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 $\lambda(n_1, n_2)$ 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.

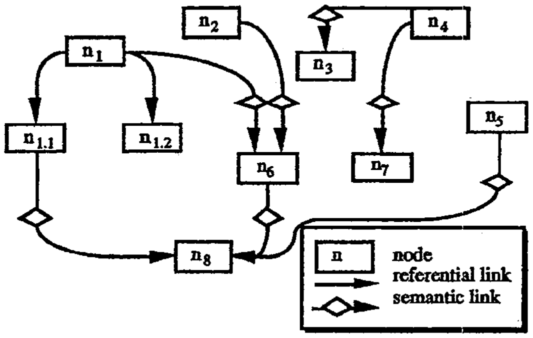

Fig. 1: Referential and Semantic Links

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 $0 \le i < m$ are the term weights of node n,
$\vec{\lambda}$	$= < \ell_0, \ldots, \ell_{m-1} >$	, where ℓ_i with $0 \le i < m$ 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 $0 \le i < m$ 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 $\lambda\,(n_i, n_j)$ or a link $\lambda\,(n_j, n_i)$. Given a link $\lambda\,(n_i, n_j)$, 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}
        $RSV_0^{q_n}$ := similarity (q, n)                            {initial RSV}
    end for
    for all nodes n of collection do                          {hypertext retrieval}
        rsv := $RSV_0^{q_n}$
        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 $\le$ 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} $\forall$ $0 \le i < d$, 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.

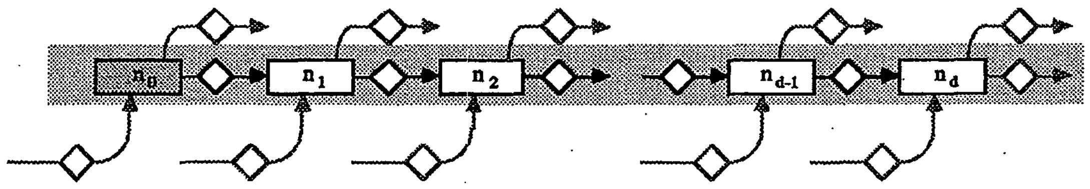

Fig. 2: Navigation Distance

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]:

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

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 $\vec{\lambda}$:

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

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:

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

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).

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.