INdex Navigator for Searching and Exploring the WWW

B.C.M. Wondergem, M. van Uden, P. van Bommel, and Th.P. van der Weide
Computing Science Institute, University of Nijmegen
Toernooiveld 1, NL-6525 ED, Nijmegen, The Netherlands
Tel: +31 24 3653147, Fax: +31 24 3553450
E-mail: bernd@cs.kun.nl

Keywords: Information Retrieval, Hypertext, Query by Navigation, World Wide Web, Index Expressions.

Technical Report CSI-R9917

Abstract

Searching information from a large and dynamic information space causes several problems, concerning, for instance, dynamic and vague information needs, too broad queries, and correctness and sensibility of descriptors. These problems may be attacked by navigational query formulation strategies which are available for stratified architectures. However, stratified architectures cannot be easily constructed for large and dynamic information spaces. In this article, we show how navigational query formulation and exploration can be employed on the WWW by using linguistic (as opposed to statistical) refinements. Grounded in the theory of navigational networks for index expressions, we introduce our tool, the INdex Navigator (INN), for searching and navigating the WWW. The INN is a dynamic electronic service system for the WWW.

1 Introduction

Searching information from a large and dynamic information space causes several serious difficulties. A number of these concern query formulation. According to [4, 14], a major problem is (caused by) the inherent *vagueness* of information needs. Therefore, formulating the information need concisely, without an explicit description of the expected result, is very difficult. In order to increase the user's knowledge about the field of interest, IR systems should enable users to explore topics of interest. Related to an increase in knowledge are *shifts in interests* ([4]). Retrieval systems should thus support interactive reformulation. A third major problem concerns constructing (syntactically) *correct* and (semantically) *sensible* complex descriptors ([12, 13]). Letting the user select

descriptors from a set of (correct) options bypasses this problem. Fourth, *broad queries* often result in low precision. IR systems should thus aim at preventing imprecise queries.

Formulating queries by navigation in an abstraction of the information space eases the mentioned problems ([11, 6, 16]). To this end, stratified architectures have been developed containing an ancillary layer that forms an abstract description of the contents of the information space ([1, 6]). This meta layer can depict an overview of the concepts present. This helps users in exploring their field of interest. Searchers can then formulate their need by recognizing rather than formulating relevant concepts. The vagueness in the information need can be further decreased by concept exploration: by inspecting actual documents that correspond to a concept. In this way, the user can learn what the concept means. This is the second way in which navigation aids exploration. Since the IR system generates the overview, it can guarantee correctness and sensibility of the offered descriptors. For instance, the descriptors can be taken from available documents. Shifts in interest are naturally supported by selecting a different direction during navigation in the overview. Finally, navigational formulation techniques enable users to iteratively select more specific descriptors. In general, this eases the way to descriptors of proper specificity. Concluding, we advocate the combination of searching and exploration based on navigation in an ancillary structure. This integration of hypermedia and information retrieval is also advocated by, for instance, Waterworth & Chignell ([15]) and Lucarella & Zanzi ([11]).

However, the size and dynamics of certain information spaces imply that a complete and up to date abstraction cannot be constructed. Consequently, navigational query formulation in ancillary layers is not directly applicable to these information spaces.

In this article, we show how a very large and highly dynamic information space, the World Wide Web, can be abstracted and navigated. We developed the INdex Navigator (INN), a dynamic information system for query formulation on and exploration of the WWW. The INN is based on Query by Navigation (QBN). QBN ([6]) is a navigational way of query formulation in a stratified architecture based on index expressions ([7]). Navigational networks for index expressions allow navigation over linguistically motivated subexpression links. Although the INN is developed for the WWW, our approach is mutatis mutandis applicable to all (dynamic or static) information spaces. The required changes only involve the communication of the INN system with the search facilities used to access the information space.

This article has the following structure. In section 2, related approaches are described. Section 3 explains the construction of a stratified architecture based on index expressions. Section 4 explains QBN. Insight in the construction of the INN system is given is section 5. Its workings are illustrated by an example session. We evaluate preliminary experiences with the INN in section 6 and provide ways for further research in section 7.

2 Related Approaches

Another system based on the approach taken for the INN is the HyperIndex Browser (HIB) ([10]). The first author of our paper implemented the first version of this HIB. Reports describing a rather general introduction to the HIB and experimental results on the cognitive load imposed by the HIB ([8]) have appeared. However, a thorough analysis of the technical fundamentals lacks. This is provided in this article. Compared to the HIB, the INN system uses different techniques for constructing the stratified architecture. In particular, the INN offers a broader notion of subexpressions. Furthermore, the INN system makes use of different search engines than the HIB. In particular, the INN serves the Dutch information community by providing access to two Dutch search engines.

More loosely related approaches include systems for meta searching, systems using statistically computed refinements, and other hypertext systems exploiting subexpression links. Systems for meta searching, like MetaCrawler and MetaSearch, fire off a query to several search engines and combine their results. Although meta search does offer an overview, it lacks abstraction: the results are simply merged, rather than abstracted.

Statistically computed terms for refining a query are offered by, for instance, AltaVista and Excite. Term co-occurrence frequencies are used to produce a set of related terms. The user can click on (groups of) these terms to add them to the query under construction. These query refinements are statistically rather than linguistically motivated.

In addition, many other hypertext systems exist that use subexpression links. Under the most well-known links are hypernyms (e.g. is-a) and meronyms (e.g. part-of). Those semantical relations between descriptors are, in general, generated from knowledge bases. This means that this kind of semantical approach is domain dependent.

Many stratified architectures, synthesising information retrieval and hypertext, have been proposed. A good introduction and overview is provided in [2]. Our approach is based on a stratified architecture for index expressions ([6]).

3 Stratified Architecture for Index Expressions

3.1 Search Support using Layers

The stratified architecture augments a set of documents with an ancillary structure, called the *hyperindex*. This hyperindex forms a more abstract description of the contents of the documents. It provides a conceptual overview of the information carried in the documents. The special form of our hyperindex, which is explained in the next section, allows structural navigation. Concept exploration is supported by transferring between the hyperindex and the actual documents that correspond to a concept in the hyperindex.

The stratified architecture, as depicted in figure 1, consists of two layers, the *base layer* and the hyperindex, which are connected through the *beam relation*.

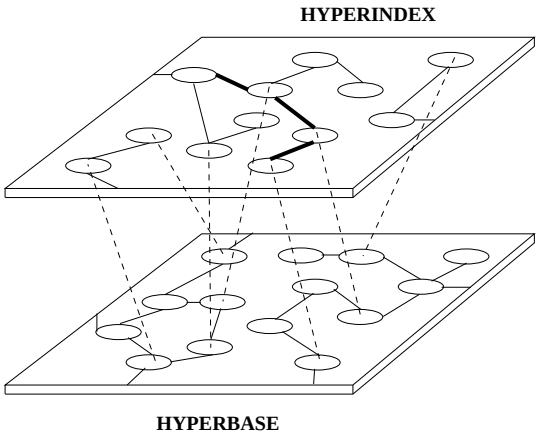

Figure 1: Stratified Architecture

The base layer contains the available documents. These documents may be linked, for example by hyperlinks. By traversing these links, as is usual in WWW context, navigation in the base layer may take place. We will not study this further in this article. Each document is indexed, or *characterised*, which results in a set of descriptors, a *characterisation*, per document. These descriptors reside in the hyperindex.

The beam relation connects the base layer with the hyperindex. That is, the beam relation connects documents in the base layer with their characterisation. In general, the links between both layers are be bidirectional, allowing traversal in both directions. From a document, the user may transfer himself to one of the descriptors of the characterisation (beam up). From a descriptor in the hyperindex, the user can perform a beam down, which transfers him to documents that are about the descriptor.

The hyperindex forms an overview of the documents based on their characterisations. If index expressions are used for characterisations, a hyperindex with a fine-grained structure is constructed. This structure is used in step-wise navigation.

3.2 Index Expressions for Document Characterisation

In our case, the descriptors that form document characterisations are index expressions. In the hyperindex, index expressions are connected with their subexpressions and superexpressions

Index expressions are built from terms and connectors by structural composition. Terms denote keywords, concept names, adjectives, gerunds, and attribute values. Connectors denote relations between terms in the form of prepositions (showing place, position, time, or method) and some present participles (e.g. using, having, and being). Structural composition, using the structural operator add, gives index expressions their (tree-like) structure.

Index Expressions
For given sets of terms T and connectors C, *index expressions* are defined by the following two cases

(Terms) If $t \in T$, then t is an index expression,

(Composition) if I and J are index expressions and $c \in C$ is a connector, then $\mathsf{add}(I, c, J)$ is an index expression.

The composed index expression $\mathsf{add}(I, c, J)$ is obtained by adding subexpression J to I via connector c. Using nesting of the structural operator, index expressions can be made more specific, by deepening the tree structure, or more general, by broadening the tree structure. This is illustrated in the following example.

Example 3.1

Index Expressions

Consider as terms conference, information, *and* retrieval. *Furthermore, consider as connectors* on *and* $\circ$, *the so called null-connector. The phrase* conference on information $\circ$ retrieval *is an index expression. It is denoted as*

$$\mathsf{add}(\mathsf{conference}, \mathsf{on}, \mathsf{add}(\mathsf{information}, \circ, \mathsf{retrieval})) \quad (1)$$

As another example index expression, consider

$$\mathsf{add}(\mathsf{conference}, \mathsf{on}, \mathsf{add}(\mathsf{IT}, \mathsf{in}, \mathsf{Belgium}))$$

Note that, in terms of structure and meaning, this differs from

$$\mathsf{add}(\mathsf{add}(\mathsf{conference}, \mathsf{on}, \mathsf{IT}), \mathsf{in}, \mathsf{Belgium})$$

Using brackets in the textual representation, these index expressions can be written as conference on (IT in (Belgium)) *and* conference on (IT) in (Belgium). □

A parsing algorithm for index expressions is provided in [6]. It distinguishes connectors with high and low priority. The priority of a connector either leads to a broadening or a deepening of the structure. This is called *structure detection*. In tests performed on the titles of the CACM document collection it was found that the algorithm produced well-formed structures in approximately ninety percent of the cases. The erroneously parsed index expressions, however, might not cause severe problems since non-sensical descriptors are unlikely to be selected by users.

Index expressions can be broken down into subexpressions. Roughly speaking, the subexpressions of an index expression are all index expressions that it contains. In [17], several notions of subexpressions are provided. Direct subexpressions can be obtained from an index expression by removing a single leaf or, if possible, the head. All subexpressions can be obtained by repeating this defoliation several times.

Example 3.2

Subexpressions

Index expression 1 from example 3.1 has two direct subexpressions. The first, information ∘ retrieval *is obtained by removing its head. The second,* conference on information*, is obtained by removing its only leaf. These, in turn, have a number of direct subexpressions, all consisting of a single term. These single terms, for instance* retrieval*, are no direct subexpressions of index expression 1. However, they are (general) subexpressions of index expression 1.* □

Subexpressions are used to build a fine-grained navigational network of index expressions. This network, called a *lithoid*, is based on an initial set of index expressions. In case of the stratified architecture, the union of characterisations constitutes this initial set. Uniting index expressions is done by sharing term sharing. The lithoid consists of nodes and links. Each node is an index expression. The index expressions included in a lithoid are all index expressions from the characterisations of the initial set of documents plus all their subexpressions. The links of the lithoid connect index expressions. In order to obtain fine-grained links, nodes are connected with their direct subexpressions.

Example 3.3

Lithoid *Consider again index expressions* conference on (information ∘ retrieval) *and* conference on (IT) in (Belgium)*. Based on their subexpressions, the lithoid in figure 2 is constructed. Note that by node sharing,*

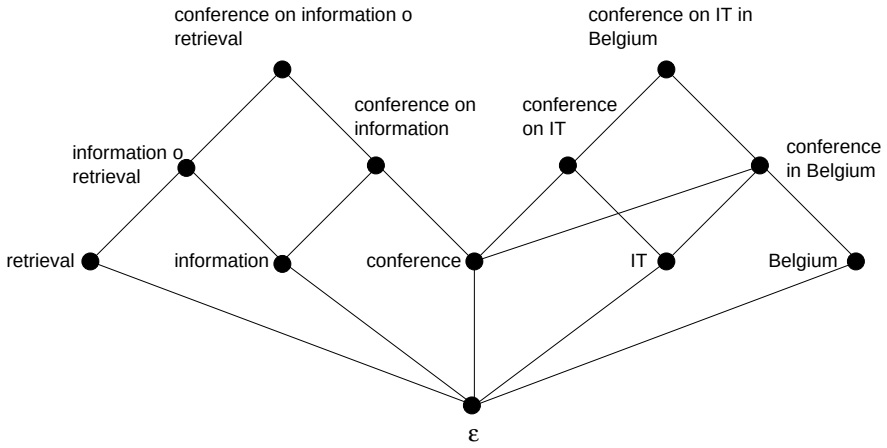

Figure 2: An example lithoid.

see e.g. node conference*, a connection is made between the two concepts. The empty index expression ϵ, which is a subexpression of every index expression, is used as the most general descriptor.* □

Lithoids form networks consisting of nodes (index expressions) that are connected through links. Therefore, lithoids can be navigated in order to formulate

queries.

4 Query by Navigation

Formulating an information need aims at finding a descriptor that properly describes it. Query by Navigation (QBN) ([7, 5]) is a navigational way of query formulation in the stratified architecture for index expressions. By structurally navigating in the hyperindex, users formulate a query. During QBN, documents may be explored by transfers between hyperindex and base layer (see also section 3.1).

QBN identifies two types of actions: navigational actions in the hyperindex and beam operations for traveling from the hyperindex to the base layer and vice versa.

4.1 Navigating the Hyperindex

QBN starts by selecting a single node (index expression) in the hyperindex. The current node in the hyperindex is called the *focus*. The user may continue navigation by selecting one of the neighbours of the focus. The selected neighbour then becomes the focus and the selection process is repeated. Navigation thus essentially is a repetitive selection of neighbours. Navigation ends if a satisfactory index expression is reached. That is, if documents that satisfy the information need have been found.

QBN exploits the special form of the hyperindex, i.e. a lithoid, by allowing fine-grained navigation steps. In figure 3, an abstract picture of a lithoid is given in which one of the nodes is marked as focus. The neighbours of the focus depict the direct choices for QBN. They thus give an overview of the concepts available for selection.

In a lithoid, the neigbours of a node are either *refinements*, direct superexpressions, or *enlargements*, direct subexpressions. Refinements, residing directly above the focus in the lithoid, denote more specific concepts. For example, conference on (IT) in (Belgium) is a refinement of both conference on IT and conference in Belgium. Since refinements contain one node more than the focus, they denote the smallest possible more specific concepts. This guarantees the fine-grained nature of the navigation steps in QBN. By selecting a refinement, the user formulates his need more concisely.

Enlargements denote less specific concepts than the focus. In fact, they denote a subconcept of the focus. For example, IT is an enlargement of conference on IT, which, in turn, is an enlargement of conference on (IT) in (Belgium). By selecting an enlargement, the user thus obtains a broader description. Enlargements can, for instance, be selected in order to recover from a previously selected refinement in order to change direction in the hyperindex. In this way, shifts in interests are fluently dealt with.

Navigation in the hyperindex consists of selecting a number of refinements and enlargements. In addition, the user can transfer himself from the hyperindex

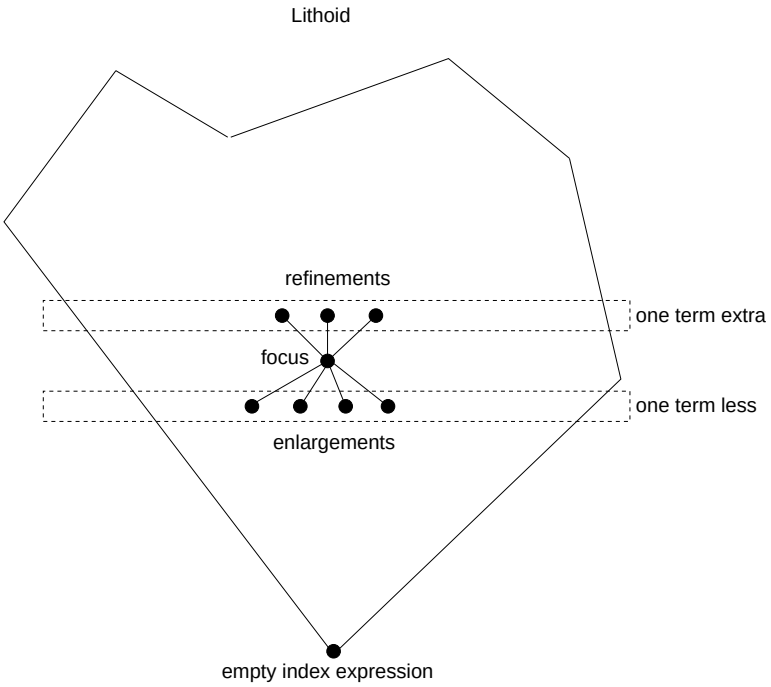

Figure 3: Neighbours in Query by Navigation

to the base layer to inspect actual documents. If required, the user can transfer himself back to the hyperindex, where navigation can be resumed.

4.2 Beaming between Layers

Inspecting actual documents is important for a user to ascertain the relevance of documents (searching) and for concept learning (exploration). Inspecting documents is enabled by an operation, *beam down*, that transfers the user from hyperindex to hyperbase (see figure 1).

By traveling the beam relation downward, the user is presented with the documents relevant to the focus in the hyperindex. The user can inspect these documents, and, if links between documents are available in the hyperbase, browse through them.

If satisfied, the user can end the QBN session. In searching, for example, this may be the case if the user has satisfied his information need by rendering some relevant documents. In exploration, this may happen if the user estimates his knowledge of the field of interest is now sufficient.

In addition, the user may transfer himself back to the hyperindex (*beam up*) to resume navigation. However, since the characterisation of a document may contain several index expressions, a document can be linked to several nodes in the hyperindex. Therefore, the target node for the beam up may not be

clearly defined. This problem, called *ambiguity*, may lead to user disorientation, especially when the user first follows a few browsing steps. Therefore, the INN system provides a rather basic back-button which guarantees that the user returns in the hyperindex at the same node where he left it.

5 INN System

The INN system forms an intermediary between users and the WWW. It makes use of existing search engines to access information on the WWW. The overall architecture of the INN system is sketched in figure 4.

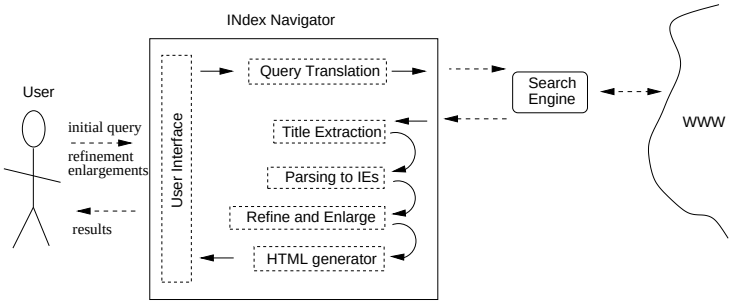

Figure 4: Overview of INN architecture

The path of control of the INN system is as follows. After the user has formulated an initial query, this is translated into the query language of the selected search engine. From the documents returned by the search engine, the titles are stripped. These titles are then parsed, so that index expressions are obtained. Refinements and enlargements in the parsed titles are computed and presented in a HTML page. If the user selects one of the navigational options, a refinement or an enlargement, the process is repeated with a new focus. The user may also go to the documents about one of the presented topics (beam down). Those documents then constitute the result of the system.

By providing an example session, we will discuss the workings of the INN system in more detail.

5.1 Getting Started

The initial screen of the INN system is given in figure 5. The components of this screen are explained below.

The About INN button provides information about the background and workings of the INN system. For example, the idea of QBN, using refinements and enlargements, is explained. The INN Home button leads the user to the home page of the INN system.

To start navigating the WWW, four steps have to be followed by the user.

Figure 5: Initial Screen of INN

1. **Select Search Engine.** First, a search engine has to be selected from a list. In figure 5, two well-known search engines, Alta Vista and Lycos, are available as well as two Dutch engines, Zoek and Ilse.

   Other search engines can easily be incorporated by slight changes in the communication between the INN system and the search facility. This also means that different information spaces can be navigated and explored via the INN system. Figure 4 shows that the communication between the INN system and the information space is mediated by a search facility. This suggests that only changes are needed in the communication between the INN system and this search facility. The user query needs to be translated into a form that the search facility supports. For textual information, this is mostly keyword-based. Since this is already supported, it means that the query translation part need not be changed. The only part that might

need changes then is the title extractor. Since most search facilities clearly mark titles in their output, modifying the title extractor is a rather easy task.

2. **Set Size of Result Set.** Second, the size of the result set produced by the search facility must be set. That is, the number of documents that will be used for producing the overview should be specified. In this way, the user can steer the coverage of the overview. In addition, the user gains control over the response time of the system.

   The size of the result set can be adapted for each step during navigation. This is a nice property since a larger query generally means that less documents are returned. By increasing the size of the result set, the chances that suitable refinements can be generated also increase.

3. **Provide Initial Query.** Third, an initial query has to be provided. It is interpreted as index expression. The initial query may be of any size. However, it is recommended that a small initial query is provided in order to start with a broad overview.

4. **Send Request.** Finally, the user sends his request by clicking on the send button.

The query is fed to a search engine and the resulting documents are processed. In our example session, the user types the query retrieval.

5.2 Navigating on the Fly

The next page shown to the user, see figure 6, gives an overview of the navigational options. It consists of four parts, which are generated on the fly.

Focus. The previously given query, i.e. retrieval, functions as current point in the hyperindex (focus). By clicking on the magnifying glass next to the focus, the user goes directly to the relevant documents (beam down). This option is offered for all descriptors.

Refinements. Next, all refinements of the focus are given. For the example query, refinements include data retrieval, information retrieval, retrieval links, and storage and retrieval. The refinement give an overview of the topic retrieval.

Refinements are computed within the titles of the documents that were returned by the search facility. Every direct superexpression of the focus that is contained in any of the titles is listed.

The refinements of the focus *in* another index expression (title) thus need to be computed. This is done by using the *twigs* of an index expression. Twigs are elementary subexpressions of the form $t_1 c t_2$, where t_1 and t_2 are terms and c is a connector. Both focus and titles are broken down into twigs. This is done by the twigs operator ([17]):

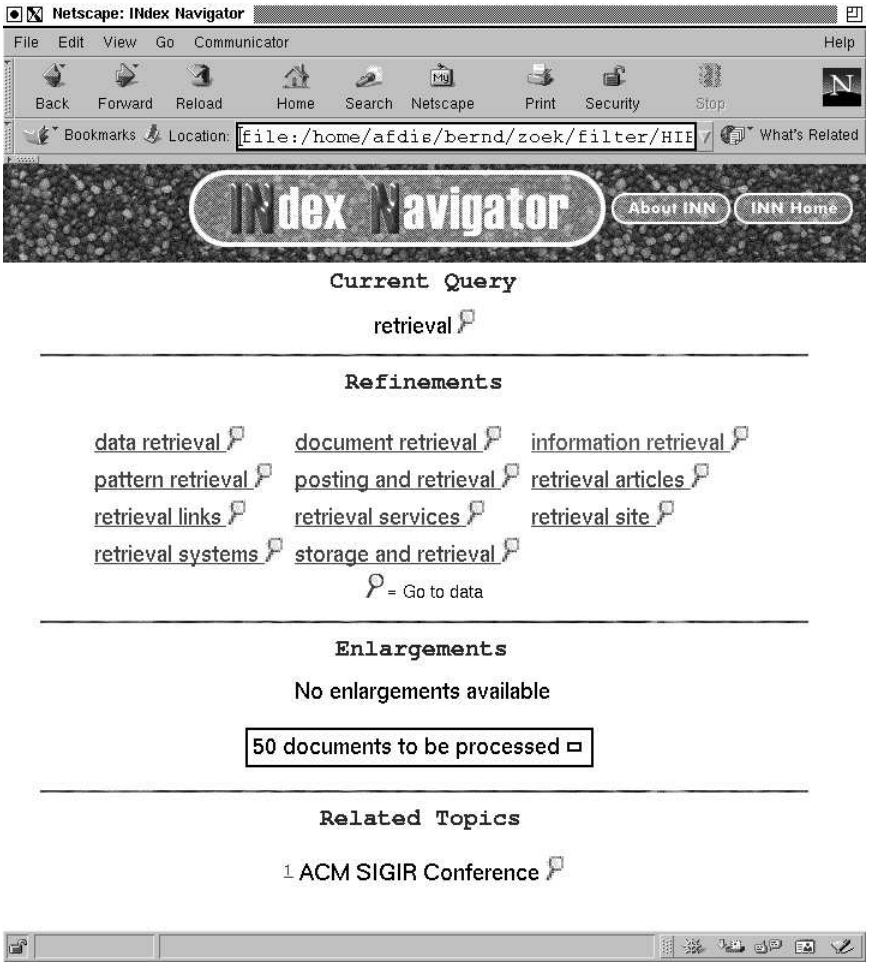

Figure 6: Overview Screen of INN

$$
\begin{array}{lcl}
\mathsf{twigs}(t) & = & \varnothing \\
\mathsf{twigs}(\mathsf{add}(I, c, J)) & = & \{\mathsf{add}(\mathsf{Head}(I), c, \mathsf{Head}(J))\} \\
 & & \cup\ \mathsf{twigs}(I) \cup \mathsf{twigs}(J)
\end{array}
$$

Function Head returns the head (root) of an index expression. Note that when every term in an index expression is different, we can reconstruct the index expression from its twigs.

Refinements can only be made if the focus is contained in the title. This is the case if the set of twigs of the focus forms a subset of the twigs of the title. This implements a more general notion of subexpressions than

used in the HIB (see figure 7). In the left of this figure, an example index expression is given. In the middle, the notion of contiguous subexpressions, as used in the HIB, is abstractly depicted. The solid lines depict direct connections, i.e. those consisting of a single connector. The dashed lines show the places where subexpressions may be added so that the example index expression is a subexpression. On the right, our twig-based notion of subexpressions is sketched. Where subexpressions in the HIB must form a contiguous part, our notion allows different parts of the focus to appear in different parts of the title. In addition, in the HIB, the order of the subexpressions is relevant. Our twig-based refinements allow differences in this order. We claim that the order of subexpressions in many cases is irrelevant for their meaning. For example, conference on IT in Belgium and conference in Belgium on IT may very well be considered equivalent. We claim that users are supported well if they are allowed to specify which notion of subexpressions they would like to use. The INN prototype supports twig-based refinements.

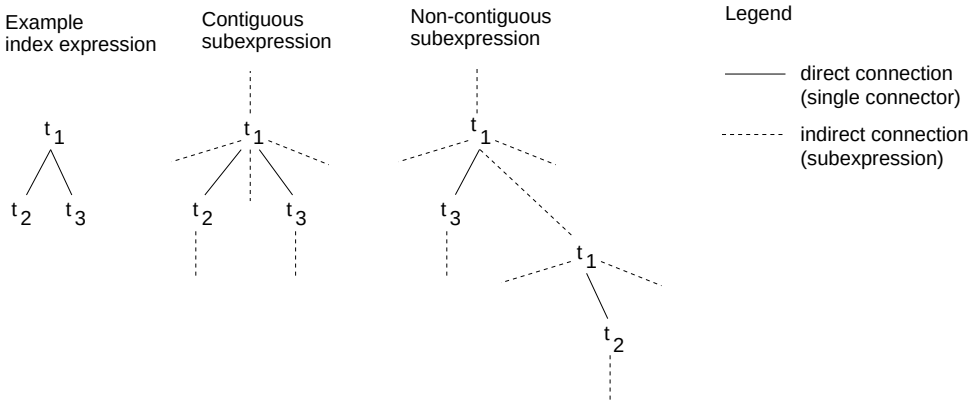

Figure 7: Non-contiguous subexpressions.

Refinements can then be obtained by augmenting the focus with new twigs from the title. Every such twig leads to a refinement. A twig is new if it is contained in the title but not in the focus. Furthermore, the new twig $t_1 c t_2$ should share exactly one of its terms with the focus. The shared node forms the anchor for augmentation.

Enlargements. The enlargements of the focus are computed by defoliation ([17]). An enlargement of an index expression is obtained by removing a single leaf and its connector. In addition, if the root of the focus has only one subexpression, the root (and its connector) can also be removed to obtain an enlargement.

The empty index expression is not included in the INN system since it bears no information. This means that single terms have no enlargements

in the INN system. Since the example query is a single term, figure 6 contains no enlargements.

Related topics. Some of the titles do not (literally) contain the focus and thus do not lead to refinements. However, since they were rendered by the search facility, they may very well be relevant to the user. Therefore, the top 10 (according to the search engine's relevance estimates) documents are included. By clicking on the title, direct access to the document is provided. In addition, the magnifying glass is also given which uses the search facility to render documents that are related (about) the title. The selected search engine is used for this.

For the example query, a reference to a page about the Sigir conference on IR is given. Clearly, this topic is related.

5.3 Beaming between Layers

When a satisfactory descriptor has been reached or when an unknown concept is arrived at, the user can transfer himself to the documents that are about that descriptor (beam down). In this way, the user is enabled to see if the documents satisfy his information need and he can learn what the concept is about. The back button of the browser enables the user to beam up again.

In the presentation of the documents, the result of the search engine is included as a frame. In this frame, the user can use the facilities offered by the selected search engine. In addition, the user is enabled to return to the INN homepage to start a new navigation session.

6 Experiences with the INN System

6.1 When No Relevant Documents are Available

For most topics, queries that contain six or more terms do not provide refinements. These queries are too specific. Of course, at a certain point, no new documents are available. However, this point can be delayed by including more index expressions in the characterisations. This means that not only the titles but also (a part of) the contents of documents should be parsed to index expressions. Since most titles are index expressions, we were able to use a simple one-pass parser. If index expressions are to be extracted from text, index expressions must first be located in the text. This can be done by part-of-speech tagging (see e.g. [3]), which labels words by their part-of-speech, or type (verb, noun, adjective, etc.). When index expressions are located, our parsing technique can be applied.

6.2 Notion of Subexpression

Two notions of subexpressions, contiguous and non-contiguous, were described earlier in this article. In certain cases, however, these notions do no result in

linguistically correct subexpressions. For instance, on path expressions, index expressions where every node has branching degree one, the last term often denotes the main concept. The prior terms modify this concept. As an example, consider little ∘ green ∘ men, where the main concept is men which is modified by green and little, respectively. A linguistically correct refinement of attack in attack by little ∘ green ∘ men is attack by men but not attack by little. In addition, post modification can co-occur with pre-modification. It is thus important to locate the main concept, which can be tried by using knowledge about word forms. At this moment, this remains an issue for further research.

7 Conclusions

In this article, we showed how navigational tools for searching and exploring large and dynamic information spaces can be based on Query by Navigation. The size and dynamic nature of the information space, in our case the WWW, was coped with by creating the required part of the navigational network on the fly.

The contributions of this article are the following. First, we showed how our tool, the INdex Navigator, is grounded in the theory about navigational networks for index expressions. Second, we provided insight in the workings of the tool. As added new functionality in relation to similar tools, we included short cuts to related topics, Fourth, we developed a new technique to compute subexpressions based on the twigs of index expressions.

Further research can be directed in a number of ways. First, a graphical user interface may enhance the feeling of navigation. An good overview of graph visualisation techniques and applications can be found in [9].

Second, extracting index expressions from the contents of documents enhances the effectiveness of the tool as well. This may require additional linguistic knowledge in the form of, for instance, a part-of-speech tagger, extractor, and parser ([3]).

Third, the log file containing user actions that is created can be examined. This may lead to insight in the (relative) frequencies of the use of different actions (e.g. refinements and enlargements). In addition, it may be possible to obtain patterns in user behaviour, based on which user support during navigation may be supplied. User support may then aim at decreasing the user's cognitive load by filtering out topics that are likely to be irrelevant or at decreasing the expected search length by providing short cuts to expected destinations.

References

[1] M. Agosti, G. Gradenigo, and P.G. Marchetti. A Hypertext Environment for Interacting with Large Textual Databases. *Information Processing and Management*, 28(3):371–387, 1992.

[2] M. Agosti and A.F. Smeaton. *Information Retrieval and Hypermedia.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[3] A. T. Arampatzis, T. Tsoris, C. H. A. Koster, and Th. P. van der Weide. Phrase-based Information Retrieval. *Information Processing & Management*, 34(6):693–707, December 1998.

[4] N.J. Belkin, R.N. Oddy, and H.M. Brooks. ASK for information retrieval. Part I. Background and theory. In *Journal of Documentation*, volume 38, pages 61–71, 1982.

[5] F.C. Berger. *Navigational Query Construction in a Hypertext Environment.* PhD thesis, Department of Computer Science, University of Nijmegen, September 1998.

[6] P.D. Bruza. *Stratified Information Disclosure: A Synthesis between Information Retrieval and Hypermedia.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[7] P.D. Bruza and Th.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[8] S. Dennis, R. McArthur, and P. Bruza. Searching the World Wide Web Made Easy? The Cognitive Load Imposed by Query Refinement Mechanisms. In *Proceedings of the 3rd Australian Document Computing Symposium*, Sydney, Australia, 1998.

[9] Herman, I. and Melancon, G. and Marshall, M.S. Graph Visualisation in Information Visualisation. In B. Falcidieno and J. Rossignac, editors, *Proceedings of Eurographics '99*, Aire-la-Ville, 1999.

[10] R. Iannella, N. Ward, A. Wood, H. Sue, and P. Bruza. The open information locator project. Technical report, Resource Discovery Unit, Resource Data Network, Cooperative Research Centre, University of Queensland, Brisbane, Australia, 1995.

[11] D. Lucarella and Z. Zanzi. Information Retrieval from Hypertext: An Approach using Plausible Inference. *Information Processing & Management*, 29(3):299–312, 1993.

[12] I. Ounis and M. Pasca. RELIEF: Combining Expressiveness and Rapidity in one System. In W.B. Croft, A. Moffat, C.J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 266–274, Melbourne, Australia, August 1998. ACM Press.

[13] R. Ragas. The effect of linguistic form on user's behaviour and performance in free-text retrieval. Master's thesis, Univerity of Nijmegen, Nijmegen, The Netherlands, 1996.

[14] D.R. Swanson. Historical Note: Information Retrieval and the Future of an Illusion. *Journal of the American Society for Information Science*, 32:92–98, 1988.

[15] J.A. Waterworth and M.H. Chignell. A Model for Information Exploration. *Hypermedia*, 3(1):35–58, 1991.

[16] R. Wilkinson and M. Fuller. Integrated Information Access via Structure. In M. Agosti and A. Smeaton, editors, *Hypertext and Information Retrieval*, pages 257 – 271, Boston, U.S.A, 1996. Kluwer.

[17] B.C.M. Wondergem, P. van Bommel, and Th.P. van der Weide. Nesting and Defoliation of Index Expressions for Information Retrieval. *Knowledge and Information Systems*, 1999. To appear.