A Framework for Integrating Browsing and Searching in Hypertext Systems

Dario Lucarella, Antonella Zanzi

1. Introduction

Hypertext can be defined as a system for managing a collection of information that can be accessed non sequentially. It consists of a network of nodes and logical links between nodes. The variety of nodes and links that can be defined make hypertext a very flexible structure in which information is provided both by what is stored in each node and by the way the information nodes are linked to each other [23].

Current hypertext systems provide sophisticated user interface tools that enable the reader to inspect the node content and to navigate through the network by selecting the path to follow, on the basis of interests emerging along the way. Many systems also provide facilities for the readers so that they can add their own links reflecting their association criteria. Anyway, the links defined by the author as well as those added by the user can be conceived as hardwired links in contrast to inferential links discussed further on.

On the other hand, much research has been carried out in the field of information retrieval systems. A comprehensive survey of such efforts can be found in [27]. Given a document collection, information retrieval systems are designed to provide, in response to a user request, references to a set of documents that are likely to contain the information desired by the user. So the emphasis here is on query facilities and related search strategies to locate useful information items.

We could say according to Frisse [2] that information retrieval systems emphasize the relative autonomy of text nodes providing sophisticated techniques for retrieving nodes as answer to user requests. Conversely, in hypertext systems, the semantic link structure is central for a system that provides sophisticated tools for graph traversal and node presentation. The former approach emphasizes searching, the latter emphasizes browsing.
It is reasonable to expect that the interaction of hypertext and information retrieval systems can bring obvious advantages to the process of knowledge transfer. On the one hand, hypertext tends to be user-directed allowing users to pick and choose their own path through the information map, ignoring some things and exploring others in depth. On the other hand, a retrieval system tends to steer the user down a path that is determined by the answer to the user request. By combining the features of both kinds of systems, it is possible for two way communication to take place. The system can present the user with information and guidance in a way that its retrieval strategy suggests. The user can arbitrarily go off to explore or learn more about pieces of knowledge along the way.

Some approaches to the integration of query-based retrieval strategies and browsing in hypertext network have been proposed recently [9,11,12]. In this context, the purpose of the current paper is twofold: a) to approach the retrieval task as an inference process relying on uncertain knowledge; b) to show that such a knowledge can be effectively embedded into a hypertext network.

The paper has the following structure. Section 2 emphasizes briefly the differences between browsing and searching. In section 3 a hypertext network is regarded as a knowledge base and in section 4 inference rules for retrieval are presented. A detailed description of the search strategy is reported in section 5. In section 6 an overview of the prototype system is given. Experimental results are summarized in section 7 and further research perspectives are outlined in the final section.

2. Browsing vs. Searching

As reported in [4], browsing can be characterized as going from where to what (presumably you know where you are in the database and you want to know what is there) whereas searching can be characterized as going from what to where (presumably you know what you want and you wish to find where in the database it is).

It is interesting to note that both styles of information selection (conventional retrieval versus hypertext) have their analogies in the process of retrieval from human memory. In cued retrieval, a person is given a cue and is asked to provide the information specified by that query. In contrast, associative retrieval returns information that is in a certain way related by
analogy to the starting information rather than an explicit answer to a query.

Assume, for example the user is reading a text passage and consequently would like to locate semantically related items. He or she should simply mark the passage that has stimulated such interests and activate a search button. The system, using the passage as a cue and the hypertext as a knowledge base, returns to the user a set of related nodes arranged in decreasing order of presumed semantic closeness. Essentially new links are inferred dynamically on the basis of the explicit link structure embedded into the hypertext network. So we can distinguish between two different kinds of links: structural links which are preset by the author or added by the reader and inferential links which are deduced automatically by the system.

Once we understand the essential difference between hypertext and conventional text retrieval, it is actually misleading to question which method is better: the two approaches serve different goals and, just for that, it is reasonable to combine them together. Hypertext may supplement conventional methods of information retrieval by allowing users to discover retrieval cues that successively can be used for query formulation, while search facilities may supplement conventional hypertext by providing the user with a set of relevant nodes for graphical browsing.

The problem centers on defining a model of an integrated retrieval system and how to implement a search strategy which is both effective and efficient. In the last years, van Rijsbergen [25] and Croft [7,10] have proposed a model of retrieval based on plausible inference. In this approach the retrieval process is regarded as the problem of determining an implication relationship between a document and a query and assessing the plausibility of that implication. Semantic inference made while processing the query may be considered as the main task in proving the logical implication between the document and the query.

In further detail, the approach introduced by van Rijsbergen is based on non-classical logic [25] whereas the approach of Croft is based on Bayesian inference nets [10]. In the same direction, Biswas et alii explore the use of Dempster-Shafer theory for evidence combination [3]; Tong et alii describe a production system that can perform evidential reasoning with queries expressed in a language of rules [29]; Watters proposes a logic framework for information retrieval [30]; and Lucarella presents an approach based on fuzzy set theory [18,19]. In the remaining part of this paper, we will show how such an approach fits into hypertext and satisfies the retrieval requirements emphasized above.
3. Hypertext as a Knowledge Base

Due to the emphasis placed on its unstructured nature, hypertext has generally been regarded as an informal way of providing information that can be browsed. Actually this is a rather limited view. Hypertexts are in fact extremely flexible knowledge representation tools that are analogous to semantic networks in that consisting of nodes and links. Different types of knowledge formalism can be implemented in hypertext by structuring and defining the basic types of nodes and links in different ways. The effect being that the highly connected structure of hypertext can be exploited as a knowledge base and can be used to build intelligent retrieval systems.

We can consider a basic set of information nodes: text, picture, sound. These nodes represent information, rather than interpreted knowledge. They are designed to be read, viewed and heard and are linked by structural links. From now on, we refer to them as document nodes. In contrast, concept nodes, which represent meaningful entities in the domain, represent organized knowledge. Each of these nodes consists of: (a) a concept name (i.e. a noun phrase); (b) a set of links to those document nodes in which the concept appears; (c) a set of links to other concept nodes that are semantically related.

The resulting structure can be regarded at two levels of abstraction [1,2,8], with the concept network conceived as an index to the document network (Fig. 1). Essentially, it is possible to regard hypertext nodes as facts and links as rules. This inferential form of hypertext would then function as an inference network. In this context, the links may be implicit and may be deduced by rules activation. Moreover, the links may be inexact, that is, they can have attached certainty values.

Different types of relationships with different properties can be defined in order to model the semantic association between concepts [5,22]. Links are labelled by the name of the relationship to reflect the nature of the semantic association and are assigned a weight to reflect the strength of the semantic association.

More formally, the resulting concept network, which is embedded into the hypertext structure, can be viewed as a directed graph \((C, T, L)\) where \(C\) is the set of the concepts that represent meaningful entities in the domain, \(T\) is the set of relation types, and \(L \subseteq C \times T \times C\) is the set of links between concepts. Accordingly, given \(L \subseteq C\), a link can be defined as a binary fuzzy relation:

\[
l = \{\mu_l(c, r) / (c, r) \mid c, r \in C; l \in L\}
\]
with a membership function $\mu_i : C \times C \rightarrow [0,1]$ indicating for each pair of concepts $(c, r)$ a measure of the strength of the semantic link between them. The notation $(c, l, r)$ means that $c$ and $r$ are linked to a degree given by $\mu_l (c, r)$.

The link relationship is defined to be fuzzy transitive, that is, if $(c / r)$ and $(r / s)$ then $(c / s)$ and $\mu_l (c, s) = \min (\mu_l (c, r), \mu_l (r, s))$. From this definition of composition, it follows that the strength of the chain linking the nodes $c_1, \ldots, c_n$ is given by the strength of its weakest link.

During the query processing, a propagation activity takes place making inferences about the goals of the user and, thus, finding information that the user did not explicitly request but that is likely to be useful. Besides the documents which satisfy requests on a specific topic, the basic assumption is that additional documents might be found on a semantically related topic, and the likelihood of support depends on the relationship between the topics. The availability of a knowledge base enables the system to emulate the reasoning process followed by an expert who can understand and expand the original query supplied by the user. For example, if an expert in the area of artificial intelligence is asked to retrieve documents concerning with «plausible inference», she or he would also retrieve documents that discuss «approximate reasoning» since the two concepts are semantically related in the domain under consideration.
The system works by spreading activation from the original query concepts through the network \([6,26]\). The node activation process starts locating a node which represents a concept included in an initial query formulation. The spreading action first affects those nodes located closest to the starting node and spreads through the network, one link at a time. Each node is assigned a special activation weight which depends on the starting activation weight and the weight associated to the link and the nodes traversed in the activation process. Therefore, given the query \(q\), for each concept \(c \in q\), the corresponding node is selected, and then, the inference rule is activated as described in the following section. Distance constraints may be imposed in the activation process by stopping the activity at some specified distance from the original node \([24]\). In addition, considering the large number of nodes involved in the spreading process activated by the query a heuristic strategy will be introduced to constrain the search algorithm in order to favour particular pathways through the network and to terminate search along other ones.

4. Retrieval as Inference

Inference from uncertain or imprecise premises and the combination of several uncertain or imprecise evidences relative to the same matter can be considered as two basic patterns of reasoning which are needed in intelligent information systems. Several mathematical models have been proposed to approach the formalization of reasoning under uncertainty \([14,15]\).

Multi-valued logics offer a framework for extending the logical inference to premises graded by degrees of truth. Of particular concern are logics in which each proposition is supposed to have a degree of truth \(c(p) \in [0,1]\) and the degree of truth of a compound proposition is only a function of the degrees of truth of the components of the proposition. Thus, for the implication, we have \(\forall (P \rightarrow Q) = f(\forall P, \forall Q)\). A large collection of implication operators \(f\) have been considered in the literature \([14]\). In our application we define such a function as the minimum. This choice, in combination with the aggregation scheme reported below, does not generate inconsistencies in presence of cycles in the concept network, thus removing the shortcoming reported in \([10]\).

In order to introduce the inference rules for retrieval, we define the following predicate symbols:

- \(Q(q, c)\): “the query \(q\) is about the subject \(c\)”
- \(D(d, c)\): “the document \(d\) is about the subject \(c\)”
\( L(c, r) \): "the subject \( c \) is linked to the subject \( r \)"
\( RC(d, c) \): "the document \( d \) is relevant to the subject \( c \)"
\( RQ(d, q) \): "the document \( d \) is relevant to the query \( q \)"

The truth values of these predicates generally range in the unit interval \([0, 1]\).

In the following, we consider a query about a single subject, since our discussion can be extended to more complicated queries by applying it to each subject constituting them. The first retrieval rule is:

\[
(Q(q, c) \land D(d, c)) \rightarrow RC(d, c)
\]

If a query is formulated to be about the subject \( c \) and the document \( d \) is described as being about the subject \( c \) then the document \( d \) is relevant. Adopting this rule, we regard \( d \) as relevant to \( c \) if and only if \( d \) is about the subject \( c \). If \( d \) is not about the subject \( c \), we cannot conclude whether \( d \) is relevant to \( c \) or not under extended material implication. The rule restricts the range of possible values of \( RC(d, c) \) to an interval given by:

\[
v(RC(d, c)) \in [\min(v(Q(q, c)), v(D(d, c))), 1]
\]

We proceed now by spreading across the network starting from the concept \( c \) and applying the following rule:

\[
(L(c, r) \land D(d, r)) \rightarrow RC(d, c).
\]

That is, even if \( d \) is not about the subject \( c \), \( d \) is regarded as relevant to \( c \) if there exists a link between \( c \) and \( r \) and \( d \) is about \( r \). This restricts possible values of \( RC(d, c) \) to the interval:

\[
v(RC(d, c)) \in [\min(v(L(c, r)), v(D(d, r))), 1].
\]

For each \( r \in C \) related to \( c \) through the link relationship, we have \( |L_c| \) intervals as pieces of evidence where \( L_c \) is the set of concepts inferred by \( c \) applying the last rule.

Defining \( \alpha_c = (v(L(c, r)), v(D(d, r))) \), we have for each piece of evidence \( v(RC(d, c)) \in [\alpha_c, 1] \), and we have to aggregate this evidence to evaluate the relevance of a document \( d \) to the subject \( c \). The proposed aggregation scheme is:

\[
v(RC(d, c)) = \max_{r \in L_c} v(RC_r(d, c)).
\]

This function determines how we combine the inferred relevance values from rules activated by the same query concept. Although alternative aggregation functions can be specified, with the current choice, we assume an
implied disjunction of evidence. If, for example, there exists at least one piece of evidence with a high degree, then the document is regarded as highly relevant to the concept. The set $L_\epsilon$ represents the maximal set which gives evidence. Obviously, it contains many meaningless pieces of evidence. Thus, we need an appropriate strategy to restrict this set as discussed in the following section.

So far, we have only looked at single concept queries. Generally, the query is concerned with more than a single subject and we have defined previously $C_q$ the set of concepts on which the query is mapped. For each $c \in C_q$ we have a piece of evidence $v(RC(d, c))$ and we have to aggregate these evidence to evaluate the overall relevance of a document to the query. Since we assume that the concepts are not linked by and/or connectives, we can use an averaging function. Thus, we have:

$$v(RQ(d, q)) = \frac{1}{N} \sum_{c \in C_q} v(RC(d, c)).$$

The set $C_\epsilon$ is the set which gives evidence and $N$ is a normalization factor (for details, see [18]).

Given a query $q$, on the basis of the presented retrieval and aggregation rules, we get the retrieved set $R_q$. In order to limit the response to only those nodes which are characterized by the highest scores, we can extract elements with a membership value greater than a fixed threshold $\alpha \in [0,1]$. A ranked output can be returned to the user by arranging the retrieved nodes in decreasing order according to their degree of membership. Alternatively, in order to reduce the output size, we can allow users to fix the number $r$ of documents that they want to be returned. Since each threshold value $\alpha \in [0,1]$ corresponds monotonically to a certain number $r$ of returned documents, limiting the size of the response either by fixing the threshold value or the maximum number of documents is the same.

5. Document Retrieval and Ranking

When approximate reasoning is used as in our case, search problems can be critical as the size of the concept and document bases increases. This is because the system has to consider imprecise solutions as well as precise ones and a large number of very imprecise, and hence useless, solutions may be generated. It follows that some means of reducing the search space is essential to the design of efficient reasoning systems. We have developed bounding procedures on the relevance function that help to reject potentially useless solutions.
The objective is to locate only the set of top ranking documents, minimizing the number of documents that are to be evaluated.

Given $c \in C$, we can easily define the fuzzy subset $D_c$ of documents about the concept $c$. This will be referred to as the document list associated with the concept $c$ given the set of documents dealing with such a concept. For each concept $c \in C$, the list $D_c$ gives a set of pairs consisting of a) the reference to the document $d$ including the concept $c$ in its description b) the weight corresponding to the degree of membership of the concept $c$ in the description $d$.

Thus, the concept network can be regarded as an index to the documents, since each document is linked into the network to those nodes of the network that represent concepts to which it is related.

Given a query $q$ the set $P_q$ of documents which possibly satisfy the query is given by the union of the document lists associated with each of the concepts appearing in the query or inferred by the system. It is the set of documents which have a non-zero value for the relevance function. Such a function should be computed for each document in the set $P_q$.

The cardinality of the set $P_q$ may be large, since there is a considerable number of documents having at least a concept in common with the expanded query. That requires the useless calculation of many low values for the relevance function with the result that a large fraction of the document collection must be inspected.

A heuristic search strategy has been devised to further restrict the set of documents to be evaluated to a subset of $P_q$ while still ensuring that the documents at the top of the ranking are identified. The approach described here is based on the previous work on best-match searching reported by Lucarella [17]. For a generalized approach to the best-match problem see Sashe and Wang [28].

5.1. Document selection

Let $N_q = \{c_1, \ldots, c_k\}$ be the total number of concept nodes (included in the query and inferred by the system) and let $R_q = \{d_1, \ldots, d_k\}$ be the relevance set consisting of the r top-ranking documents to be returned, with $r$ defined by the user. The search process can be described in the following steps.

For each concept $c \in N_q$, the associated document list is accessed. For each document $d$ in the list, a variable $\mu(d)$ is allocated and set to the minimum between the values $\nu(Q(q,c_i))$ and $\nu(D(d,c_i))$. When the same document appears once again in a subsequent list, its corresponding variable
is incremented by the minimum of the new values. After having processed all the document lists associated with the concepts, each variable contains the numerator of the membership function, i.e.:

\[ \mu (d) = \sum_{i=1}^{k} \min (v (D (d, c_i)), v (Q (q, c_i))) \]

An evident advantage of this procedure is that we do not have to access the document database but we can work only on the concept network.

Now, let us see how to avoid processing all of the lists with relative documents. Assume that we have processed \( j \) document lists out of \( k \), keeping track of the top \((r+1)\) variables computed so far:

\[ <\mu (d_1), ..., \mu (d_j) > \mu (d_{r+1}). \]

We have to process the \((j+1)^{th}\) document list. For each document \( d \) referenced in the new list, an upperbound for its membership function \( U (\mu (d)) \) can be computed as it will be shown below. It represents the maximum degree of membership that the document \( d \) can reach, assuming it should happen to deal with all of the remaining \((k-j)\) concepts. If the computed upperbound for \( d \) is less than the membership degree associated with \( d_r \), it means that the document \( d \) will never reach the relevance set, so that \( d \) can be removed from further consideration, saving memory space and computing time.

In addition, the procedure can be terminated, without examining the remaining lists, if the computed upperbound for the membership degree of the document \( d_{r+1} \), the first out of the relevance set, is less than the membership degree associated with \( d_r \), the last in the set \( Rq \). The stopping condition is:

\[ U (\mu (d_{r+1})) \leq \mu (d_r). \]

Considerable improvements can be obtained if, instead of stopping when the set of documents to be returned to the user is completely determined, the procedure stops when the majority of the most relevant documents are guaranteed to be in the retrieved set. This means that we accept that only the top \( s \) best documents are returned with \( s < r \), while the remaining \((r-s)\) are simply good documents:

\[ <\mu (d_1), ..., \mu (d_j), ..., \mu (d_r) > \mu (d_{r+1}). \]

This heuristic implies changing the previous stopping condition to:

\[ U (\mu (d_{r+1})) \leq \mu (d_s). \]
Since $\mu(d) > \mu(d')$, the stopping condition works better in terms of dropping a higher number of concepts, hence, constraining the activation of nodes in the knowledge base.

5.2. Upperbound evaluation

Now, let's show how to evaluate an upperbound for the membership function. After having processed $j$ concepts out of $k$, let $\mu(d)$ be the current score for the $d^b$ document. Then, the membership degree can be bounded assuming the worst condition that all the remaining uninspected $(k - j)$ concepts are common to the document and the query:

$$U(\mu(d)) \leq \mu(d) + \sum_{i=j+1}^{k} \min(v(D(d, c_i)), v(Q(q, c_i))).$$

Even if we do not know $v(D(d, c_i))$, since we do not access the document file, an upperbound can be easily computed considering that the retrieval function requires the evaluation of the minimum between the two values.

We can assume that $\min(v(D(d, c_i)), v(Q(q, c_i))) = v(Q(q, c_i))$. If $v(D(d, c_i)) \geq v(Q(q, c_i))$, than it corresponds to the exact value; whereas if $v(D(d, c_i)) < v(Q(q, c_i))$, it still gives an upperbound. The resulting expression becomes:

$$U(\mu(d)) = \mu(d) + \sum_{i=j+1}^{k} v(Q(q, c_i)).$$

The conclusion is that an upperbound can be determined taking into account only the membership degrees of the remaining concepts in the query $q$ without needing to take account of the document $d$. Generally, the effect of the introduced heuristic is a performance optimization, although returned documents may not be precisely ranked.

6. THE PROTOTYPE SYSTEM

Different approaches can be followed in order to set up a system which integrates browsing and searching approaches as discussed above. We have chosen a loosely-coupled architecture. It presents some weak points but gives the advantage of implementing a quick prototype since the hypertext system and the information retrieval system are independent with only some form of communication between them. The two subsystems share
the hypertext network as a common data structure and communicate through a module underlying the user interface (Fig. 2).

The network browser supports the user (author/reader) with canonical hypertext facilities and tools for navigating through the network structure consisting of nodes and links. The search engine acts as a question-answering system that, given the request, returns the best matching answer reasoning on the network as a knowledge base. The hypertext network, as discussed in the previous section, can be regarded as a two-layered structure that can be browsed without any constraint on possible entry points and movements. Conversely, during a search activity, the concept network is regarded as an index to the documents, since each document is linked into the network to those nodes that represent concepts to which it is related. The functional integration is guaranteed through the task scheduler.

The document is managed by the system as a structured object composed by the following elements: <header, content, type, body>. The header collects a set of fixed attributes qualifying the document; the content gives a

---

**Fig. 2. The Architecture of the System**

[Diagram of the architecture showing the User Interface, Task Scheduler, Network Browser, Text Analysis, Search Engine, Network & Data Management, Concept Network, and Document Network.]
summary of the document subject and will be addressed in content-based queries; the type defines document type (text, image, picture, voice) and has an implication for object manipulation and presentation; the body is the full extension of the document and it can be further structured.

Objects from different information sources can be imported into and exported from the system under the control of the user interface. Essentially the system manages multimedia documents enabling the user to locate relevant ones on the basis of the associated textual description. When documents are loaded, the content is processed by the text analysis component with the support of the concept base; relevant concepts are identified and each document is linked to those nodes of the network representing such concepts.

The natural language text, associated to documents or queries respectively, is processed in order to restrict the conceptual area where the semantics of the text can be framed. In our opinion, the appropriate level of focus for topic analysis lies between the wide-focus text understanding approach and the narrow-focus term selection approach. The proper conceptual unit is at the level of the noun phrase, namely, a multi-word string. Thus, the concept recognition problem can be transformed into the problem of comparing multi-word strings and measuring the resemblance between them.

Concepts included into the concept base are indexed by their constituent terms. Each phrase is extracted from the text and parsed at a cursory level in order to remove non-content bearing words and function words. Remaining words are stemmed by means of a conflation algorithm and are used to access the concept index and identify potential candidates. Selected concepts are matched against the phrase extracted from the text by applying a similarity function to evaluate matched expressions. Thus a weight can be associated to each concept. The procedure in use is a slight modification of an improved procedure for nearest neighbour searching already presented in a different context [16].

Concepts identified as being of interest to the user are returned along with the estimated weight, enabling the user to adjust the weight simply by moving, with the mouse, the corresponding slider on the screen. During this interactive phase, the user can contribute by adding concepts to the concept base, and linking such concepts to the existing structure with the support of the browser. It is important to recall that the domain knowledge is assumed to be dynamic and largely derived by the system-user interaction. Indeed, we emphasize the importance of an individual description of a domain rather than a global description completely available at the time when documents and queries are processed.
The graphical user interface was developed using the Microsoft Windows Software Development Kit, while other modules of the system were implemented in PDC-Prolog by Prolog Development Center. Concept and document network have been stored in the PDC Prolog's external database. Major details about the system architecture are reported in [20], and related ideas can be found in [8,13].

6.1. User interface

The integration of browsing and searching approaches into a unified environment meets the users' needs very well. The system is responsive to novice users as well as to expert ones, and is flexible enough to accommodate different styles of information demands that can be characterized by the following canonical situations:

- I know exactly what I want: the user specifies the value of one or more attributes qualifying the requested item(s) exactly;
- I have a rough idea of what I am looking for: the user enters a query specifying by means of a natural language text the subject of interest;
- I realize I am interested in something when I see it: the user browses through the concept network marking interesting concepts met along the way and the system collects them to form a query;
- Some passages of the text I am reading suggest further exploration: the user marks them and the system, using the passage as a query, retrieves further documents related to the topic of interest.

To give a better idea of such capabilities, we show two examples of user-machine interaction in order to retrieve relevant information.

In Fig. 3, the window «Retrieval» presents the template to be filled in by the user; the window «Document card» return items ranked in a decreasing order of relevance; and the window «Document browser» enables the user to inspect the full text of the document.

Queries have the same structure as the items to be retrieved, and they are entered as natural language texts in order to relieve the user from tricky Boolean formulations and to provide a transparent and easy to use interface. More specifically, when the query facility is entered, the system displays a template of the structure of the item to be retrieved and the user simply has to fill in the template. The more attributes are filled in the template, the more specific is the query, and the more selective is the answer. Parts addressed in the query are both the header and the content which are typed in natural language statements describing in a rough way the topic of interest. The
system returns all of the objects which satisfy the conditions on predefined attributes, if specified, and whose subject is plausibly related to the addressed topic. Attributes specified in the original query remain unchanged; attributes missing are filled in with the attributes of the retrieved item; and the text of the query is replaced by the summary of the retrieved item.

The order in which objects fulfilling the query are returned to the user reflects the assessment made automatically by the system about the degree of relevance of the retrieved items to the query. So the effect is to restrict the attention to a manageable subset of relevant objects. Then the user can browse, view, and mark any part of the object submitting it as a query for further exploration.

Fig. 4 shows the browsing approach and points out as the same document previously obtained through searching can be reached by browsing. The window «Concept browser» presents the concept network from which the user can select one or more concepts marking them with the mouse. After a selection, activating the search button, the system uses concepts collected by the user to compose the query and then executes it. The retrieved
documents are presented in the same way as described above, and the "Document Browser" is entered when one wants to read the retrieved document.

7. Experimental Results

In order to get an evaluation of the system we have used a mix of queries generated by browsing and searching sessions. These have been processed against an experimental hypertext network.

We have used a subset of a test document collection in the area of Applied Mathematics and Computer Science, available at the Automation Research Center of ENEL, the Italian Electricity Company. Relevant characteristics are reported in Table 1. The last row of the table gives the number (on average) of concepts we get after the spreading activation process has taken place.

In order to evaluate the effectiveness of the system, we used standard measures of precision, the proportion of retrieved items actually relevant
and recall, the proportion of relevant items actually retrieved. In our system, which returns a ranked set of documents, such measures can be computed by setting the value of the relevance threshold ($\alpha$-level) and considering as retrieved all documents that have a relevance score higher than the threshold.

**Table 1. Experimental Data**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of document nodes</td>
<td>300</td>
</tr>
<tr>
<td>Number of concept nodes</td>
<td>175</td>
</tr>
<tr>
<td>Number of queries</td>
<td>10</td>
</tr>
<tr>
<td>Average concepts per document</td>
<td>14.8</td>
</tr>
<tr>
<td>Average concepts per query</td>
<td>7.2</td>
</tr>
<tr>
<td>Average concepts per expanded query</td>
<td>37.6</td>
</tr>
</tbody>
</table>

**Table 2. System Effectiveness**

<table>
<thead>
<tr>
<th>$\alpha$-level</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.52</td>
<td>1</td>
</tr>
<tr>
<td>0.3</td>
<td>0.79</td>
<td>0.87</td>
</tr>
<tr>
<td>0.4</td>
<td>0.98</td>
<td>0.8</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0.63</td>
</tr>
<tr>
<td>0.6</td>
<td>1</td>
<td>0.24</td>
</tr>
</tbody>
</table>

For each query we have five runs corresponding to different values for the relevance threshold, $0.2 \leq \alpha \leq 0.6$. Table 2 reports average values for recall and precision at different cut-off levels, giving an indication of the effectiveness of the system. It follows that the best choice for the threshold value is $\alpha = 0.4$ which gives high precision with good recall.

The optimization in the efficiency of the search process can be estimated by computing the number of nodes that do not have to be explored due to the introduced dynamic bounding. Given the set of sample queries, we have to take note of the number $N_b$ of documents that are examined in a complete proof of the query, and then to take note of the number of documents that are actually examined with the bounding method. Furthermore, the runtime of the proofs in the two cases ($T$ and $T_b$ respectively) is stored in order to obtain a figure of the real improvement in computation. This is a crude measure of the elapsed time between the request and the answer.
The same set of queries has been used with the returned relevant set consisting of ten good documents \((r = 10)\) while only five documents \((r' = 5)\) were guaranteed to be the best ones. Each query was processed twice, first with the basic procedure and then with the bounded version.

Table 3 illustrates the relative performance of the bounded procedure. It reports the average values of \(N_b / N\) and \(T_b / T\) ratios. In order to relate such measures to the effectiveness of the system, average values of \(P_b / P\) (Precision) and \(R_b / R\) (Recall) have been reported as well. As already remarked, the effect of the introduced optimization is a possible decrease in the ordering precision among the retrieved documents since they are ranked in the order of their partial membership. Consequently, the last row gives a measure of the retrieval accuracy reporting the average discrepancy in ranking order introduced with the bounded procedure. Such a value is obtained computing for each run the number of misplaced documents over the number of returned documents.

**Table 3. Relative Performance of the Bounded Procedure**

<table>
<thead>
<tr>
<th>documents examined (N_b / N)</th>
<th>0.71</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtime (T_b / T)</td>
<td>0.66</td>
</tr>
<tr>
<td>precision (P_b / P)</td>
<td>1</td>
</tr>
<tr>
<td>recall (R_b / R)</td>
<td>0.89</td>
</tr>
<tr>
<td>ordering discrepancy</td>
<td>0.4</td>
</tr>
</tbody>
</table>

The experiments demonstrate that, on average, an improvement can be reached applying the bounding method without a significant decrease in the system effectiveness (the precision is the same and there is a 11% recall degradation). As we could expect, the relationship between the number of documents processed and the runtime is not linear, thus the runtime optimization is larger. Conversely the ordering precision is affected since, on average, four documents are misplaced.

Although this is a significant improvement, further research is required to better constrain the search process and further experiments must be carried out to draw final conclusions.

8. **Conclusion**

We have presented a model for an hypertext-based retrieval system using plausible reasoning and we have given a possible approach for the
integration of search facilities in hypertext systems. Moreover, we suggested a constrained search strategy which allocates documents to given queries, restricting as much as possible the set of candidates to be evaluated.

We outlined the architecture of a prototype system based on this model; addressed problems of user-machine interaction; and presented some issues of the user interface.

A preliminary set of experiments carried out seem to prove the power of combining these two paradigms. Although a research prototype, we believe it shows considerable promise as an advanced information retrieval environment.

Work in progress includes an object-oriented model that, combining browsing and searching techniques, presents a uniform framework for direct manipulation of multimedia information [21]. In this model units of information that are arbitrarily diverse in form and content are connected by links to form a complex objects. It would be clearly useful to allow the incorporation of this structural information in the construction of content-based queries.

REFERENCES


