Hypermedia Links and Information Retrieval

Zhuoxun Li, Hugh Davis and Wendy Hall
The Department of Electronics and Computer Science
The University of Southampton

Abstract

Link creation in most existing hypertext / hypermedia products is a time consuming task. Microcosm is an open hypermedia system, in which various dynamic techniques have been used to attempt to ease the task of linking large bodies of information. This paper introduces the Microcosm model, and focuses on a technique for using computational power for dynamically creating links, known as retrieval links. The algorithm used for creating retrieval links is described and some preliminary experiments to assess the value of this facility are discussed.

1. Introduction

Hypertext / hypermedia systems have many potential applications in the field of information retrieval, since they provide new, and potentially very powerful, interfaces to information systems. The impact of hypertext / hypermedia on information retrieval is likely to be greatest on the development of multimedia information systems. This paper describes work being undertaken at the University of Southampton to integrate the two processes of hypermedia link following and information retrieval in a novel way.

Most hypermedia information systems depend upon specifically anchored links that have been manually created to allow the user to navigate through the information. However it is doubtful whether this method of linking information will scale up successfully to large information systems. Microcosm is an open hypermedia system that has been developed with the intention of dealing with large amounts of multimedia material **[Fountain90]**, and permits the manual creation of dynamically anchored links which may be followed from any point in the system. It is also possible to request that the system computes dynamic links to offer to the user. Both of these facilities reduce the amount of manual effort that is involved in creating links, but at the possible cost of reducing the quality of information available in the destination of the link. The implementation and integration of these computed links, known as retrieval links, is the primary topic of this paper.

2. The Microcosm Model

Microcosm is an open hypermedia System. Within Microcosm it is possible to browse through large bodies of multimedia information by following links from one place to another. It is also possible for the user to add links and further information to the system. In this respect Microcosm provides all the services that would be expected in any hypermedia system. However, Microcosm adds many significant features to this basic model, which place it at a higher level than most currently available hypermedia systems, and make it a particularly suitable environment for integrating data and processes. In order to understand the facilities that Microcosm provides it is necessary to examine the underlying model.

Microcosm consists of a number of autonomous processes which communicate with each other by a message passing system. No information about links is held in the document data files in the form of mark-up. All data files remain in the native format of the application that created them. Instead, all link information is held in link databases, which hold details of the source anchor (if there is one), the destination anchor and any other attributes such as the link description. This model has the advantage that it is possible for processes to examine the complete link database as a separate item, and also it is possible to make link anchors in documents that are held on read only media such as CD-ROM and video disk.

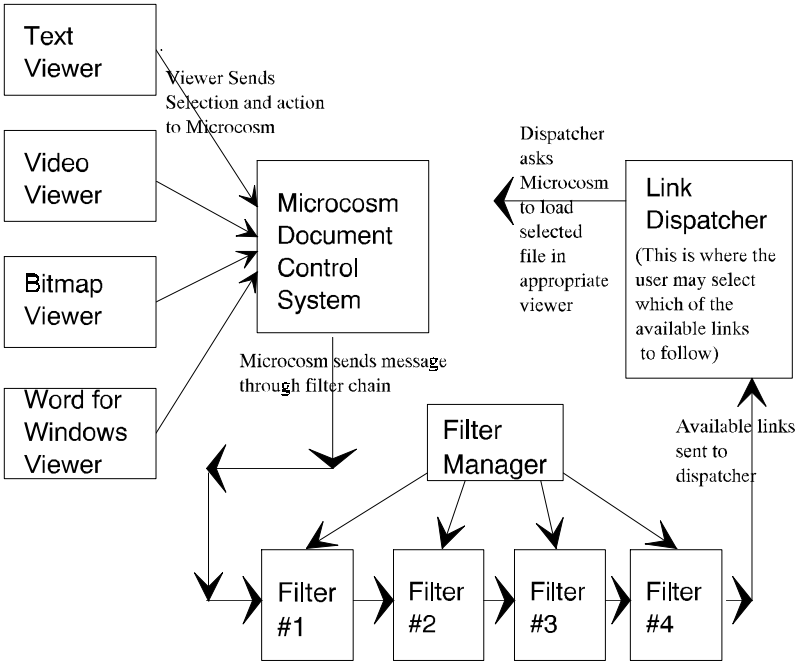
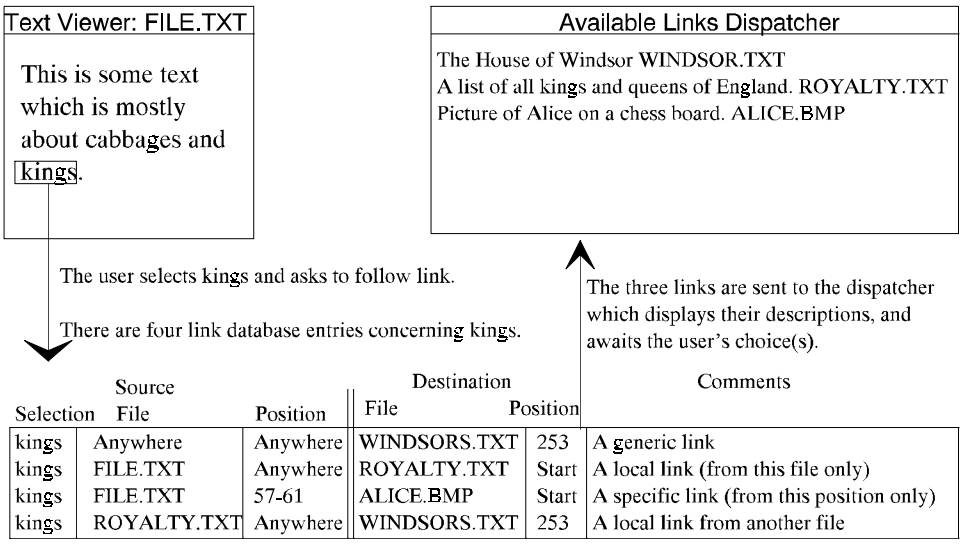


Figure 1: *The Microcosm Model*

Microcosm allows a number of different actions to be taken on any selected item of interest, so consequently use of the system involves more than simply clicking on buttons to follow links. In Microcosm the user selects the item of interest (e.g. a piece of text) and then chooses an action to take. We may see this as selecting an object then sending it a message. A button in Microcosm is simply a binding of a specific selection and a particular action. A particular feature of Microcosm is the ability to generalise source anchors. In most hypertext systems the source anchor for any link is fixed at a particular point in the text. In Microcosm it is possible for the author to specify three levels of generality of link sources.

1) **The generic link**. The user will be able to follow the link after selecting the given anchor at *any* point in any document.
2) **The local link**. The user will be able to follow the link after selecting the given anchor at any point in the current document.
3) **The specific link**. The user will be able to follow the link only after selecting the anchor at a specific location in the current document. Specific links may be made into buttons.

Generic links are of considerable benefit to the author in that a new document may be created and immediately have access to all the generic links that have been defined for the system.



Selection	Source File	Position	Destination File	Position	Comments
kings	Anywhere	Anywhere	WINDSORS.TXT	253	A generic link
kings	FILE.TXT	Anywhere	ROYALTY.TXT	Start	A local link (from this file only)
kings	FILE.TXT	57-61	ALICE.BMP	Start	A specific link (from this position only)
kings	ROYALTY.TXT	Anywhere	WINDSORS.TXT	253	A local link from another file

Figure 2: *Following Links*

The basic Microcosm processes are viewers and filters.

2.1. Viewers.

Viewers are programs which allow the user to view a document in its native format. Included with Microcosm are viewers for ten formats including various forms of text, bitmaps, video and audio.

The task of the viewer is to allow the user to peruse the document, to make selections and to choose actions. Typical actions are *follow link, make link* and *complete link* (where links may be to processes as well as to documents). The actions themselves are not effected by the viewer. The viewer is responsible for binding the information into a message, which is sent on to the filter chain where it will look for one or more processes that can satisfy this request. Any Windows application might be used as a viewer, with the proviso that it is possible to select objects, and at the least copy them to the clipboard. Microcosm is capable of taking actions upon objects on the clipboard. In cases (such as Word for Windows and Superbase) where the application has some level of programmability, it is sometimes possible to add a menu to the application so that the application is a viewer in its own right.

A major strength of Microcosm is its ability to integrate other applications. In fact Microcosm may be seen as an umbrella environment, allowing the user to make links from documents in one application package to documents in another application package. While many other hypermedia systems, for example Guide, allow the user to follow links *from* a hypermedia document *to* an external application, the facility to follow links from an external application into a hypermedia application, and therefore on to another external application is unusual. Microcosm provides a link service which allows other applications to appear to have hypermedia functionality.

2.2. Filters

Filters are processes which are responsible for receiving messages, taking any appropriate actions, and then handing the message on to the next filter in the chain. The actions that filters take will be of the nature of changing the message, or adding or removing messages. The order that the filters appear in the chain is under user control, and may be dynamically re-ordered and filters may be installed and removed. Some of the filters that are provided with Microcosm include:

Link Databases.
Link Databases hold all the information referring to links. More than one database may be installed at a time. It possible to have a concept of public and private databases.

Show Links.
When a user selects a chunk of text and uses the *Show Links* action this filter will organise the display of *all* available links, including those links which are not buttons, such as generic links.

Compute Links
Sometimes no links have been defined for a particular subject. On these occasions it is desirable to offer the user some further assistance. Microcosm has a facility that allows a user to batch a set of text files and to index these documents. Once this indexing has been done a block of text may be selected and the action, compute link, may be chosen. The system will very rapidly return a number of other documents within the system that have a similar vocabulary to the selected block, in the order of best match. This facility is analysed in greater detail in Section 3.

Other Navigational Filters
A well documented problem in hypermedia is that of remaining oriented while navigating about a network of nodes. (e.g. [**Hammond87**]) Microcosm provides some tools to assist users with navigation. There is a history mechanism that keeps a list of all documents that have been visited, and allows the user to return to a chosen point, a Mimic filter which allows the user to follow a tour through a set of documents pre-defined by an author and a book mark mechanism which allows users to mark document windows that they wish to keep on the screen. Further research is currently being conducted into methods of integrating other navigational tools, such as maps, into an open hypermedia system.

2.3. Implementations.
Currently Microcosm is implemented on Windows 3.x, and will require at least a 286 machine to run but is best on a 386 machine with at least 4 Mega Bytes of memory. A Beta test version of most of this software is available. Versions of Microcosm are under development for Apple Macintosh computers and for Unix machines running X-Windows.

3. Classification of links

Most hypertext / hypermedia systems only support links that allow the user to traverse the link from a specific source anchor point in a document to a specific destination. In designing Microcosm we have created three distinct kinds of links. The different classifications of links allow for different levels of access to the information in the system.

Some Terminology:

- A *static link* is a link for which both the source and the destination of the link have been well defined before it is used. The attributes of the static links are physically stored in the hypertext system. They may either be embedded in the information units (documents) or be stored separately.

- A *dynamic link* is a link for which the source and/or the destination of the link have not been defined until the link is followed. The source and/or the destination of a dynamic link is decided according to some rules at the time that the link is followed.

- *Computed links* are created by the system according to some predefined rules. Computed links may be either static links or dynamic links, depending on when the link creation is completed.

From the point of view of link creation, there are *manually created* links and *computed links*. From the point of view of following links, there are *static links* and *dynamic links*. A static link may be created either manually or by computing, whereas a dynamic link can not be created manually.

In the above terminology, most hypertext systems support manually created static links; the link attribute information is usually stored within the information unit or document in the form of hidden mark-up. However, by separating the link attributes from the documents we achieve certain advantages.

- The ability to implement Generic links. These are manually created, dynamic links, in that the source anchor is not fixed.

- The system remains open in the sense that the documents remain unchanged by Microcosm, and may be viewed, edited and processed by other applications.

- It is possible to batch process all the information in the link database.

- Link databases may be passed between users.

Clearly manually created static links (e.g. buttons) give the most specific access to information, since the author of the link has clearly defined the link between two specific pieces of information. This is why we call such links *specific links* within Microcosm. However such links also require a considerable amount of manual effort to create a suitable network through the information units.

Where specific links are not available, we have generic links. These allow us to follow a link from any point at which a given item, such as a word or phrase, occurs, to a specific destination. Clearly the quality of such information may be lower, as the author can only create the destination, and users may find ways of following such links from inappropriate sources. However, the manual effort is decreased as the author need only create the destination of the link once, and then the link will be available from any document.

Where no specific or generic links are available, we have *retrieval-links*. These are computed links which are dynamically created. Such links require no intervention from the author, but the quality of the information found by following such links may be lower than manually created links.

The remainder of this section is devoted to explaining how retrieval-links have been implemented within Microcosm.

3.1. Requirements of the retrieval process

In designing retrieval-links, one needs to consider how to integrate information retrieval into the hypertext environment and to convert the retrieval processes into dynamic links. One needs also to consider the proper design of information retrieval processes so that the advantages of hypertext systems and information retrieval can both be attained.

1. *Automatic indexing.*
Automatic indexing is a process that uses machine power to produce document identifiers which would be used to match with the identifier obtained from queries. Since one of our purposes of designing retrieval-links is to use machine power to replace some manual work, the ability to automatically index is essential.

2. *Fast retrieval speed.*
Because hypertext systems work interactively, it is not feasible to keep users waiting for a long time in order to follow a retrieval-link. Generally, a "follow link" operation should give a response as soon as possible. Research has shown that

"the difference between one system with a response of several seconds and another with sub-second response is so great as to make them seem qualitatively different." **[Akscyn88]**

3. Ease of use.
Users of a hypertext system may have various backgrounds. To avoid the necessity of learning extra skills before using a retrieval-link, a retrieval mechanism should be designed to be easy to use. In other words, the retrieval operations should be similar to the operations of following an ordinary link. This requires that the information retrieval queries should be expressed in natural language so that users can directly select part of a document as the source of a retrieval-link.

4. User controlled results.
The environments provided by hypertext / multimedia systems are suitable for browsing quite a large number of documents efficiently. This means that users are more involved in the retrieval process, and are able to assist in judging the usefulness of the retrieved documents. This user involvement also means that the retrieval process may have higher recall and a relative lower precision.

In the Microcosm implementation, the retrieved documents are ranked before being presented to users. The ranking is based on the similarity between the text in the query and the textual content of each document. The number of retrieved documents is controlled by a set of rules to avoid too many documents being offered to the user.

Here, the most important things to be considered, when designing an information retrieval mechanism for a hypertext system, are the speed of retrieval and the ability to automatically index. Conklin **[Conklin87]** pointed out that the most distinguishing characteristic of hypertext system is its machine support links, and another essential characteristic is the speed with which the system responds to link referencing.

3.2. Retrieval-links Design

The index information about documents is stored in an inverted file. The inverted file currently used is based on the single term frequency obtained from each document. Each term has a group of weights to mark the importance of the term in representing the documents in a collection. We believe that this information can reasonably represent the content of documents

Single term weightings in an inverted file can be extracted from documents by automatic indexing processing, can be stored in a simple data structure and then can be accessed quickly for retrieval calculation.

The following steps are used to form such an inverted file.

(1) Separate the words from the documents.
(2) Eliminate stop words by consulting a stop words list.
(3) Reduce words variants to a single form which is called the stem.
(4) Count the stem frequency for all remaining stems in each document and create weighted values for each stem in each document.

It is easy to see that all words remained after removing the stop words are included in the inverted file. This obviously can increase recall of retrieval, and as addressed previously, this is what we need.

3.2.1. Stop words

The stop words are those that are frequently used in textual documents but have no real meaning for retrieval. These words appear in almost all the documents, but they are not suitable to represent the contents of documents. We prepared our own stop words list that contains about 300 words. Experiments

showed that removing these words from documents could reduce the document's length by 30 to 60 percent. Obviously, the more stop words contained in stop words list, the less the size of inverted file.

3.2.2. Stemming algorithm

A stemming algorithm is used to reduce the variation in word forms. There are two advantages to using a stemming algorithm. First, stemming makes various words that share same stem look the same. This can improve the recall of a retrieval, since stemming increases the chance of matching. Second, stemming reduces the length of words used to characterise each document so the space used to store these stems is reduced as well.

After studying the possible choices, we chose the stemming algorithm designed be Lovins **[Lovins68]**.

This stemming algorithm is based on the longest-match. To stem a word means to remove the ending from the word. In order to do so, an ending list is used. Comparing endings of a word with the endings list, if more than one ending provides a match, then the ending which is longest should be removed. According to Lovins, to remove the longest ending is better than to remove several short endings one by one.

To cope with the spelling exceptions, two steps are used in stemming a word. First, the longest ending will be removed from a word, and then, the rest part of the word will be processed by another routine called the recording procedure. This will modify some of the remaining part according to 34 prefixed rules **[Lovins68]**.

3.2.3. Weighting documents' stems

It is the weighted values of stems that is stored in the inverted file to represent the content of the documents. A weighted values is produced according to the frequency of occurrence of a stem in a document. To avoid the length of a document affect the weighting, a stem frequency would be divided by the document length.

To save storage space used by the inverted file, a weighted values is stored by just one byte. This means there could be a maximum of 256 different weighted values. In other words, any weight value would finally be mapped to one of 256 values. We used a logarithmic function to achieve this mapping, rather than a linear function. This has the effect of giving much greater significance to the difference between a stem occurring once or twice, than to the difference between a stem occurring 99 or 100 times. If a query stem occurs twice in a document, the document is probably twice as significant as a document in which the stem only occurs once, whereas there is little difference in significance between documents containing 99 and 100 occurrences, which will both be heavily weighted anyway.

3.2.4. Query stem weighting

Research has shown that showed that weighted query stems can lead to more effective retrieval than unweighted query stems. Jones **[Jones72]** gave a weighting method called inverse document frequency. The idea was that the importance of a stem in retrieval was based on how this stem can distinguish one document from another. If a stem was shared by most of the documents in a collection, then it was less useful in distinguishing the documents. If a stem was only used by a few documents, then it was very useful in distinguishing these documents from others.

Let's suppose there are K stems in a query, then if the inverse document frequency method was used, the weight for stem i should be: $w_i=log(N/N_i)$ where N is the total number of documents in the collection and the N_i is the number of documents that shared the stem i. This weighting method, by favouring stems

shared by less documents, emphasises the function of stems as devices to distinguish documents from the collection.

In the retrieval procedure designed for retrieval-links, the weight of the query stem *i* was defined as: $w_i=N/N_i$, instead of $w_i=log(N/N_i)$. This change gave more emphasis to stems shared by less documents and simplified the similarity calculation. Since documents would be indexed by all the stems appearing in each document, it is important to stress the effect of the important stems.

3.2.5. Similarity function

The match between a query and documents is based on the weighted values associated with the document stems and the query stems. The similarity function we used was defined as:

$$S(q,d_j)=\sum_{i=1}^{k} w_i \times stem_{ij}$$

where *q* stands for a given query that contains *k* stems, d_j stands for document *j*, w_i is weight for query stem *i*, $stem_{ij}$ is weighted value of stem *i* in document *j*.

The similarity between the query and all documents needs to be computed in order to rank the documents according to their similarity to the query. Since in the inverted file all $stem_{ij}$ are stored continuously for a given stem *i*, the process of calculating all $S(q,d_j)$ using the above formula is very fast.

3.2.6. Results control

For each query, similarities between the query and all the documents in the collection is calculated. If none of the query stems were used in document *i*, then $S(q,d_i) = 0$, which means that the document *i* is unlikely relevant to the query. That does not means when $s(q,d_i) > 0$, document *i* is relevant to the query. As the similarity function we used does not produce normalised similarity values, there is no fixed threshold that can be used to decide whether a document is relevant to a query. So, the following rules are designed to produce a dynamic threshold.

(1) Calculate the average value of the similarities of all documents. Use the average value as a sub-threshold; only the documents whose similarity values are above the average value would be passed to next step.

(2) To avoid too many documents becoming destinations of a retrieval link, another sub-threshold whose value is defined as *M* is used to control the maximum number of destinations. Considering a document collection containing a large number of documents may have more relevant documents to a query than a small document collection, another sub-threshold whose value is defined as percentage *P* is used to control the maximum number of destinations of a retrieval-link.

(3) Rank the documents according to their similarity values in decreasing order. Then the $max(M,P)$ documents that have bigger than average similarity values are chosen as the destinations of the retrieval-link.

At present $M=5$ and $P=10\%$. For a collection that has 35 documents, a maximum of 5 destinations would be permitted. For a collection that has 100 documents, a maximum of 10 documents could become destinations.

In the above procedure, step one has the ability to control the relative quality of the retrieval results, step two is mainly used to avoid the relevant documents being buried in too many other documents.

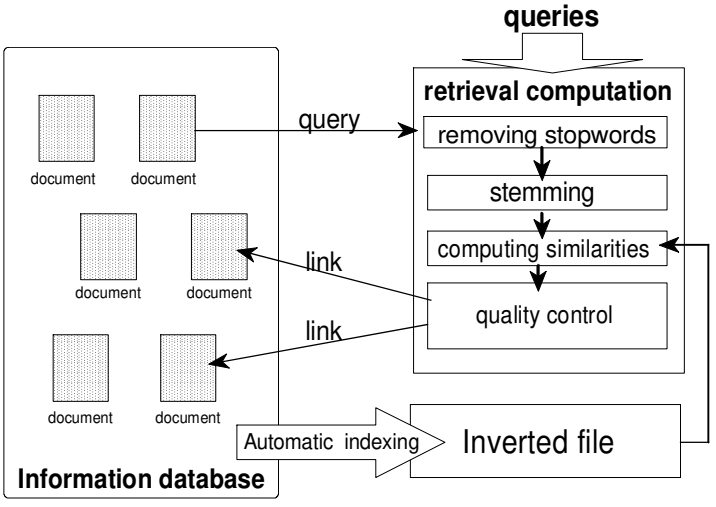


Figure 3: *Retrieval Links*

3.3. Experimental Results

To evaluate the design and performance of retrieval-links, experiments were carried out in retrieval effectiveness, response time and the time needed to create the inverted file.

To examine the performance of retrieval links, 20 departmental CSTRs (Computer Science Technical Reports) were used as a collection of documents.

As designed, the destinations of retrieval links are a group of documents. We expect that the destinations of retrieval-links are relevant to the source of the retrieval-links.

In one experiment, the *titles* of the 20 CSTRs were used as sources of the retrieval-links. In this case, we would expect that the destinations of retrieval-links would contain the document (expected-document) whose title was used as the source of the retrieval-links. Since the destinations of retrieval-links are ranked according their similarities to the source, we also expect that the expected-document will rank above all the other documents. The table below gives the experimental results.

Ranking of expected document	number of documents that satisfied the ranking	percentage
1	17	85%
2	2	10%
3	0	0%
4	1	5%

Figure 4: *Experimental results: using document titles as the sources of retrieval-links.*

In the above table, we can see that in most cases (85%), the expected-documents were ranked top of the destinations of retrieval links. Notice that there were no special weights for titles, authors names, and abstracts of documents in the inverted file, so a title received no more weight than a sentence in the document. In another experiment with 100 RFC network protocol documents, 92% of the expected

documents were ranked first. In a further experiment, some selected keywords were used as the source of retrieval-links and the results were also satisfactory. The decision to chose all stems from documents as keywords was made based on a similar consideration. Such results shows that the indexing procedures and similarity function calculations used by the retrieval mechanism produce satisfactory results, and further work will be conducted to verify these results using the CACM collection.

The response time of retrieval-links was tested for a document collection containing 176 documents. Experiments showed that when several words were used as the source of a link, there was no significant delay to follow a retrieval-link. For a query with about one hundred words, it needs a couple of seconds to produces the destinations of retrieval-links.

The retrieval mechanism is based on the inverted file which contains extracted information about each document. Since the retrieval process needs to access weighted values of a stem on all documents, while inverted file creation process can only process the documents one by one, a data structure for the inverted file must be chosen that either ensures a quick retrieval process or makes the creation of the inverted file simple. To ensure fast retrieval response, the data structure of the inverted file is optimised in the form that can be quickly accessed by the retrieval process. This makes inverted file creation hard work. By carefully designing the creation process, we can now create inverted files in reasonable time. For instance, to index 176 documents (total size over 2MB) takes about 15 minutes.

4. Conclusions and Further Work

Microcosm has been in use for around two years. The facility to create generic links has been welcomed by authors, but early studies indicate that users need greater prompting to investigate the possible existence of such links. Retrieval links have only been added to the system within the last few months. Once users discover this facility they tend to make considerable use of it. **[Davis92]**

There are a number of possible further improvements that we are currently investigating.

- It would be possible to allow authors a tool to use retrieval links to suggest links that might be followed, then to choose which links would made into static links by including them into the link database.

- One of the link attributes, stored in the link database is a short textual description of the data that will be found by following the link. We are investigating the possibility of using this information for either enhancing the current computed links accuracy, or for offering some level of computed links in an information set that has not been pre-indexed.

- Retrieval links currently offer destinations that are whole documents. We are investigating the possibility of enhancing the system to automatically offer destinations within the document when such a link is followed.

- We are investigating the use of a synonym filter, that would increase the probability that a search for a given word in an index would be successful.

- We are attempting to extend the index to include certain word pairs (phrases). It seems likely that where a phrase occurs in a query and a document there is a higher probability that this is a useful match than in the case where the words appear separately in the document. The current system is unable to distinguish between these two cases.

It is our belief that a computed links facility, enhanced as described above would make an extremely accurate and useful facility for both authors and users, and will enable hypermedia systems to make the transition into dealing with large information systems.

References

[Akscyn88] Robert M. Akscyn, Donald L. McCracken, and Elise A. Yoder. *KMS: A distributed hypermedia system for managing knowledge in organisations.* Communications of the ACM, 31:820-835, July 1988.

[Conklin87] J. Conklin. *Hypertext: an introduction and survey*. IEEE trans. computer, 1987.

[Davis92] Hugh Davis, Wendy Hall, Gerard Hutchings, David Rush and Rob Wilkins. *Hypermedia and the Teaching of Computer Science: Evaluating an Open System*. in David Bateman and Tim Hopkins (eds). *The Proceedings of the Conference on Developments in the Teaching of Computer Science*. The University of Kent. 1992.

[Fountain90] Andrew M. Fountain and Wendy Hall and Ian Heath and Hugh C. Davis. *MICROCOSM: An Open Model for Hypermedia With Dynamic Linking* in A.Rizk and N.Streitz and J. Andre (eds). *Hypertext: Concepts, Systems and Applications. The Proceedings of The European Conference on Hypertext, INRIA, France, November 1990*. Cambridge University Press 1990.

[Hammond88] Hammond, N.V and Allinson, L.J. T*ravels around a learning support environment: rambling, orienteering or touring?* In Soloway, E, Frye, D and Sheppard, S.B. (eds), *CHI '88 Conference Proceedings: Human Factors in Computer Systems*, ACM Press: New York, 269-273. 1988

[Jones72] Karen Sparck Jones. *A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation*, 28:11-21, 1972

[Lovins68] Julie Beth Lovins. *Development of a stemming algorithm. Mechanical Translation and Computational Linguistics*, 4:22-31, 1968.