ayuda a reutilizar información en grandes organizaciones. La Gran Memoria construye grupos (o clusters) de documentos HTML; estos documentos poseen una estructura particular que puede ser aprovechada por el proceso de clustering. Luego de realizar una recopilación de estrategias de clustering, cuyos resultados muestro en el presente trabajo, compruebo que no existen estrategias especificas para documentos HTML. Los algoritmos de clustering utilizan medidas de similitud para documentos. Diseño una mecanismo para incorporar información referente a la estructura de los documentos HTML a medidas de similitud para este tipo de documentos. A partir de una medida de similitud este mecanismo genera una versión extendida de esta medida de similitud. Comparo la calidad de una medida de similitud estándar con la de su par extendida mostrando que la utilización de la estructura HTML puede ayudar a calcular la similitud entre documentos HTML. De este modo un algoritmo de clustering que utilice una medida de similitud enriquecida por este mecanismo estará optimizado para documentos HTML.

1 Introducción

Este trabajo comenzó en el centro de investigación IBM Almaden Research Center, San Jose, California. Uno de los motivos fundamentales para trabajar en dicho centro era conocer el funcionamiento de un centro de investigación en los Estados Unidos. Con este objetivo busqué un proyecto de investigación en el que pudiera participar y, luego de haber examinado algunos proyectos, comencé a trabajar en Web Browser Intelligence (WBI de aquí en mas), proyecto a cargo de Paul Maglio y Robert Barrett. En la segunda sección de este trabajo presento una breve descripción de WBI. [Barrett et al., 1997] describe detalladamente las características de WBI.

Utilizando WBI desarrollo "La Gran Memoria" una aplicación que ayuda a reutilizar información en grandes organizaciones. La Gran Memoria almacena documentos HTML recorridos por usuarios de la organización. Luego agrupa estos documentos en clusters. Finalmente, cuando el usuario esta browseando la Web, La Gran Memoria agrega una notificación al comienzo de la pagina HTML que cuando es clickeada presenta una lista de clusters relacionados al documento corriente.

El siguiente escenario muestra como La Gran Memoria ayuda a reutilizar información en grandes organizaciones:

La Gran Memoria almacena los documentos HTML

1998

browseados por los usuarios de La Gran Organización S.A. Juan Pérez, un miembro de La Gran Organización S.A., esta buscando información acerca de la transformada de Fourier. En cuanto llega al primer documento que trata sobre dicho tema pide a La Gran Memoria clusters relacionados al documento corriente. La Gran Memoria devolverá el cluster construido automáticamente a partir de los documentos recorridos por María López mientras buscaba información sobre dicho tema la semana anterior.

Luego de realizar una recopilación de estrategias de clustering, cuyos resultados presento en la sección 4, compruebo que no existen estrategias específicas para documentos HTML. Estas estrategias son importantes pues los documentos HTML poseen una estructura particular que puede ser aprovechada por el proceso de clustering. Por ejemplo, los documentos HTML poseen un campo TITLE y si dos documentos comparten una palabra en este campo probablemente sean mas similares que si la comparten en el campo BODY. Los documentos HTML poseen referencias de hipertexto (o links) y si un documento apunta a otro posiblemente sean similares.. Estos son algunos ejemplos de características de los documentos HTML que pueden ser aprovechadas para determinar su similitud.

Desarrollo un mecanismo para incorporar información acerca de la estructura de HTML a medidas de similitud para documentos HTML. A partir de una medida de similitud dada este mecanismo construye una versión extendida de la misma. Usando evaluaciones del tipo precisión vs recuerdo comparo la calidad de una medida de similitud estándar con la de su par extendido mostrando que el uso de la estructura HTML puede ayudar a determinar la similitud entre documentos HTML.

En la sección 2 introduzco WBI. En la sección 3 describo La Gran Memoria. Luego, en la sección 4, presento los resultados de la recopilación sobre estrategias de clustering. El algoritmo de clustering implementado para La Gran Memoria es presentado en la sección 5. En la sección 6 describo la medida de similitud compuesta por distintos criterios HTML, el algoritmo utilizado para encontrar la combinación óptima de estos criterios y la evaluación implementada para evaluar la calidad de la medida de similitud. En esta sección también muestro los resultados de la evaluación y presento algunas conclusiones sobre la calidad de la medida de similitud. Finalmente en la sección 7 indico algunas direcciones interesantes de trabajo futuro.

UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

1998

THESIS PROJECT:

About Agents that Personalize the Work with Internet, Document Clustering and the Use of the Structure of HTML Documents In Document Similarity Measures

AUTHOR:

Joaquín Rapela
Lu: 652/91
rj2a@dc.uba.ar

ADVISOR:

Dr. Pablo M. Jacovkis
Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

CO-ADVISOR:

Dr. Paul Maglio
IBM Almaden Research Center

To my parents for growing me up

Abstract

Using Web Browser Intelligence (WBI), an agent architecture that works on behalf of the user to personalize the work with the World Wide Web, I develop "The Big Memory", an application that helps to reuse information in a big organization environment. The Big Memory builds groups (or clusters) of HTML documents; these documents are different from standard digital documents in that they are structured documents. After making a survey on available document clustering methods, whose results are described in the present work, I realized that there were not specific strategies for clustering HTML documents. Clustering algorithms are based on a document similarity measure. I designed a mechanism for incorporating information about the structure of HTML documents to a given similarity measure. This mechanism transforms a similarity measure into an enriched similarity measure. I compare the performance of a standard similarity measure to the performance of its enriched counterpart showing that using the structure of HTML documents can help to find their similarity. A clustering algorithm using the enriched similarity measure will be optimized for HTML documents.

1 Introduction

This work began while I was working at the IBM Almaden Research Center, San Jose, California. One of the main reasons for working there was to get in touch with the research community of that center and a good way of doing so was to work with researchers in a common project. So after trying different projects I began working in Paul Maglio's and Robert Barrett's Web Browser Intelligence (WBI here on). In the next section I will present a brief overview of WBI. A detailed description can be found in [Barrett et al., 1997].

Using WBI I develop "The Big Memory", an application that helps to reuse information in a big organization environment. The Big Memory stores HTML documents as they are browsed by web users in the organization. Afterwards it automatically builds clusters out of these documents. Now when the user is browsing the Web, The Big Memory adds a notification at the top of each page and when this notification is clicked it presents a list of clusters related to the current page.

The following scenario shows how The Big Memory can help to reuse information in a big organization environment:

> *The Big Memory stores HTML documents browsed by members of The Big Company Inc. Now John Higgins, a member of The Big Company Inc., is looking for information about the Fast Fourier Transform (FFT) algorithm. After reaching the first document related to this topic he asks The Big Memory for clusters related to the current document. The Big Memory will answer a cluster automatically built out of documents browsed by JoAnn Smith while she was looking for information on the FFT algorithm last week.*

After making a survey on the available document clustering strategies, which I show in section 4, I realized that there were not specific clustering strategies for HTML documents. These strategies are important because HTML is a structured language and this structure can help the clustering process. For example HTML documents have a title field and if two documents share words in this field, they are probably going to be more similar than if they share them in the body field. Besides HTML documents contain links and if one document points to another then they are probably going to be quite similar. Moreover, HTML documents belong to specific servers and if two documents belong to the same server, then they are probably going to be quite similar. These are some examples showing how the structure of HTML documents can be used for finding their similarity.

I designed a mechanism for adding information about the structure of HTML documents to a given similarity measure. This mechanism transforms a similarity measure to an enriched similarity measure. I use a recall-precision evaluation to compare the performance of a standard similarity measure to the performance of its enriched counterpart showing that considering the structure of HTML documents can help to find their similarity.

In the next section I introduce WBI. I describe The Big Memory in section 3. Next, in section 4, I show the result of a survey on document clustering strategies. I describe the clustering algorithm used by The Big Memory in section 5. In section 6 I describe the mechanism for adding information about the structure of HTML documents to a given similarity measure, the experiments for comparing the performance of a standard similarity measure with the performance of its enriched counterpart and the conclusions drawn from these experiments. Finally in section 7 I suggest some directions for future research.

2 Web Browser Intelligence

The following description of Web Browser Intelligence is based on [Barrett et al., 1997].

Agents can personalize otherwise impersonal computational systems. The World Wide Web presents the same appearance to every user regardless of that user's past activity. Web Browser Intelligence (WBI) is an implemented system that organizes agents on a user's workstation to observe user actions, proactively offer assistance, modify web documents, and perform new functions. WBI can annotate hyperlinks with network speed information; record pages viewed for later access, and provide shortcut links for common paths. In this way, WBI personalizes a user's Web experience by joining personal information with global information to effectively tailor what the user sees.

Every user sees the same Web connected together in the same way. Web authors build the structure of the Web according to their interests, concerns, and tastes, and then all users rely on that same structure.

To overcome the impersonal feel of the Web, browser software generally provides "Hot Lists" or "Bookmarks" which allow users to record often-used URLs for quick access. These lists of URLs begin to take the Web more personal, pulling certain web pages closer to individual users, and thus enabling easier access. They add a personal structure to the impersonal structure of the Web. Of course, some users go further and author their own home pages that can also provide launching points for Web access. In addition, browsers can generally be configured to automatically start up with a particular page loaded. Creating hot lists, home pages, and setting start pages are steps individual users can take toward personalizing the Web.

Web Browser Intelligence (WBI) automatically personalizes the Web using agent technology. WBI provides an architecture that taps into the communication stream between a user's web browser and the Web itself. Small programs, or agents, attach themselves to this stream, observe the data flowing along the stream, and alter the data as it flows past. These agents can learn about the user, influence what the user sees by marking-up pages before passing them on, and provide entirely new functions to the user through the web browser.

In the next section I introduce WBI architecture.

2.1 WBI Architecture

The fundamental communication mechanism of the Web is the hypertext transfer protocol (HTTP), although others such as gopher and file transfer protocol (FTP) are also used and can be treated analogously. HTTP is a simple, stateless, request-response system [Berners-Lee et al., 1996]. A browser connects to a server, sends a retrieval request, the server sends the requested document and then closes the connection.

HTTP also allows a proxy to mediate this transaction: the browser sends the request to the proxy, which retrieves the document from the appropriate server and then returns the document to the browser. The proxy mechanism is often used to provide a one-way firewall for intranet security. WBI is a proxy that intercepts the HTTP stream for observation and alteration. Every web transaction flows through WBI as a request goes from the browser to the Web and the response returns back to the browser (see Figure 1).

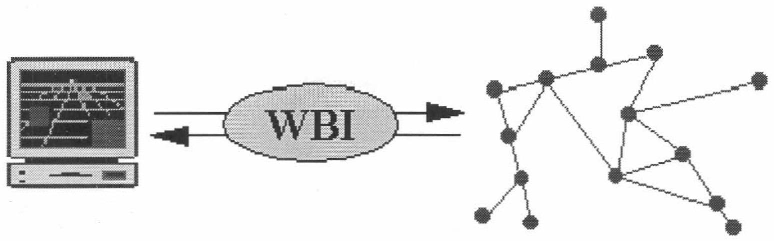


Figure 1. WBI stands between a user's web browser and the Web, monitoring, editing, and generating documents.

2.1.1 Four Kinds of Agents

WBI is composed of three types of agents which interact with this request-response stream (monitor agents, editor agents, generator agents), and one type of agent which acts independently (autonomous agents). A monitor receives a copy of the entire request-response transaction but cannot alter it. Monitors track user actions to provide information for other agents. For example, we have a page content monitor that records all of the text contained in the web pages that a user has viewed. The resulting content history can then be used to access previously viewed pages via keyword searching. Any number of monitors may observe any particular transaction.

An editor agent intercepts the communication stream, receiving information and then delivering a modified version of it. This edited data stream can be created from the incoming stream and other available information, such as a user's past history, system status, or any information obtained from the Web. Editors can connect to either the request part of the stream or to the response part. Request editors can transform a document request from the browser into another request. One simple application of a request editor is to fetch documents that WBI knows have been moved to a different URL. Response editors can modify the actual content of documents that users see. For example, response editors are used to add extra buttons or additional links to a web page, or to change colors and backgrounds. Alternatively, editors can choose to simply pass their input to their output, effecting no change in the stream. Any number of editor agents may alter a transaction.

A generator is an agent that converts a request into a response. Every transaction activates exactly one generator. The default generator is simply a Web communication program that passes the request on to the appropriate web server, retrieves the response and passes it back to the browser (i.e., simply performing the job of an HTTP proxy). Other generators are used to provide new WBI functions, intercepting requests from the browser and generating documents for the user to see in response. For example, a generator could provide a web-based way to view the state of the printer queues on a user's workstation. When the generator is activated, it produces an HTML document which describes the printer queues. Likewise, a generator can handle requests with special communication protocols, such as communicating with a firewall.

Finally, an autonomous agent is executed based on a trigger independent of the communication stream, such as a time interval. An autonomous agent simply performs its task and then terminates. For example, an autonomous agent might perform housekeeping actions, digest recorded transactions to develop user models, or explore the Web for new information.

2.1.2 How Agents are Triggered

WBI's central loop dynamically constructs a network composed of monitors, editors, and a generator to handle each request and response. At start up, each agent registers itself with the arbitrator along with its trigger rules for activation. Agents can be triggered when particular servers or domains are accessed, when particular document types are received, or at particular times or intervals.

Agents also register ordering and grouping information so that:

1. Editors can control the order in which they receive the communication stream (to work cooperatively in modifying the stream);
2. Monitors can control whether they see the response as it comes from the generator, as the user sees it, or after a particular editor has modified it;
3. Generators register the types of requests that they handle. Often generators register new URLs that do not exist on the Web but that can be addressed as normal web documents. For example, a generator that displays a workstation's printer queues might trigger on a request to http://wbi/printq.

Thus, every request is specially handled by the arbitrator, which routes it through the appropriate sequence of agents.

3 The Big Memory

The Big Memory is a WBI based application that helps to reuse information in a big organization environment. The Big Memory stores HTML documents as they are being browsed by web users in the organization. Afterwards it automatically builds clusters out of these documents. Now when the user is browsing the Web, The Big Memory adds a notification at the top of each page and when this notification is clicked it presents a list of related clusters to the current page.

The Big Memory builds clusters offline; new documents browsed by users of the big organization are not added immediately to the clusters. While users browse new documents, The Big Memory stores the new document in a special collection and at night it clusters the old and new documents together, leading to updated clusters that will be used in the next day.

The Big Memory architecture is made out of the following agents:

A monitor agent that saves each HTML page browsed by users in the big organization building a collection of HTML documents;

1. An autonomous agent that builds clusters from the previously mentioned collection of HTML documents (using the clustering algorithm described in section 5);
2. An editor agent that adds a notification of the presence of The Big Memory to every web page. When this notification is clicked a generator agent presents a ranked list of clusters related to the web page;
3. A generator agent that when activated searches the clusters collection for similar clusters to the current web page and shows the resulting clusters in a ranked list.

Figure 2 illustrates The Big Memory architecture:

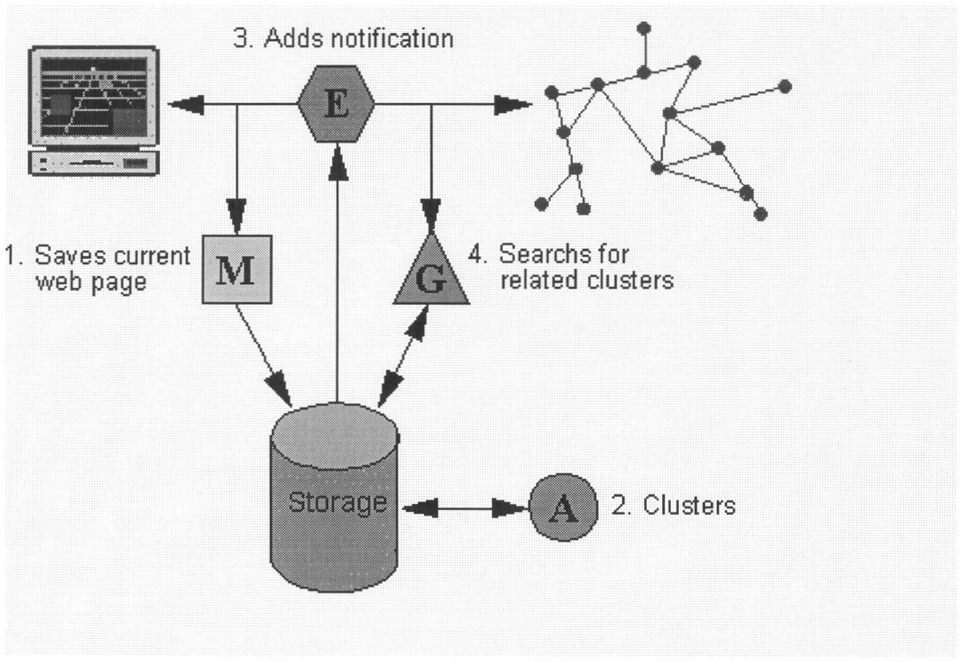


Figure 2: The Big Memory architecture

When the user reaches a web page the editor agent ads a notification of The Big Memory at the top of the page.

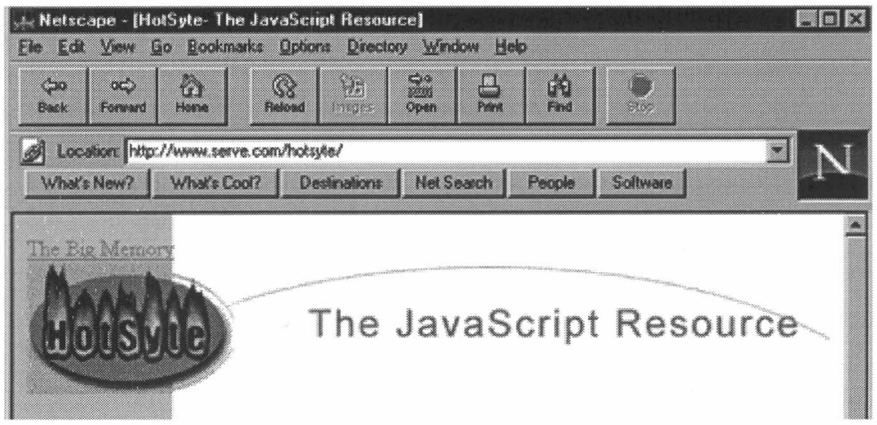
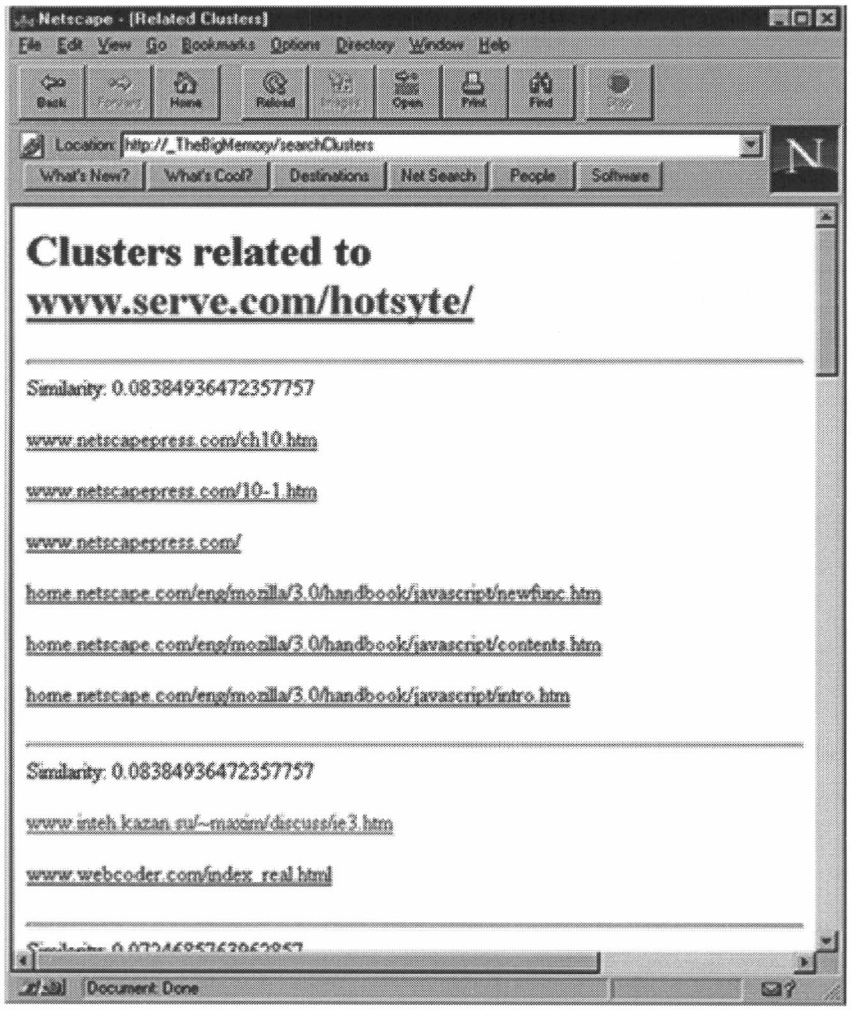

Netscape - [HotSyte- The JavaScript Resource]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://www.serve.com/hotsyte/
What's New? What's Cool? Destinations Net Search People Software
The Big Memory
HotSyte
The JavaScript Resource

Figure 3: The editor agent adding an annotation to http://www.serve.com/hotsyte

When the user clicks on The Big Memory link the generator agent is activated and shows a ranked list of similar clusters.


Netscape - [Related Clusters]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: http://_TheBigMemory/searchClusters
What's New? What's Cool? Destinations Net Search People Software
Document: Done

Clusters related to www.serve.com/hotsyte/

Similarity: 0.08384936472357757

www.netscapepress.com/ch10.htm

www.netscapepress.com/10-1.htm

www.netscapepress.com/

home.netscape.com/eng/mozilla/3.0/handbook/javascript/newfunc.htm

home.netscape.com/eng/mozilla/3.0/handbook/javascript/contents.htm

home.netscape.com/eng/mozilla/3.0/handbook/javascript/intro.htm

Similarity: 0.08384936472357757

www.inteh.kazan.su/~maxim/discuss/ie3.htm

www.webcoder.com/index_real.html

Similarity: 0.0724685763062857

Figure 4: The ranked list of clusters similar to http://www.serve.com/hotsyte

4 Survey on Document Clustering

At this point it is useful to distinguish between two major activities of the Information Retrieval field of study:

1. **Document Clustering**, which finds groups or clusters of related documents in a flat collection of documents.
   For example, given a collection of documents from a university and using document clustering we would discover clusters of documents about biology, mathematics, history, psychology, etc.
2. **Document Classification**, which given a query document and a collection of clusters, finds those clusters more similar to the query document.
   For example, given the previous collection of clusters from the university, a document about Linear Algebra and using document classification we would find that the cluster about Mathematics best matches the document about Linear Algebra.

The current survey focuses on the former topic. Document classification is an active field of research and for an interesting approach refer to [Koller and Sahami, 1997].

4.1 Why clustering?

The use of computers in numerous applications is generating data at a rate that far outstrips our ability to process and analyze it. For example, NASA satellites are expected to generate hundreds of terabytes of data per day (1 terabyte = 10^{12} bytes). Sets of financial data ranging from credit-card transactions to shipping records contain terabytes, and textual databases are growing rapidly. A great deal of effort is currently being expended to develop new hardware and software to generate, transmit, and store such data, but relatively little emphasis has been placed on developing new ways to use computers to analyze the data after it is acquired.

In response to this huge increase in the amount of electronic information techniques such as clustering have emerged. Clustering is a concept extraction technique that builds groups or clusters of conceptually related items. The underlying premise of concept extraction is that in order to interpret data, humans naturally and quickly extract and identify significant concepts. A person can contemplate a landscape, receive data through the sense organs, and can then make a reasonable judgement about whether it will rain, for example, or whether it will snow. A person can scan a magazine's table of contents and easily select the articles that relate to a subject of interest. Humans are constantly assimilating, categorizing, synthesizing, and analyzing data most often without any conscious realization of doing so.

The ability to extract concepts from sensory data is the product of millions of years of evolution and years of individual learning and experience. But humans go beyond simply recognizing and naming objects or events; we make judgements based on an overall context or quite subtle correlations among diverse elements of the available data. The great complexity, subjectivity, and ambiguity of human concepts make them extremely difficult if not impossible to define in a quantitative manner appropriate for use by computers.

However, computers accurately perform certain tasks that demand of humans too much time or concentration. For example, a librarian may wish to organize into groups the whole collection of electronic publications from a university. No matter how skilled the librarian may be, the task of building groups of electronic publications is time-consuming and tedious. In contrast computers once appropriately programmed are ideally suited to that task.

Clustering has attracted much attention in information retrieval and library science [Salton and McGill, 1983] [van Rijsbergen, 1979] as well as in pattern recognition [Duda and Hart, 1973]. In the information retrieval and library science field clustering builds groups of conceptually related documents out of plain collections of documents. In the pattern recognition field clustering helps to discover patterns in a data source; for example clustering might help to discover rural zones in a satellite image. Although the emphasis in pattern recognition is not on document clustering, it uses some methods and ideas that are applicable to the document-clustering environment.

4.2 Introduction to Document Clustering and the Vector Space Model

Document clustering has been extensively used as a methodology for improving the quality of document search and retrieval. The benefits of document clustering are based on the cluster hypothesis [Hearst and Peterson, 1996][van Rijsbergen, 1979] which states that mutually similar documents will tend to be relevant to the same queries.

When we submit a query to a plain collection of documents we only get those documents that best match the query. Besides when we submit a query to a clustered collection of documents we obtain those clusters that best match the query. These clusters will contain several documents exactly matching the query and some other documents that, although do not match the query exactly, are similar enough to the previous documents and will probably be relevant to the user. In this way document clustering has the effect of broadening the search request.

The first step in classical document clustering algorithms is to represents documents according to the Vector Space Model [Salton and McGill, 1983]. Vector-space models rely on the premise that the meaning of a document can be derived from the document's constituent terms. They represent documents as vectors of terms d=(t1, t2, …, tn) where ti ($1 \leq i \leq n$) is a non-negative value denoting the single or multiple occurrences of term i in document d. Thus, each unique term in the document collection corresponds to a dimension in the vector space.

In vector-space models each term can be individually weighted, allowing that term to become more or less important within a document or the entire document collection as a whole. Several weighting functions have been proposed [Faloutsos and Oard, 1994]:

- $FREQ_{ik}$: the occurrence frequency of term k in document i. It is easy to obtain and more effective than the binary weight;
- "Term specificity": $\log N - \log(DOCFREQ_k) + 1$ where $DOCFREQ_k$ is the number of documents that contain the term k and N is the total number of documents. It is relatively easy to obtain and it is more effective than using 0 or 1 to denote the presence or absence of the term k;
- Inverse Document Frequency: $FREQ_{ik}/DOCFREQ_k$. Similar to the previous weights, but seems to be more effective [Salton and McGill, 1983, p.105].

Also, by applying different similarity measures to compare queries to terms and documents, properties of the document collection can be emphasized or de-emphasized. For example, the dot product (or inner product) similarity measure finds the Euclidean distance between the query and a term or document in the space. The cosine similarity measure, on the other hand, by computing the angle between the query and a term or document rather than the distance, de-emphasizes the lengths of the vectors. In some cases, the directions of the vectors are a more reliable indication of the semantic similarities of the objects than the distance between the objects in the term-document space.

Similarly, a query is represented as a vector q=(t1, t2, …, tm) where term ti ($1 \leq i \leq m$) is a non-negative value denoting the number of occurrences of a term in the query (or, merely a 1 representing occurrence). Both the document vectors and the query vector provide the locations of the objects in the term-document space. By computing the distance between the query and other objects in the space, objects with similar semantic content to the query presumably will be retrieved.

Vector-space models, by placing terms, documents, and queries in a term-document space and computing similarities between the queries and the terms or documents, allow the results of a query to be ranked according to the similarity measure used. Unlike lexical matching techniques that provide no ranking or a very crude ranking scheme (for example, ranking one document before another document because it contains more occurrences of the search terms), the vector-space models, by basing their rankings on the Euclidean distance or the angle measure between the query and terms or documents in the space, are able to automatically guide the user to documents that might be more conceptually similar and of greater use than other documents.

An important contribution of the vector space model is that it transforms the Information Retrieval problem of finding the similarity between two documents or between a document and a query to the mathematical problem of finding the distance between points in a vector space.

Also vector-space models often provide an elegant method of implementing relevance feedback. In a retrieval system supporting relevance feedback the user, after receiving the results of a query, can point any document that seems relevant to the query. The retrieval system will enlarge the query adding important terms in the marked document and issue the query again. The resulting documents from the reformulated query will still be relevant to the original query but more similar to the pointed document.

In the next section I describe classical approaches used in document clustering and in section 4.4 I describe some recent trends.

4.3 Classical Cluster Generation Methods

The goal of a cluster generation method is to partition a set of documents into groups.

The partitioning procedure should ideally meet two goals: it should be theoretically sound and efficient. The criteria of theoretical soundness are ([van Rijsbergen, 1979], p. 47):

- The method should be stable under growth, i.e.; the partitioning should not change drastically with the insertion of new documents;
- Small errors in the description of the documents should lead to small changes in the partitioning;
- The method should be independent of the initial ordering of the documents.

Many cluster generation methods have been proposed. Unfortunately, no single method meets both requirements for soundness and efficiency. Thus, we have two classes of methods:

- "Sound" methods, which are based on the document-document similarity matrix;
- Iterative methods, that are more efficient and proceed directly from the documents vectors.

4.3.1 Methods Based on Similarity Matrix

Retrieval is usually accelerated using hierarchic clustering methods that result in binary treelike classifications in which small clusters of documents that are judged to be strongly similar to each other are nested within larger and larger clusters containing documents that are less similar. The single cluster containing the entire collection is represented by the root of the tree while the individual documents reside in the leaves; nodes in the body of the tree correspond to clusters that are formed during the operation of the clustering procedure.

These methods usually require O (n^2) time (or more) where n is the number of documents and apply graph theoretic techniques. A document-to-document similarity function, as the ones showed in section 6, has to be chosen.

Given the document similarity matrix, a simplified version of such clustering method would work as follows ([Duda and Hart, 1973], p.238): An appropriate threshold is chosen and two documents with a similarity measure that exceeds the threshold are assumed to be connected with an edge. The connected components (or the maximal cliques) of the resulting graph are the proposed clusters.

Two main strategies are used for the construction of hierarchies of clusters: agglomerative or divisive strategies. An agglomerative strategy proceeds through a total of N-1 fusions for a collection of N documents and results in the classification being built upwards from the leaves, with the smallest clusters being generated first and with the final fusion resulting in the root of the tree. Besides in a divisive strategy a single initial cluster is subdivided into smaller and smaller groups of documents.

Divisive algorithms result in monothetic classifications where every document belonging to a cluster must contain a certain term as a necessary condition for cluster membership. Agglomerative algorithms result in polythetic classifications where there are not specific terms required for cluster membership and documents in a given cluster have several terms in common with one or more documents in the same cluster. Monothetic classifications and divisive algorithms are of less use than polythetic classifications and agglomerative algorithms.

The most famous of the agglomerative clustering methods is single linkage, or nearest neighbor, in which clusters are formed on the basis of the similarity between the most similar pair of documents, one of which is in each of a pair of clusters. The clusters formed by this method have the property that any cluster member is more similar to at least one member of that cluster than it is to any member of any other cluster; hence the name nearest neighbor. In graph theoretic terms, the single linkage clusters at some similarity level are the connected components of a graph. A characteristic of this method is its tendency to form loosely bound clusters with little internal cohesion, a phenomenon referred to as chaining and that has resulted in several attempts to produce related methods that do not suffer from this defect [Wishart, 1969] [Jarvis and Patrick, 1973].

An alternative formulation [Gordon, 1981] of the single linkage method assumes that an N x N inter-document similarity matrix is available where the element SIM[K,L] contains the similarities between two documents K and L. The K and L are defined as belonging to the same cluster at some similarity threshold T if there is a chain of intermediate documents

$$SIM[K,X_1], SIM[X_1,X_2] \dots SIM[X_p,L] \ (1 \leq P \leq N)$$

linking them together where each similarity is greater than T. Thus, the number of documents in the cluster will increase as T is decreased, with the documents in the cluster at some threshold being a subset of those in the cluster at some lower similarity threshold.

Closely related to single linkage clustering is the concept of a minimal spanning tree, or MST. Given the set of N(N –1)/2 inter-document similarities, a spanning tree is a set of N – 1 similarities that links all of the N objects in the dataset together into a connected graph without any circuits; the MST is that spanning tree for which the sum of the N – 1 similarities is the maximum [Lee, 1981]. Gower and Ross show that the single linkage clusters at some threshold T can be obtained by deleting all coefficients from the MST for which the similarity was less than T [Gower and Ross, 1969]. MSTs have been used for the clustering of index terms in studies of query expression [van Rijsbergen, 1977] [van Rijsbergen et al., 1981]; however, they do not seem to have been used for the clustering of documents.

An alternative and popular approach to the description of clustering methods is to use the algorithms that are used to the implementation of the methods. Single linkage is just one of a group of hierarchic agglomerative clustering methods that have been used extensively over the years and that includes the complete linkage, group average, and Ward methods inter alia. The following general algorithm may describe all of these methods:

```
FOR I := 1 TO N-1 DO
FOR J := I=1 TO N DO calculate SIM[I, J];
REPEAT
search SIM to identify the most similar remaining pair of clusters;
fuse this pair, K and L, to form a new cluster KL;
update SIM by calculating the similarity between KL and each of the remaining clusters
UNTIL there is only a single cluster
```

The various hierarchic agglomerative methods differ in the definition of similarity that is used for the selection of the most similar pairs of clusters and for the updating of SIM in this algorithm.

The complete linkage, or furthest neighbor, method is the converse of the single linkage, since the least similar pair of documents forms the basis for the definition of inter-cluster similarity. Thus, each cluster member is more similar to the least similar document of that cluster than to the least similar document in any other cluster. This definition of cluster membership is very much stricter than that for single linkage and large numbers of small tightly bound groupings here replaces the large straggly clusters in the latter case. This form of classification that may be just as inappropriate as the extended single linkage clusters in some applications. In graph theoretical terms, complete linkage clustering corresponds to the identification of the maximally complete subgraphs at some threshold similarity.

The group average method results in clusters such that each cluster member has a greater average similarity to the remaining members of that cluster than it does to all members of any other cluster. It thus represents a midpoint between the two extremes represented by single linkage and complete linkage but has been criticized for the quality of the produced clusters [Edelbrock, 1979].

Ward's method joins together those two clusters whose fusion results in the least increase in the sum of the distances from each document to the centroid of its cluster. Ward's method has proved to be an extremely powerful grouping mechanism, and Wishart suggests that it is probably the most generally useful hierarchic procedure [Wishart, 1978]. It has, however, been criticized for tending to produce spherical clusters, which may not accurately reflect the true shape of the clusters present in the data set. Moreover, it is only defined explicitly when the Euclidean distance is used for the calculation of the inter-document similarities; the use of an association coefficient (e.g., the Dice coefficient) will not result in an exact Ward classification.

4.3.2 Iterative Methods

Iterative algorithms begin with a set of k centroid documents usually chosen by the user. First, the documents are partitioned into k clusters: a document x becomes a member of cluster i if z_i is the centroid document closest to x. For each cluster a centroid document is computed averaging the documents in that cluster. In this way the assignment of documents to clusters and the re-computation of the centroid documents is repeated until centroids reach a stable value.

Iterative algorithms are thus similar to fitting routines, which begin with an initial "guess" for each fitted parameter and then optimize their values. Algorithms within this family differ in the details of generating and adjusting the partitions [Faber, 1994].

These algorithms operate in less than quadratic time (that is, O(nlogn) or $O(n^2/\log n)$) on the average, they are based directly on the object (document) descriptions and they do not require the similarity matrix to be computed in advance. The price for the increased efficiency is the sacrifice of the "theoretical soundness"; the final classification depends on the order the objects are processed and the results of errors in the document descriptions are unpredictable. The proposed algorithms are based on heuristics and they also need a number of empirically determined parameters such as:

- The number of clusters desired;
- A minimum and maximum size (i.e., number of documents) of each cluster;
- A threshold on the document-to-cluster similarity measures, below which a document will not be included in the cluster;
- The control of overlap between clusters;
- An arbitrarily objective function to be optimized.

4.4 Recent trends in Document Clustering

In the previous section I presented traditional strategies in document clustering. The continuous growth in the amount of information we need to manage makes document clustering an active field of research in information retrieval. In this section I describe some novel approaches to document clustering.

Although many alternatives to traditional strategies have been developed a new and definitely better approach has not yet been discovered. [Faloutsos and Oard, 1994] compared traditional methods for information retrieval to modern developments that try to capture semantic information such as neural nets and Latent Semantic Indexing (LSI) and they conclude that there are not clear advantages towards the modern developments. They state that indexing on phrases provides some small improvements (from negative up to 20% savings [Croft et al., 1991]) on the precision/recall performance, at the expense of more elaborate preprocessing of the documents (full or partial parsing and syntax analysis). They also show that LSI provides improvements on precision/recall, requiring (a) the availability of a training corpus, on which to build the term-document matrix and perform the Singular Value Decomposition (SVD) and (b) a large amount of computer time, since SVD of an m x n matrix is worse than quadratic on the smallest dimension.

Modern clustering applications demand rapid response times while utilizing data sets too large for standard clustering algorithms. The bottleneck in clustering text documents is calculating the distance between term vectors. This calculation takes time proportional to the number of distinct terms in the smaller document. One obvious way to speed up clustering, then, is to discard unimportant words from documents or to project the term space into a smaller subspace containing important words for the documents. For another application of clustering, word sense disambiguation, it has been shown that projection onto a smaller subspace does not affect performance. This is the main motivation for the use of term space projection techniques described in the next section.

Apart from the traditional use of clusters in improving the recall of searches new applications of document clustering are appearing. [Cutting et al., 1992] describe an application of document clustering where the clusters are used for accessing the contents of the collection of documents. They describe a browsing method called Scatter/Gather, which uses document clustering as its primitive operation. This technique is directed to information access with no specific goals and serves as a complement to more focused techniques. I introduce Scatter/Gather in the section 4.4.2.

4.4.1 Term space projections

At its heart, the clustering of text documents consists of clustering m vectors in an n-dimensional space, where m is the number of documents and n is the number of terms. For a given vector d, the value d_t, is the number of times term t occurs in document d.

Projecting is the act of converting some non-zero values of d_t to 0, possibly first modifying the vector d in an arbitrary way. If we choose not to convert any value, then we obtain the trivial projection, called FULL for "full profiles".

One common approach to projecting is a method called truncation, where for each document we excise a number of "unimportant" terms. This type of projection is called local because each document is projected onto a different subspace. As an example of truncation we could mention weighting where each term in a document is assigned a weight based on its frequency in that document and, possibly, in other documents and terms with the lower or the higher weight are then deleted.

The alternative to truncation is called dimension reduction, in which the terms to delete are chosen first, and then these terms are deleted from each document. This type of projections is called global because all documents are projected onto the same subspace. A good example of dimension reduction is called Latent Semantic Indexing (LSI) where documents are mapped from the term space to an orthonormal space by means of Latent Semantic Indexing (LSI), an application of Singular Value Decomposition (SVD) to this problem. An advantage of the orthonormal space (which is called "LSI space") is that, if the dimensions are ordered, projecting the set of documents onto the d lowest dimensions is guaranteed to have, among all possible projections to a d dimensional space, the lowest possible least-square distance to the original documents. In this sense, LSI finds an optimal solution to dimensionality reduction. See [Deerwester et al., 1990] for further discussion of LSI and [Berry, 1992] for a description of SVD and algorithms used for compute it.

[Schütze and Silverstein, 1997] show that projecting documents via LSI and truncation offers a dramatic advantage over full-profile clustering in terms of time efficiency. The improved efficiency, surprisingly, is not accompanied by a loss of cluster quality.

4.4.2 Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections

According to [Cutting et al., 1992] the standard formulation of the information access problem presumes a query, the user's expression of an information need. The task is then to search a corpus for documents that match this need. However, we can imagine a situation in which it is hard, if not impossible, to formulate such a query precisely. For example, the user may not be familiar with the vocabulary appropriate for describing a topic of interest, or may not be able to commit himself to a particular choice of words. Indeed, the user may not be looking for anything specific at all, but rather may wish to discover the general information content of the corpus. Access to a document collection in fact covers an entire spectrum: at one end is a narrowly specified search for a particular document, given something as specific as a its title, at the other end is a browsing session with no well defined goal, satisfying a need to learn more about the document collection. It is common for a session to move across the spectrum, from browsing to search: the user starts with a partially defined goal that is refined as he finds out more about the document collection. Standard information access techniques tend to emphasize the search end of the spectrum. A glaring example of this emphasis is cluster search, where clustering, a technology capable of topic extraction, is submerged from view and used only to assist near-neighbor search.

[Cutting et al., 1992] propose an alternative application for clustering in information access inspired from the access methods typically provided with a conventional textbook. If one has a specific question in mind, and specific terms, which define that question, one consults the index, which directs one to passages of interest. However, if one is simply interested in gaining an overview, or has a general question, one peruses the table of contents, which lays out the logical structure of the text. The table of contents gives a sense of what sort of questions might be answered by a more intensive examination of the text, and may also lead to specific sections of interest. One can easily alternate between browsing the table of contents and searching the index. A browsing method, called Scatter/Gather, is proposed which uses a cluster-based, dynamic table-of-contents metaphor for navigating a collection of documents; and one or more word-based, directed text search methods, such as near neighbor search or snippett search [Pedersen et al., 1991]. The browsing component describes groups of similar documents, one or more of which can be selected for further examination. This can be iterated until the user is directly viewing individual documents. Based on documents found in this process, or on terms used to describe document group, the user may, at any time, switch to a more focused search method. The browsing tool will not necessarily be used to find particular documents, but may instead help the user formulate a search request, which will then be serviced by some other means. Scatter/Gather may also be used to organize the results of word-based queries that retrieve too many documents.

In the basic iteration of the proposed browsing method, the user is presented with short summaries of a small number of document groups.

Initially the system scatters the collection into a small number of document groups, or clusters, and presents short summaries of them to the user. Based on these summaries, the user selects one or more of the groups for further study. The selected groups are gathered together to form a sub-collection. The system then applies clustering again to scatter the new sub-collection into a small number of document groups, which are again presented to the user. With successive iteration the groups become smaller, and therefore more detailed. Ultimately, when the groups become small enough, this process bottoms out by enumerating individual documents.

5 Implemented Clustering Algorithm

5.1 Implementation of the cluster generation method

I implemented a cluster generation method for building clusters out of the collection of HTML documents collected by The Big Memory.

I chose a similarity matrix based algorithm for two reasons. On the one side in this kind of algorithms I could use a similarity measure that takes into consideration particular features of HTML documents. On the other side I was more concerned with the quality of the resulting clusters, and therefore in the theoretical soundness of the clusters generation method, rather than in the speed of the clusters generation method.

I used a hierarchical algorithm because non-hierarchical algorithms generally involve a number of input parameters to control the clustering (e.g., the number of clusters required, minimum or maximum cluster sizes, and threshold document-cluster similarity levels) and I wanted a fully automatic document-clustering algorithm requiring no user collaboration. Moreover non-hierarchical algorithms are rather arbitrary in operation since the final clusters may depend on the order in which the document collection is processed, the random selection of documents as initial cluster centers, or the exact parameter values that are used.

I implemented the complete linkage algorithm found in [Voorhees, 1986]:

The algorithm used to construct the complete link hierarchy is due to Chris Buckley. In the worst case the algorithm requires $O(N^2)$ space and $O(N^3)$ time, but the worst case should not be expected to arise due to implementation considerations.

The complete link algorithm uses the similarity measure that I describe in section 6.

I developed the whole program in Java using Java Database Connectivity (JDBC) for connecting to a relational database where all the indexing and clustering information was stored. As Java is a multi-platform programming language, this program can run on any platform supporting Java and can connect to any relational database supporting JDBC or ODBC (Open Database Connectivity).

The program was developed using an object-oriented methodology[1] and the database was designed using both an entity-relation and a relational model[2].

5.1.1 Sample output of the cluster generation method

I run the clustering algorithm with a collection of HTML documents collected by The Big Memory while I was browsing the Web. This collection contained several documents related to advertising on the Internet. In what follows I show a sub-tree of the generated clusters hierarchy related to this topic. The leaves of the sub-tree represent actual HTML pages of the collection and the intermediate nodes represent hierarchical clusters built at the specified similarity level.

[1] Refer to Appendix A for some comments on the object-oriented methodology.

[2] Refer to Appendix B for some comments on the use of models.

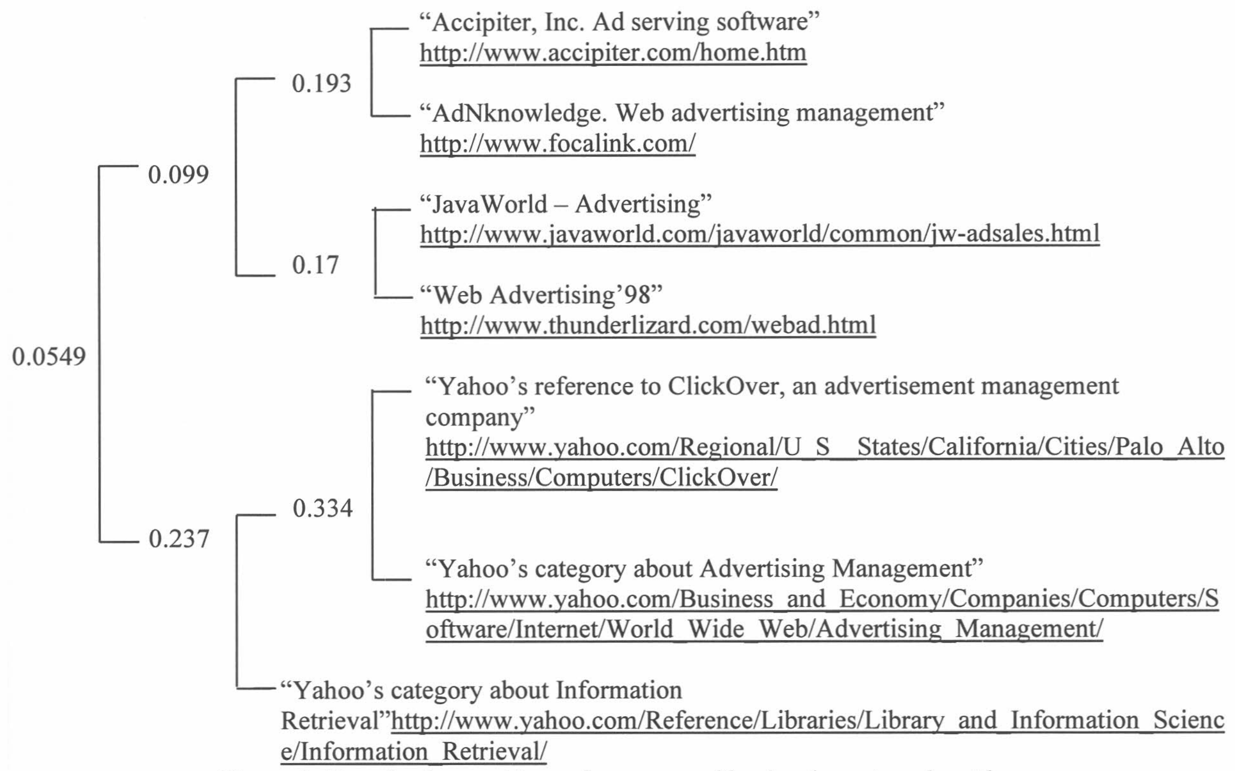

0.0549
0.099
0.193
"Accipiter, Inc. Ad serving software"
http://www.accipiter.com/home.htm
"AdNknowledge. Web advertising management"
http://www.focalink.com/
0.17
"JavaWorld – Advertising"
http://www.javaworld.com/javaworld/common/jw-adsales.html
"Web Advertising'98"
http://www.thunderlizard.com/webad.html
0.237
0.334
"Yahoo's reference to ClickOver, an advertisement management company"
http://www.yahoo.com/Regional/U_S__States/California/Cities/Palo_Alto/Business/Computers/ClickOver/
"Yahoo's category about Advertising Management"
http://www.yahoo.com/Business_and_Economy/Companies/Computers/Software/Internet/World_Wide_Web/Advertising_Management/
"Yahoo's category about Information Retrieval"http://www.yahoo.com/Reference/Libraries/Library_and_Information_Science/Information_Retrieval/

Figure 5: Sample clusters hierarchy generated by the clustering algorithm. Documents in this hierarchy are related to advertising on the Internet. Leaf nodes represent HTML pages and internal nodes represent hierarchical clusters built at the specified similarity level

5.2 Implementation of the cluster searching method

I implemented a simple cluster searching method that given a query document answers a ranked list of similar clusters. This method is used by The Big Memory for searching similar clusters to the currently browsed HTML document.

The implemented cluster searching method first prunes the cluster hierarchy at a given similarity level building a collection of plain clusters. For example the implemented cluster searching method would prune the cluster hierarchy shown in Figure 5 at a similarity level of 0.083 into the two clusters illustrated in Figure 6.

Cluster1
http://www.accipiter.com/home.htm
http://www.focalink.com/
http://www.javaworld.com/javaworld/common/jw-adsales.html
http://www.thunderlizard.com/webad.html

Cluster2
http://www.yahoo.com/Regional/U_S__States/California/Cities/Palo_Alto/Business/Computers/ClickOver/
http://www.yahoo.com/Business_and_Economy/Companies/Computers/Software/Internet/World_Wide_Web/Advertising_Management/
http://www.yahoo.com/Reference/Libraries/Library_and_Information_Science/Information_Retrieval/

Figure 6: Plain clusters obtained by pruning the cluster hierarchy in figure 5 at a similarity level of 0.083

Then it builds a centroid document for each plain cluster; this centroid document summarizes the information of all documents in the cluster.

Finally it answers the ranked list of clusters for which the similarity between the query document and the cluster's centroid exceeds a fixed similarity level. These clusters are ranked by the similarity between the query document and the cluster's centroid.

The problem of finding the cluster more closely related to a given document is a document classification problem (refer to section 4 for a brief definition of document classification). In the current work I took a simplistic solution to this problem and implementing a new strategy is an excellent direction for future work.

6 Information about the Structure of HTML Documents

6.1 Document Similarity Measures

A document similarity measure is a function that given a pair of documents answers a real number representing the similarity between the two documents. A good similarity measure should return a high value for similar documents and a low value for dissimilar ones.

Numerous coefficients of association have been described in the literature, see for example Goodman and Kruskal [Goodman and Kruskal, 1954], Kuhns [Kuhns, 1965], Cormack [Cormack, 1971] and Sneath and Sokal [Sneath and Sokal, 1973].

There are five commonly used measures of association in information retrieval. Since in information retrieval documents and requests are most commonly represented by term or keywords lists, I shall simplify matters by assuming that an object is represented by a set of keywords and that the counting measure |.| gives the size of the set. We can easily generalize to the case where the keywords have been weighted, by simply choosing an appropriate measure (in the measure theoretic sense).

The simplest of all association measures is:

$$|X \cap Y| \qquad \text{SimpleMatchingCoefficient}$$

which is the number of shared index terms. An important drawback of this coefficient is that it does not take into account the sizes of X and Y. Evaluating this similarity measure on a pair of documents A and B, each one containing 100 words and sharing 10 words, will answer a value of 10. Also evaluating this measure on a pair of identical documents C and D, each one containing and sharing 10 words, will answer a value of 10. Although the pair (C, D) is more similar than the pair (A, B) this similarity measure is answering the same value for both pairs.

The following coefficients which have been used in document retrieval take into account the information provided by the sized of X and Y.

$$2 * \frac{|X \cap Y|}{|X| + |Y|} \qquad \text{Dice Coefficient}$$

$$\frac{|X \cap Y|}{|X \cup Y|} \qquad \text{Jacard Coefficient}$$

$$\frac{|X \cap Y|}{|X|^{1/2} * |Y|^{1/2}} \qquad \text{Cosine Coefficient}$$

$$\frac{|X \cap Y|}{\min(|X|, |Y|)} \qquad \text{Overlap coefficient}$$

These may all be considered to be normalized version of the simple matching coefficient.

It is worth noting that the previous similarity measures could be used for comparing any pair of objects (X, Y) whenever these objects can be described as vectors of features. For example the previous similarity measures could be used to compare a pair of gray level images X and Y because gray level images can be described as a vectors of size 256: X[i]=n if and only if image X has n pixels of gray level i. In this context |X| means the number of pixels in image X, $|X \cap Y|$ means the number of common colors in X and Y and $|X \cup Y|$ means the total number of colors in X and Y.

In what follows whenever I mention any of this similarity measures I will mean the application of the similarity measure to pairs of documents.

6.2 *Focusing on HTML features*

When computing similarities between HTML documents there is more information to be used than merely the simple word occurrences. For example we could consider the document field where common words appear because two HTML documents are probably going to be more similar if they share words in the title than if they share words in the body. We could also consider the server where the documents belong because two documents belonging to the same server are probably going to be quite similar. Finally if one document has a link to another document then they will probably be quite similar. These are several examples showing how the structure of HTML documents can be used in finding their similarity.

I develop a mechanism that incorporates information about the structure of HTML documents into a given similarity measure. The structure of HTML documents is incorporated through the HTML criteria that I describe in the next section.

6.2.1 HTML Criteria

An HTML criterion is a function that, given a pair of HTML documents, answers a real number in the range [0..1]. Each criterion focuses on a specific feature of HTML documents.

I have implemented the following criteria:

Title Criterion: for computing this similarity measure between documents X and Y this criterion first extracts the words in the title of document X (titleX) and the words in the title of document Y (titleY). It filters both titleX and titleY with the stop list[3] and extracts their stems[4]. Then it computes the intersection between titleX and titleY (*intersection*) answering the following value:

$$2\frac{|\text{int}\,er\,\sec\,tion|}{|titleX|+|titleY|} \text{ (Dice Coefficient)}$$

As an example consider that document X has the title THE CAR RENTAL HOMEPAGE and document Y has the title JERIN'S CARS RENTAL. Then when I filter the title of document X using the stop list I get CAR RENTAL and the stop list does not filter any word from document Y. The stemming algorithm does not modify title X and it transforms titleY to JERIN CAR RENTAL. The intersection will contain CAR RENTAL so the returned similarity will be 2*2/(2+3)=4/5.

H1 Criterion: for computing this similarity measure between documents X and Y this criterion first extracts the words from the H1[5] field building two collections of words, h1X and h1Y. It filters both h1X and h1Y with the stop list and extracts their stems. Then it computes the intersection between h1X and h1Y (*intersection*) answering the following value:

---

[3] A stop list is a negative dictionary used to remove common words (such as the, and, before, until) that are not useful in characterizing the similarity between two documents. I've build a negative dictionary adding Web specific words such as "web" and "homepage" to a standard information retrieval (IR) negative dictionary found at (http://local.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils).

[4] A stemming algorithm extracts common prefixes and suffixes from words and produces stems. For example a stemming algorithm will reduce the word cars to the stem car. I'm using the well known Porter stemming algorithm (http://local.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils).

[5] If an HTML document contain the following tag *<H1>Text</H1>* then I call text in the H1 field to *Text*.

$$2\frac{|intersection|}{|h1X|+|h1Y|} \text{ (Dice Coefficient)}$$

H2 Criterion: for computing this similarity measure between documents X and Y this criterion first extracts the words from the H2[6] field building two collections of words, h2X and h2Y. It filters both h2X and h2Y with the stop list and extracts their stems. Then it computes the intersection between h2X and h2Y (*intersection*) answering the following value:

$$2\frac{|intersection|}{|h2X|+|h2Y|} \text{ (Dice Coefficient)}$$

Links text Criterion: for computing this similarity measure between documents X and Y this criterion first extracts the words from the text of the links[7] building two collections of words, linksTextX and linksTextY. It filters both linksTextX and linksTextY with the stop list and extracts their stems. Then it computes the intersection between linksTextX and linksTextY (*intersection*) answering the following value:

$$2\frac{|intersection|}{|linksTextX|+|linksTextY|} \text{ (Dice Coefficient)}$$

Links Criterion: for computing this similarity measure between documents X and Y this criterion extracts all the links[8] in document X and document Y. Now if a link of document X points to document Y or if a link in document Y points to document X then this criterion answers 1 otherwise this criterion computes the intersection (*intersection*) and the union (*union*) between the links of document X and document Y answering the following number:

$$\frac{|intersection|}{|union|} \text{ (Jacard Coefficient)}$$

URLs Criterion: this criterion is based on the URLs from where the documents come. Let's assume the following notation for describing an URL: http://server/p1/p2/…/pn (in http://www.dc.uba.ar/materias/optativas.html n will be 2, p1=materias, p2=optativas.html). If both URLs belong to the same server this criterion will answer the proportion of common tokens in the path (pi's).

Consider, as an example, the following pair of URLs: http://www.dc.uba.ar/research/IR/papers.html and http://www.dc.uba.ar/research/NeuralNets/publications.html. As both URLs belong to the same server, then the server criterion will answer the proportion of common tokens in the path. These URLs have the token research in common in the set of path tokens {research, IR, papers.html, NeuralNets, publication.html} then this criterion will answer (1/5).

---

[6] If an HTML document contain the following tag *<H2>Text</H2>* then I call text in the H2 field to *Text*.

[7] If an HTML document contains the following tag *<a href="http://server/path">Text</a>* then I call text of the link to *Text*.

[8] If an HTML document contains the following tag *<a href="http://server/path">Text</a>*then I call link to *http://server/path*

6.2.2 Exhaustive versus specific similarity measures

Standard similarity measures, such as the Dice or Jacard coefficients, use all the words in a pair of documents to find their similarity. These similarity measures perform an exhaustive comparison of document features (words) for finding their similarity so we call them exhaustive.

HTML criteria are also document similarity measures because they are functions that given a pair of documents answer a real value representing their similarity. Contrary to standard similarity measures, the HTML criteria only consider specific words of the documents being compared; for example, the Title Criterion only considers the words in the title field. So we call them specific similarity measures.

6.3 *Mechanism for Adding Information about the Structure of HTML documents*

Given an exhaustive similarity measure this mechanism builds an enriched similarity measure that adds information about the structure of HTML documents to the given similarity measure.

6.3.1 The Enriched Similarity Measure

The enriched similarity measure combines the exhaustive similarity measure with a group of HTML criteria. The overall similarity measure for a pair of documents X, Y is:

$$Enriched\,(X,Y) = \frac{We * EXHAUSTIVE\,(X,Y) + W1 * C1(X,Y) + \ldots + Wn * Cn(X,Y)}{|Wd| + |W1| + \ldots + |Wn|}$$

where:

Enriched(X, Y):	enriched similarity for documents X and Y
EXHAUSTIVE(X, Y)	exhaustive similarity for documents X and Y
C1(X, Y)	similarity between documents X and Y according to criterion 1
…	
Cn(X, Y)	similarity between documents X and Y according to criterion n
We	Weight assigned to the exhaustive similarity measure
W1	Weight assigned to criterion 1
…	
Wn	Weight assigned to criterion n

The HTML criteria and the exhaustive similarity measure return values in the range [0..1] and weights are real numbers with the only limitation that cannot be all simultaneously zero. So the enriched similarity measure return values in the range [-1..1].

In the next section I show how to find the weights values, W's, that optimize the enriched similarity measure.

6.3.2 Finding the Weights

Different collections of documents have particular characteristics.

If the collection of documents is very general, including different topics, then the words in the title field will be a good indicator about the similarity between two documents while if the collection if very specific, then the words in the title field will not be so useful. As an example of a general collection consider the documents in Yahoo where if two documents share the word "Biology" in the title field then they will probably be very similar. Now as an example of a specific collection consider the documents in the digital library of the Biology Department at a university; here the fact that two documents share the word "Biology" in the title field does not mean that the documents are similar.

Besides if the collection of documents is densely linked then the link information could be a good indicator about the similarity between documents in the collection.

Moreover, if the collection of documents contains several documents from the same server then the information about the source server could be useful when finding the similarity between pairs of documents in the collection.

I implemented an algorithm that, given some training data from a collection of documents, automatically finds the optimal weight for the Dice Coefficient and for each criterion in the enriched similarity measure. In this way the resulting similarity measure will learn which features should be considered important and which not for the specific collection of documents.

In finding the optimal weights I use some training data provided by end users. This training data consists of a set of document queries and for each document query a set of relevant documents. Figure 7 shows the descriptions and URLs of a sample document query and its relevant documents.

Query
"Immigration Statute and Regulation index" http://www.oalj.dol.gov/public/ina/refrnc/istatin.htm
Relevant Documents
"Hot Links in Immigration" http://wohfa.asiamarket.com/community/hotlink.htm
"American Policy Regarding Foreign Workers" http://www.geocities.com/CapitolHill/9302/main.htm
"Report about an American company violating immigration law" http://www.infoark.com/softpac/news/immigration/aas.sep15
"The New Americans: Economic, Demographic, and Fiscal Effect of Immigration (1997)" http://www.nap.edu/readingroom/records/0309063566.html
"Immigration Bulletin for May 1996" http://www.ohboy.com/~gsiskind/96may/index.html
"Siskind's Immigration Bulletin" http://www.ohboy.com/~gsiskind/bulletin.html
"US Borders Must Be Protected" http://www.spotlight.org/html/immigration.html
"Foreign Student Immigration Issues; U.S. Immigration" http://www.wave.net/upg/immigration/students.html

Figure 7: Descriptions and URLs of a query document and its relevant documents

I build a retrieval system that given a query document and using the enriched similarity measure answers a ranked list of documents similar to the query.

The set of weights is optimized using the approach detailed in [Bartell et al., 1995]. They propose a method that explicitly optimizes the ability of the retrieval system to rank documents well for a finite training set of users' judgments of the desired ordering of retrieved documents. The retrieval system is optimized by automatically adjusting the weights of the similarity measure. The weights are adjusted by numerically optimizing a criterion of how well the system is ranking documents with respect to the users' target ranking for the set of queries. I use Guttman's Point Alienation, a rank order static motivated by techniques in the field of Multidimensional Scaling (MDS) [Shepard, 1962; Kruskal, 1964; Borg and Lingoes, 1987], as a useful criterion for how well the system is matching the users' target rankings. Optimizing the criterion automatically via gradient descent techniques adjusts the set of weights so that the system ranks documents more similarly to the users' own preference order.

The immediate goal of the method is to match the users' own document preferences by ranking more relevant documents before less relevant ones for the training set of queries.

The larger goal of the method is to find a set of weights that result in improved performance for novel queries – queries not considered during optimization. This ability for the system to generalize to new queries is what makes the optimized system of greater value than the non-optimized system.

$\Re_{\theta q}(d)$ is the enriched similarity measure which is to be optimized. $\Re_{\theta q}(d)$ provides the single relevance estimate for each document d for a query q. $\Re_{\theta q}(d)$ is called the ranking function because its scores determines the order by which retrieved documents are resented to the user. Larger values of $\Re_{\theta q}(d)$ imply that the retrieval system estimates that d is likely relevant to the document query; small values imply that d is less relevant. Of course, the particular values generated by $\Re_{\theta q}(d)$ are of less interest; rather, it is the rank order of documents implied by the values which is important.

θ, in $\Re_{\theta q}(d)$, is the set of weights of the similarity measure so according to the notation used in the previous section $\theta=(Wd, W1, \ldots, Wn)$ where Wi is the weight corresponding to criterion Ci.

The goal is to find values of θ such that $\Re_{\theta q}(d)$ best ranks the documents for a set of queries. The notion of "best ranking" is with respect to a desired ordering over the documents for each query. This "desired ordering" is formalized using the notation of Wong and Yao [Wong and Yao, 1990]: $>_q$ is defined as the Document Preference Relation, a binary relation of document pairs for a given query, q. For documents d and d' from the collection of documents D:

$$d >_q d' \Leftrightarrow \text{the user prefers d to d'}$$

$\Re_{\theta q}(d)$ has successfully ranked the documents for query q when the ordering implied by the values of $\Re_{\theta q}(d)$ correspond to the user preference partial ordering. The optimization method heuristically search the space of parameters seeking the θ such that:

$$\forall\, d, d' \in D,\ \Re_{\theta q}(d) > \Re_{\theta q}(d') \text{ whenever } d >_q d'$$

Note that the particular value given by $\Re_{\theta q}(d)$ for a query and document is irrelevant; it is only the relative order of the $\Re_{\theta q}(d)$ values that is of concern.

6.3.2.1 Criterion for Optimization

I optimize a criterion, described in [Bartell et al., 1995], which measures how well the ordering implied by $\Re_\theta$ corresponds to the target document preference relation $>_q$. The criterion is derived from Guttman's Point Alienation measure [Guttman, 1978], a statistical estimate of the rank correlation between two variables. Our criterion is:

$$J(R_\theta) = \frac{-1}{|Q|}\sum_{q\in Q}\frac{\sum_{d>_q d'}(R_{\theta,q}(d) - R_{\theta,q}(d'))}{\sum_{d>_q d'}|(R_{\theta,q}(d) - R_{\theta,q}(d'))|}$$

Q is the set of training queries, and d, d' are from the collection of documents D. This function is an average, over the queries in Q, of the rank match between $>_q$ and $\Re_{\theta q}$ for the documents in D. Note that when $\Re_{\theta q}$ perfectly orders the elements with respect to $>_q$, the numerator and denominator are equivalent, since the sum of the differences $\Re_{\theta q}(d) - \Re_{\theta q}(d')$ is equivalent to the absolute value of the sum of differences. In this case, the ratio is 1.0 and the criteria takes on its minimum value –1.0. When $\Re_{\theta q}$ is completely disordered, the numerator is the negative of the denominator, the ratio is –1.0, and the criterion is maximized at 1.0. The goal is to minimize the criterion.

Figure 8 provides a small example of the evaluation of J. We assume that the similarity measure has two criteria C1 and C2 and that there is one query document in Q , Q={q1}, and 3 documents in D, D={d1, d2, d3} and the preference relation is given.

Preference Relation
{d1, d2} $>_{q1}$ {d3}

Preference relation for query q1

Query	Document	Similarity From Dice	Similarity From C1
Q1	D1	0.7	0.5
Q1	D2	0.8	0.6
Q1	D3	0.9	1.0

Similarities between query q1 and documents in D

$$J(R_\theta) = -1\times\frac{\frac{0.7\theta_1+0.5\theta_2}{|\theta_1|+|\theta_2|} - \frac{0.9\theta_1+1.0\theta_2}{|\theta_1|+|\theta_2|} + \frac{0.8\theta_1+0.6\theta_2}{|\theta_1|+|\theta_2|} - \frac{0.9\theta_1+1.0\theta_2}{|\theta_1|+|\theta_2|}}{\left|\frac{0.7\theta_1+0.5\theta_2}{|\theta_1|+|\theta_2|} - \frac{0.9\theta_1+1.0\theta_2}{|\theta_1|+|\theta_2|}\right| + \left|\frac{0.8\theta_1+0.6\theta_2}{|\theta_1|+|\theta_2|} - \frac{0.9\theta_1+1.0\theta_2}{|\theta_1|+|\theta_2|}\right|}$$

Figure 8: Example of the evaluation of the criterion J

It is worth mentioning that there are numerous other possible criteria which could be used in place of the Guttman's Point Alienation. Alternatives include Kruskal's stress function [Kruskal, 1964], a variation of Pearson's correlation coefficient [Borg and Lingoes, 1987], and the monotonicity coefficient of Guttman [Guttman, 1978]. In addition, the Perceptron Learning Rule [Rosenblatt, 1962] is a well known criterion from the neural network literature that can also be used as a measure of rank correctness, as has been done by Wong and Yao [Wong and Yao, 1990, Wong et al., 1993] in their adaptive information retrieval approach. Squared Error, used by Fuhr & Buckley [Fuhr and Buckley, 1991], is an additional option, though it is applicable only to binary relevant/irrelevant relevance assessments and not to the more general quasi-orders. Guttman's Point Alienation has a good mix of favorable features. In particular, it coincides well with a standard goodness measure in information retrieval, average precision; is reasonable efficient; and it is a well know and used statistical measure.

6.3.2.2 Numerical Optimization Method

Since $J(\Re_\theta)$ is differentiable with respect to $\Re_\theta$ everywhere except on a finite number of points, any number of standard numerical optimization methods which make use of the gradient can be used (see [Press et al., 1988] for a good reference). In the current work I implemented the conjugate gradient method.

$J(\Re_\theta)$ has critical points at which the derivative is degenerated for the purpose of optimization. One such critical point occurs when all $\Re_{\theta q}(d)$ are equivalent for $d \in D$. This is an extreme condition and is very infrequent. A more prevalent critical point occurs when $\Re_{\theta,q}(d)=\Re_\theta(d')$, as the derivative of $|x|$ at $x=0$ is required. In this case, we define $\partial|x|/\partial x=-1.0$ for $x=0$; this has the effect of forcing $\Re_{\theta,q}(d)$ to be strictly greater than $\Re_{\theta,q}(d')$ when $d>_q d'$.

A limitation of the numerical optimization method advocated here is that it may not always find the global optimum of the criterion function. This is because it essentially does local hill climbing, i.e., it finds the optimal set of respect to the starting point. It may therefore get caught in a local hill rather than finding the true optimum. In the event that local minimum do cause problems for these optimization methods, alternatives are available (e.g., Simulated Annealing [Press et al., 1988]) which can increase the likelihood of finding the global optimum, though typically at greater computational expense.

6.4 Evaluation

The performance of a similarity measure is typically evaluated using a recall-precision evaluation. Using this evaluation I compare the performance of a classical document similarity measure, the Dice Coefficient, to the performance of its enriched counterpart. Afterwards I use a statistical test to evaluate whether the difference in performance is significant or not. I describe recall-precision evaluations in the next section and I describe the statistical test in section 6.4.2.

6.4.1 Description of Precision vs. Recall Evaluation

A retrieval system is a software program that given an input query and using a similarity measure answers a collection of documents relevant to the query. A retrieval system using a good similarity measure should be able to retrieve a large part of the relevant information contained in the corpus, while rejecting a large part of the extraneous information.

Retrieval systems performance is often measured by using recall and precision values, where recall measures the ability of the system to retrieve useful documents, while precision conversely measures the ability to reject useless materials. In an operational situation, where information needs may vary from user to user, some customers may prefer high recall, that is, the retrieval of most everything that is likely to be of interest, while others may prefer high precision, that is, the rejection of everything likely to be useless. A good system is one that exhibits both a high recall and a high precision.

A recall-precision evaluation uses the following input data:

1. A collection of documents;
2. A collection of queries;
3. Relevance-feedback information: for each query in the queries collection the relevance feedback information contains the set of documents relevant to the query.

The standard recall-precision evaluation measures the efficiency of a query-document similarity measure, a similarity measure that gives the similarity between a word sequence or query and a document. For example a query-document similarity measure will return the similarity for a query "UNIVERSITIES IN THE US" and the document DOC1.

In the current work I want to evaluate a document-document similarity measure, a similarity measure that gives the similarity between a pair of documents. Therefore, in the implemented recall-precision evaluation, a query instead of being a sequence of words will be a document and the relevance information will contain documents relevant to each query document.

If, for the evaluation of a given query, a cut is made through the document collection to distinguish retrieved items from non-retrieved ones on the one hand, and if procedures are available for separating relevant items from non-relevant on the other, the standard recall and standard precision may be defined as:

$$PRECISION = \frac{\textit{number}\text{ of items retrieved and relevant}}{\text{total retrieved}}$$

$$RECALL = \frac{\text{number of items retrieved and relevant}}{\text{total relevant in collection}}$$

In conventional retrieval systems the search requests are presented as Boolean combinations of search terms. The retrieved document set consists of all documents exhibiting the exact combination of keywords specified in the query. That is, each query produces an unordered set of documents that are either relevant or non-relevant. Hence for each query a single precision and a single recall value can be obtained. Pairs of recall-precision figures can be compared for two searches i and j, and whenever $\text{RECALL}_i \leq \text{RECALL}_j$ and $\text{PRECISION}_i \leq \text{PRECISION}_j$ the results of search j are judged to be superior to those for search i. Unfortunately, problems arise when $\text{RECALL}_i < \text{RECALL}_j$ and $\text{PRECISION}_i > \text{PRECISION}_j$ [van Rijsbergen, 1979]. In these cases, a judgment of superiority depends on the user's orientation. That is, the user must determine if the principal interest is in recall or in precision and assess the importance of differences between the recall and precision values. In typical retrieval systems, the recall will increase as the number of retrieved documents increases; at the same time, the precision is likely to decrease. Hence users interested in high recall tend to submit broad queries that retrieve many documents, whereas high-precision users will submit narrow and specific queries.

Some retrieval systems can produce varying amounts of output. A different recall-precision pair can be obtained for each separate output amount. The finer the division in quantity of output, the greater the number of available recall-precision pairs. For example, the retrieval decision can be based on the number of matching terms between queries and documents. A partial ranking can then be defined for the retrieved document set by first retrieving all items that exhibit at least some arbitrary k matching terms with the query for some judiciously chosen number k. Next all items with k – 1 matching terms are retrieved, followed by those with k – 2 matching terms, and so on down to the items that have no terms in common with the query. In each case the greater the number of matching query terms the higher the rank of the document in the list of retrieved documents. In such a system several different pairs of recall-precision values can be computed depending on the number of matching terms between queries and documents leading to a recall-precision curve.

In the current work for obtaining a recall-precision curve, a document query is submitted to the retrieval system. The retrieval system answers a ranked list of documents similar to the query. The documents ranked higher are more similar to the query than those documents ranked lower. From this ranked list I build mutually including sub-lists according to the rank order. If, for example, the retrieval system answers a ranked list of 50 documents for a given query then I build 5 mutually including sub-lists: the first sub-list will containing the 10 documents ranked higher, the second sub-list will containing the 20 documents ranked higher, …, the fifth sub-list will containing the 50 documents ranked higher. Then for each sub-list a recall-precision pair is computed obtaining several recall-precision points. Using these points I plot a recall-precision curve.

The precision of a sub-list is computed as the ratio of the number of relevant documents in the sub-list and the total number of documents in the sub-list. When the size of the sub-list is small then all the documents in the sub-list are ranked highly by the retrieval system and most of them will be relevant to the query. So the proportion of relevant documents in a small sub-list will be close to one. As the size of the sub-list increases the rank of the new documents in the sub-list will be smaller and more irrelevant documents will be included in the sub-list. So the proportion of relevant documents in a bigger sub-list will decrease which means that as the size of a sub-list increases its precision decreases.

The recall of a sub-list is computed as the ratio of the number of relevant documents in the sub-list and the total number of relevant documents in the collection. A small sub-list will contain a small number of relevant documents and as the sub-list becomes bigger the number of relevant documents will increase. This means that as the size of a sub-list increases its recall also increases.

From the previous two paragraphs we can conclude that small values of recall correspond to small sub-lists and that small sub-lists correspond to high precision. Also high values of recall correspond to big sub-lists and big sub-lists correspond to small values of precision. So small values of recall correspond to high values of precision and high values of recall correspond to small values of precision leading to a recall-precision curve that decreases monotonically.

For example suppose that we submit the query q_1 to the retrieval system and it answers the following ranked list:

Rank	Document
1.0	d1
0.95	d2
0.9	d3
0.8	d4
0.6	d5
0.4	d6
0.2	d7

The ranked list is partitioned into the following lists:

L1={d1, d2, d3}

L2={d1, d2, d3, d4, d5}

L3={d1, d2, d3, d4, d5, d6, d7}

Considering that documents {d1, d2, d4, d6} are relevant to q1, for each list recall-precision values are computed:

L1:

Precision = 2/3 (d1 and d2 are the two relevant documents in L1 and L1 has a size of 3)

Recall = 2/4 (d1 and d2 are the two relevant documents in L1 and there are 4 relevant documents in the collection)

L2:

Precision = 3/5 (d1, d2 and d4 are the three relevant documents in L2 and L2 has a size of 5)

Recall = 3/4 (d1, d2 and d4 are the three relevant documents in L2 and there are 4 relevant documents in the collection)

L3:

Precision = 4/7 (d1, d2, d4 and d6 are the four relevant documents in L3 and L3 has a size of 7)

Recall = 4/4 (d1, d2, d4 and d6 are the four relevant documents in L3 and there are 4 relevant documents in the collection)

We can use these recall-precision values to plot the recall-precision curve for query q1. In this example we can verify that the recall-precision curve decreases monotonically.

Figure 9 illustrates recall-precision curves for two queries (extracted from [van Rijsbergen, 1979]):

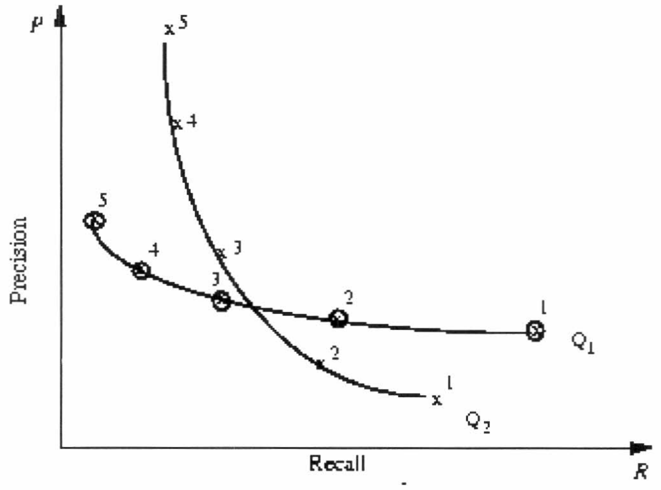


Figure 9: recall-precision curves for two queries where the ordinals indicate the values of the control parameter (extracted from [van Rijsbergen, 1979])

To measure the overall performance of a system, the set of curves, one for each query, is combined to produce an average curve. There are several techniques for averaging curves so in what follows I describe the implemented averaging technique.

If $RETREL_i$ is defined as the number of items retrieved and relevant, $RETNREL_i$ is the number retrieved but not relevant, and $NRETREL_i$ is the number relevant but not retrieved for query I then the $RECALL_i$ for query i, and the $PRECISION_i$ are defined as:

$$RECALL\,\mathrm{i} = \frac{RETREL\,\mathrm{i}}{\mathrm{RETRELi} + \mathrm{NRETRELi}}$$

$$PRECISION\mathrm{i} = \frac{RETREL\mathrm{i}}{RETREL\mathrm{i} + \mathrm{RETNRELi}}$$

If Q is the set of queries and A_q is the set of documents relevant to query q then:

$$A = \sum_{q \in Q} A_q$$

If $B_{\lambda q}$ is the set of documents retrieved by the retrieval system for query q at or above the rank level λ then:

$$B_\lambda = \sum_{q \in Q} B_{\lambda q}$$

The points (R_λ, P_λ) are now calculated as follows:

$$R_\lambda = \sum_{s \in S} \frac{A_s \cap B_{\lambda s}}{A} \quad P_\lambda = \sum_{s \in S} \frac{A_s \cap B_{\lambda s}}{B_\lambda}$$

By varying the rank level λ different pairs of recall-precision values can be computed leading to a recall-precision curve. Figure 10 (extracted from [van Rijsbergen, 1979]) shows graphically what happens when two individual precision-recall curves are combined in this way.

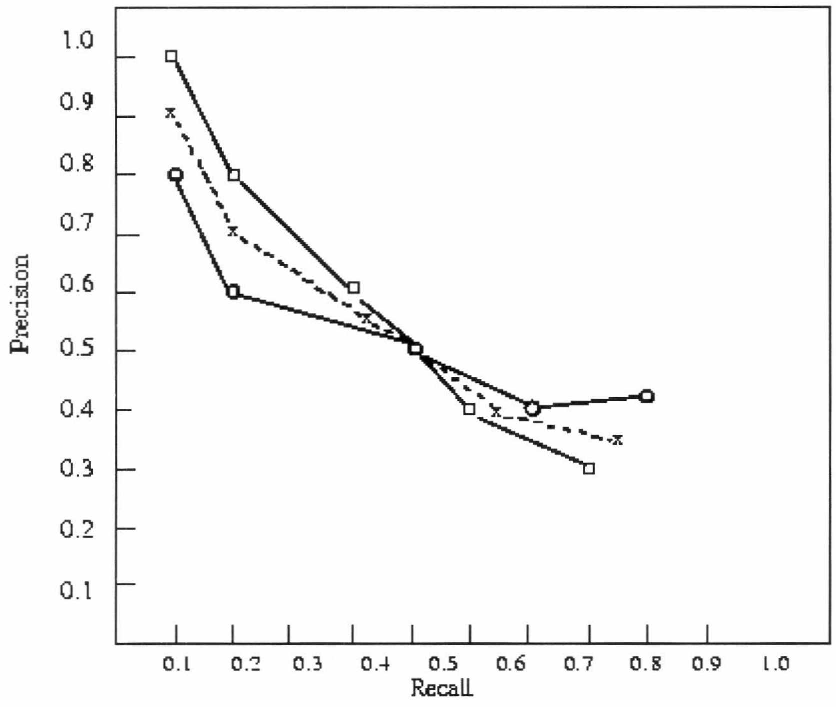


Figure 10 (extracted from [van Rijsbergen, 1979]) shows graphically what happens when two individual precision-recall curves are combined

6.4.2 Significance test

Once we have our retrieval effectiveness figures we want to establish whether the difference in performance is statistically significant or not. Since it is difficult to judge the significance of the differences between two performance curves, it is helpful to furnish statistical evidence indicating whether a given difference between two averages is in fact significant. Most standard statistical significance test based on paired comparisons will produce statistical evidence giving the probability that differences between the two sets of sample values as great as, or greater than, those observed would occur by chance. When the computed probability is small enough – for example, less than or equal to 0.05 – one concludes that the two sets of sample values are significantly different. If, on the other hand, the computed probability is greater than 0.05, the presumption is that the observed differences could have been obtained by chance – that the original pairs of values might in fact haven been derived from the same distribution.

The pairs of measurements being compared are the precision values at fixed recall levels {0.1, 0.2, …, 0.9}.The pairs of measurements are compared using the sign test [Siegel, 1956]. The way this test works is as follows: Let {$Z_a(Q_1)$, $Z_a(Q_2)$, …}, {$Z_b(Q_1)$, $Z_b(Q_2)$, …} be our two sets of measurements, under conditions a and b respectively. Within each pair ($Z_a(Q_i)$, $Z_b(Q_i)$) a comparison is made, and each pair is classified as '+' if $Z_a(Q_i)> Z_b(Q_i)$, as '-' if $Z_a(Q_i)< Z_b(Q_i)$ or 'tie' if $Z_a(Q_i)= Z_b(Q_i)$. Pairs that are classified as 'tie' are removed from the analysis thereby reducing the effective number of measurements. The null hypothesis we wish to test is that:

$$P(Z_a > Z_b)= P(Z_a < Z_b)=0.5$$

Under this hypothesis we expect the number of pairs which have $Z_a > Z_b$ to equal the number of pairs which have $Z_a > Z_b$. Another way of stating this is that the two populations from which Z_a and Z_b are derived have the same median.

The probability associated with the occurrence of a particular number of '-' can be determined by reference to the binomial distribution with P=Q=0.5 where N is the number of compared pairs. If a matched pair shows no difference (i.e. the difference being zero) it is dropped from the analysis and N is thereby reduced.

So for testing the significance of the differences I count the number of '-' differences and I look for the probability that a binomial distributed variable with P=Q=0.5 has this amount of successful outcomes in N trials. If this probability is smaller than the significance level (α=0.05) then the null hypothesis is rejected and the difference is considered significant [Siegel, 1956].

In IR this test is usually used as a one tailed test, that is, the alternative hypothesis prescribes the superiority of retrieval under condition *a* over condition *b* or vice versa. The use of the sign test raises a number of interesting points. The first of these is that it only assumes that the Z's are measured on an ordinal scale, that is, the magnitude of $Z_a - Z_b$ is not significant. This is a suitable feature since we are only seeking to find which strategy is better in an average sense and do not want to be unduly influenced by excellent retrieval performance on one query. The second point is that some care needs to be taken when comparing Z_a and Z_b. Because our measure of effectiveness can be calculated to infinite precision we may be insisting on a difference when if fact it only occurs in the tenth decimal place. It is therefore important to decide beforehand at what value of $\in$ we will equate Z_a and Z_b when $Z_a - Z_b \leq \in$.

Finally, although I have just explained the use of the sign test in terms of single number measures, it is also used to detect a significant difference between precision recall curves. We now interpret the Z's as precision values at a set of standard recall values. Let this set be SR={0.1, 0.2, …, 1}, then corresponding to each $R \in SR$ we have a pair ($P_a(R)$, $P_b(R)$). The P_a's and P_b's are now treated in the same way as the Z_a's and Z_b's. Note that when doing the evaluation in this way, the precision-recall values will have already been averaged ver the set of queries by one of the ways explained before.

6.4.3 Experiments

6.4.3.1 Input Data

The input data to be used in these experiments is required to have:

1. A collection of HTML documents, D;
2. A collection of HTML document queries, Q;
3. For each document query $q \in Q$ the the documents in D that are relevant to q.

Additionally the collection of documents and queries must be partitioned into a training set and an evaluation set.

The experiments in the current work used three different corpuses of input data:

Corpus from Yahoo

Yahoo[9] is a tree like hierarchy of document categories. Categories near the root of the hierarchy correspond to broad topics, such as Arts or Education, and categories near the bottom of the hierarchy correspond to more specific topics, such as History of Photojournalists or Preparation Centers for the GRE. Categories in Yahoo are manually built by user experts so they are an objective mean for evaluation purposes.

From Yahoo I selected 253 documents grouped into 14 categories.

I used the following categories for building the training set:

1. Biostatistics;
2. Boxing;
3. Education in Mathematics;
4. Finance;
5. Sex: activities and practices;

I used the following categories for building the testing set:

6. Vegetarianism;
7. Irish Studies;
8. Java;
9. Kung Fu;
10. Literature;
11. Megaliths;
12. Photography;
13. Triathlon;
14. Immunology;

From each category I chose documents playing the role of queries for the retrieval system. Then for each query I selected those documents belonging to the same category as its relevant documents. This is a reasonable approach since the selected categories are considerably orthogonal so only the documents belonging to the same category as the query are similar to it.

Corpus from Yahoo for cross-validation

This corpus is identical to the previous one except that here the training set becomes the evaluation set and the evaluation set becomes the training set.

Corpus from Browsing Sessions

I used "The Big Memory" for saving HTML documents while I was working with the Internet. Afterward I grouped these saved documents into different categories according to their topics. Finally from each category I chose several query documents and I set the documents belonging to that category as the documents relevant to the query.

For example, on Monday I asked "The Big Memory" to start saving documents. During that week I browsed documents about different topics: Java programming language, HTML, universities for doing a Ph.D., Lotus Notes, etc. and all these documents were saved by "The Big Memory" to my hard disk. On Friday I manually grouped the documents browsed during that week according to their topics building a group of categories. Finally from each category I selected several documents as query documents and I set the documents belonging to that category as relevant to the query documents. For the category of Java programming language, for example, I chose the document from http://www.javasoft.com as a query document and I

[9] http://www.yahoo.com

set all the documents belonging to the category Java programming language as relevant to http://www.javasoft.com.

In this way I build a collection of 174 documents manually grouped into 15 categories.

I used the following categories for building the training set:

1. GRE;
2. Inmigration;
3. Soccer;
4. Tango;
5. Universities Ph.D.

And I used the following categories for building the testing set:

6. Advertisements;
7. Games;
8. Information retrieval;
9. Javascript;
10. Net-Commerce;
11. Lotus Notes;
12. Recommendations Ph.D.;
13. Skying;
14. Swimming;
15. Web Browser Intelligence.

The corpus from the browsing sessions vs. the corpus from Yahoo

For building a category in the corpus from the browsing sessions I went to different sites and I navigated through different documents about a fixed topic. For example for building the PHD APPLICATION category I first went to the application information page of Stanford, then using a link I went to the financial aid information page at Stanford and finally to the research areas page also at Stanford. I repeated this process at Berkeley and MIT. All these documents made the PHD APPLICATION category of the corpus from the browsing sessions.

The way of building a category in the corpus from the browsing sessions shows that in this corpus related documents will probably share the same server. As the category has been built by browsing documents in several sites (using hypertext links) then a query and its related documents will probably be linked.

Documents about the same topic from the same site will probably share a common vocabulary. Due to the way of building the corpus from the browsing sessions many related documents will belong to the same site. So related documents in the corpus from the browsing sessions will share a common vocabulary.

Besides, Yahoo as a design principle does not include in a single category two documents from the same site. This means that, for example, in a category about Java programming we will not find two documents from http://www.javasoft.com.

This design principle has important implications. First a query and its relevant documents will belong to the same category then they will not belong to the same server. As documents in any category of the corpus from Yahoo do not belong to the same server then a query and its relevant documents will probably be less linked than in the corpus from the browsing sessions. Moreover, as a query and its relevant documents belong to different sites then they will probably use a slightly different vocabulary for describing their topic.

So we can compare the two corpuses along three dimensions:

1. Server: a query and its relevant documents in the corpus from the browsing sessions will probably belong to the same server while in the corpus from Yahoo they will belong to different servers.
2. Links: a query and its relevant document in the corpus from the browsing sessions will probably be densely linked while in the corpus from Yahoo they will probably be poorly linked.
3. Vocabulary: a query and its relevant documents will use a vocabulary more similar in the corpus from the browsing session than in the corpus from Yahoo.

The previous information is summarized in the following table.

	Yahoo	Browsing Sessions
Server	Not same server	Same server
Links	Poorly linked	Densely linked
Vocabulary	Not so common	Common

Table 1: Characteristics of a query and its relevant documents in the corpus from Yahoo vs. in the corpus from the browsing sessions

6.4.3.2 Design

The main objective of the following experiments is to show that the structure of HTML documents can help to find their similarity.

The purpose of experiment 1 is to compare the performance of a similarity measure containing only HTML criteria[10] – no exhaustive similarity measure – with the performance of the Dice Coefficient.

In experiment 2 I compare the performance of the Dice Coefficient to the performance of its enriched counterpart.

Experiment 3 repeats experiment 2 with a different training and evaluation set. It is a cross validation intended to show that the performance improvements of experiment 2 do not depend on particular characteristics of chosen training or evaluation set.

The enriched similarity measure could contain certain HTML criteria performing poorly and degrading the performance of the whole similarity measure. In such a case it would be better to discard these criteria, leading to a better similarity measure. I use experiment 4 to show that no criterion makes the enriched similarity measure perform worse.

In Figure 11 I show the dependency graph of all the experiments in this evaluation.

[10] The similarity measure will be a weighted average of the Titles, H1, H2, Links Text, Links and URLs criterion.

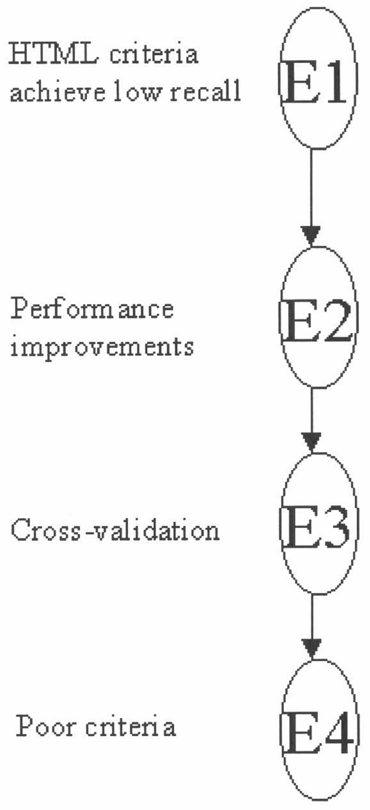


Figure 11: Dependency graph of experiments in this experience

6.4.3.3 Experiment 1

The objective of this experiment is to compare the performance of a similarity measure containing only HTML criteria – no exhaustive similarity measure – with the performance of the Dice Coefficient.

First I build a similarity measure containing all the HTML criteria. I train this similarity measure using the methodology described in section 6.4 and the documents and queries in the training set. Then I run a recall-precision evaluation using the documents and queries in the evaluation set, obtaining the average recall-precision curve for the similarity measure containing all the HTML criteria.

Next I run a recall-precision evaluation for the Dice Coefficient using the documents and queries in the evaluation set obtaining the average recall-precision curve for the Dice Coefficient.

Finally I compare the performance of the previous recall-precision curves and I draw some conclusions.

I repeat the previous experience with the corpus from Yahoo and with the corpus from the browsing sessions.

Corpus from Yahoo

After training the similarity measure containing all the HTML criteria (with the documents and queries in the training set of the corpus from Yahoo) and running a recall-precision evaluation (with the documents and queries in the evaluation set of the corpus from Yahoo) I obtained the recall-precision curve that I plot in Figure 12.

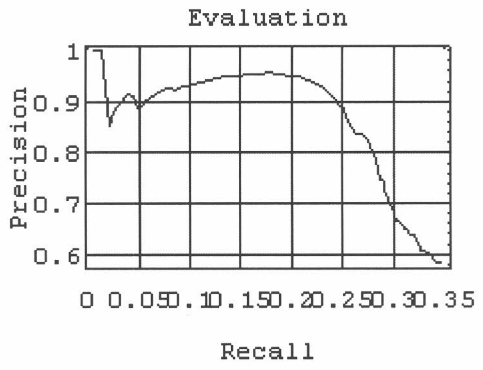


Figure 12: Average recall-precision curve for a similarity measure containing all the HTML criteria and using the corpus from Yahoo

After running a recall-precision evaluation of the Dice Coefficient (with the documents and queries in the evaluation set of the corpus from Yahoo) I obtained the average recall-precision curve that I plot in Figure 13.

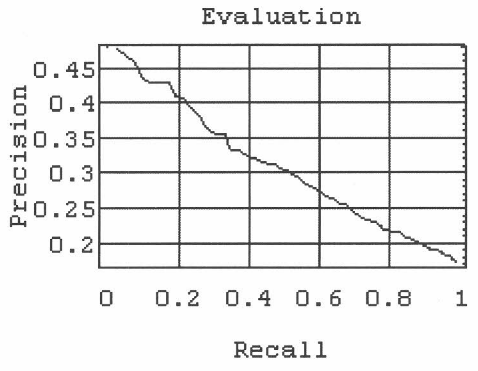


Figure 13: Average recall-precision curve for the Dice Coefficient using the corpus from Yahoo

In Figure 14 I plot the average recall-precision curves of both, the similarity measure containing all the HTML criteria and the Dice Coefficient.

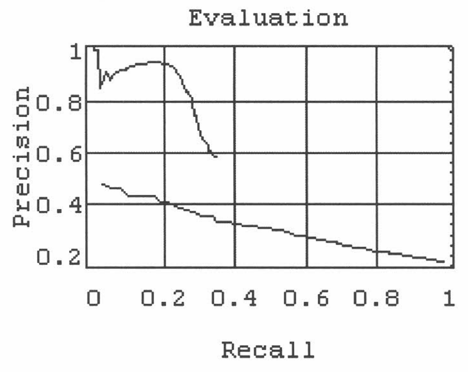


Figure 14: Average recall-precision curves for both, a similarity measure containing all the HTML criteria and the Dice Coefficient, using the corpus from Yahoo

Figure 14 shows that with the corpus from Yahoo the HTML criteria achieve low recall values compared to the Dice Coefficient and that for low recall values the HTML criteria are more precise than the Dice Coefficient.

Corpus from browsing sessions

After training the similarity measure containing all the HTML criteria (with the documents and queries in the training set of the corpus from the browsing sessions) and running a recall-precision evaluation (with the documents and queries in the evaluation set of the corpus from the browsing sessions) I obtained the recall-precision curve that I plot in Figure 15.

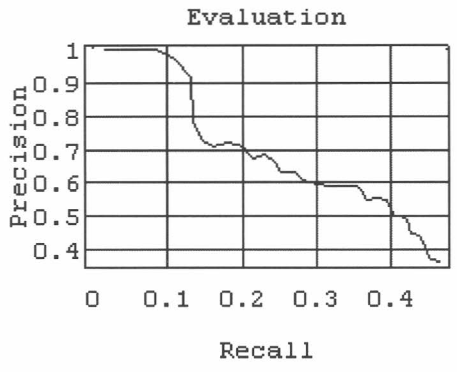


Figure 15: Average recall-precision curve for a similarity measure containing all the HTML criteria using the corpus from the browsing sessions

After running a recall-precision evaluation of the Dice Coefficient (with the documents and queries in the evaluation set of the corpus from the browsing sessions) I obtained the average recall-precision curve that I plot in Figure 16.

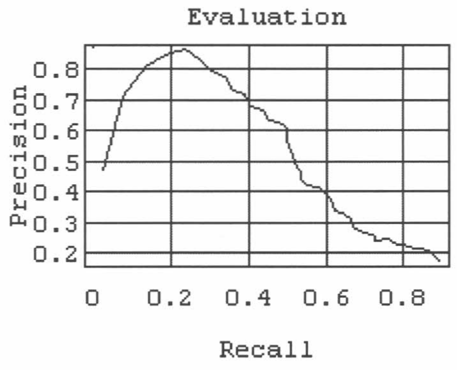


Figure 16: Average recall-precision curve for the Dice Coefficient using the corpus from the browsing sessions

In Figure 17 I plot the average recall-precision curves of both, the similarity measure containing all the HTML criteria and the Dice Coefficient.

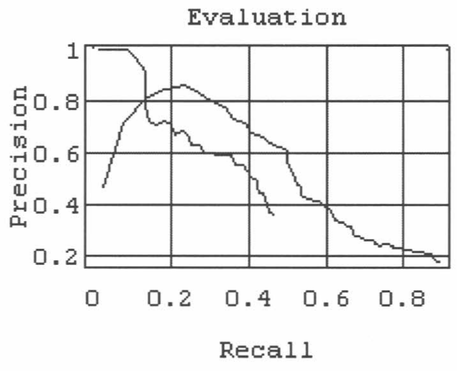


Figure 17: Average recall-precision curves for both, the similarity measure containing all the HTML criteria and the Dice Coefficient, using the corpus from the browsing sessions

Figure 17 shows that again in the corpus from the browsing sessions the HTML criteria achieve low recall values compared to the Dice Coefficient and that the HTML criteria are more precise than the Dice Coefficient for low recall values.

Conclusions from experiment 1

The low recall performance of a similarity measure containing only HTML criteria is certainly intuitive. The Titles Criterion, for example, will retrieve a document when it shares common words in the title field with the query document. The corpus will probably contain many relevant documents not sharing words in the title field with the query document. These documents will not be retrieved by the Title Criterion, thus the proportion of retrieved documents within the relevant documents will be low, producing a low recall value.

To explain the higher precision for low recall values of the HTML criteria compared to the Dice coefficient it is useful to remember from section 6.4.1 that the documents defining low recall values are ranked highly by the similarity measure. If a document is ranked highly by the HTML criteria then it will probably share with the query document many words in the title, H1 or H2 field, it could be linked to the query document, it could share words with the query document in the links or it could belong to the same server as the query document. Besides, if a document is ranked highly by the Dice Coefficient then it will share many words with the query document. So the relevance information for highly ranked documents is stronger for the HTML criteria than for the Dice Coefficient leading to higher precision values.

This experiment also shows that a similarity measure containing only HTML criteria will not achieve high recall values. So for achieving high recall values the HTML criteria should work as an enhancement of an exhaustive similarity measure. In the next experiments the HTML criteria will work as an enhancement of the Dice Coefficient.

6.4.3.4 Experiment 2

The objective of this experiment is to compare the performance of the Dice Coefficient with the performance of its enriched counterpart. The enriched similarity measure is composed of the Dice Coefficient and the following criteria:

1. Titles Criterion
2. H1 Criterion
3. H2 Criterion
4. Links Text Criterion
5. Links Criterion
6. URLs Criterion

I call this similarity measure WHOLE SIMILARITY MEASURE.

First I train the WHOLE SIMILARITY MEASURE using the documents and queries in the training set and I show the obtained weights. Then I run a recall-precision evaluation using the documents and queries in the evaluation set obtaining the average recall-precision curve for the WHOLE SIMILARITY MEASURE.

Next I run a recall-precision evaluation for the Dice Coefficient using the documents and queries in the evaluation set obtaining the average recall-precision curve for the Dice Coefficient.

Finally I compare the performance of the previous recall-precision curves and I use a statistical test to verify whether the performance difference is statistically significant or not.

I repeat the previous experience with the corpus from Yahoo and with the corpus from the browsing sessions.

Corpus from Yahoo

First I trained the WHOLE SIMILARITY MEASURE using the documents and queries in the training set from Yahoo obtaining the set of weights that I show in Table 2.

Criterion	Weight
Title	1140.03
Dice	183.0
H1	170.81
Links	24.85
H2	6.95
Links Text	-26.85
URLs	-684.83

Table 2: Weights found for the criteria in the WHOLE SIMILARITY MEASURE using the training set of the corpus from Yahoo

According to the weight values the Title, Dice and H1 criteria seem very important for the enriched similarity measure while the Links, H2 and Links Text criteria do not seem very important. A particular fact is the weight assigned to the URLs criterion, which is negative with a high absolute value. This can be explained by noting that Yahoo, as a design principle, does not allow two documents from the same server to belong to the same category. So the similarity measure learned that the URLs criterion identifies irrelevant documents in the corpus from Yahoo.

I run the recall-precision evaluation, using the methodology described in section 6.4.1 and the documents and queries in the evaluation set, obtaining the average recall-precision curve that I plot in Figure 18.

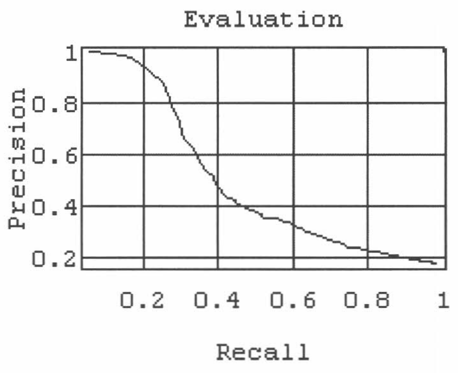


Figure 18: Average recall-precision curve for the WHOLE SIMILARITY MEASURE using the corpus from Yahoo

Next I evaluated the Dice Coefficient using the same evaluation set obtaining the average recall-precision curve that I plot in Figure 19.

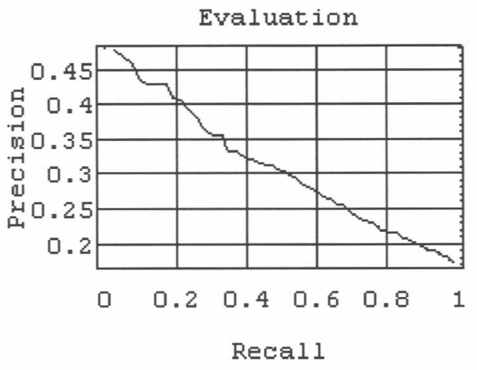


Figure 19: Average recall-precision curve for the Dice Coefficient using the corpus from Yahoo

In Figure 20 I plot the average recall-precision curves of both, the enriched similarity measure and the Dice Coefficient.

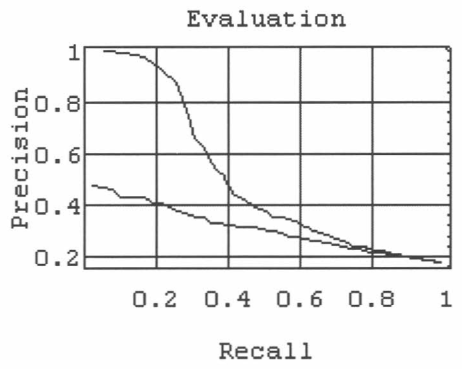


Figure 20: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the Dice Coefficient, using the corpus from Yahoo

Although it seems evident that the enriched similarity measure outperforms the Dice Coefficient I perform the sign test over the curves plotted in Figure 20. I use the standard recall values illustrated in the column labeled 'Recall' and I consider two precision values identical if [Precision of Multi-Criterion] – [Precision of Dice Coefficient] ≤ 0.05.

Recall	Precision of Multi-Criterion	Sign	Precision of Dice Coefficient
0.05	1	+	0.47
0.1	0.99	+	0.43
0.15	0.97	+	0.43
0,2	0.92	+	0.4
0.25	0.87	+	0.38
0.3	0.66	+	0.35
0.35	0.56	+	0.33
0.4	0.46	+	0.32
0.45	0.41	+	0.31
0.5	0.37	+	0.3
0.55	0.35	+	0.29

0.6	0.32	+	0.27
0.65	0.29		0.26
0.7	0.27		0.24
0.75	0.24		0.23
0.8	0.22		0.22

The probability that a binomial distributed variable (P=Q=0.5) shows 0 successful outcomes in 12 trials is not significant at 3 decimal places, so the sign test gives enough evidence that the enriched similarity measure outperforms the Dice Coefficient.

As the training set and evaluation set do not have documents in common, the latest evaluation also shows that the trained enriched similarity measure will perform well with novel documents.

This experiment validates the hipothesis that HTML features can help to determine the similarity between HTML documents.

Corpus from browsing sessions

Again I trained the WHOLE SIMILARITY MEASURE obtaining the set of weights that I show in Table 3.

Criterion	Weight
Title	95.0
Links	94.35
H1	76.47
H2	71.62
URLs	65.45
Dice	57.55
Links Text	10.99

Table 3: Weights found for the criteria in the WHOLE SIMILARITY MEASURE using the training set of the corpus from the browsing sessions

An important characteristic about these weights is that they are significantly different from the ones in the corpus from Yahoo. In Yahoo the weights of the different criteria were notably dissimilar, ranging from –684.83 for the URLs Criterion to 1140.03 for the Title Criterion. In the corpus from the browsing sessions the weights of the different criteria are more homogeneous, ranging from 10.99 for the Links Text Criterion to 95.0 for the Title Criterion. Moreover the relative order of the criteria according to their weights is also different.

I run the recall-precision evaluation obtaining the average recall-precision curve that I plot in Figure 21.

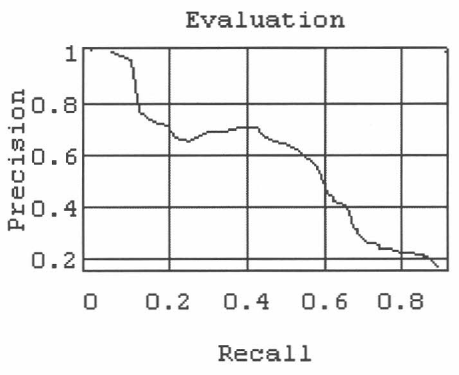


Figure 21: Average recall-precision curve for the WHOLE SIMILARITY MEASURE using the corpus from the browsing sessions

Next I evaluated the Dice Coefficient using the same evaluation set obtaining the average recall-precision curve that I plot in Figure 22.

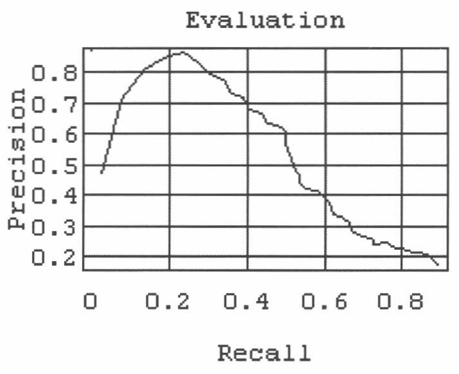


Figure 22: Average recall-precision curve for the Dice Coefficient using the corpus from the browsing sessions

The Dice Coefficient will perform better when a query and its related documents use a common vocabulary. As explained in section "6.4.3.1 Input Data" a query and its related document in the corpus from the browsing sessions use a vocabulary that is more common than related documents in the corpus from Yahoo. Thus, the performance of the Dice Coefficient is expected to be better in the corpus from the browsing sessions than in the corpus from Yahoo. This explanation is validated in the previous experiments where Figure 22 shows that the Dice Coefficient performs better than with the corpus from the browsing sessions than with the corpus from Yahoo as illustrated in Figure 19.

In Figure 23 I plot the average recall-precision curves of both, the enriched similarity measure and the Dice Coefficient.

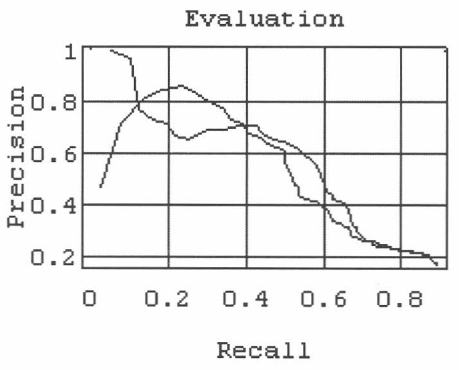


Figure 23: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the Dice Coefficient, using the corpus from the browsing sessions

For high precision values, which means high similarity between queries and retrieved documents, the enriched similarity measure is more precise than the Dice Coefficient. For recall values ranging from 0.14 to 0.39 the Dice Coefficient outperforms the enriched similarity measure and for high recall values the enriched similarity measure slightly outperforms the Dice Coefficient.

Significance test

Again I perform the sign test for determining whether the performance differences are significant or not. I use the standard recall values illustrated in the column labeled 'Recall' and I consider two precision values identical if [Precision of Multi-Criterion] – [Precision of Dice Coefficient] ≤ 0.05.

Recall	Precision of Multi-Criterion	Sign	Precision of Dice Coefficient
0.05	1	+	0.47
0.1	0.97	+	0.8
0.15	0.73	-	0.84
0,2	0.72	-	0.86
0.25	0.66	-	0.83
0.3	0.69	-	0.8
0.35	0.69		0.73
0.4	0.7		0.68
0.45	0.68	+	0.63
0.5	0.64	+	0.57
0.55	0.59	+	0.42
0.6	0.48	+	0.38
0.65	0.4	+	0.32
0.7	0.27		0.26
0.75	0.23		0.24
0.8	0.22		0.22

The probability that a binomial distributed variable (P=Q=0.5) shows 4 succesful outcomes in 11 trials is 0.274 > α=0.05, so the sign test does not give enough evidence that the enriched similarity measure outperfroms the Dice Coefficient for the corpus from browsing sessions.

Altough the performance improvements due to the HTML criteria are not statistically significant, Figure 23 shows that the HTML criteria do not degrade the performance of the exhaustive similarity measure.

Conclusions from experiment 2

This experiment shows that when the exhaustive similarity measure performs poorly, as in the corpus from Yahoo, the performance improvement due to the HTML criteria is statistically significant. Besides when the exhaustive similarity measure performs well, as in the corpus from the browsing sessions, although the performance improvements due to the

HTML criteria are not statistically significant, the HTML criteria do not degrade the performance of the exhaustive similarity measure.

This experiment also shows that the set of optimal weights significantly change as the characteristics of the document collection used for training the similarity measure change. When, for example, the training collection contains similar documents from the same server, the URLs criterion is weighted heavier. This shows that the similarity measure is learning from the training collection those important features of the corpus to be considered when finding the similarity between HTML documents.

6.4.3.5 Experiment 3

In this experiment I want to verify that the improvement in performance shown in the previous experiment for the corpus from Yahoo[11] does not depend on particular characteristics of the chosen training or evaluation set. So the next evaluations use the corpus from Yahoo for cross-validation, that has a different training set and a different evaluation set than the corpus from Yahoo.

First I trained the WHOLE SIMILARITY MEASURE using the documents and queries in the training set of the corpus from Yahoo for cross-validation. Then I run a recall-precision evaluation using the documents and queries in the evaluation set, obtaining the average recall-precision curve that I plot in Figure 24.

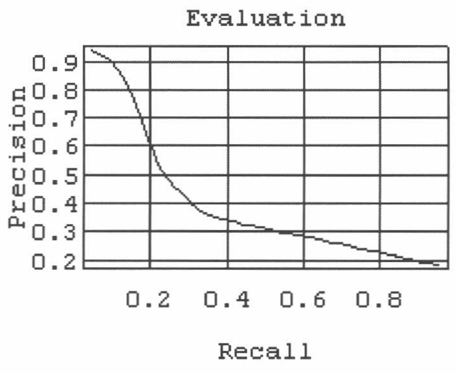


Figure 24: Average recall-precision curve for the WHOLE SIMILARITY MEASURE using the corpus from Yahoo for cross-validation

Next I evaluated the Dice Coefficient using the same evaluation set obtaining the average recall-precision curve that I plot in Figure 25.

[11] I do not use the corpus from the browsing sessions because in this corpus the performance improvements are not statistically significant.

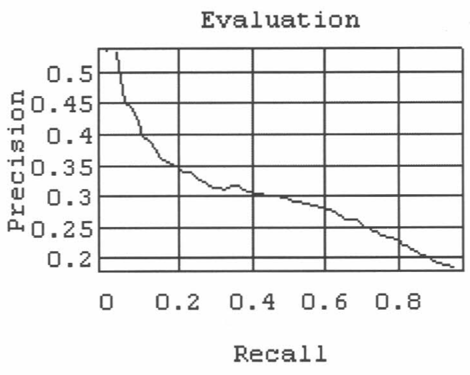


Figure 25: Average recall-precision curve for the Dice Coefficient using the corpus from Yahoo for cross-validation

In Figure 26 I plot the average recall-precision curves of both, the enriched similarity measure and the Dice Coefficient.

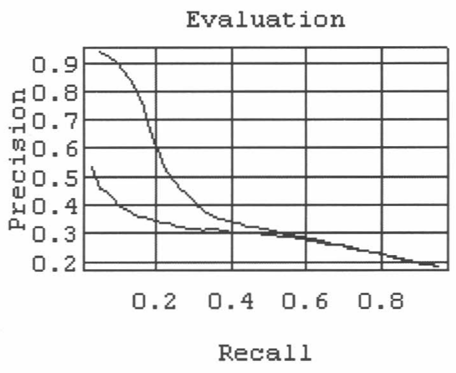


Figure 26: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the Dice Coefficient, using the corpus from Yahoo for cross-validation

Figure 26 shows that once again the WHOLE SIMILARITY MEASURE clearly outperforms the Dice coefficient. So we can conclude that the advantage of the WHOLE SIMILARITY MEASURE in the corpus from Yahoo does not depend on the particular choice for the training or evaluation set.

6.4.3.6 Experiment 4

It could happen that the inclusion of a new HTML criterion into the enriched similarity measure makes it get confuse and not only does not improve its performance but makes it get worse. This could happen if we assign a high weight to a criterion that retrieves more irrelevant documents than relevant ones.

The first objective of this experiment is to show that no criterion makes the enriched similarity measure perform worse.

Within a corpus of documents a criterion will be labeled important when its presence in the enriched similarity measure improves its performance as measured by the recall-precision evaluation.

Then if we want to build a fast and simple similarity measure we could just discard those unimportant criteria and be sure that the performance of the similarity measure will not change.

The second objective of this experiment is to find the importance of the different criteria in the enriched similarity measure for the corpus from Yahoo.

For each criterion, C_i, I build a enriched similarity measure, M_i, by removing C_i from the WHOLE SIMILARITY MEASURE and I compare the performance of M_i to the performance of the WHOLE SIMILARITY MEASURE. If the performance is different the criterion C_i is labeled important.

I repeat the previous experience with the corpus from Yahoo and the corpus from the browsing sessions.

Corpus from Yahoo

I first built a similarity measure by removing the Titles criterion from the WHOLE SIMILARITY MEASURE. I trained the new similarity measure using the queries and documents in the training set. I run a recall-precision evaluation using the documents and queries in the evaluation set obtaining the average recall-precision curve, which I plot in Figure 27.

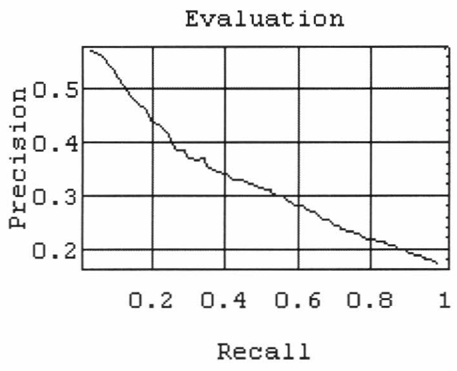


Figure 27: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the Title Criterion using the corpus from Yahoo

In Figure 28 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

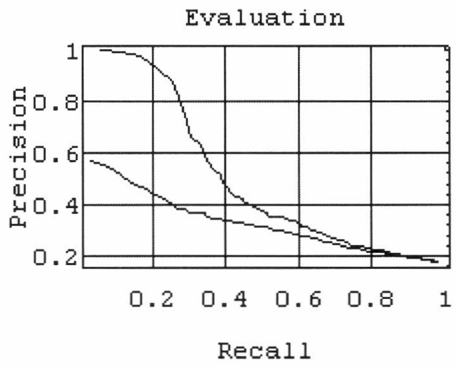


Figure 28: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Title Criterion, using the corpus from Yahoo

Figure 28 shows that the presence of the Title Criterion significantly changes the performance of the WHOLE SIMILARITY MEASURE so the Title Criterion is labeled important.

Next I built a similarity measure by removing the URLs criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 29.

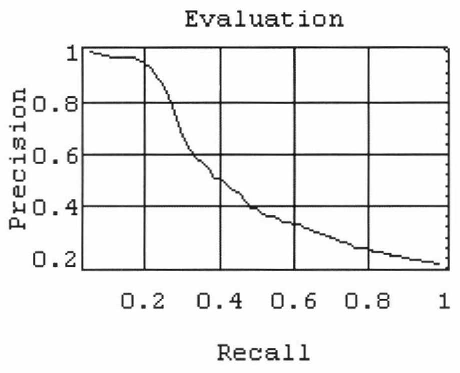


Figure 29: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the URLs Criterion using the corpus from Yahoo

In Figure 30 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

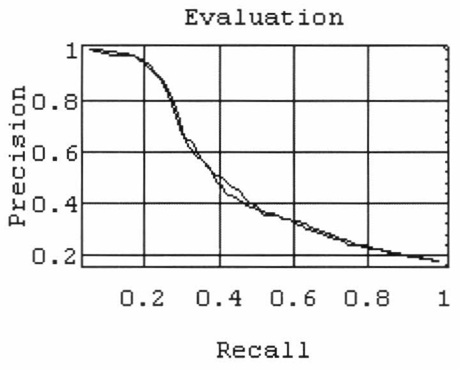


Figure 30: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the URLs Criterion, using the corpus from Yahoo

Figure 30 shows that the presence of the URLs Criterion does not change significantly the performance of the WHOLE SIMILARITY MEASURE so the URLs Criterion is labeled unimportant.

Next I built a similarity measure by removing the Links criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 31.

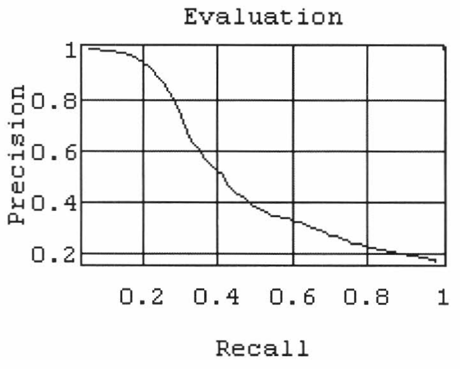


Figure 31: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the Links Criterion using the corpus from Yahoo

In Figure 32 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

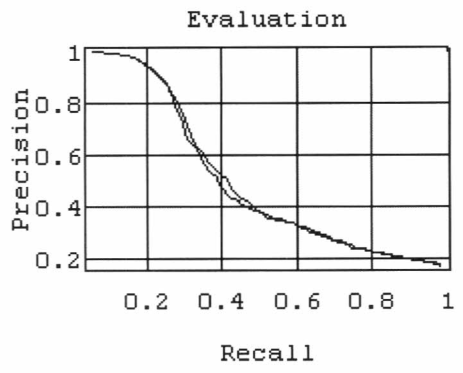


Figure 32: Average recall-precision curve for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Links Criterion, using the corpus from Yahoo

Figure 32 shows that the presence of the Links Criterion does not change significantly the performance of the WHOLE SIMILARITY MEASURE so the Links Criterion is labeled unimportant.

Next I built a similarity measure by removing the Links Text criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 33.

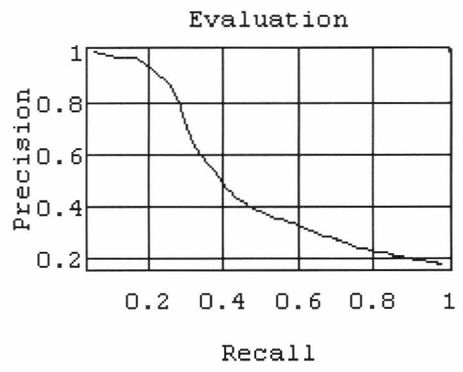


Figure 33: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the Links Text Criterion using the corpus from Yahoo

In Figure 34 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

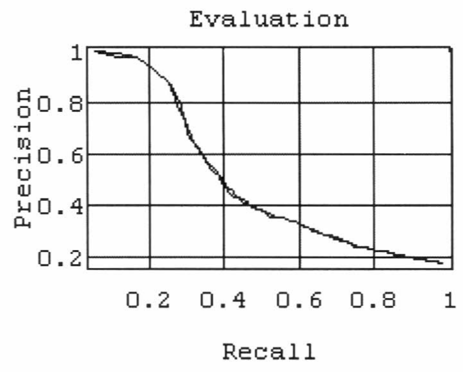


Figure 34: Average recall-precision curve for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Links Text Criterion, using the corpus from Yahoo

Figure 34 shows that the presence of the Links Text Criterion does not change significantly the performance of the WHOLE SIMILARITY MEASURE so the Links Text Criterion is labeled unimportant.

Next I built a similarity measure by removing the H1 criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 35.

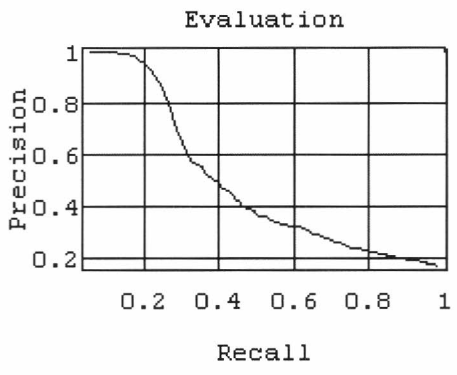


Figure 35: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the H1 Criterion using the corpus from Yahoo

In Figure 36 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

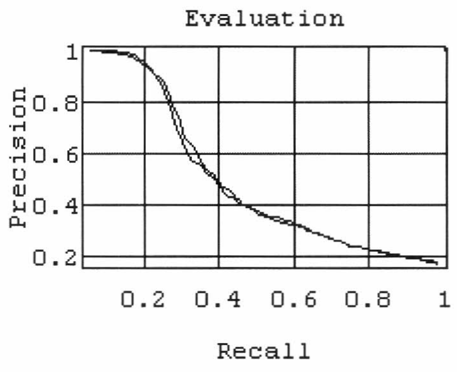


Figure 36: Average recall-precision curve for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the H1 Criterion, using the corpus from Yahoo

Figure 36 shows that the presence of the H1 Criterion does not change significantly the performance of the WHOLE SIMILARITY MEASURE so the H1 Criterion is labeled unimportant.

Next I built a similarity measure by removing the H2 criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 37.

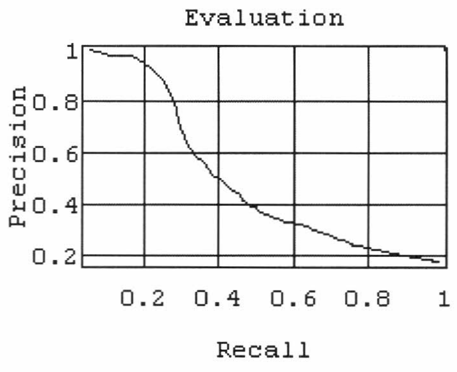


Figure 37: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the H2 Criterion using the corpus from Yahoo

In Figure 38 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

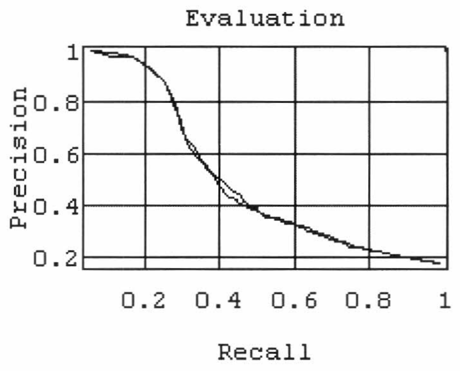


Figure 38: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Title Criterion, using the corpus from Yahoo

Figure 38 shows that the presence of the H2 Criterion does not change significantly the performance of the WHOLE SIMILARITY MEASURE so the H2 Criterion is labeled unimportant.

So the Title Criterion is the only criterion labeled important for the corpus from Yahoo and the remaining criteria are labeled unimportant. In any case in the corpus from Yahoo no criterion makes the performance of the WHOLE SIMILARITY MEASURE perform worse.

Finally I built a similarity measure containing the Dice Coefficient and the Title Criterion, the only important criterion for the corpus from Yahoo. I trained and evaluated this last similarity measure obtaining the average recall-precision curve that I plot in Figure 39.

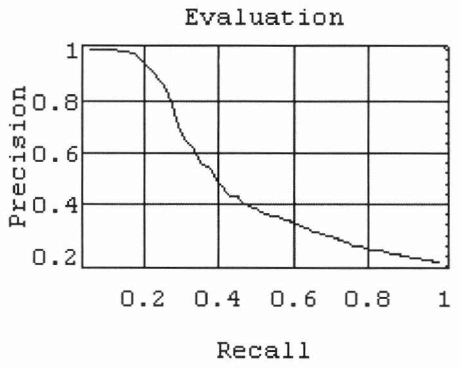


Figure 39: Average recall-precision curve for a similarity measure containing the Dice Coefficient and the Title Criterion using the corpus from Yahoo

In Figure 40 I plost the average recall-precision curve from both the previous similarity measure with the WHOLE SIMILARITY MEASURE.

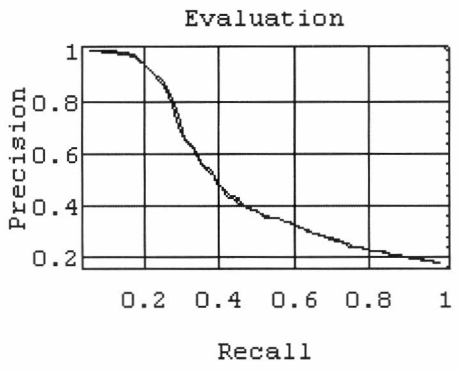


Figure 40: Average recall-precision curves for both, a similarity measure containing the Dice Coefficient and the Title Criterion and the WHOLE SIMILARITY MEASURE, using the corpus from Yahoo

Figure 40 shows that in the corpus from Yahoo we can build a new enriched similarity measure by discarding unimportant criteria from the WHOLE SIMILARITY MEASURE and that the new similarity measure will perform similarly than the WHOLE SIMILARITY MEASURE but it will be faster and simpler. Also the previous evaluations show that no criterion makes the WHOLE SIMILARITY MEASURE perform worse in the corpus from Yahoo.

Corpus from browsing sessions

I first built a similarity measure by removing the Titles criterion from the WHOLE SIMILARITY MEASURE. I trained the new similarity measure and I run a recall-precision evaluation obtaining the average recall-precision curve, which I plot in Figure 37.

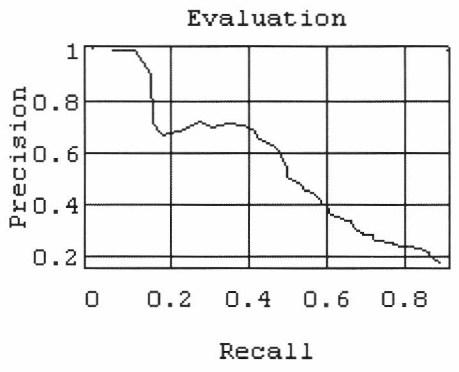


Figure 41: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the Title Criterion using the corpus from browsing sessions

In Figure 42 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

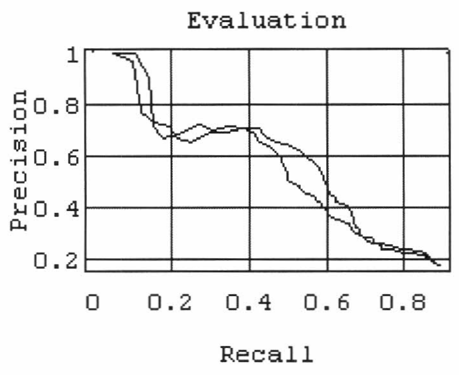


Figure 42: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Title Criterion, using the corpus from browsing sessions

Figure 42 shows that for high precision values it would be better to discard the Title Criterion but for high recall values it would be better to consider it. So, if we were seeking a similarity measure archiving high precision we would label the Title Criterion unimportant but if we are seeking a similarity measure archiving high recall we would label the Title Criterion important.

Next I built a similarity measure by removing the URLs criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 43.

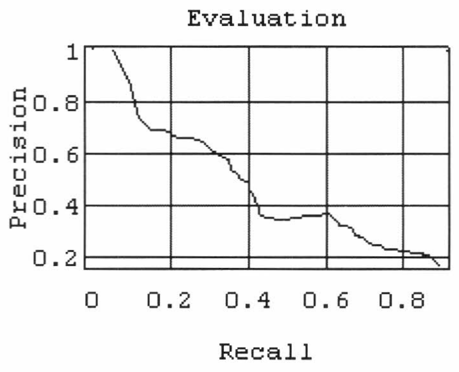


Figure 43: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the URLs Criterion using the corpus from browsing sessions

In Figure 44 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

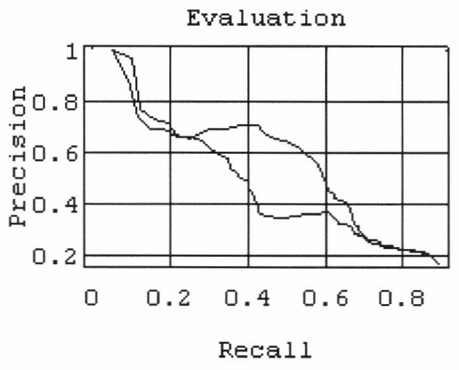


Figure 44: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the URLs Criterion, using the corpus from browsing sessions

Figure 44 shows that the presence of the URLs Criterion changes the performance of the WHOLE SIMILARITY MEASURE, specially for middle and low recall values, so the URLs Criterion is labeled important.

Next I built a similarity measure by removing the Links criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 45.

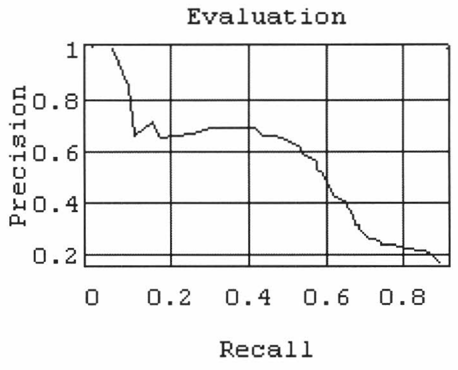


Figure 45: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the Links Criterion using the corpus from browsing sessions

In Figure 46 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

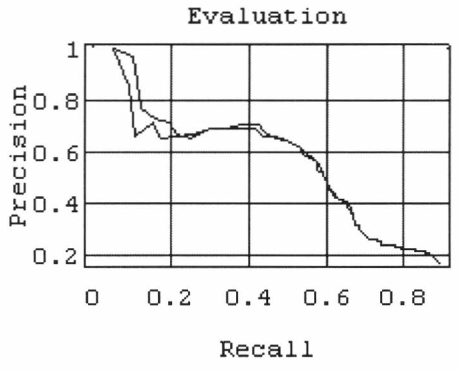


Figure 46: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Links Criterion, using the corpus from browsing sessions

Figure 46 shows that the presence of the Links Criterion changes the performance of the WHOLE SIMILARITY MEASURE for high precision values so the Links Criterion is labeled important.

Next I built a similarity measure by removing the Links Text criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 47.

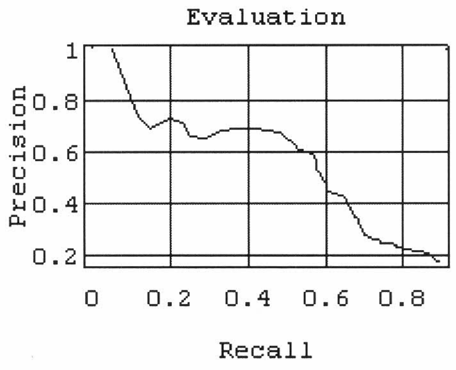


Figure 47: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the Links Text Criterion using the corpus from browsing sessions

In Figure 48 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

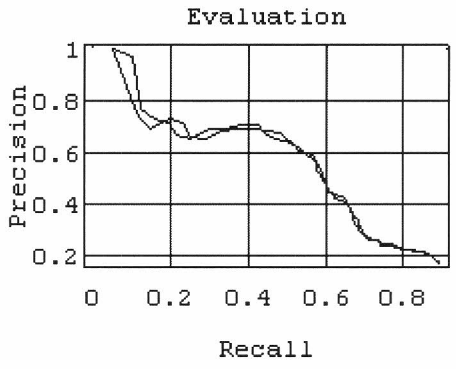


Figure 48: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the Links Text Criterion, using the corpus from browsing sessions

Figure 48 shows that the presence of the Links Text Criterion changes slightly the performance of the WHOLE SIMILARITY MEASURE for high precision values so the Links Text Criterion is labeled important.

Next I built a similarity measure by removing the H1 criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 49.

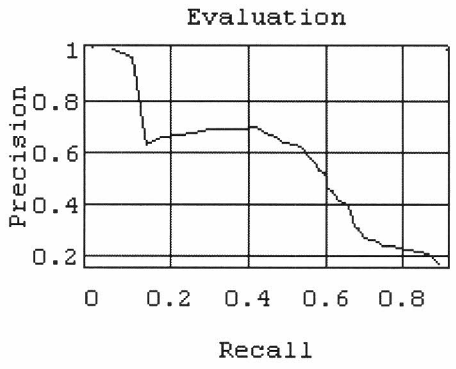


Figure 49: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the H1 Criterion using the corpus from browsing sessions

In Figure 50 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

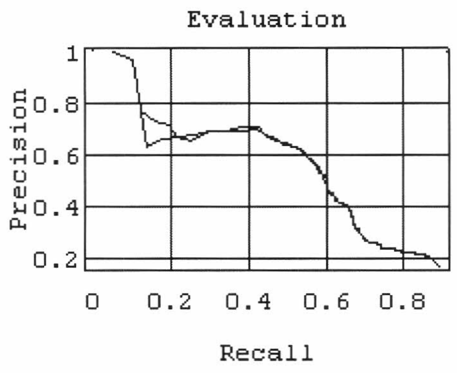


Figure 50: Average recall-precision curve for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the H1 Criterion, using the corpus from browsing sessions

Figure 50 shows that the presence of the H1 Criterion changes slightly the performance of the WHOLE SIMILARITY MEASURE so the H1 Criterion is labeled important.

Next I built a similarity measure by removing the H2 criterion from the WHOLE SIMILARITY MEASURE. I trained and evaluated the similarity measure obtaining the average recall-precision curve, which I plot in Figure 51.

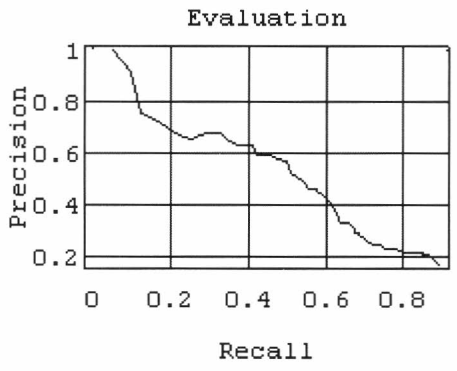


Figure 51: Average recall-precision curve for the WHOLE SIMILARITY MEASURE without the H2 Criterion using the corpus from browsing sessions

In Figure 52 I plot the average recall-precision curves from the evaluation of the previous similarity measure and from the evaluation of the WHOLE SIMILARITY MEASURE.

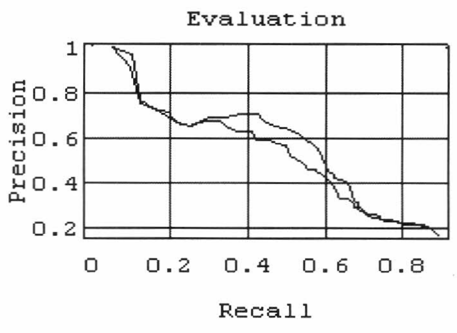


Figure 52: Average recall-precision curves for both, the WHOLE SIMILARITY MEASURE and the WHOLE SIMILARITY MEASURE without the H2 Criterion, using the corpus from browsing sessions

Figure 52 shows that the presence of the H2 Criterion changes the performance of the WHOLE SIMILARITY MEASURE, specially for middle and high recall values, so the H2 Criterion is labeled important.

This experiment shows that if we were seeking a similarity measure archiving high precision we would label all the criteria but the Title Criterion important and if we were seeking a similarity measure archiving high recall we would label all the criteria important. In any case no criterion makes the WHOLE SIMILARITY MEASURE perform worse.

This experiment also shows that the set of important criteria for the enriched similarity measure depends on the characteristics of the corpus used for the training and evaluating.

Conclusions from experiment 4

This experiment shows that no HTML criterion degrades the performance of the whole similarity measure. This result is extremely important since it shows that HTML criteria can be added to the base criterion. In some occasions this addition will result in a performance improvement, in some others it will not make any difference but it will never make the base criterion perform worse.

I develop a mechanism for finding unimportant criteria in such a way that if we want to build a fast and simple similarity measure we could just discard those unimportant criteria and be sure that the performance of the similarity measure will not change.

6.4.4 Conclusions

I showed that a similarity measure containing only HTML criteria achieves low recall values and that the HTML criteria should work as an enhancement of an exhaustive similarity measure.

When the exhaustive similarity measure performs badly, as in the corpus from Yahoo, the performance improvement, due to the HTML structure, is statistically significant. Besides, when the exhaustive similarity measure performs well, as in the corpus from the browsing sessions, although the performance improvement is not statistically significant, the addition of HTML criteria does not degrade the performance of the exhaustive similarity measure.

It could happen that the inclusion of a new HTML criterion into the enriched similarity measure makes it get confuse and not only does not improve its performance but makes it get worse. So I showed that the inclusion of no HTML criterion degrades the performance of the enriched similarity measure. This result is extremely important since it shows that HTML criteria can be added to the base criterion. In some occasions this addition will result in a performance improvement, in some others it will not make any difference but it will never make the base criterion perform worse. As the computational cost involved in using the HTML structure is considerably low, the inclusion of HTML criteria in a similarity measure is recommended.

I developed a mechanism that, given a collection of documents, finds those unimportant HTML criteria for the enriched similarity measure and for the given collection of documents. Discarding unimportant criteria from an enriched similarity measure leads to a new similarity measure that performs as well as the original one but that is simpler and faster.

The set of optimal weights significantly change as the characteristics of the document collection used for training the similarity measure change. When, for example, the training collection contains similar documents from the same server, the URLs criterion is weighted heavier. This shows that the similarity measure is learning from the training collection those important features of the corpus to be considered when finding the similarity between HTML documents.

More experiments with new document collections are suggested for correcting or increasing the confidence and validity on the previous results.

7 Future work

The following items seem interesting directions for future research:

1. Install The Big Memory in a big organization environment. In particular the usage of The Big Memory will show the important advantages of reusing information in the big organization. In general it will show the importance of using agent technology for working with the Web;
2. Implement a classification algorithm for finding relevant clusters for a given query document. This algorithm could be used by The Big Memory for finding clusters similar to the currently browsed document instead of the currently implemented strategy;
3. Evaluate the performance of the enriched similarity measure with new document collections. New evaluations will help to increase the confidence on the previous results or they will discover new characteristics about the enriched similarity measure;
4. Try more sophisticated techniques, such as Latent Semantic Indexing or Indexing Based on Phrases, in combination with HTML criteria and see whether it improves the performance of the retrieval system.

8 Summary

The current work contains three sections:

On the first section I worked with a new topic in computer science, namely agents that personalize the work with the World Wide Web. Using WBI, an agent architecture, I developed The Big Memory, an application that helps to reuse information in a big organization environment.

A survey is a fundamental part of many research projects. In the second section I describe the survey I have done on available strategies for document clustering.

In the third section appears the research work. I found a way to incorporate the structured information of HTML documents to standard document similarity measures and I show that this information can help to find the similarity between HTML documents.

9 References

[Barrett et al., 1997]	Barrett, R.; Maglio, P.P. and Kellem, D.C. How to personalize the Web. Human factors in Computing Systems (CHI '97). New York, NY: ACM Press, 1997.
[Bartell et al., 1995]	Bartell, B.T.; Cottrell, G.W. and Belew, R.W. Optimizing Similarity Using Multi/Query Relevance Feedback. University of California, San Diego, 1995.
[Berners-Lee et al., 1996]	Berners-Lee, T.; Fielding, R. and Frystyk, H. Hypertext transfer protocol: HTTP/1.0. Tech.Rep.RFC 1945, MIT, May 1996.
[Berry, 1992]	Berry, M.W. Large-scale sparse singular value computations. The International Journal of Supercomputer Applications, 6(1):13-49, 1992.
[Booch, 1994]	Booch, Grady. Object-Oriented Analysis and Design with Applications 2nd ed. The Benjamin/Cummings Publishing Company, 1994.
[Borg and Lingoes, 1987]	Borg, I. and Lingoes, J. Multidimensional Similarity Structure Analysis. Springer/Verlag, New York, 1987.
[Cormack, 1971]	Cormack, R.M., A review of classification, Journal of the Royal Statistical Society, Series A, 134, 321-353, 1971.
[Croft et al., 1991]	Croft, W.B.; Turtle, R. and Lewis, D.D. The use of phrases and structured queries in information retrieval. Proc. Of ACM SIGIR, 32-45, 1991.
[Cutting et al., 1992]	Cutting, D.R.; Karger, D.R.; Pederson, J.O.; and Tukey, J.W. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections, ACM SIGIR, 1992.
[Deerwester et al., 1990]	Deerweser, S.; Dumais, S.T.; Furnas, G.W.; Landauer, T.K. and Harshman, R. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6):391-407, 1990.
[Duda and Hart, 1973]	Duda, R.O. and Hart, E. Pattern Classification and Scene Analysis. Wiley, New York, 1973.
[Edelbrock, 1979]	Edelbrock, C. Mixture model tests of hierarchical clustering algorithms: the problem of classifying everybody. Multivariate Behavioral Research 14:367-384, 1979.
[Faber, 1994]	Faber, V. Clustering and the Continuous k-Means Algorithm. http://lib-www.lanl.gov/pubs/la-sci/clustering-and-kmeans.html, 1994.
[Faloutsos and Oard, 1994]	Faloutsos, C. and Oard, D. A Survey of Information Retrieval and Filtering Methods. University of Maryland, 1994.
[Fuhr and Buckley, 1991]	Fuhr, N. and Buckley, C. A probabilistic learning approach for document indexing. ACM Transactions on Information Systems, 9(3):223-248, 1991.
[Gamma et al., 1994]	Gamma, E.; Helm, R.; Johnson, R. and Vlissides, J. Design Patterns: elements of reusable object-oriented software. Addison-

	Wesley, 1994.
[Goodman and Kruskal, 1954]	Goodman, L. and Kruskal, W. Measures of association for cross classifications, Journal of the American Statistical Association, 49, 732-764, 1954.
[Gordon, 1981]	Gordon, A.D. Classification. London: Chapman and Hall, 1981.
[Gower and Ross, 1969]	Gower, J.C. and Ross, G.J.S. Minimum spanning trees and sigle linkage cluster analysis. Applied Statistics 18:54-64, 1969.
[Guttman, 1978]	Guttman, L. What is not what in statistics. The Statistician, 26:81-107, 1978.
[Hearst and Peterson, 1996]	Hearst, M.A. and Pedersen, J.O. Reexamining the cluster hypothesis. In Proceeding of SIGIR '96, 76-84, 1996.
[Jarvis and Patrick, 1973]	Jarvis, R.A. and Patrick, E.A. Clustering using a similarity measure based on shared nearest neighbors. IEEE Transactions on Computers C-22:1025-1034, 1973.
[Koller and Sahami, 1997]	Koller, Daphne; Sahami, Mehran. Hierarchical classifying documents using very few words. Stanford University, 1997.
[Kruskal, 1964]	Kruskal, J.B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, 29(1):1-27, 1964.
[Kuhns, 1965]	Kuhns, J.L., The continuum of coefficients of association. In Statistical Association Methods for Mechanized Documentation, (Edited by Steven et al.) National Bureau of Standards, Washington, 33-39, 1965.
[Lee, 1981]	Lee, R.C.T. Clustering analysis and its applications. Advances in Information Systems Science 8:169-292,1981.
[Pedersen et al., 1991]	Pedersen, J.O.; Cutting, D.R. and Tukey J.W. Snippet search: a single phase approach to text access. In Proceedings of the 1991 Joint Statistical Meetings. American Statistical Asociation, 1991. Also available as Xerox PARC technical report SSL-91-08.
[Press et al., 1988]	Press, W.H.; Flannery, B.P.; Teukolkky, S.A. and Veterling, W.T. Numerical Recipes in C: The Art of Scientific Computing. Cambridge Univeristy Press, 1988.
[Rosenblatt, 1962]	Rosenblatt, F. Principle of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington, D.C., 1962.
[Salton and McGill, 1983]	Salton, G. and McGill, M.J. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.
[Schütze and Silverstein, 1997]	Schütze, H and Silverstein, C. Projections for Efficient Document Clustering. SIGIR 97, 74-81, 1997.
[Shepard, 1962]	Shepard, R.N. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. Psychometrika, 27(2):125-139, 1962.

[Siegel, 1956]	Siegel, S. Nonparametric Statistics for the behavioral sciences. McGraw-Hill. Book Company Inc, 1956.
[Sneath and Sokal, 1973]	Sneath, P.A. and Sokal, R.R. Numerical Taxonomy: The Principles and Practice of Numerical Classification, W.H. Freeman and Company, San Francisco, 1973.
[van Rijsbergen et al., 1981]	van Rijsbergen, C.J. Harper, D.J. and Porter, M.F. The selection of good search terms. Information Processing and Management 17:77-91, 1981.
[van Rijsbergen, 1977]	van Rijsbergen, C.J. A theoretical basis for the use of co-occurrence data in information retrieval. Journal of Documentation 33:106-119, 1977.
[van Rijsbergen, 1979]	van Rijsbergen, C.J. Information Retrieval. Butterworths, London, England, 1979, 2nd edition.
[Voorhees, 1986]	Voorhees, E. M., Implementing Agglomerative Hierarchic Clustering Algorithms for Use in Document Retrieval, Information Processing & Management, Vol. 22 No. 6, 465-476, 1986.
[Wishart, 1969]	Wishart, D. Mode analysis, a generalization of nearest neighbor which reduces chaining, in Numerical Taxonomy. (Edited by A.J. Cole) London: Academic Press, 1969.
[Wishart, 1978]	Wishart, D. CLUSTAN IC User Manual. Edinburgh; Edinburgh University Program Library Unit, 1978.
[Wong and Yao, 1990]	Wong, S.K.M. and Yao, Y.Y. Query formulation in linear retrieval models. Journal of the American Society for Information Retrieval, 41(5): 334-341, 1990.
[Wong et al., 1993]	Wong, S.K.M.; Cai, Y.J. and Yao, Y.Y. Computation of term associations by a neural network. In Proceedings of the ACM SIGIR, Pittsburgh, PA, 1993.

Appendix A: Comments on Object-Oriented methodology

In the design and implementation of the current work I have used object-oriented methodology. Although discovering a good object oriented design is not an easy task and is time consuming I believe object oriented methodology is a valuable methodology in research projects which involve software development.

While developing software in a research project requirements are changing continuously. Some changes I faced during this projects were:

1. Initially I was using files to store the indexing information of the HTML documents. Afterwards I realized that for building any significant evaluation I would need to use database support;
2. Documents are pre-processed before being inserted in the document collection: words are indexed, weighted and filtered by a stop list and a stemming algorithm. The way in which this pre-processing is done changed several times during the development process;
3. Criteria were added, removed and modified.

According to [Booch, 1994]:

"... large systems tend to evolve over time, a condition that is often incorrectly labeled software maintenance. To be more precise, it is maintenance when we correct errors; it is evolution when we respond to changing requirements; it is preservation when we continue to use extraordinary means to keep an ancient and decaying piece of software in operation ..."

The development of software for a research project is an ever evolving process because based on a initial idea you develop the initial implementation. Now based on the results of the implementation new ideas appear and the software needs to evolve. Because process is repeated continuously an object oriented methodology is becomes important in a research project. As [Booch, 1994] notes:

" *...we find that the process that leads to the successful construction of object-oriented architectures tends to be both iterative and incremental. Such a process is iterative in the sense that it involves the succesive refinement of an object-oriented architecture, from which we apply the experience and results of each release to the next iteration of analysis and design..."*

When developing software for a research project it is very important to build software that can evolve easily because when researchers think of new ideas they do not need to be concerned with the difficulty of implementing the new idea in the software program. This means an important degree of freedom for researchers and allows more ideas to be tested during the research project.

Design patterns

[Gamma et al., 1994] introduce their book of designing patterns as follows:

"*Designing object-oriented software is hard, and designing reusable object-oriented software is even harder. You must find pertinent objects, factor them into classes at the right granularity, define class interfaces and inheritance hierarchies, and establish key relationships among them. Your design should be specific to the problem at hand but also general enough to address future problems and requirements. You also want to avoid redesign, or at least minimize it. Experienced object-oriented designers will tell you that a reusable and flexible design is difficult if not impossible to get right the first time. Before a design is finished, they usually try to reuse it several times, modifying it each time.*

....

One thing expert designers know not to do is solve every problem from first principles. Rather, they reuse solutions that have worked for them in the past. When they find a good solution, they use it again and again. Such experience is part of what makes them experts. Consequently, you'll find recurring patterns of classes and communicating objects in many object-oriented systems. Therese patterns solve specific design problems and make object-oriented designs more flexible, elegant, and ultimately reusable. They help designers reuse successful designs by basing new designs on prior experience. A designer who is familiar with such patterns can apply them immediately to design problems without having to rediscover them."

In the next section I describe some design patterns used in this project. For completely understand this patterns the user is required to know basic notions on design patterns [Gamma et al., 1994] and on the Unified Modeling Language, modeling language used in the Unified Method by James Rumbaugh and Grady Booch.

Abstract Factory

Within this project this pattern allows the system to be independent of the type of storage.

All the classes in the system depend on the abstract classes StoreMgr, IndexesMgr and LinksMgr and if an instance of ODBCStoreMgr is created then the whole system would be using database storage through ODBC and if an instance of FileStoreMgr is created then it would be using file storage.

Moreover, if in the future I need to add a new type of storage, optical disk storage for instance, I will have to create a new subclass of StoreMgr, IndexesMgr and LinksMgr and the whole system will be using the new type of storage.

The following figure illustrates the class diagram of this pattern.

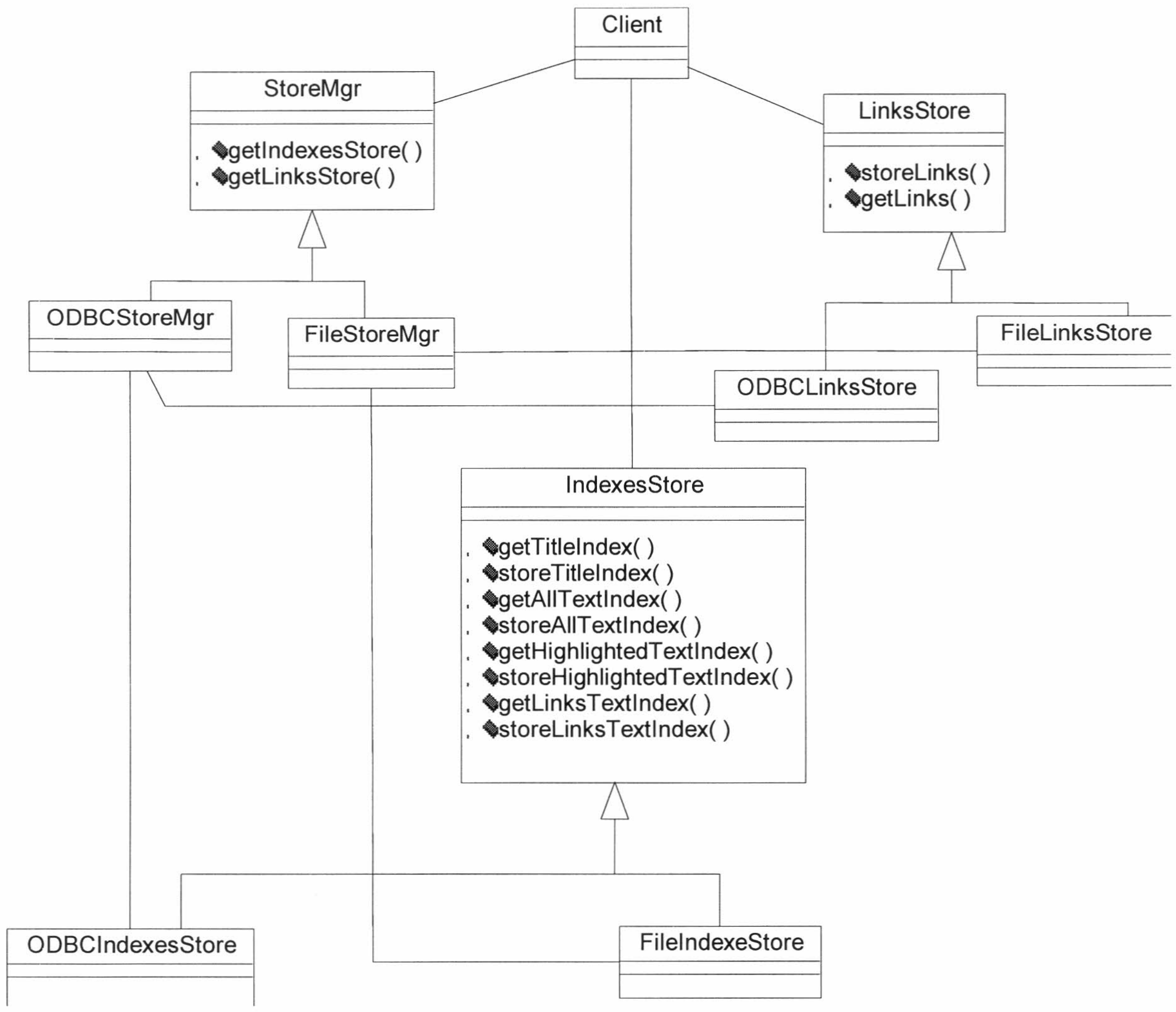


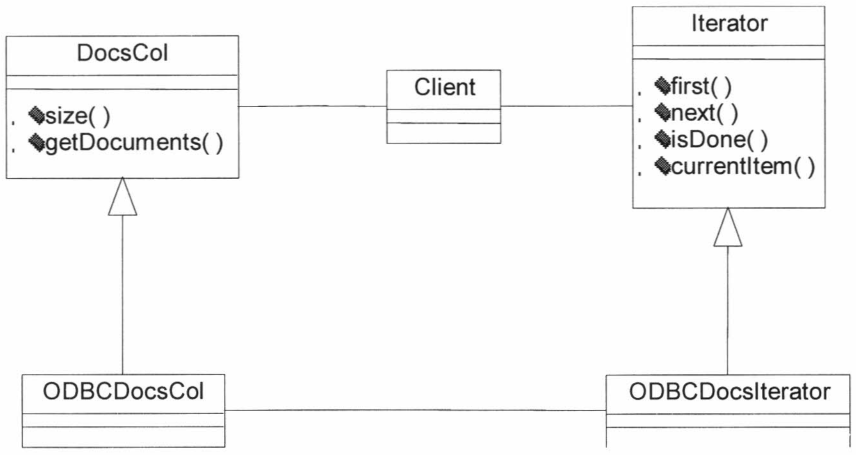
Figure 49: Class Diagram representing the classes involved in the application of the Abstract Factory pattern

Iterator

An aggregate object[12] such as a collection should give you a way to access its elements without exposing its internal structure. A document collection should give access to its documents in the same way whether it stores its documents in a database or in a file.

The following class diagrams illustrates the implementation of this pattern. The first class diagram shows a document collection using a database (ODBCDocsCol) and the second class diagram shows a document collection using a file. In both cases when the message getDocuments is called an Iterator will be returned and the client will not know whether the underlying storage is a file or a database.

[12] An aggregate object is an object composed of several other objects. For instance a collection of names is an aggregate object composed of several name objects.



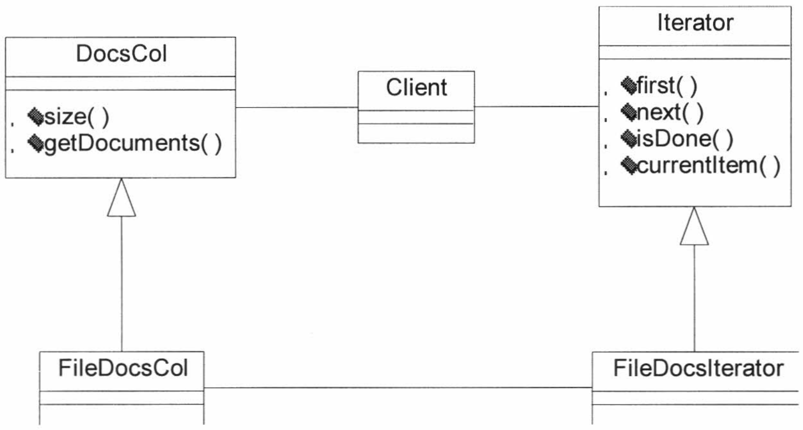


Figure 50: Class Diagram representing the classes involved in the application of the Iterator pattern

Appendix B: Some comments on the use of models

The use of models is common in every discipline dealing with complex systems because models help to increase the level of abstraction and better understand the whole system.

In the current work I used the object model [Booch, 1994] for modeling the software. I used class hierarchy diagrams for representing the static relations between classes in the system and I used interaction diagrams for representing interaction between objects in the system.

In Figure 51 I show a class hierarchy diagram representing classes involved in the manipulation of documents:

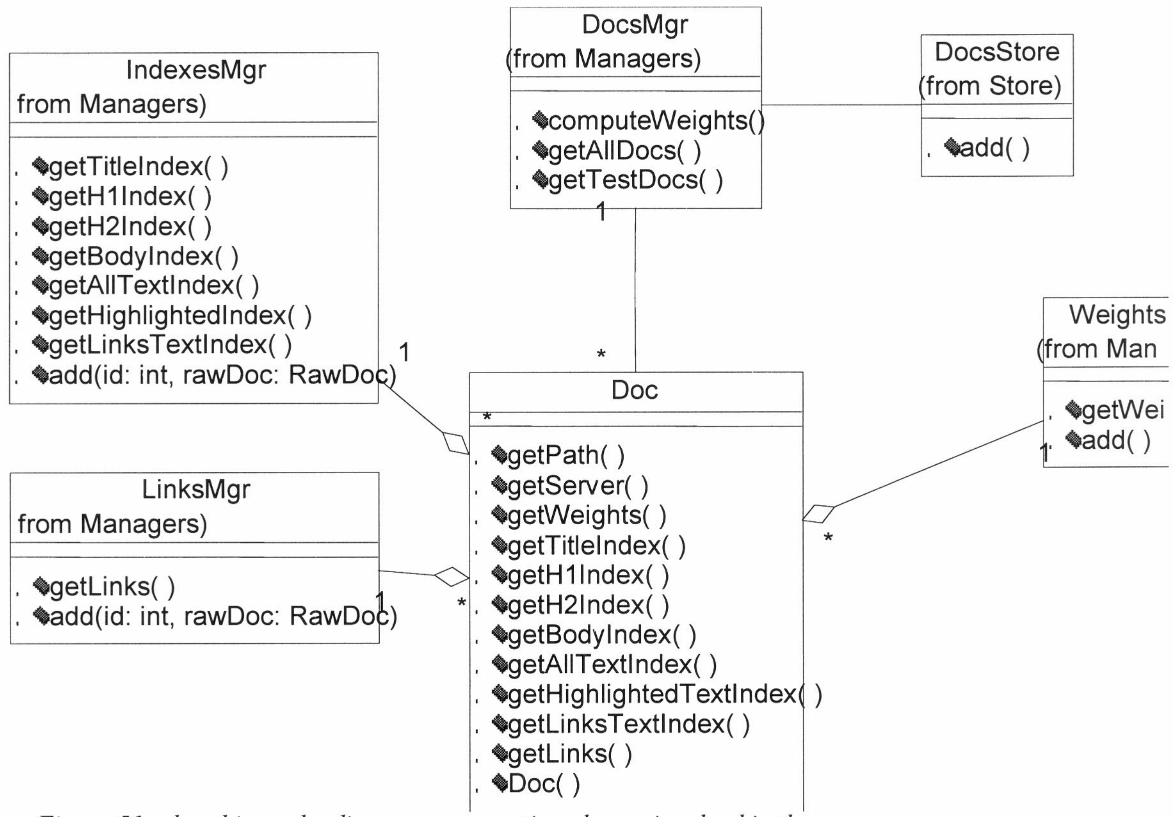


Figure 51: class hierarchy diagram representing classes involved in the manipulation of documents

In Figure 52 I show an interaction diagram representing the collaboration of objects in building the title criteria similarity between two documents:

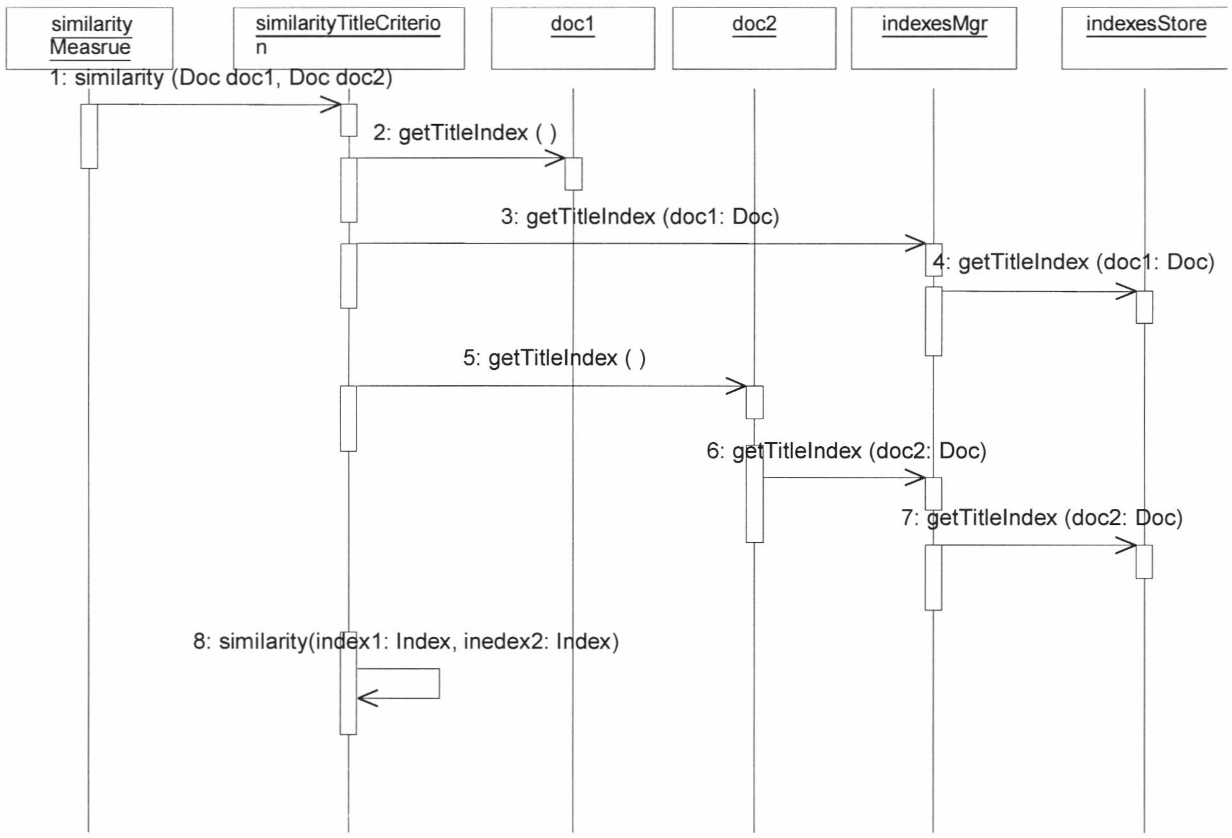

similarity Measrue
similarityTitleCriterion
doc1
doc2
indexesMgr
indexesStore
1: similarity (Doc doc1, Doc doc2)
2: getTitleIndex ( )
3: getTitleIndex (doc1: Doc)
4: getTitleIndex (doc1: Doc)
5: getTitleIndex ( )
6: getTitleIndex (doc2: Doc)
7: getTitleIndex (doc2: Doc)
8: similarity(index1: Index, inedex2: Index)

Figure 52: Interaction diagram representing the collaboration of objects in building the title criteria similarity between two documents

I also used the entity-relational model and the relational model in building the database.

Figure 53 shows a section of the entity-relation model:

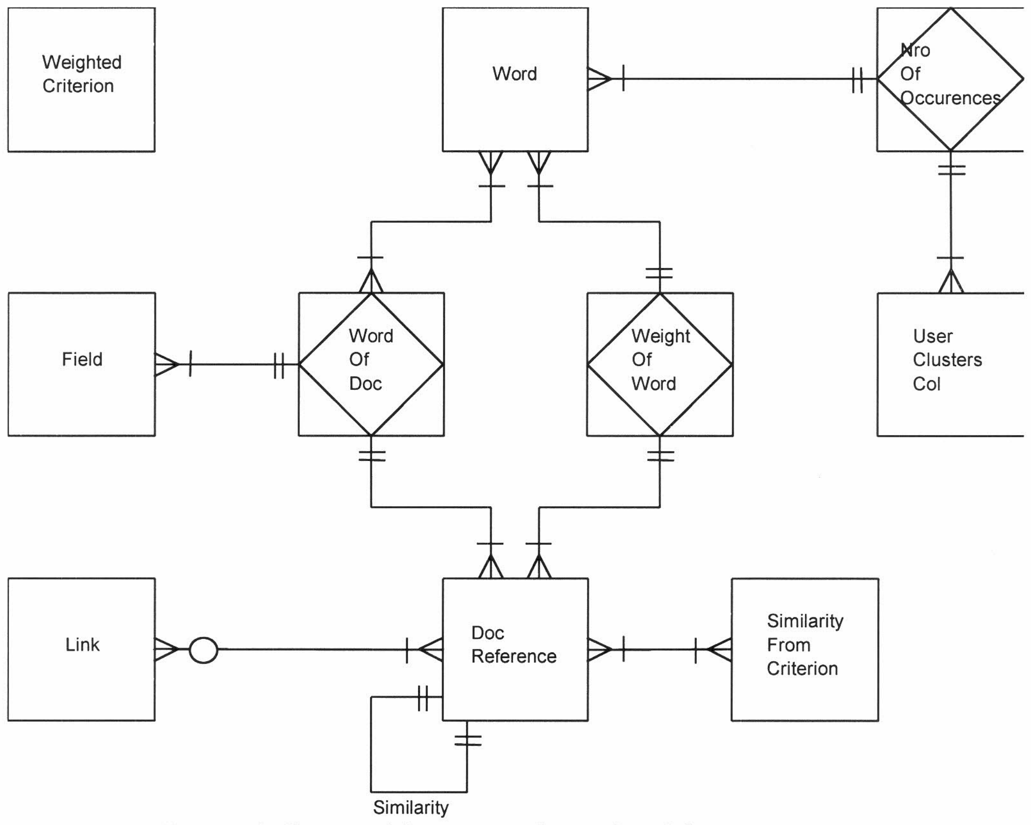


Figure 53: Section of the Entity-Relational model

Figure 54 shows the relational model:

DocReference(<u>refNro</u>, server, path, hClusterID, uClusterID);
Word(<u>wordNro</u>, text);
WordInDoc(<u>refNro, field, wordNro</u>, nroOccurrences);
WeightOfWordInDoc(<u>refNro, wordNro</u>, weight);
NroOfOccurrences(<u>wordNro, uClustersColName</u>, nroOfOccurrences)
Link(<u>linkNro</u>, server, path);
LinkOfDoc(<u>refNro, linkNro</u>);
SimilarityFromCriterion(<u>refNro1, refNro2, CriterionName</u>, SimilarityValue);
Similarity(<u>refNro1, refNro2</u>, value);
SimilarityMatrixRow(<u>rowNro, uClustersColName</u>, refNro1, refNro2);
SameClusterMathVectorElement(<u>rowNro, uClustersColName</u>);
UserCluster(<u>uClusterId</u>, name, uClustersColName);
Query(<u>refNro</u>, uClusterId);
QueryResultSet(<u>resultSetID</u>, refNro);
DocInQueryResultSet(<u>refNro, resultSetID</u>, similarity);
QueryResultSetInCol(<u>resultSetID, colName</u>)
RPEvaluationItem(<u>name, coordinationLevel</u>, precision, recall);
HCluster(h<u>ClusterID</u>)
HMultiCluster(u<u>ClusterID</u>, uClusterID1, uClusterID2, similarityValue)
WeightedCriterion(<u>measureName, criterionName</u>, weight)

Figure 54: Relational model