HypIR: Hypertext Based Information Retrieval

Yuan Lee[*] Fazli Can[†]

[*] Miami University, commons-admin@lib.muohio.edu
[†] Miami University, commons-admin@lib.muohio.edu

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE
& SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-1992-007

HypIR: Hypertext Based Information Retrieval

Yuan Ming Lee and Fazli Can

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

HypIR: Hypertext Based Information Retrieval

by

Yuan Ming Lee Fazli Can
Systems Analysis Department
Miami University
Oxford, Ohio 45056

Working Paper #92-007 August, 1992

HypIR: HYPERTEXT BASED INFORMATION RETRIEVAL*

Yuan Ming LEE **Fazli CAN+**

Department of System Analysis
Miami University
Oxford, OH 45056

August 1, 1992

Abstract

Information Retrieval (IR), which is also known as text or document retrieval, is the process of locating and retrieving documents that are relevant to the user queries. In hypertext environments, document databases are organized as a network of nodes which are interconnected by various types of links. This study introduces a hypertext-based text retrieval system, HypIR. In HypIR, the semantic relationships among documents are obtained using a clustering algorithm. A new approach providing the advantages of system maps and history list is introduced to prevent the user from being lost in the IR hyperspace. The paper presents the underlying concepts and implementation details. HypIR is based on the object-oriented paradigm and its execution platform is HyperCard.

1. INTRODUCTION

Information Retrieval (IR) is the process of locating and retrieving documents that are relevant to the user queries [SALT89]. In general, IR is accomplished using document representatives or surrogates. Whatever the representation of the documents, the major problem in IR is the query formulation. This is why several retrieval techniques are available in the IR literature [BELK87]. Among these techniques, the hypertext approach, which allows the user to navigate and inspect the database documents according to his own wishes, is the most intuitive one [NIEL90].

In hypertext environments, databases are organized as a network of nodes which are interconnected by various links. Through the links, the user can navigate or browse the documents in a non-sequential manner. This network browsing process is totally controlled by the user .

* This work was supported in part by MUCFR grant.

+ (513) 529-5950, fc74sanf@miamiu.bitnet

In a hypertext-based IR system, documents have multiple entries and numerous connections as shown in Figure 1. In Figure 1, computed links are constructed whenever the user submits a query. On the other hand, embedded links may be provided by system functions. In such a system, the browsing of hypertext is triggered by some "optimal" starting nodes, which are the documents with high similarity to the user query. The user may then navigate among documents following the original query or, alternatively, the user may utilize system functions to find documents containing the same keywords or documents written by the same author, etc. Consequently, the user can become more and more familiar with the system and his information need. Thus, a query which is more accurate then the initial query can be formulated and more relevant documents can be found.

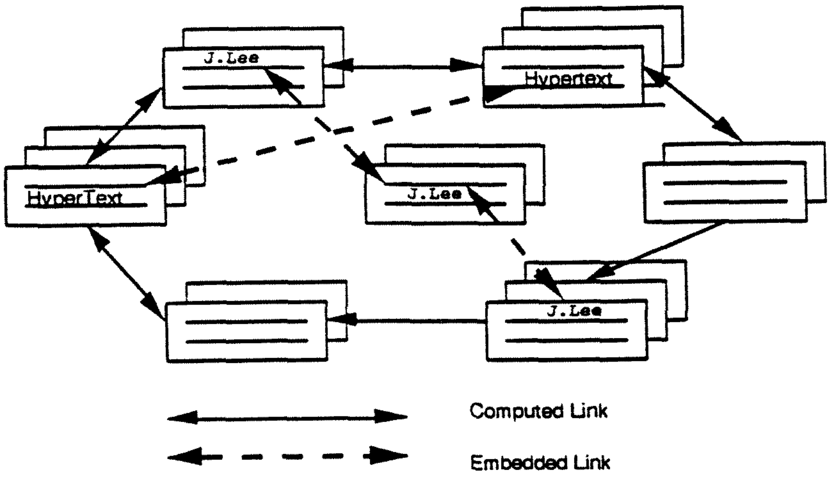


Figure 1. An example hypertext structure.

The recent IR literature contains various examples of hypertext-based IR systems sometimes with multimedia support. For example, American Memory is a multimedia integrated system which provides electronic images of selected collections of the Library of Congress. The system provides archival material related to the American culture and history on CD-ROM discs and laser videodiscs. The multimedia database covers motion pictures, photographs, cartoons, speeches, songs and text [LIBC91].

The News Retrieval Tool (NRT), built at University of Glasgow in Scotland, is based on a probabilistic retrieval model. This system covers a collection of articles from Financial Times, and is designed to test retrieval improvement for users of the existing profile retrieval services [HARM92].

CANSEARCH, which is an application of artificial intelligence techniques, provides intelligent access to on-line information. It is designed to enable doctors to retrieve cancer-therapy-related documents from the MEDLINE database. To use CANSEARCH, the user must have sophisticated medical knowledge, but little IR experience [GAUC92].

I^3R (Intelligent Interface for Information Retrieval) is a knowledge base system which allows the user to find information using various means. To retrieve documents, both natural language query and Boolean query formulation can be used [CROF87, THOM89].

HYPERLINE, which has been developed by the Information Retrieval Service of European Space Agency, is based on a two-level conceptual architecture for the construction of a hypertext environment for interacting with large textual database. In HYPERLINE, the collection of documents of interest is placed in the first level and the semantically related concepts are placed in the second level. Meanwhile, various functions such as semantic association, navigation, sequential reading, backtracking and history list are also provided [AGOS92].

Another example of hypertext-based IR system with hierarchical cluster browsing capability is implemented by Crouch and his co-workers. This system allows the user to browse the nodes within a single link clustering structure. When using this system, the user is not expected to access best-matching documents directly. Instead, the user can utilize the similarity values and single link structure to decide which clusters should be visited [CROU89].

An application of the hypertext and IR techniques on a medical handbook is defined by Frisse. In his system the links are defined by already existing hierarchical relationships of different sections of the handbook [FRIS88].

In this paper we introduce a hypertext-based text retrieval system, HypIR. As in any other information system, efficiency and effectiveness are the main concerns. Efficiency and effectiveness are, respectively, associated with the time and space required for searching and with the quality of retrieval. The implementation principles of HypIR are proven to be both effective and efficient [CAN90, CAN92a, CAN92b]. In HypIR, since the documents are independently created, the semantic relationships among documents are obtained using the Cover Coefficient-based Clustering Methodology (C^3M). This algorithm generates statistically valid clusters (i.e. groups of documents that are strongly associated with each other) which are appropriate for IR [CAN90]. The selection of documents from the generated clusters is performed using inverted index search techniques. HypIR is implemented using HyperCard and THINK Pascal. The system has a dynamic nature and documents can easily be added and deleted.

The paper is organized as follows. Section 2 briefly introduces the concepts of document representation and query-document matching process. In hypertext browsing, Full Search (FS) and Cluster-based Search (CS) provide the so-called "optimal" browsing starting points. Section 3 covers the principles of FS and the details of CS. Section 4 and 5, respectively, provide the data and file structures, and the design principles used for the implementation of HypIR. Finally, a conclusion is given in Section 6.

2. DOCUMENT REPRESENTATION AND QUERY-DOCUMENT MATCHING

In the design and implementation of IR systems, some decisions should be made for the techniques of document representation, query-document matching and searching strategies. Thus, before getting into the details of HypIR, we would like to introduce the techniques that are adopted in HypIR and the reasons that support the selection of these techniques. This section considers document representation and query-document matching, and the next section considers the search strategies.

2.1 Generation of Document Representatives

In IR, two common approaches for document representative generation are document signatures and the vector space model [SALT89]. The document signature approach uses a bit map array for each document whose entries are set by a hash function using the words of documents as its input [FALO85]. In the vector space model, the approach used in this study, each document is represented by a document vector describing the words, or terms, which appear in the associated document. This model is simple and appropriate for hypertext environments [SALT89, CROU89].

According to the vector space model, a document database simply becomes a document, D matrix. For a database of m documents defined by n terms, an entry in the D matrix in row i (document i) at column j (term j), d_{ij} ($1 \leq i \leq m$, $1 \leq j \leq n$), represents the weight, or frequency of term j in document i (i.e. the number of occurrences of term j in document i).

When constructing the D matrix, a stemming algorithm should be adopted to reduce the size of the D matrix. For the documents or queries written in natural English, it is known that terms with a common stem will usually have similar meanings such as the following.

attract, attracted, attraction and attractive

Thus, if the IR system can recognize the various suffixes (-ed, -ion, -ive, etc.) and remove them from the stem, "attract," the complexity of the system and the storage requirement of the database can both be reduced. In HypIR, the stemming program is coded using Porter's algorithm [PORT80]. Porter's algorithm is simple, compared to other stemming algorithms, but effective [HARM91].

2.2 Query - Document Matching

No single search strategy can satisfy all users' queries. Therefore, it is desirable that an IR system should have more than one search strategy. Two common search techniques are Full Search (FS) and Cluster-based Search (CS). FS has the best performance in terms of retrieval effectiveness and CS facilitates document browsing. For both, a query matching (similarity) function, also described as a search machine, determines which documents or clusters potentially relevant (i.e. match the query) and should be returned to the user.

Several matching functions based on term weighting components of document and query terms have been introduced in the IR literature. Term weighting consists of three components, the term frequency component (TFT), the collection frequency component (CFC), and the normalization component (NC). Both the weights of terms in a document and a query (denoted by w_{dj} and w_{qj}, $1 \leq j \leq n$) can be derived by multiplying the term weights of these three components. After obtaining the term weights, the similarity between a document d and a query q can be defined as follows [SALT89].

$$\text{similarity}(d, q) = \sum_{k=1}^{n} w_{dk} \cdot w_{qk}$$

where n is the number of terms.

According to Salton and Buckley's research, 1800 different combinations of document-query term weight assignments (i.e. matching functions) can be derived. Among these combinations, 287 were found to be distinct and six of them were recommended [SALT88]. The results of the experiments reported in [CAN90] indicate that the matching function labeled as TW2 (tfc.nfx in [SALT88]) is the most effective one. Thus, TW2 is used as the search machine of HypIR.

3. SEARCH STRATEGIES

3.1 Full Search

Full Search (FS) is implemented using inverted index search (IIS). In IIS, each distinct term in the system has a list of documents in which that term appears. Each document is represented by its document number and associated with the weight of the corresponding term. By traversing the list of those query terms, the similarity values of all database documents are calculated [SALT89]. The documents with the highest similarity values are then selected to answer the user's query. It is known that IIS is both effective and efficient [CAN92b, SALT89].

3.2 Cluster - based Search

In IR, there is a hypothesis known as the "clustering hypothesis," which states that "closely related documents tend to be relevant to the same query" [VANR79]. It is this hypothesis that supports the Cluster-based Search (CS) strategy. In CS, the documents are divided into several homogeneous groups (clusters). In a typical CS, the user queries are first compared with the cluster representatives (centroids). Then, after selecting best-matching clusters, detailed query-by-document comparison is performed within the selected clusters. (Note that this is a conceptual explanation. The actual implementation may be different.)

Although the selected clusters may not contain the best-matching documents, generally speaking, CS and clustering provide several advantages.

(1) In a clustered document environment, the user may choose to browse the cluster of any retrieved document. This provides some expansion of recall ability, as not all documents in a cluster are relevant, but they are related in ways not always accessible through a query. Furthermore, during the process of cluster browsing, the user creates a better image of his information need and can submit a better query to the system.

(2) In a multi-search IR system, CS constitutes a good alternative to FS.

(3) The results of FS and CS can be combined to increase the system effectiveness. For instance, the combination of FS and CS may provide a precision improvement of up to 25 percent [CAN92a] (precision is defined as the ratio of the number of retrieved relevant documents to the number of retrieved documents).

(4) In a clustered environment, the documents of a cluster can be put into close physical proximity in secondary storage to decrease I/O time, and therefore, to increase system efficiency [SALT89].

3.3 Clustering Algorithm

In HypIR, the semantic relationships among documents is obtained using the Cover-Coefficient-based Clustering Methodology (C^3M). In C^3M, some of the documents are selected as the cluster initiators (seeds) then the nonseed documents are assigned to one of the clusters initiated by the seed documents. C^3M produces a single-level partitioning type clustering structure. The number of clusters, n_c, is determined using the Cover-Coefficient (CC) concept. According to CC, for an m document by n term D matrix, the value range of n_c and the average cluster size (d_c) is as follows.

$$1 \leq n_c \leq \min(m, n), \quad \max(1, m/n) \leq d_c \leq m$$

In C^3M, an m by n D matrix is first mapped into an m by m C matrix using the following formula.

$$c_{ij} = \alpha_i . \sum_{k=1}^{n} (d_{ik} . \beta_k . d_{jk})$$

where $1 \leq i, j \leq m$ and α_i and β_k are the reciprocals of the ith row sum and kth column sum. This asymmetric C matrix shows the relationships among the documents of a database. The diagonal entries of the C matrix determine the number of clusters, n_c, and is used for the selection of cluster seeds. The relationships between a nonseed document (d_i) and a seed document (d_j) is determined by calculating c_{ij} entry of the C matrix. The whole clustering process requires the calculation of $(m+(m-n_c).n_c)$ entries of the total m^2 entries of the C matrix. This is a small fraction of m^2, since n_c is much less than m. A detailed discussion of C^3M and its complexity analysis are available in [CAN90]. In a dynamic document environment the clusters of C^3M can easily be updated without initiating a reclustering process [CAN92a]. The CS effectiveness of C^3M is reported in [CAN90]. The mentioned study shows that the effectiveness of C^3M is 15.1 to 63.5 (with an average of 47.5) percent better than four other clustering algorithms in CS.

3.4 Implementation of Cluster-based Search

Cluster-based Search (CS) is conventionally implemented in the following two ways.

CVDV: Match the query vectors with all the centroid vectors (CV) and the document vectors (DV) of the members of the best-matching clusters.

ICDV: Match the query vectors with the inverted indexes of centroids (IC) and the document vectors (DV) of the members of the best-matching clusters.

In addition to these conventional methods, the following method of implementation for CS has been introduced in [CAN92b].

ICIIS: Match the query vectors with the inverted indexes of centroids (IC) and the inverted indexes of all documents.

In ICIIS, the system first retrieves the best-matching clusters by matching the query vector with the inverted indexes of centroids. After obtaining the best-matching clusters, the documents of these clusters are selected using the results of IIS performed on the complete database. In other words, by using the ICIIS algorithm, the IR system can also provide the results of FS without extra effort. By definition the efficiency of ICIIS is independent of the number of clusters to be selected and the number of documents to be displayed to the user for browsing purposes. It is shown that ICIIS is much more efficient than other conventional CS implementation methods. Its efficiency is due to shortness of the query vectors, and it is especially suitable to very large databases [CAN92b].

4. SYSTEM DATABASE AND FILES

4.1 System Database

The document database of HypIR is the TODS/TOIS database covering the papers published in both *ACM Transactions on Database Systems* and *ACM Transactions on Information* Systems* . In HypIR, each document of TODS/TOIS is represented by a document card consisting of the title, author(s), and the abstract of the corresponding article. For clustering purposes, the database is defined with a D matrix using an indexing software. The relevant statistics of the current TODS/TOIS database are listed in Table I.

Table I. Characteristics of the TODS/TOIS Database.

No. of documents (m)	: 524
No. of terms (n)	: 3668
No. of clusters (n_c)	: 65
No. of nonzero entries in the D matrix	: 28343
Average No. of terms per document	: 54.09

The TODS/TOIS database is currently adopted by a mainframe IR system, ANIRS, and is available on the Miami, IBM, environment [CAN92c]. The updates of TODS/TOIS and its clustering structure are conducted periodically [CAN92a].

4.2 System Files

Several files are used in the implementation of HypIR. They are described in the following.

(1) IID file :contains the inverted index representation of the D matrix and is needed for the selection of the best-matching documents using IIS.

(2) DV file :is a direct access file of document vectors of the D matrix and is used for the creation of IID file and the implementation of "Nearest Neighbor" browsing technique which returns the documents most similar to a located document.

(3) IC file :contains the inverted index representation of the centroid vectors and is needed for the selection of the best-matching clusters.

(4) CV file :is a direct access file of centroid vectors and is needed for the update of the IC file.

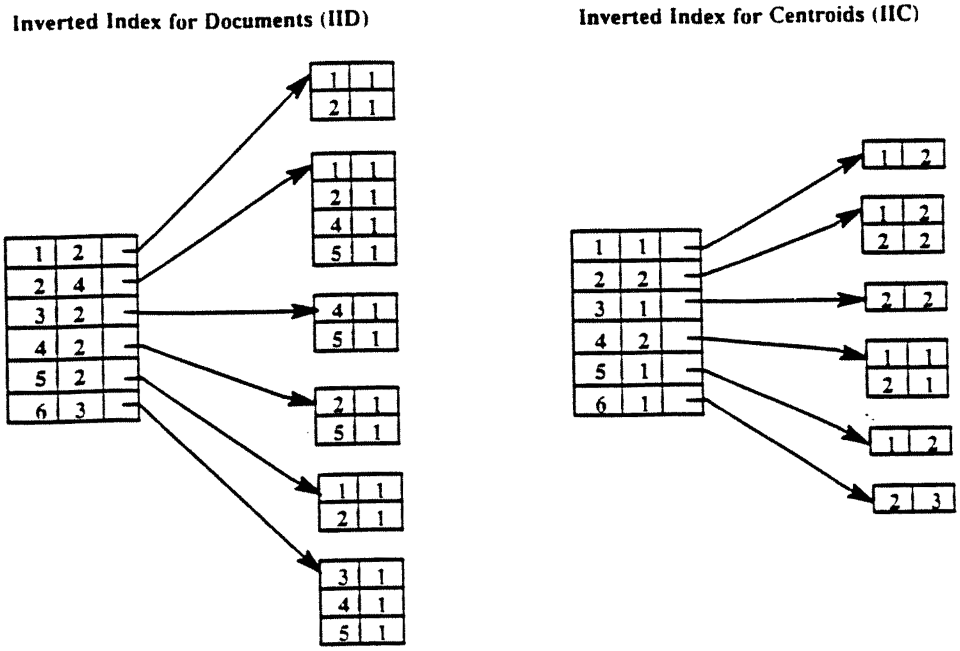
For better understanding of the major file structures, IID and IC, let us consider the example D matrix of Figure 2.A. The application of C^{3}M to this D matrix produces the

*Formerly ACM Transactions on Office Information Systems.

cluster $C_1 = \{d_1, d_2\}$ and $C_2 = \{d_3, d_4, d_5\}$ [CAN90]. The centroids and templates of IID and IC for the example D matrix are depicted in Figure 2.B. In Figure 2.B, IID shows that t_1 appears in two documents: d_1 and d_2; since the D matrix is binary, the term weights are ones. IC provides the same information for centroids. For example, the header information for t_2 indicates that it appears in two centroids (c_1 and c_2). In cluster C_1 both members, d_1 and d_2, contain t_2, that is why the weight of t_1 in c_1 is two. Similarly, in C_2, two members, d_4 and d_5, contain t_2.

$$D = \begin{array}{cccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & \\ \left[\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right. & \begin{array}{c}1\\1\\0\\0\\0\end{array} & \begin{array}{c}1\\1\\0\\1\\1\end{array} & \begin{array}{c}0\\0\\0\\1\\1\end{array} & \begin{array}{c}0\\1\\0\\0\\1\end{array} & \begin{array}{c}1\\1\\0\\0\\0\end{array} & \begin{array}{c}0\\0\\1\\1\\1\end{array} & \left.\begin{array}{c} \\ \\ \\ \\ \\ \end{array}\right] \begin{array}{c}d_1\\d_2\\d_3\\d_4\\d_5\end{array} \end{array}$$

Figure 2.A. an example of D matrix.



Templates of the IID and IC files:
IID: << Term No., No. of Documents Using This Term> --> <Document No., Term Weight>$^+$>$^+$
IC : << Term No., No. of Centroids Using This Term> --> <Centroid No., Term Weight>$^+$>$^+$
+ : indicates one or more occurrences of the enclosed information.
--> : indicates a pointer.

Figure 2.B. IID and IC file structures and templates.

5. USING HYPERTEXT FOR IR AND HypIR

5.1 Links and Nodes

It is commonly agreed that the major advantage of hypertext model is the non-sequential organization of the information. In a hypertext-based system, each piece of information on the screen can provide several links through which the user can browse the entire database and retrieve increasingly useful information. In general, the structures containing all the nodes and links in a hypertext system can be divided into two categories, hierarchical structures and network structures, as shown in Figure 3.

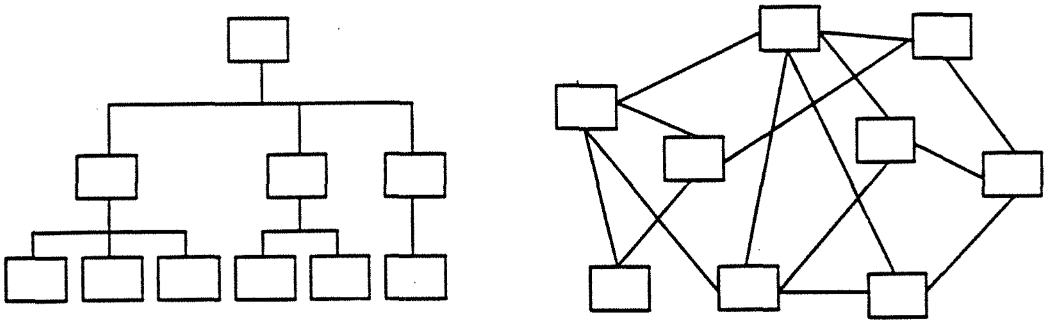


Figure 3. Hierarchical and network hypertext models.

In a hierarchical structure, the information is divided into several levels. Each lower level of the hierarchy provides more detailed information. Hence, the information can be searched and presented in a semantic way. For a bibliography database adopted in HypIR, the network structure is a better choice, since the hierarchical relationships are expensive to construct [WILL89]. In a network structured system, nodes are connected in a non-hierarchical manner. The links between the nodes may be embedded in the names of authors, the key terms, or an index list in the end of a node. Thus, the user's navigation is totally free in network environments.

5.2 Hypertext Browsing

Browsing is a heuristic search through a well connected collection of documents in order to find information relevant to the user's needs [THOM89]. It provides two major advantages for IR.

(1) By browsing the user's information need can become more and more clear. Hence, more appropriate queries can be created.

(2) Since the related documents are linked together, a user can evaluate a large database in a rapid manner.

In a hypertext-based IR system, the user may browse

(1) the documents written by the authors of the located document,

(2) the references of the located document,

(3) the documents in the same publication,

(4) the documents that are very similar to the located document (i.e. the nearest neighbors),

(5) the documents containing the same key terms.

In HypIR, all of these browsing methods are implemented and future modification or enhancement will be conducted according to users' feedback.

A major consideration when designing a hypertext-based IR system is to prevent users from getting lost during browsing. Currently, two major designs, "system map" and "history list", have been used in some hypertext systems [AGOS92, NIEL90]. The "system map" method, which is usually adopted in the hierarchical hypertext environment, is implemented by dividing the entire hierarchy of the system into several maps according to different subjects. Each map is then inserted into the links between the nodes. Thus, the user can navigate the entire collection according to the system maps. If the user is lost, he can still find the subjects which he was in. "History list" is usually a special node in a hypertext system. It keeps a list of all the navigated nodes for the user. Whenever, the user is lost, he can still find a specific document by checking such a list.

Both of these designs are not used in HypIR due to the following reasons.

(1) In a system adopting system maps, all the documents must have a kind of semantic relationship which can be clearly expressed graphically.

(2) System maps can not help the user find a specific document conveniently. In a large database, finding a navigated document may still consume a lot of time.

(3) History list is a test for the user's memory, since it expects the user to memorize the titles of the documents without providing any hint.

(4) History list does not provide any browsing ability.

In HypIR, a new approach is used. The detail of this design is provided in section 5.4.

5.3 HyperCard's Object-Oriented Philosophy

HypIR is implemented using the HyperCard graphic programming package, because of its hypertext nature, object-oriented philosophy, powerful scripting language and the ability to facilitate general purpose languages such as Pascal and C. The object-oriented philosophy of HyperCard visualizes the well known "object" in traditional object-oriented programming languages such as C++ or Smalltalk. In HyperCard, when a user needs a new object, it is not necessary to use a specific statement provided in the language.

Instead, a graphic tool is provided to actually draw an object on the screen. The objects provided in HyperCard are stacks, backgrounds, cards, buttons, and fields. Each of these objects can be associated with a script which enables the object to respond to a HyperCard message. A system based on hierarchical inheritance and message passing can be constructed by combining these objects . The inheritance of HyperCard is based on the relationships between the objects as shown in Figure 4.

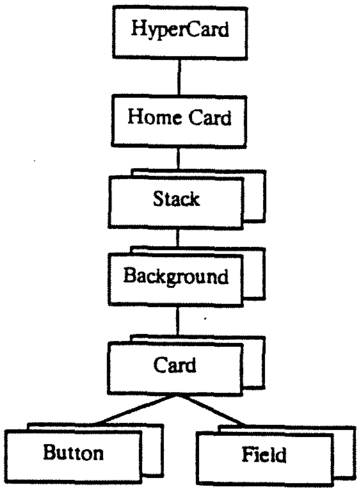


Figure 4. The hierarchy of objects in HyperCard.

The "stacks" and "cards" are actually HyperCard terminology to represent HyperCard files and screens, respectively. As in Figure 4, a stack may have several different backgrounds. A background can be shared by numbers of cards. Each card has several buttons and fields which distinguish itself from the other cards with the same background. Furthermore, from the bottom level to the top each object inherits all the HyperCard properties of the higher level object. For example, all the cards with the same background look similar to each other. All the backgrounds within a stack have the same size on the screen. Figure 4 also expresses the message passing strategy adopted in HyperCard. Whenever a message has been generated by an object, it will be either sent to a specific object if any exists, or it will be passed through the hierarchy until it is interrupted by an object on a higher level.

5.4 HypIR System Design and Implementation

The architecture of HypIR, is based on HyperCard's object-oriented philosophy. The whole system is accomplished by using three stacks, the HypIR stack, the help stack and the index stack as shown in Figure 5.

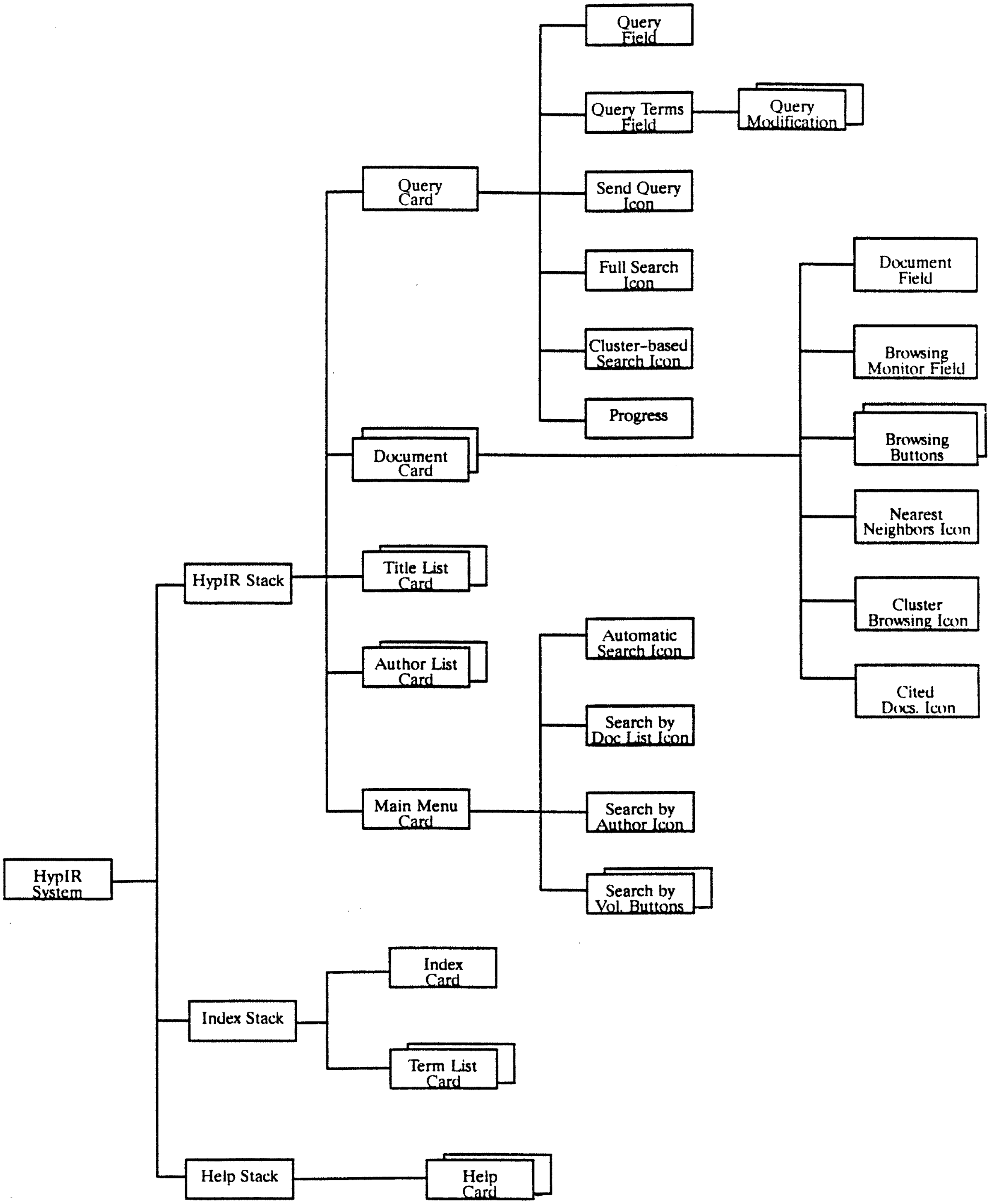


Figure 5. HypIR system design.

The HypIR stack is the main body of the entire system which contains a main menu card, a query processing card, and several title list cards, author list cards and document cards. After accessing the system, the user is first led to the main menu card (Figure 6). On this card, the user can make a decision among several searching strategies and start the navigation in the system. The user may click on

(1) The "Automatic Search" icon to access query processing card,

(2) The "Doc List" icon to browse the titles of the entire collection of documents (In this case, we assume that the user does not have any particular destination to start the searching.),

(3) The "Authors" icon and then type in the last name of a particular author in a pop-up dialog box. consequently, a list of the author's papers are provided by the system,

(4) One of the "books" and obtain a list of documents in a certain volume of the collection.

All the previous choices, except (1), provide the user a starting point to begin with his navigation. For example, after obtaining a list of documents (Figure 7), the user then browses through the document titles. Whenever an interesting document title is found, the user can click on the title and access the content of this document (Figure 8).

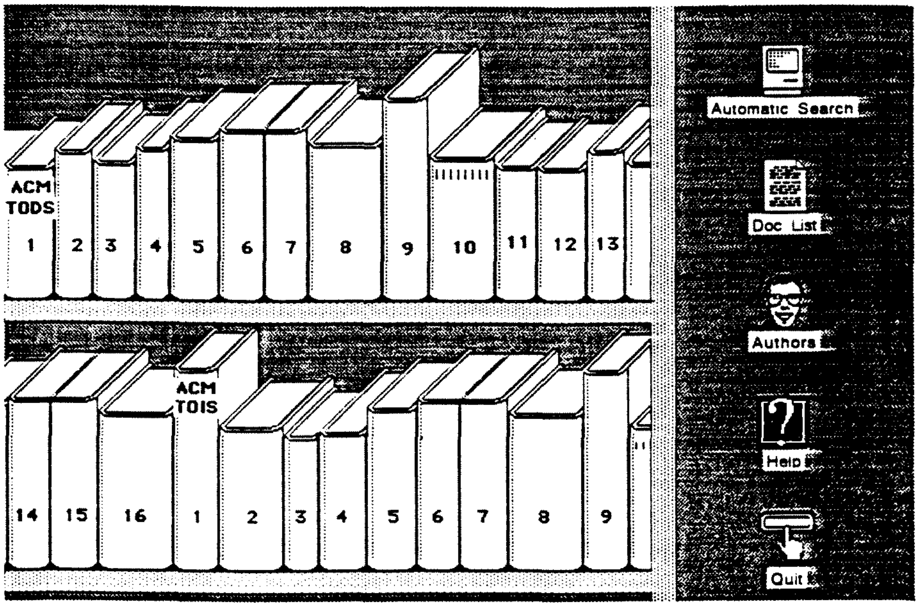
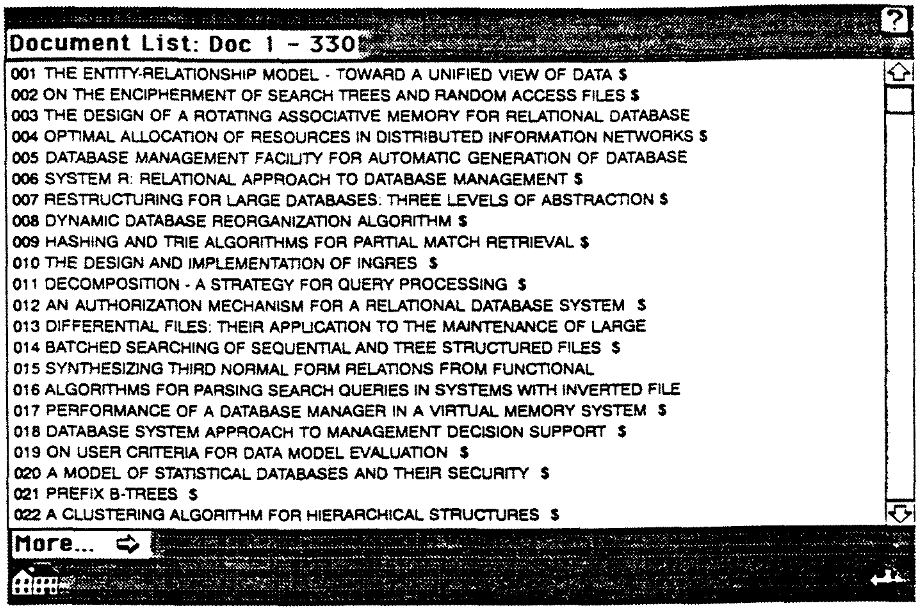


Figure 6. Main menu in HypIR.



Document List: Doc 1 - 330

001 THE ENTITY-RELATIONSHIP MODEL - TOWARD A UNIFIED VIEW OF DATA $

002 ON THE ENCIPHERMENT OF SEARCH TREES AND RANDOM ACCESS FILES $

003 THE DESIGN OF A ROTATING ASSOCIATIVE MEMORY FOR RELATIONAL DATABASE

004 OPTIMAL ALLOCATION OF RESOURCES IN DISTRIBUTED INFORMATION NETWORKS $

005 DATABASE MANAGEMENT FACILITY FOR AUTOMATIC GENERATION OF DATABASE

006 SYSTEM R: RELATIONAL APPROACH TO DATABASE MANAGEMENT $

007 RESTRUCTURING FOR LARGE DATABASES: THREE LEVELS OF ABSTRACTION $

008 DYNAMIC DATABASE REORGANIZATION ALGORITHM $

009 HASHING AND TRIE ALGORITHMS FOR PARTIAL MATCH RETRIEVAL $

010 THE DESIGN AND IMPLEMENTATION OF INGRES $

011 DECOMPOSITION - A STRATEGY FOR QUERY PROCESSING $

012 AN AUTHORIZATION MECHANISM FOR A RELATIONAL DATABASE SYSTEM $

013 DIFFERENTIAL FILES: THEIR APPLICATION TO THE MAINTENANCE OF LARGE

014 BATCHED SEARCHING OF SEQUENTIAL AND TREE STRUCTURED FILES $

015 SYNTHESIZING THIRD NORMAL FORM RELATIONS FROM FUNCTIONAL

016 ALGORITHMS FOR PARSING SEARCH QUERIES IN SYSTEMS WITH INVERTED FILE

017 PERFORMANCE OF A DATABASE MANAGER IN A VIRTUAL MEMORY SYSTEM $

018 DATABASE SYSTEM APPROACH TO MANAGEMENT DECISION SUPPORT $

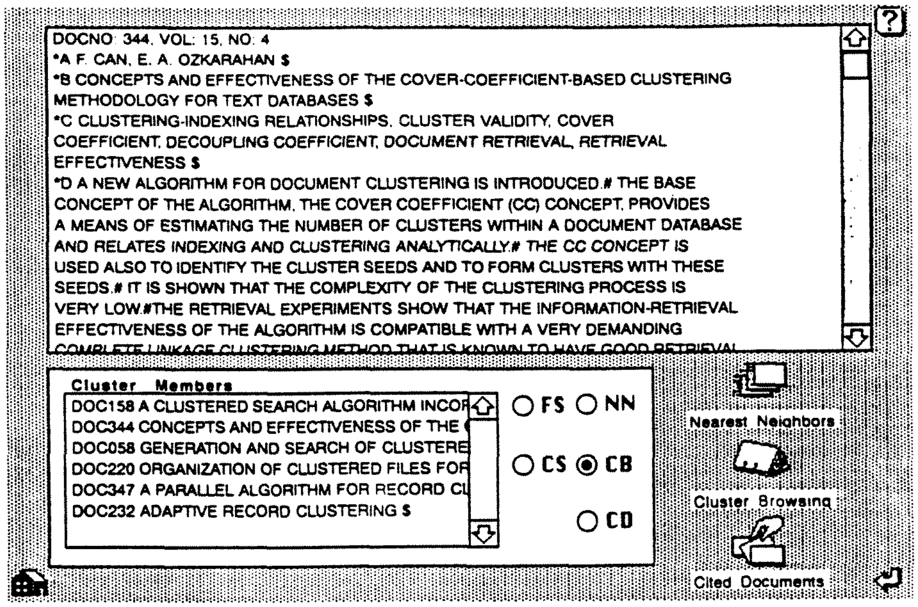
019 ON USER CRITERIA FOR DATA MODEL EVALUATION $

020 A MODEL OF STATISTICAL DATABASES AND THEIR SECURITY $

021 PREFIX B-TREES $

022 A CLUSTERING ALGORITHM FOR HIERARCHICAL STRUCTURES $

Figure 7. An example of document list in HypIR.



DOCNO: 344, VOL: 15, NO: 4

*A F. CAN, E. A. OZKARAHAN $

*B CONCEPTS AND EFFECTIVENESS OF THE COVER-COEFFICIENT-BASED CLUSTERING METHODOLOGY FOR TEXT DATABASES $

*C CLUSTERING-INDEXING RELATIONSHIPS, CLUSTER VALIDITY, COVER COEFFICIENT, DECOUPLING COEFFICIENT, DOCUMENT RETRIEVAL, RETRIEVAL EFFECTIVENESS $

*D A NEW ALGORITHM FOR DOCUMENT CLUSTERING IS INTRODUCED.# THE BASE CONCEPT OF THE ALGORITHM, THE COVER COEFFICIENT (CC) CONCEPT, PROVIDES A MEANS OF ESTIMATING THE NUMBER OF CLUSTERS WITHIN A DOCUMENT DATABASE AND RELATES INDEXING AND CLUSTERING ANALYTICALLY.# THE CC CONCEPT IS USED ALSO TO IDENTIFY THE CLUSTER SEEDS AND TO FORM CLUSTERS WITH THESE SEEDS.# IT IS SHOWN THAT THE COMPLEXITY OF THE CLUSTERING PROCESS IS VERY LOW.#THE RETRIEVAL EXPERIMENTS SHOW THAT THE INFORMATION-RETRIEVAL EFFECTIVENESS OF THE ALGORITHM IS COMPATIBLE WITH A VERY DEMANDING COMPLETE LINKAGE CLUSTERING METHOD THAT IS KNOWN TO HAVE GOOD RETRIEVAL

Cluster Members

DOC158 A CLUSTERED SEARCH ALGORITHM INCOR

DOC344 CONCEPTS AND EFFECTIVENESS OF THE

DOC058 GENERATION AND SEARCH OF CLUSTERE

DOC220 ORGANIZATION OF CLUSTERED FILES FOR

DOC347 A PARALLEL ALGORITHM FOR RECORD CL

DOC232 ADAPTIVE RECORD CLUSTERING $

○ FS ○ NN

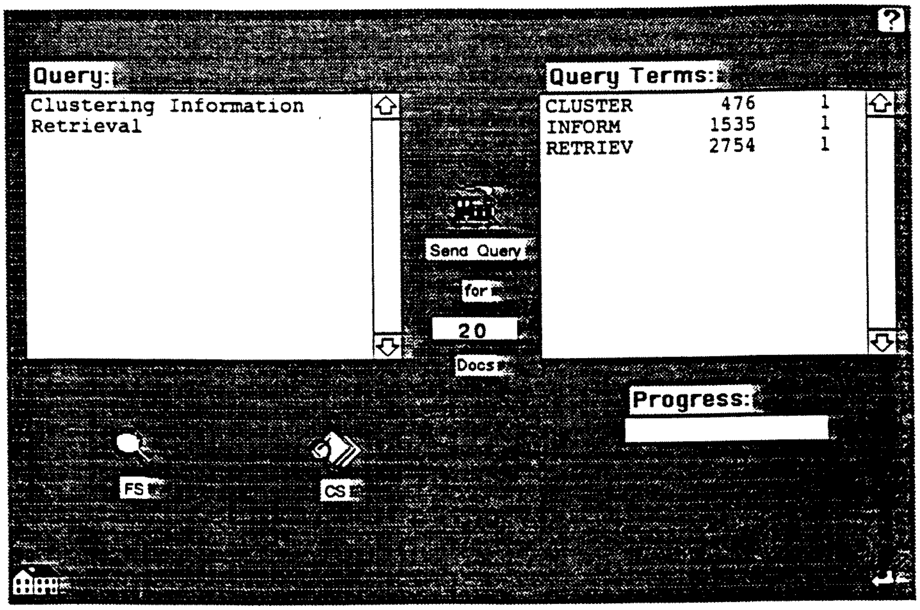
○ CS ● CB

○ CD

Figure 8. Document content in HypIR.

"Automatic Search" icon displays the query processing card (Figure 9). On this card, the user can first type in his query in natural language, click on the "Send Query" icon and obtain the corresponding query vector from the "Query Terms" field. If the user wants to modify the query vector, HypIR provides a simple method. To delete a query term, the user can click on the term itself. By doing so, the system will pop-up a dialog box (Figure 10) to request an action from the user. After the user clicks on the "Delete

Term" button, the specified term will be erased from the query vector. A similar approach is also provided for the modification of a term weight. The only difference is that the user needs to type in the term weight in another dialog box (Figure 11). To add a new term, the user can click on any spot on the "Query terms" field. After responding to the same dialog box shown in Figure 10, the "Index" stack will appear in a new window (Figure 12). By clicking on the letters in the window and using the same approach as checking a dictionary, the user can easily locate desired term and add it to the query vector.



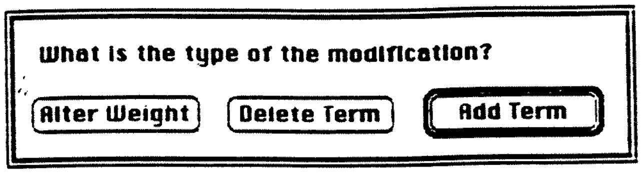
Query:

Clustering Information Retrieval

Query Terms:

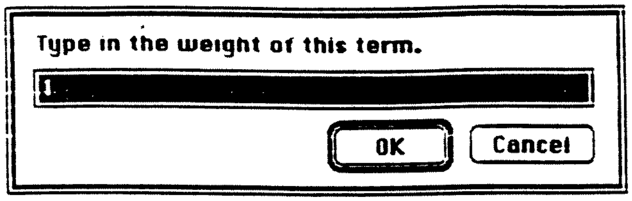
```
CLUSTER      476      1
INFORM      1535      1
RETRIEV     2754      1
```

Figure 9. Query processing card in HypIR



What is the type of the modification?

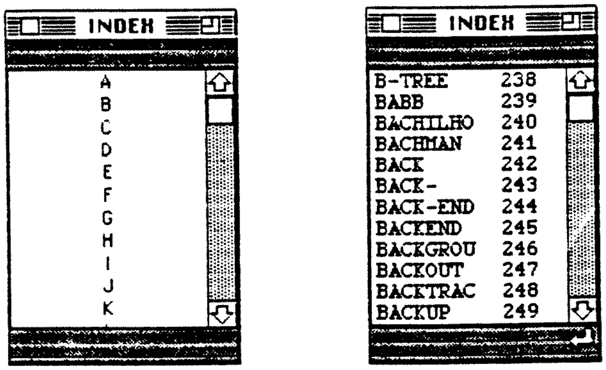
Figure 10. Query modification dialog box.



Type in the weight of this term.

1

Figure 11. Term weight dialog box.



INDEX

A
B
C
D
E
F
G
H
I
J
K

INDEX

B-TREE	238
BABB	239
BACHILHO	240
BACHMAN	241
BACK	242
BACK-	243
BACK-END	244
BACKEND	245
BACKGROU	246
BACKOUT	247
BACKTRAC	248
BACKUP	249

Figure 12. Index windows

The "FS" and "CS" icons represent the searching techniques, Full Search and Cluster-based Search, respectively. As defined before, FS and CS provide a list of the best-matching documents of the query vector in the entire database and the selected clusters, respectively. Again, the user accesses the content of the documents by clicking on their titles.

The document card as shown in Figure 8 contains several icons to support different types of browsing. The "Nearest Neighbors" icon provides the function that sends the entire document as a query and returns the nearest neighbors of it as the result. The "Cluster Browsing" icon provides a way that the user reveals all the members of the cluster in which the located document resides. The "Cited Documents" icon provides all the titles of the documents cited by the located document. Furthermore, browsing by key terms is also available in HypIR. The "Document" field contains a script that allows the user to navigate the entire database by clicking on any word which appears in the located document.

The field appearing on the bottom left corner of the document card is responsible for displaying the results of all the previous mentioned functions. It keeps track of the results of FS (Full Search), CS (Cluster-based Search), NN (Nearest Neighbors), CB (Cluster Browsing) and CD (Cited Documents). For instance, the user may first locate a document using the FS function, then traverse the database by using CB function or links of key terms. After all of these, the user may still want to see other documents found by FS. To do so only requires a click on one of the five radio buttons. In this case, "FS" button is the appropriate choice, of course. By doing so, the user can go back to the original path, once again. Due to the functionality of this field, it is named as *browsing monitor*.

The major reason that we adopted browsing monitor is to prevent the user from getting lost in HypIR. This design actually contains both of the advantages of system maps and history lists. It not only tracks all the navigated documents, but also groups the documents according to the retrieval function. Since the switching among different groups is so handy, we believe that browsing monitor provides a more convenient and practical aid to the user than system maps and history lists.

6. CONCLUSION

Information retrieval (IR), also known as text or document retrieval, is the process of locating and retrieving documents that are relevant to the user queries. In hypertext environments, document databases are organized as a network of nodes which are interlinked by various types of links. Through the links, the documents are navigated or browsed in a non-sequential manner which is totally controlled by the user. In a hypertext-based retrieval system, the browsing of documents should be triggered by so-called "optimal" starting nodes.

In this paper, we introduced HypIR, a hypertext-based IR system which is implemented in the HyperCard environment of Macintosh. In HypIR, the semantic relationships among documents are obtained using the C^3M clustering algorithm which is known to have good IR performance. For information retrieval, the users are provided with various search and browsing tools. These include full search (FS), cluster-based search (CS), nearest neighbors (NN) search, cluster browsing (CB) and others. In HypIR, the user enters natural language queries and can easily modify them. The paper introduces the underlying concepts, implementation details and the object-oriented nature of the HypIR system.

To prevent the user from being lost in the IR hyperspace, a new approach, browsing monitor, is introduced. Browsing monitor provides the advantages of system maps and history lists without their difficulties of usage.

In future research, we are planning to modify and enhance the system utilizing users' feedback. Two other future research possibilities are the design and implementation of an on-line thesaurus for better query formulation, and the development of performance measurement methods that would be appropriate for hypertext-based IR systems.

REFERENCES

[AGOS92] Agosti, M., Gradenigo, G., Marchetti, P. G. A hypertext environment for interacting with textual database. *Information Processing & Management.* 28, 3 (1992), 371-387.

[APPL88] Apple Computer, Inc. *Inside Macintosh I-VI.* Addison Wesley, Reading, MA 1988.

[BELK87] Belkin, N. J., Croft, W. B. Retrieval techniques. In Annual Review of Information Science and Technology, ARIST. Vol. 22, M. E. Williams, Ed. Elsevier Science, Amsterdam, The Netherlands, 1987, 109-145.

[CAN90] Can, F., Ozkarahan, E. A. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems.* 15, 4 (Dec. 1990), 483-517.

[CAN92a] Can, F. Incremental clustering for dynamic information processing *ACM Transactions on Information Systems,* to appear.

[CAN92b] Can, F. On the efficiency of best-match cluster searches. (Tentatively accepted by *Information Processing and Management.*)

[CAN92c] Can, F., McCarthy, K. J. Implementation of an information retrieval system (ANIRS) with ranking and browsing capabilities. Working paper 92-001, Dept. of System Analysis, Miami Univ., Oxford. Ohio, April 1992.

[CROF87] Croft, W. B., Thompson, R. H. I^{3}R: A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science.* 38, 6 (Nov. 1987), 389-404.

[CROU89] Crouch, D. B., Crouch, C. J., Andreas, G. The use of cluster hierarchies in hypertext information retrieval. *Hypertext '89 Proceedings.* (November 1989), 225-237.

[FALO85] Faloutsos, C. Access methods for text. ACM Computing Surveys. 17, 1 (Mar. 1985), 49-74.

[FRIS88] Frisse, M. E. Searching for information in hypertext medical handbook. *Communications of the ACM.* 31, 7 (July 1988), 880-886.

[GAUC92] Gauch, S. Intelligent information retrieval: An introduction. *Journal of the American Society for Information Science.* 43, 2 (March 1992), 175-182.

[HARM91] Harman, D. How effective is suffixing? *Journal of the American Society for Information Science.* 42, 1 (Jan. 1991), 7-15.

[HARM92] Harman, D. User-friendly system instead of user-friendly front-ends. *Journal of the American Society for Information Science.* 43, 2 (March 1992), 164-174.

[LIBC91] Library of Congress. *American Memory Instruction Manual.* Washington D.C., October 1991.

[NIEL90] Nielsen, J. *Hypertext and Hypermedia.* Academic Press, San Diego, CA, 1990.

[PORT80] Porter, M. F. An algorithm for suffix stripping. *Program.* 14, 3 (July 1980), 130-137.

[SALT88] Salton, G., Buckley, C. Term-weight approaches in automatic text retrieval. *Information Processing and Management*. 24, 5 (May 1988), 513-523.

[SALT89] Salton, G. *Automatic Text Processing.* Addison Wesley, Reading, MA, 1989.

[SHAF91] Shafer, D. *The Complete Book of Hypertalk 2.* Addison Wesley, Reading, MA, 1991.

[THOM89] Thompson, R. H., Croft, W. B. Support for browsing in an intelligent text retrieval system. *Int. J. of Man-Machine Studies.* 30, (1989) 639-668.

[VANR79] Van Rijsbergen, C. J. *Information Retrieval,* 2nd ed. Butterworths, London, 1979.

[WILL89] Willett, P. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management.* 24, 5 (1989), 577-597.