Making Use of Hypertext Links when Retrieving Information

H.P. Frei, D. Stieger
Swiss Federal Institute of Technology (ETH) Zurich
Department of Computer Science
8092 Zurich, Switzerland

Abstract

Hypermedia links were invented to support the manual browsing through large hypertext or hypermedia collections. However, retrieving specific portions of information in such a collection cannot be achieved by browsing only; retrieval mechanisms are necessary. In this paper we show how to use the semantic content of hypertext links for retrieval. We present special purpose indexing and retrieval algorithms that exploit the node and link content. First retrieval results in a hypertext test collection are presented: the results are clearly better than those obtained when the links are ignored. The hope is that these results can be extended to hypermedia information and that they can be improved by more sophisticated indexing algorithms.

1 Introduction

The main idea of hyper documents is that documents — or parts thereof — can be brought into relation to each other and that additional information may be attached to any part of a document. The document parts are called nodes. When these nodes are connected by links a hyper document web results. Each hypermedia node constitutes an information item. Complex nodes are partitioned into simpler ones and — at the lowest level — are simple linearly organized hypermedia nodes of a single media type. These nodes and the links connecting them constitute a directed graph. A link λ (n_1, n_2) represents a connection from the source node n_1 to the destination node n_2.

Most of the conventional Information Retrieval (IR) algorithms have been developed for searching in large linear text collections [11]. They are not suited to retrieve information in non-linearly organized hyper collections. If they are applied to the individual linear nodes of hyper collections, the hyper structure — that contributes a great deal to the content of a hyper collection — is simply ignored and the retrieval results are accordingly poor. In addition, conventional IR algorithms are not suited to retrieve non-textual information as they employ textual descriptors as indexing features. To retrieve information in multimedia environments, suitable features have to be introduced for every individual medium as done for speech documents [5].

This paper focuses on the problem of exploiting the links when specific content-related information is to be retrieved. We present new IR algorithms that make use of the semantic content of the links involved. Currently, we are considering text information and textual descriptors exclusively. We hope that the methods developed are general enough to be extended to non-textual features and thus to real hypermedia collections.

2 Hypertext Nodes and Links

2.1 Hypertext Information

We already pointed out that hypertext information consists of two parts. First, there is the information contained in the hypertext *nodes*. Second, the interconnections between the nodes, the *links*, define the structure of the hyper document and the nature of every particular interconnection [6]. Therefore, IR algorithms have to deal with both parts: nodes and links. In this paper, we concentrate on using the semantic content of links and employ well-known IR algorithms to deal with the textual information contained in the nodes.

Links allow the user to discover relationships that are difficult to determine without a hyper structure. We distinguish two types of links [3]:

- referential links,
- semantic links.

The main purpose of referential links is comfortable reading of the document. The purpose of semantic links is to point to similar, more detailed, or additional information. The reasons for establishing such semantic links contribute to the *topic description* of the link.

Fig. 1 depicts a small example of a hypermedia or hypertext collection. The small part of the collection shown includes the two hypernets $N_1 = \{n_1, n_{1.1}, n_{1.2}, n_2, n_5, n_6, n_8\}$ and $N_2 = \{n_3, n_4, n_7\}$. The nodes $n_{1.1}$ and $n_{1.2}$ are sub-nodes of n_1; they structure node n_1 similarly to the way two paragraphs structure a chapter. The links connecting n_1 with these two nodes are referential links. On the other hand, the nodes n_3 and n_7 contain information related to the information of node n_4. This is the reason for the semantic links pointing from n_4 to n_3 and from n_4 to n_7.

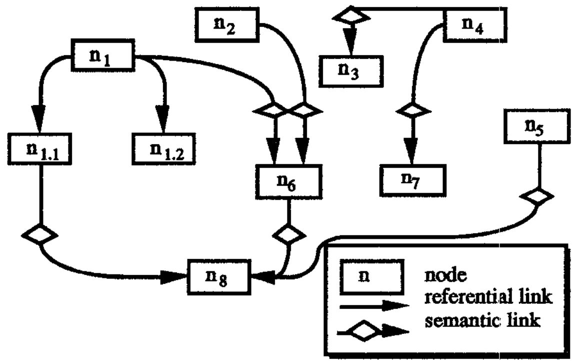

Fig. 1: Referential and Semantic Links

To facilitate content-specific retrieval, nodes *and* semantic links have to be indexed. The problem of how to index independent text nodes and queries has been discussed thoroughly in the IR literature [11]. This is why we concentrate here on the indexing of links and on the subsequent usage of indexing information.

2.2 The Nature of Links

Referential links serve the same purpose as foreign keys in a domain of a relational database. Therefore, they do not provide additional information to the topic of the document. They are plain pointers to improve both reading and browsing. Their use for information retrieval purposes is not discussed in this paper.

Conversely, semantic links provide additional information on the topic of the hyper document web. Semantic links point to nodes which would be difficult to find otherwise. Such nodes contain special annotations, corrigenda, or similar, contradicting, generalizing, specializing, or simply additional information. Every link has associated some well-structured attributes like creation time, author name, and the like. The information associated with a link may be consulted both by a 'reader' when browsing through the document and by a retrieval algorithm when processing a query.

A link consists of the following components:

$\lambda = \langle t, I, ls, ld\rangle$,

where

t	is the link type
I	is a set of structured link attributes (auxiliary link information)
ls	signifies the source node of the link λ
ld	signifies the destination node of the link λ

The *link type* t specifies whether the link is of type referential or semantic. In addition, the parameter t could be used later to distinguish more than these two types, in particular to distinguish several subtypes of semantic links. The intention is to restrict ourselves to a few link types so that their semantics may be understood fully by authors and users. This is in sharp contrast to other hypermedia paradigms which are based on up to 80 different kinds of links [15].

2.3 The Link Description

We associate a *link description* $\vec{\lambda}$ with every semantic link. This description contains mainly the content related reasons for the existence of the link, usually expressed by some topic descriptors (features). It is the result of an indexing procedure $\lambda \rightarrow \vec{\lambda}$ that takes the neighboring nodes of the link into account, in particular the source and destination nodes. In addition, the content-specific link description can be expanded or even changed by users when they read or browse through the hypernet. New links can be established by users when content-related relations were not recognized by the initial authoring and indexing processes.

The link description $\vec{\lambda}$ is a vector with feature weights as components. In the case of purely textual nodes (e.g. hypertext), the features are usually *weighted terms or phrases*.

As hyper collections result from a dynamic process, some nodes are more densly linked than others. Specifically, older nodes tend to be well linked whereas newer ones are often poorly linked. Thus we believe that the absence of a (semantic) link does not imply necessarily the absence of referential or semantic relations between the nodes. If there is no link we assume that possible dependencies are unknown. This is in contrast to Frisse's interpretation that says "the absence of a link (or a path of links) between two cards asserts that the utilities of the two cards are conditionally independent" [4]. The problem of missing links becomes apparent when a hyper document collection is in use: Poorly linked nodes are less likely to be found when browsing through the collection. Sparsely linked nodes also hamper those automatic retrieval methods that use links.

In what follows we assume that the indexing vocabulary contains m terms. The node description $\vec{n}$ and the link description $\vec{\lambda}$ are therefore given by:

$\vec{n} = < n_0, \ldots, n_{m-1} >$, where n_i with $0 \leq i < m$ are the term weights of node n,

$\vec{\lambda} = < \ell_0, \ldots, \ell_{m-1} >$, where ℓ_i with $0 \leq i < m$ are the topic descriptor weights of the link λ.

The link description $\vec{\lambda}$ depends mainly on the link type $t \in T$, on the source node ls, on the destination node ld, and possibly on an user expansion $\vec{u}$ represented by a feature vector:

$\vec{u} = < u_0, \ldots, u_{m-1} >$, where u_i with $0 \leq i < m$ are the weights of descriptors provided by users.

We define a simple link indexing function i:

$$i : T \times \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^m,$$

$$(t, \vec{ls}, \vec{ld}) \mapsto \alpha\,(f\,(t, \vec{ls}, \vec{ld}))$$

where $f : T \times \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ is a function modelling why the semantic link was established; the link type t may play an amplifying or negating role. The purpose of this function is to find *common abstract concepts* (expressed by weighted indexing terms) in the source and destination nodes. A similar idea was followed by Croft and Turtle in their probabilistic hypertext retrieval model [2, p. 219].

$\alpha : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is a (non-injective) mapping function reducing small components of the vector $\vec{\lambda}$ to 0. As the number of links in a hyper document collection is usually much larger than the number of nodes, the aim is to keep link descriptions as compact as possible.

The correction function i_u takes an existing link description $\vec{\lambda}$ and a user expansion $\vec{u}$ in order to create a modified link description $\vec{\lambda}'$:

$$i_u : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}^m,$$

$$(\vec{\lambda}, \vec{u}) \mapsto \alpha\,(\vec{\lambda} + \vec{u})$$

More elaborate link indexing functions would also take information outside of the immediate neighborhood of the two nodes into account. In this way, a weak destination node followed by promising descendants would also get a chance to be considered by a retrieval algorithm.

3 Retrieval Strategies exploiting Hypertext Links

3.1 General Considerations

When users browse, they normally follow only a small number of the existing links. Conversely, a retrieval algorithm may follow many links and may create both high retrieval costs and doubtful results. Retrieval experiments in a collection of bibliographic references showed that following citations — a kind of referential links—produces ambiguous results [8, 12]:

... An evaluation of the process shows that many useful content words can be extracted from related document titles, as well as many terms of doubtful value ... *[12, pp. 385]*

The hope is that our semantic links contain the information necessary to decide whether a further node should be visited by the retrieval algorithm or not. The proposed automatic navigation through the hyper web is governed by the following considerations:

- the further away from the initial node the IR algorithm searches the less likely it is to find suitable information;
- links are only followed when they promise to point to nodes containing information relevant to the query;
- it may become mandatory to visit a specific node (e.g., because of given restrictions during the retrieval process), depending on the retrieval resources available.

In this way the descriptions of semantic links control the navigation process of the retrieval algorithm. The existence of such link descriptions and their use for retrieval purposes constitute the main difference between our retrieval algorithm and approaches described elsewhere [4, 8, 9, 13].

3.2 Retrieval Algorithm and Retrieval Costs

A node n_i is said to be adjacent to node n_j if and only if there exists a link λ (n_i, n_j) or a link λ (n_j, n_i). Given a link λ (n_i, n_j), n_j is said to be the destination node of this link (or, less precisely, a destination node of the node n_i), conversely n_i is a source node of n_j. The *outdegree* of a node n is the number of links with n as source node, the *indegree* is the number of links with n as destination node.

The retrieval algorithm determining a Retrieval Status Value (RSV) between the query q and the hypernet node n includes an initialization and a navigation phase:

Initialization phase, step ①

```
①  for all nodes n of collection do                         {standard retrieval}
       $RSV^{q,n}_0$ := similarity (q, n)                      {initial RSV}
   end for
   for all nodes n of collection do                         {hypertext retrieval}
       rsv := $RSV^{q,n}_0$
       navigation (n, rsv, 1)              {navigation procedure described below}
       InsertList (q, n, rsv)                             {result, to be sorted}
   end for
```

- Navigation phase, iteration of decision step ② and navigation step ③.

```
   procedure navigation (in n, in/out rsv, in distance)
②  if (outdgree (n) > 0) and (distance $\leq$ maximum_distance) then
       for all destination nodes n' of node n do
           if sim (q, $\lambda$ (n, n')) > threshold_value then
③              update rsv
               navigation (n', rsv, distance+1)
           end if
       end for
   end if
```

We use the definitions for *walk, trail,* and *path* in the same sense as they are used in graph theory. If (n_0, n_1, ..., n_d) is a sequence of nodes of a hypernet H, such that n_i is adjacent to n_{i+1} $\forall$ $0 \leq i < d$, then these nodes and the corresponding links are called a *walk of length d.* If the links are distinct, this walk is called a *trail of length d.* If all the nodes of a trail are distinct, the trail is called a *path of length d.*

The *navigation distance* is the minimum path length from a given reference node n in the hyper web to any node that can be reached from this node. If a maximum navigation distance is given, the retrieval algorithm ignores those nodes whose distance from the reference node exceeds the maximum distance.

We assume a homogeneously linked hyper collection with nodes of an average outdegree of k. If a node traversal starts from a given reference node n, the number of traversed nodes of all possible paths with length d grows exponentially. To limit the retrieval costs, a navigation strategy with restrictive propagation must be found.

Two different *hypermedia retrieval strategies* can be distinguished:

a) A user who knows little about the hyper collection is looking for an entry point convenient to start browsing. Here the aim of the retrieval strategy is to retrieve nodes that represent the interest of the user. For this purpose an *exhaustive search* through the network is necessary. The potentially large search effort can be limited by taking into account the descriptions of the semantic links.

b) The user is "sitting" on a node satisfying some of her or his information needs. In this case an automatic *navigational search* along the hyper structure is performed to find further useful nodes.

In what follows we describe these two strategies, namely, the exhaustive search and the navigational search along semantic links.

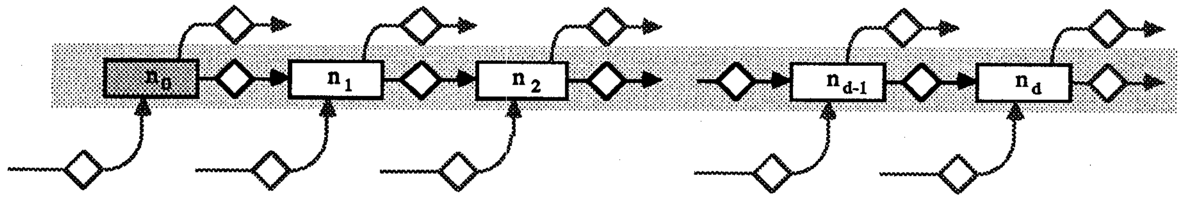

Fig. 2: Navigation Distance

3.3 Exhaustive Search

The aim of the exhaustive search algorithm is to retrieve nodes that are in the focus of the user's interest. Each node in the hyper collection is regarded as a reference candidate from where a search may start.

① Initialization phase:

An initial $RSV_0^{q,n}$ is obtained by determining the similarity between the reference node n and the query q given by a m-dimensional real vector $\vec{q} = < q_0, \ldots, q_{m-1} >$. This can be done by applying a conventional retrieval function ρ [11]:

$$\rho : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}, (\vec{q}, \vec{n}) \mapsto \rho(\vec{q}, \vec{n})$$

This retrieval function was called 'similarity' in 3.2.

② Decision step:

The algorithm decides if a navigation step to the next node has to be performed. This is the case if the link description signalizes the existence of information related to the query in adjacent nodes. This can be determined by the similarity function σ between the query description $\vec{q}$ and the link description $\vec{\lambda}$:

$$\sigma : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}, (\vec{q}, \vec{\lambda}) \mapsto \sigma(\vec{q}, \vec{\lambda})$$

If $\sigma(\vec{q}, \vec{\lambda})$ is smaller than a threshold value v, navigation is prevented. In addition, a predicate P on the attribute set I limits navigation. In this way the function sim of chapter 3.2 becomes:

$$\big((\sigma(\vec{q}, \vec{\lambda}) > v) \text{ AND } P\,(I)\big)$$

③ Navigation step:

Each time the algorithm navigates, the RSV with respect to the reference node is modified. This modification depends primarily on the destination node and on the distance from the current reference node. Furthermore, it can depend on the net topology.

The modification of the current RSV when a navigation step is performed from a node of distance d to a node of distance d+1 is calculated as follows (n_i^{d+1} denominates the destination node i of node n at distance d+1):

$$RSV_{d+1}^{q,n} := RSV_d^{q,n} + w_d \cdot \sum_i RSV_0^{q,n_i^{d+1}}$$

w_d is a *propagation factor* depending on the navigation distance previously covered and on the net topology. We are considering to use the 'link quality' — a function of σ — as an auxiliary weighting factor.

The method presented for the exhaustive search approach — in particular the choice of specific retrieval functions and weighting factors — will be discussed in chapter 4 where an example is introduced.

3.4 Navigational Search

The navigational search deals with the situation where a user is 'sitting on' a node essential to the scope of her or his information need. The assumption then is that further interesting nodes can be found in the neighborhood of this reference node provided that the retrieval algorithm follows suitable semantic links. The retrieval algorithm first determines a subset of further nodes to be visited depending on the maximum navigation distance (cf. 3.2).

Subsequently, a search is initiated as described in the previous chapter. It is to be noted that this navigation is limited by the propagation threshold value v and the predicate P as described in the previous chapter.

4 Some Experiments

4.1 Test Collection and Indexing Functions

To show the effects of the method presented, we used a conventional test collection which was expanded automatically to a hypertext. The collection consists of a subset of the INSPEC collection comprising 2472 documents and 65 queries [7]. The expansion to a hypertext was made by considering documents to be nodes and establishing *two* directed links between nodes with *common phrases* in their manual descriptions (paragraph 'DE'). The result was a hypertext consisting of a large net of 2397 nodes and 7 independent small nets (1 of 5, 2 of 4 and 4 of 2 nodes). In addition, there are 54 single independent nodes without any links to the rest of the collection. The average *outdegree* is 23.07, the average path length between two arbitrary nodes within a net is 4.2. This represents a *densly linked* hyper collection.

The node and the query descriptions were obtained by applying Porter's word stemming [10] and a term weighting of *tf·idf* [11, p.63]. As all links are interpreted as *semantic links*, the link type t was omitted. Therefore, the link indexing function i is reduced to

$$i_{simple} : (\vec{ls}, \vec{ld}) \mapsto \alpha_{20}\,(f\,(\vec{ls}, \vec{ld}))\ ,$$

where $\vec{ls}$ and $\vec{ld}$ are vectors whose components are the frequencies of the terms occuring in source and destination nodes respectively. For simplicity reasons, α_{20} restricts the resulting vector to the 20 highest valued weights (all other vector components are set to 0).

Three different link indexing functions f are applied and their performance is compared:

f_1 :	$\vec{ls} + \vec{ld}$	topics covered by both nodes
f_2 :	$\vec{ls} + \vec{ld} + \vec{u}$	with additional link information (by 'user')
f_3 :	$\vec{u}$	'user' link information exclusively

The function f_1 represents a simple attempt to provide a first link description. The idea is to express the common topics of both involved nodes.

The 'user' link information $\vec{u}$ was *not* provided by a real user in our particular case but was determined automatically. It was obtained by decomposing the phrase of paragraph 'DE' — which established the link — into reduced words (applying Porter's algorithm [10]).

Where function f_2 simulates a plausible real situation — an initial link description is modified by an auxiliary *manual* description — the function f_3 is only of academic interest. In real hyper collections the 'user' link information will probably be provided by relevance feedback mechanisms as pointed out in [14] and only in very few and special cases by a manual link description. Conversely, the function f_3 allows us to compare the simple automatic approach f_1 with an elaborated manual link description.

For the following experiments the functions ρ (initialization phase) and σ (link similarity function) are identical to the cosine measure [11, p.121].

4.2 Basic Evaluations

Fig. 3 shows the result of applying the link description function f_1 (i.e. $\vec{ls} + \vec{ld}$). The propagation distance was restricted to 1. The effectiveness was measured by taking the *average* number of relevant nodes of the 20 nodes with the highest RSVs (we chose this simple measure for reasons of simplicity in the data representation, as each of the following graphs represents the evaluations of at least 600 complete retrieval evaluations of 65 queries on 2472 nodes). The other two axes show the parameters 'propagation factor' w_d (chapter 3.3) and 'threshold value' v (chapter 3.3) determining if a propagation has to take place.

The constant 'platform' for large values of v shows the effectiveness level of the algorithm. High values of v prevent the propagation. Low values of w_d, in particular $w_d = 0$, prevent the RSV to be incremented. In both cases the initial retrieval status value $RSV_0^{q,n}$ (cf. 3.2) remains unchanged.

A threshold value of v = 0 returns the best results. Although 39.6% of the links show a similarity of σ ($\vec{q}$, λ) = 0 (and therefore are not taken into account), the retrieval costs are very high for v = 0 as the algorithm visited 13.9 links per node on the average.

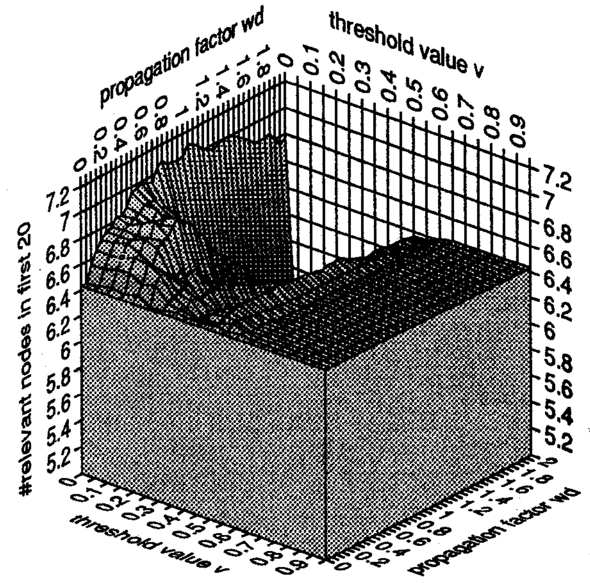

Fig. 3: Evaluation $f_1 = \vec{ld} + \vec{ls}$

It turned out that taking *all* links into account (on the average 23 links) in this experiment does not provide significantly different retrieval results compared to the ones with threshold value v = 0 (not shown in Fig. 3). Conversely, newer results of experiments performed on another collection [14] that suggest link descriptions (based on f_1) may enhance the retrieval effectiveness even when taking significantly less links into account. Taking *fewer* links — and therefore reducing the retrieval cost — provides satisfactory results only for small propagation factors w_d.

The main problem is to balance retrieval cost against retrieval effectiveness. We are measuring the effectiveness by comparing a method A with a method B. If no method B is explicitely specified, we compare a given method with the retrieval that does not take links into account.

As an example let us look at v = 0.2 and w_d = 0.45: the improvement is only 7.7% on the average (compared with 10.3% at v = 0 and w_d = 1.05). In this example only 0.56 links per node are followed on the average.

Fig. 4 and 5 show the improvement by taking an auxiliary user description into account. Fig. 4 shows the result when the additional features are weighted with their term frequency in the phrase.

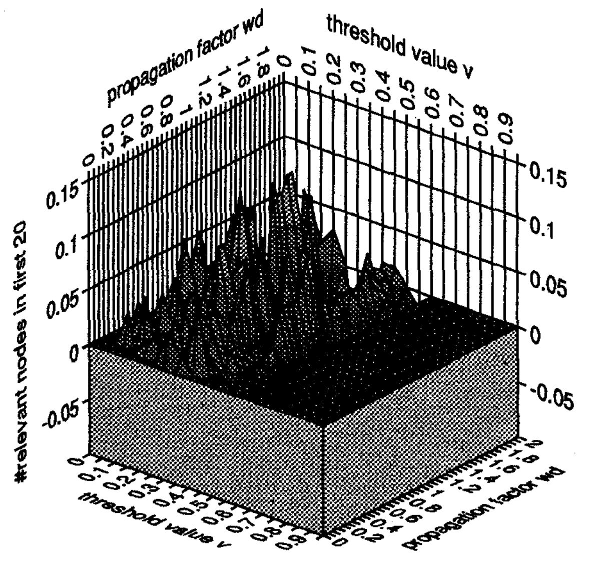

Fig. 4: Improvement by a 'Virtual' User Description: $f_2 = \vec{ls} + \vec{ld} + \vec{u}$, simple weight

Fig. 5 shows the result when the weights of the auxiliary user features are doubled to increase their influence.

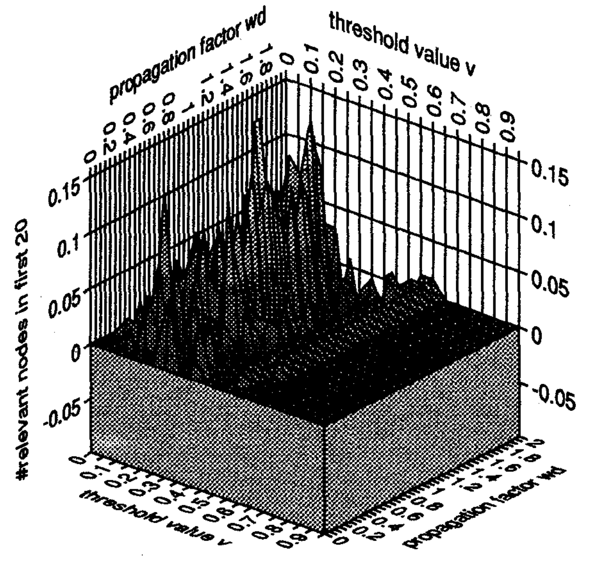

Fig. 5: Improvement by a 'Virtual' User Description: $f_2 = \vec{ls} + \vec{ld} + \vec{u}$, stressed weight

The improvement compared to the evaluation of f_1 is relatively small in both cases. This result is not very surprising. As mentioned in 4.1 our 'user description' was derived automatically from phrases similar to the way descriptors were obtained in the Cranfield Project [1]. These Cranfield experiments showed that retrieval effectiveness is not increased when a controlled vocabulary and phrases are used compared to using single terms automatically extracted from the documents. Also, our 'user description' originated from a manual indexing of the nodes instead of being a real description of the links. Also it is debatable whether a decomposition of phrases is suitable because of the loss of information.

Fig 6. shows the effect of applying the link indexing function f3:

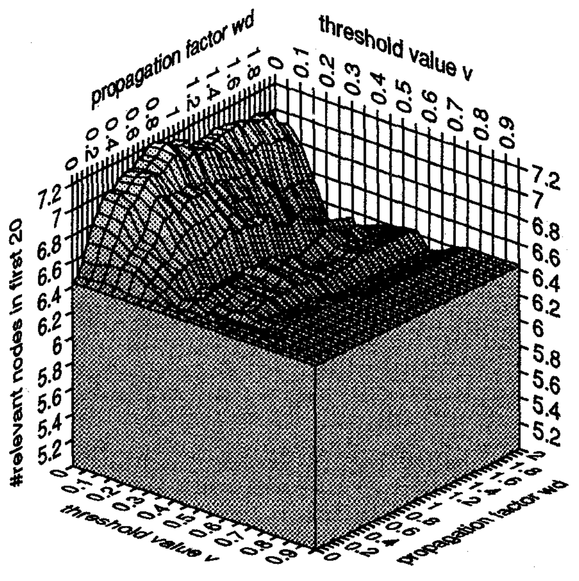

Fig. 6: User Description Only: $f_3 = \vec{u}$

A small improvement of the retrieval effectiveness can be gotten by using the indexing function f_3, i.e. the additional user description only. This provides two further advantages:

- The retrieval costs are significantly lower:
  Only 1.79 nodes are visited on the average for a threshold value of v = 0. With v = 0.1, only 1.25 and with v = 0.2, only 0.53 nodes are visited.
- The method is less sensible to variations of w_d and v:
  The small number of — in the major part — specific features in the semantic link description allows only a very restrictive navigation; the visited nodes seem more useful than the nodes that are visited when applying the indexing functions f_1 and f_2. As far as we know there are no problems with outliers. Applying a median function instead of the average function during the navigation step (cf. 3.2) does not modify the results significantly for the indexing functions f_1 to f_3.

This result shows that besides the over-all effectiveness — which behaves as expected — other factors determine the quality of a link description. A good choice of the link description seems to be crucial for stable retrieval algorithms.

4.3 Visiting Nodes with Distances larger than 1

To show the effects when taking nodes up to a distance of 2, we experimented with the link indexing functions $f_2 = \vec{ls} + \vec{ld} + \vec{u}$ (Fig. 4) and $f_3 = \vec{u}$ (Fig. 6). In Fig. 7 and 8 the propagation factor w_d for the distances 1 and 2 are depicted instead of the axes w_d and v. The value axis represents the average number of relevant nodes within the 20 nodes with highest RSV.

Because *bidirectional links* are generated in our experiments, two series of experiments can be done for all parameter combinations:

- An evaluation without restriction, i.e. the algorithm may return to the node it is coming from.
- An evaluation where the backward link is blocked when the corresponding forward link is activated.

Although we did both types of experiments, only the ones with blocked return links are presented here. The results are in fact very similar, but we think that it is too early to make general conclusions.

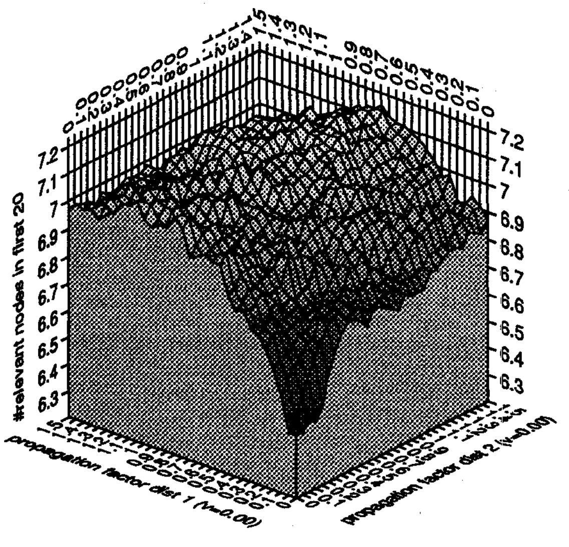

Fig. 7: Distance 2 (v=0) for function f_2

Fig. 7 shows the evaluation using the node indexing function f_2. Fig. 8 uses the function f_3. The values of v are 0 for both distances 1 and 2. The two graphs show a smaller improvement when distance 2 nodes are added as opposed to the improvement gotten when we added the distance 1 nodes. This can be explained by the decreasing similarities between the query and the nodes that are further away from the reference node. It goes without saying that the retrieval cost grows extremely fast with increasing distance.

Fig 9 and 10 show the evaluations for threshold values higher than 0 (to reduce retrieval costs).

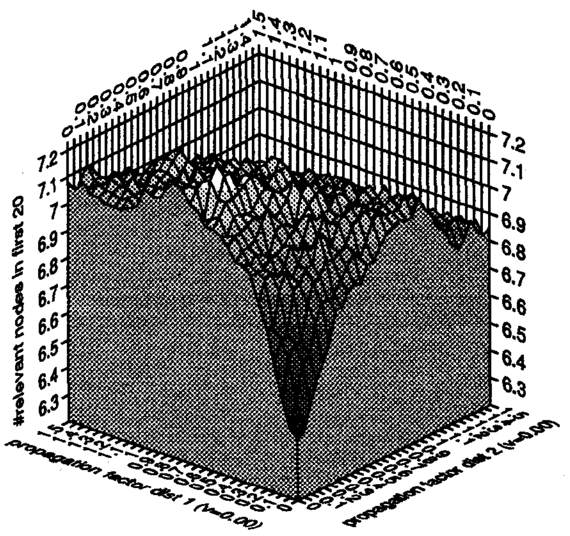

Fig. 8: Distance 2 (v=0) for function f_3

Both evaluations were performed with the link indexing function f_2. Fig. 9 shows the threshold pair $v_{dist=1}= 0.05$ and $v_{dist=2}= 0.10$; Fig. 10 the pair $v_{dist=1}= 0.10$ and $v_{dist=2}= 0.20$.

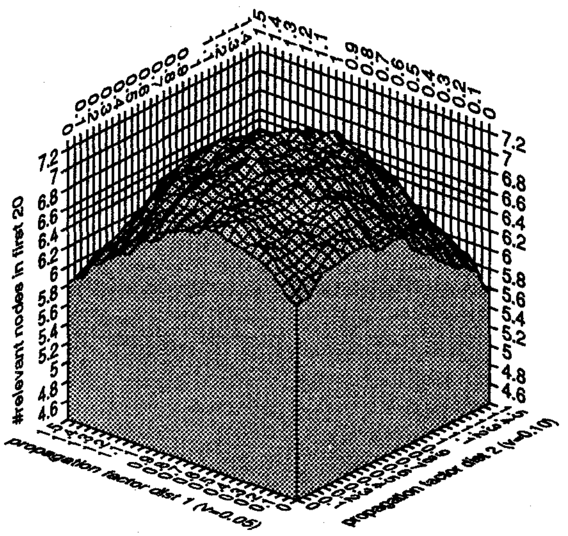

Fig. 9: Distance 2 (v=0.05/0.10); $f_2 = \vec{ld} + \vec{ls} + \vec{u}$

In our — densly linked — test collection it is questionable if it is worth propagating up to distance 2 given these circumstances. A small improvement of the effectiveness must be paid for with a significant increase of retrieval costs.

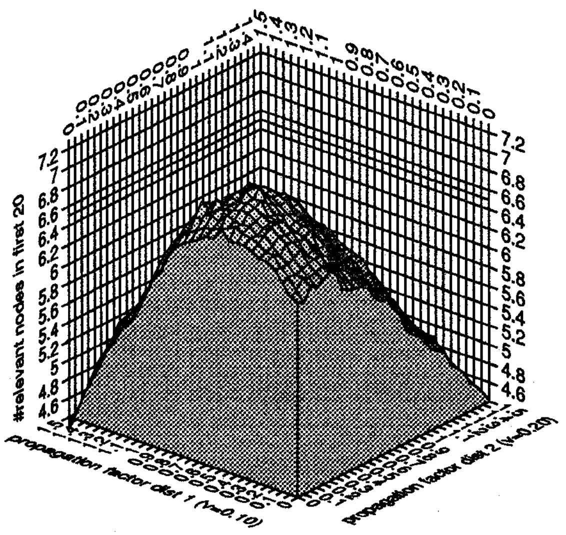

Fig. 10: Distance 2 (v=0.10/0.20); $f_2 = \vec{ld} + \vec{ls} + \vec{u}$

There is evidence that in a less densly linked collection it might be worthwhile to take distances larger than 1 into account. When we removed 50% of the links randomly in our test collection there was — of course — a smaller effectiveness improvement with distance 1. On the other hand, we still observed a significant improvement with distance 2. This effect is reproducible with all the link indexing functions presented here.

5 Current and Future Work

The experiments described in this paper showed how important it is to select the parameters properly. These parameters may cause both good effectiveness and high retrieval costs — depending on their values. The aim is to identify parameters so that efficient retrieval under time restrictions is possible. This is important as real life hyper collections will be very large in the not too distant future. Likewise, the amount of memory necessary to store the link descriptions seems high. Another area of investigation is the design of better indexing functions to generate powerful link descriptions with small memory demands.

Some experiments were performed on the Berkeley-UNIX online manual pages. The 'SEE_ALSO'-references were indexed with different indexing functions and a relevance feedback mechanism was used to improve these link descriptions [14]. The feedback experiments gave very promising results. Such a feedback mechanism is also applicable on real hypermedia collections even in cases when the link descriptions were empty at the outset.

Further experiments will be necessary to investigate the role of the source and destination nodes when links are automatically indexed. Furthermore, it should be possible to find approximations for missing node descriptions, e.g. for not indexed nodes that contain not supported media types.

6 Conclusions

We showed that it is possible to enhance the retrieval effectiveness by having the retrieval algorithms follow semantic links thus considering the content-oriented neighborhood of a reference node. Semantic links have to be established between nodes whenever appropriate for this purpose. These links must be indexed by a suitable indexing method that delivers a link description depending on the content of at least the two adjacent nodes.

We devised retrieval algorithms taking the descriptions of both the nodes and the links into account. After evaluating the similarity between the query and a reference node, these algorithms determine whether links should be followed and if so, which links are the most promising ones to follow. The experiments we carried out with a hypertext test collection showed encouraging improvements in retrieval effectiveness.

References

[1] Cleverdon, C.W., Keen, E.M.: *Factors Determining the Performance of Indexing Systems.* Vol. 1 and 2, Aslib Cranfield Research Project, Cranfield, England, 1966.

[2] Croft, W.B., Turtle, H.: A Retrieval Model for Incorporating Hypertext Links. *Proc. of the 1st European Conf. on Hypertext (ECHT) 90,* Ed. A. Rizk, N. Streitz, J. André, Cambridge University Press, 1990, pp. 213-224.

[3] Frei, H.P., Schäuble, P.: Designing a Hypermedia Information System. *Proc. DEXA '91,* Springer-Verlag, Wien, 1991, pp. 449-454.

[4] Frisse, M.E.: Searching for Information in a Hypertext Medical Handbook. *Commun. ACM 31,* No. 7, July 1988, pp. 880-886.

[5] Glavitsch, U., Schäuble, P.: A System for Retrieving Speech Documents. *Proc. 15th ACM SIGIR Conf.,* SIGIR Forum, ACM Press, June 1992, pp. 168-176.

[6] Haan, B., Kahn, P., Riley, V., Coombs, J., Meyrowitz, N.: IRIS Hypermedia Services. *Commun. ACM 35,* No. 1, January 1992, pp. 36-51.

[7] Harding, P.: Automatic Indexing and Classification for Mechanised Information Retrieval. *British Library R&D Department, Report 5273*, 1982.

[8] Kwok, K.L.: On the Use of Bibliographically Related Titles for the Enhancement of Document Representations. *Inf. Proc. & Management, Vol. 24, Nos. 2*, 1988, pp. 123-131.

[9] Lucarella, D.: A Model for Hypertext-Based Information Retrieval. *Proc. of the 1st European Conf. on Hypertext (ECHT) 90*, Ed. A. Rizk, N. Streitz, J. André, Cambridge University Press, 1990, pp. 81-94.

[10] Porter, M.F.: An Algorithm for Suffix Stripping. *Program,* Vol. 14, No. 3, 1980, pp. 130-137.

[11] Salton, G., McGill, M.J.: *Introduction to Modern Information Retrieval.* Int. Student Edition, 2nd printing, McGraw-Hill, 1984.

[12] Salton, G., Zhang Y.: Enhancement of Text Representations using Related Document Titles. *Information Processing & Management*, vol. 22, 1986, pp. 385-394.

[13] Savoy, J.: Ranking Schemes in a Hypertext Retrieval System. *Publ. 811, Dépt. d'informatique et de recherche operationelle Univ. de Montréal*, Fev. 1992 (see also Publ. 794, 799, 804, same author, 1991).

[14] Stieger, D.: Verbesserung der Kantenbeschreibungen in Hyperkollektionen durch Relevanzrückkoppelung. To appear in: *Proc. of Hypertext '93,* Springer Verlag, Zurich, March 2-3, 1993.

[15] Trigg, R.H.: *A Network-based Approach to Text Handling for the Online Scientific Community.* PhD. Thesis, University of Maryland, 1983.