Anticipatory User Interfaces for Search Result Visualization Using Query Lookahead

Bienvenido Vélez-Rivera
University of Puerto Rico-Mayagüez

Juan Torres
University of Puerto Rico-Mayagüez

Follow this and additional works at: http://aisel.aisnet.org/amcis2001

Recommended Citation
http://aisel.aisnet.org/amcis2001/234

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2001 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Abstract

This paper presents the first anticipatory user interface for search result visualization that exploits query lookahead, a technique that eagerly evaluates refined queries automatically generated from an initial user query. A selected collection of refined queries is displayed to the user as an interactive query hierarchy. Each child query refines its parent with additional indexing terms that are extracted from the documents that match the parent query. The user interface presents the results of each selected child query in advance. Interactive query hierarchies can also break down a large and imprecise result set in categories determined by the indexing terms appearing in the documents in the result set. Inforadar, our new prototype network search system, uses a novel, fast query lookahead algorithm achieving a throughput of 7000 refined queries per second in a database of about eighty thousand documents.

Keywords: Information retrieval, interactive query hierarchies, query lookahead

Introduction

Our research addresses a fundamental problem in information retrieval. Formulating precise queries requires detailed knowledge of an underlying document collection, but as the size of the collection increases, obtaining this detailed knowledge becomes impractical. Simple queries often return extremely large and imprecise result sets when used with large corpora. The lack of adequate assistance from the retrieval system renders the task of visualizing such result sets arduous.

A new information discovery technique called query lookahead invests additional computation expected from advances in processor technologies on the eager evaluation of multiple queries automatically generated from an initial user query. Query lookahead has the potential of improving search systems in at least two novel ways. First, it enables the deployment of anticipatory user interfaces capable of presenting the result sets of automatically generated refined queries ahead of time. Refined queries serve as categories upon which a large and imprecise result set can be organized. Second, query lookahead has the potential of improving the effectiveness of feature (e.g. term) selection algorithms. These algorithms can be improved by exploiting information about the result set induced by each potential term when combined with the user query. This paper presents a new network search system, Inforadar, exploiting query lookahead along these two lines. In response to a user query, Inforadar displays a hierarchically organized selection of refined queries that we call an interactive query hierarchy.

An interactive query hierarchy is a tree of queries automatically generated from an initial root query submitted by a user. Figure 1 shows an interactive query hierarchy for the query space. An interactive query hierarchy offers a selection of automatically generated queries that enable the user to quickly pursue more focused searches. The queries are arranged in a tree, where the root
is the query submitted by the user. Each child query refines its parent query with additional indexing terms automatically extracted from the documents that match the parent query. These indexing terms narrow the focus of the query and therefore help the user quickly formulate a precise specification of their information need. Interactive query hierarchies are convenient for focusing on an information need because they provide progressively finer descriptions of the pertinent information subspaces.

We have developed Inforadar as a vehicle for testing our hypothesis that interactive query hierarchies can improve information discovery effectiveness. Inforadar has three main software components: a multi-threaded Java applet, a server module and an indexing module. Inforadar supports boolean queries using a syntax borrowed from the popular Altavista (www.altavista.com) search engine. In response to a query request from the applet, the Inforadar server returns a hierarchy of queries together with their individual result sets. The applet renders the query hierarchy allowing the user to expand or collapse the nodes in the tree.

Interactive query hierarchy generation can be computationally expensive because of the potentially large number of queries that must be eagerly evaluated. For simplicity and efficiency, Inforadar computes child queries by conjoining a single indexing term to a parent query. The result set of each child query is thus a subset of the result set of its parent. This semantics permits the parallel evaluation of child queries while the indexing terms of the parent query are extracted. The result is a dramatically faster query lookahead algorithm. In experiments with a collection of eighty thousand press articles Inforadar achieves throughput of 7000 queries evaluated per second. Our query lookahead algorithm demonstrates that eager query evaluation entails computation that is within a small constant factor of that required by previous query refinement algorithms (Velez et al. 1997).

The remainder of the paper reviews related work, describes the Inforadar user interface, introduces the query hierarchy generation algorithms, discusses our experimental results and offers our conclusions and directions for future work.

Related Work

Interactive query hierarchies build on our previous work with query refinement on the Community Information System (Gifford et al. 1985) as well as on the Discover (Sheldon et al. 1995) and HyPursuit (Weiss et al. 1996) content routing (Sheldon 1995) systems. In the Community Information System, a simple theorem prover assists the user in choosing query terms that guarantee the query can be satisfied at available news wire databases. HyPursuit also offers a browsable set of more general and more specific terms, though the browsing tools are limited and the hierarchy is a simple two-level thesaurus. Query refinement (Harman 1988 and Sheldon 1995) offers a flat list of query suggestions and it does not eagerly compute the result set for each query. Therefore, query refinement cannot select terms that make up a succinct description of a result set. There is no way, for instance, to determine whether two indexing terms describe very similar subsets of the parent result set. Another important difference is that typical systems supporting query refinement force users to repetitively submit refined queries in order to examine their result sets. Query lookahead allows users to visualize the result set of each child query immediately and without additional communication with the server.

Multi-level thesauri have been used to overcome the lack of hierarchical organization in query refinement. For example, the Lexical Navigation system (Cooper and Bird 1997) uses a “context thesaurus” to suggest to the user terms related to the original query. Other approaches such as (Schatz et al. 1996) make use of multi-level subject thesauri to produce a tree of related terms. While the use of thesauri may effectively place a particular query within a hierarchical context, it is also limited. One important limitation is that a thesaurus can only produce a fixed hierarchical structure with pre-determined granularity and depth, thus unnecessarily bounding the refinement process. In addition, the construction of effective thesauri may require a large degree of human involvement, which makes indexing large collections of documents very costly.

The Open Information Locator project (Iannella et al. 1994) has also taken up query refinement. They make use of two techniques: non-monotonic reasoning which combines user-directed relevance feedback and term co-ocurrence analysis, and query by navigation which allows the user to browse a hierarchy of more general and more specific terms. Unlike using thesauri, co-occurrence analysis produces related terms directly from the result set of the original query. But it cannot place them within a hierarchy. Consequently, co-occurrence lists are best used for expanding a result set rather than categorizing and refining it. This shortcoming of co-occurrence lists has also been recognized by (Schatz et al. 1996), which combines co-occurrence lists with an additional hierarchical term list.

A number of systems have embraced clustering in the struggle to improve information retrieval. Scatter/Gather (Cutting et al. 1992 & 1993) uses agglomerative document clustering to support browsing a hierarchical representation of a query result set. Envision (Nowell et al. 1996) and Sense Maker (Wang et al. 1997) also have the ability to display clustered search results although their clustering criteria is limited to structured meta-information available from the corpus (e.g. web site URL, keywords, author). In general, this information is not available or requires extensive human involvement.
What differentiates Inforadar from all previous systems is its ability to generate dynamic query hierarchies without extensive human intervention. The hierarchical structure of the terms not only provides a context for the original query, but also presents a visual broken-down description of the result set. The hierarchy can be used to either expand or refine a particular query while providing constant feedback and interaction to aid the user in his/her search. In Inforadar browsing, searching and query formulation are all unified in a single user interaction paradigm.

A New User Interface

This section describes the user Inforadar interface with special emphasis on its use of interactive query hierarchies. The description will use the single-term query space as the initial user query. The documents returned by the search engine come from a collection of about eighty thousand Associated Press articles from 1989. This collection of articles is a subset of the TREC (trec.nist.gov) document collection (Harman 1994).

The Inforadar client is executed by using a web browser to retrieve a web page containing appropriate applet directives. Initially, the applet displays a window like the one in Figure 2. This window has the customary text region in which the user types a query. The window also contains five tabbed panels titled: New Query, Hierarchy, Basket, Preference and Help. The first time the applet is executed it displays the New Query panel.

To submit the query typed in the text region the user presses the Search button. In response, the applet sends the query to the server module, the address of which is specified as an applet parameter. Following the procedures described in detail in the next section, the server module evaluates the query against its database and returns a tree of queries together with their individual result sets.

The results returned by the server are rendered by the applet in the Hierarchy panel. Continuing with our example, Figure 3 shows the Hierarchy panel after submitting the example query space. The Hierarchy panel has three sub-panels numbered from 1 to 3 in the Figure. Sub-panel 1 (left) displays the interactive query hierarchy. Sub-panel 2 (center) displays the ranked list of documents matching the selected node in the hierarchy. Sub-panel 3 (right) displays a "panoramic" graphical rendering of the list of documents displayed in sub-panel 2. Each sub-panel is described in more detail below.

Sub-panel 1 displays the query hierarchy using labeled circular icons to represent nodes and lines to connect parent nodes to their children. Each child query is labeled by the query term that it contributes to the parent query. For instance, the query associated with the node labeled booster is the query (space ^ booster). The ^ symbol represents logical conjunction (i.e. AND). The number in parenthesis next to each query label represents the number of documents in the result set specific to the corresponding query. For instance, the query (space ^ booster) has 85 matching documents among the 500 documents that match its parent query (space).

Figure 3 illustrates how an interactive query hierarchy can organize a large number of documents matching a simple user query into categories made up by the refined queries automatically generated by the Inforadar server module. Although by no means ideal, the interactive query hierarchy shown in Figure 3 exposes several interesting subtopics related to the original query space. Some examples are booster, disaster and magellan. Other sub-topics, on the other hand, look less interesting in the sense that they convey less information about the documents that they match. What distinguishes the terms generated by Inforadar from other query refinement techniques is its application of query lookahead information to select the indexing terms used to refine the user query. This subject will be examined in more detail in the section that follows on query hierarchy generation.

Continuing with our description of the user interface, there are five operations that can be performed on query nodes: select, expand, collapse, delete and make-root. A node can be selected with a single left-click of the mouse on its label. In response to a select operation, the applet highlights the selected node and displays the corresponding query above sub-panel 2. In Figure 3 the root node, labeled space, appears selected. All other operations on query nodes can be triggered by selecting from an menu (see Figure 4) that pops up when the user right-clicks on the target node.

Expand, which can also be triggered by double clicking on the query node label, may have one of two outcomes depending on whether the node has been expanded before or not. The first time a node is expanded the applet sends a request to the server to
generate children queries for the selected node. The server conducts a procedure similar to generating children queries for the initial user query.

This time, however, the server uses the result set of the parent query, computed by a previous request, as the basis for its generation of child queries. The applet is responsible for sending back the parent result set as part of its request to the server.

If the node has been expanded before, double clicking on it toggles between displaying and hiding its children queries. The collapse operation simply hides the children of the corresponding node. A plus (+) symbol next to the circular query node icon indicates that the node contains hidden children. A minus (-) symbol indicates that the node is expanded and its children are visible. No symbol is shown when the node has never been expanded before. Figure 4 shows that the nodes labeled booster, disaster and gravity have been expanded, but only the node labeled disaster has visible children.

Delete permanently removes the node from the hierarchy. This operation can be used to prune out nodes that are either uninteresting or redundant. Make-root forces the selected node to become the root of a new interactive query hierarchy. This entails a new request to the server and is equivalent to typing the corresponding query at the New Query window. This mechanism provides an easy shortcut to users who want to focus on a single sub-query of interest.

Sub-panel 2 of the Hierarchy panel continuously displays the ranked list of documents matching the selected query. Because of its use of query lookahead, updating this display in response to a selection of a different query never entails communication with the server module. Moreover, the hierarchy of result sets can be examined in any order without having to repetitively submit queries that lie higher in the hierarchy. This later inconvenience is commonly found on search engines, like Altavista and Excite (www.excite.com), supporting plain query refinement.

Sub-panel 3 of the Hierarchy panel displays a similarity graph for the result set corresponding to the query node highlighted in sub-panel 1. This type of graph constitutes our first attempt to provide a graphical “panoramic” view of a potentially large result set. The similarity graph is essentially a bar graph with one bar for each document in the result set. The bars are arranged horizontally and their height is determined by the similarity between the document and the selected query.

The user can scroll down the document list using the slider associated with the graph. The applet keeps both the result set and the similarity graph panels synchronized. For instance, in Figure 4 both sub-panels show that document 3 in the result set for the query (space ^ disaster) is selected. We are currently investigating how these types of panoramic graphical displays of query result sets can improve information visualization. We hypothesize that these views may uncover important properties that can speed the task of sifting each result set. In particular, we are searching for graphs that could suggest the user how far down to examine a large ranked result set.

Inforadar ranks documents according to classical document-to-query similarity measures (Frakes and Baetza 1992). Double clicking on a document abstract displays the full text of the corresponding article and marks the abstract with a spy glass icon. In Figure 4 documents 1 and 3 appear with corresponding spy glass icons. Marking documents as they are read is particularly useful in Inforadar because the same
document may appear in several different result sets. After reading several documents it usually becomes hard for the user to keep track of which documents have been read.

In an interactive query hierarchy, the result sets corresponding to the child queries of a given node do not necessarily form a partition of the result set associated with their parent node. A given document may appear in the result set of more than one child query. Moreover, the union of the result sets of a collection of sibling child queries may not be equal to the result set of their parent query. The rationale for this choice of semantics for interactive query hierarchies will be explained in the next section on query hierarchy generation.

**Inforadar** takes advantage of Java's multi-threading facilities. For instance, several user interface operations can be performed simultaneously. To provide the user with adequate feedback, the applet uses colored icons to represent the status of query nodes. For instance, the icon associated with the node labeled *astronauts* in Figure 4 is yellow (rendered white in this paper) to indicate that the applet is waiting for a reply to a request to expand the node sent to the server.

The **Inforadar** user interface is unique among network search engines in that the user interface state is persistent across multiple user queries. For instance, a user can expand nodes in the hierarchy in an attempt to pursue search paths of interest. Upon exhausting a search path a user can easily backtrack arbitrarily to any upper level query in the hierarchy since all the nodes in the tree remain accessible.

Persistence is also exploited by the **Inforadar** document basket mechanism. The mechanism allows a user to collect documents in the document basket by clicking on the *Add to basket* button at the top of the Hierarchy panel. In sub-panel 2 the abstracts of documents added to the basket appear marked with a basket icon (see document 3 in Figure 4). To save space the Basket panel is not shown, but it simply allows a user to examine, modify and print the list of documents put in the basket. The document basket feature is particularly useful because it persists across multiple queries. As with the spy glass icon, use of the basket icon is important because the same document may be visited via different query nodes.

The preference panel, shown in Figure 5, allows the user to select a number or parameter used by the search engine to generate the query hierarchies. A change in the preferences panel becomes active the next time a new query is entered or a node is expanded for the first time. The hierarchy width is the maximum number of child queries that any query may have. The hierarchy depth is the length of the longest path from the root to a leaf query. The document frequency threshold parameter allows the user to control the maximum document frequency (i.e. the number of documents containing the term) allowed for terms selected as nodes. The document sample size controls the number of documents scanned for terms as well as the number of documents retrieved for the root query node.

### Query Hierarchy Generation

This section presents a top-down description of the algorithm currently used by **Inforadar** to generate interactive query hierarchies. Throughout the description we will make minimal use of mathematical notation. More specifically, the set of documents matching a query \( q \) (i.e. its result set) will be denoted as \( D(q) \). The set of indexing terms appearing documents in \( D(q) \) will be denoted by \( F(q) \).

### Algorithm Overview

The essential input to the query hierarchy generation algorithm consists of a root (user) query and two numbers representing the width and depth of the hierarchy. The width is the maximum number of children that any node may have. The depth is the length of the longest path from the root to any descendant node.

In **Inforadar** multi-level query hierarchies are generated one level at a time using Interactive Query Hierarchy, the recursive algorithm shown in Figure 6. The \(<a,b,...>\) notation is used to represent n-tuples. The actual work is performed by the Query Refinement algorithm which generates a single-level query hierarchy. Therefore, the remainder of this section will focus on the details of Query Refinement.
As shown in Figure 7, the Query Refinement Algorithm can be decomposed into four phases: result set sampling, term extraction, query lookahead, and term selection. The input to the algorithm is the root query \( q \) and its result set \( D(q) \). The output consists of a set of pairs \( \langle q_i, D(q_i) \rangle \) of child queries and their individual result sets. The following subsections describe each phase of query hierarchy generation in more detail.

Due to space limitations, we have been unable to present detailed descriptions of the different algorithm phases as well as performance evaluations. The interested reader is referred to (Velez 1999) for further details.

References


Cutting, Douglass R., David R. Karger, and Jan O. Pedersen. “Constant interaction-time scatter/gather browsing of very large document collections,” In 16th Annual International SIGIR ’93, June 1993, pp. 126-134.


