BioNav: Effective Navigation on Query Results of Biomedical Databases

Abhijith Kashyap[*1], Vagelis Hristidis[#], Michalis Petropoulos[*2], Sotiria Tavoulari[+]

[*]*Department of Computer Science and Engineering, SUNY at Buffalo*
201 Bell Hall, Buffalo, NY 14260-2000, USA
[1]rk39@cse.buffalo.edu
[2]mpetropo@cse.buffalo.edu

[#]*School of Computing and Information Sciences, Florida International University*
11200 S.W. 8th Street, Miami, FL 33199, USA
vagelis@cis.fiu.edu

[+]*Department of Pharmacology, Yale University*
333 Cedar Street, New Haven, CT 06520-8066, USA
sotiria.tavoulari@yale.edu

Abstract— **Search queries on biomedical databases like PubMed often return a large number of results, only a small subset of which is relevant to the user. Ranking and categorization, which can also be combined, have been proposed to alleviate this information overload problem. Results categorization for biomedical databases is the focus of this work. A natural way to organize biomedical citations is according to their MeSH annotations, a comprehensive concept hierarchy used by PubMed.**

In this paper, we present the BioNav system, a novel search interface that enables the user to navigate large number of query results by organizing them using the MeSH concept hierarchy. First, the query results are organized into a navigation tree. Previous works expand the hierarchy in a predefined static manner. In contrast, BioNav uses an intuitive navigation cost model to decide what concepts to display at each step. Another difference from previous works is that the hierarchy is not strictly displayed level-by-level.

I. INTRODUCTION

The last decade has been marked by unprecedented growth in both the production of biomedical data and the amount of published literature discussing it. The MEDLINE database, on which the PubMed search engine operates, contains over 18 million citations, and the database is currently growing at the rate of 500,000 new citations each year [7]. Biologists, chemists, medical and health scientists are used to searching their domain literature – such as PubMed – using a keyword search interface. Currently, in an exploratory scenario where the user tries to find citations relevant to her line of research and hence not known a priori, she submits an initially broad keyword-based query that typically returns a large number of results. Subsequently, the user iteratively refines the query, if she has an idea of how to, by adding more keywords, and re-submits it, until a relatively small number of results are returned. This refinement process is problematic because after a number of iterations the user is not aware if she has over-specified the query, in which case relevant citations might be excluded from the final query result.

As an example, a query on PubMed for "cancer" returns more than 2 million citations. Even a more specific query for "prothymosin", a nucleoprotein gaining attention for its putative role in cancer development, returns 313 citations. The size of the query result makes it difficult for the user to find the citations that she is most interested in, and a large amount of effort is expended searching for these results. Many solutions have been proposed to address this problem – commonly referred to as *information-overload* [1]-[3], [5]. These approaches can be broadly classified into two categories; ranking and categorization, which can also be combined.

The primary focus of BioNav is on categorization techniques, which are ideal given the rich concept hierarchies (e.g., MeSH[6]) available for biomedical data. We augment our categorization techniques with simple ranking techniques. BioNav organizes the query results into a dynamic hierarchy, the *navigation tree*. Each concept (node) of the hierarchy has a descriptive label. The user then navigates this tree structure, in a top-down fashion, exploring the concepts of interest while ignoring the rest.

An intuitive way to categorize the results of a query on PubMed is using the MeSH static concept hierarchy [6]. Each citation in MEDLINE is associated with several MeSH concepts in two ways: (i) by being explicitly annotated with them, and (ii) by mentioning those in their text. Since these associations are provided by PubMed, a relatively straightforward interface to navigate the query result would first attach the citations to the corresponding MeSH concept nodes and then let the user navigate the navigation tree. Fig. 1 displays a snapshot of such an interface where shown next to each node label is the count of distinct citations in the subtree rooted at that node. A typical navigation starts by revealing the children of the root ranked by their citation count, and is continued by the user expanding on or more of them, revealing their ranked children and so on, until she clicks on a concept and inspects the citations attached to it. A similar interface and navigation method is used by GoPubMed [9]

and e-commerce sites, like Amazon and eBay. For this example, we assume that the user will navigate to the three indicated concepts corresponding to three independent lines of research related to prothymosin.

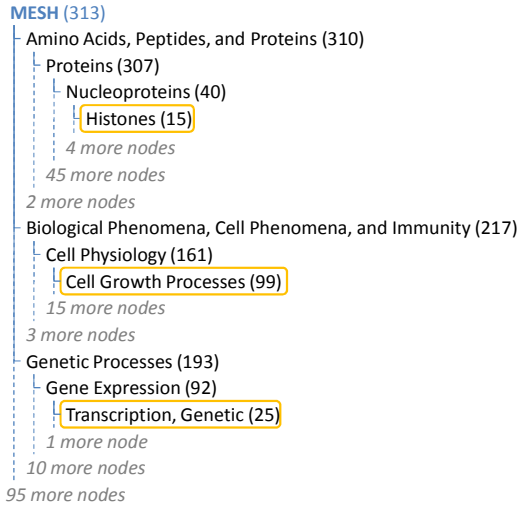

Fig. 1 Static Navigation on the MeSH Concept Hierarchy

The above *static* –same for every query result– navigation method is problematic when the MeSH hierarchy is used for categorization for the following reasons:

- The massive size of the MeSH hierarchy (over 48,000 concept nodes) makes it challenging for the users to effectively navigate to the desired concepts and browse the associated records. Even if we remove from the MeSH concept nodes with no citations attached to them, the 313 distinct query results for "prothymosin" are attached to 3,940 nodes, which is the actual size of the navigation tree in Fig. 1.
- A substantial number of *duplicate* citations are introduced in the navigation tree of Fig. 1, since each one of the 313 distinct citations is associated with several concepts.

BioNav introduces a *dynamic* navigation method that depends on the particular query result at hand and is demonstrated in Fig. 2. The query results are attached to the corresponding MeSH concept nodes as in Fig. 1, but then the navigation proceeds differently. The key action on the interface is the *expansion* of a node that selectively reveals a ranked list of descendent (not necessarily children) concepts, instead of simply showing all its children.

Fig. 2a, for example, shows the initial expansion of the root node where only 8 (highlighted) descendants are revealed compared to 98 children shown in Fig. 1. The concepts are ranked by their relevance to the user query and the number of them revealed depends on the characteristics of the query results. Next, assuming the user is interested in the "Amino Acids..." node and judging that the 310 attached citations is still a big number, she expands it by clicking on the ">>>" hyperlink next to it in Fig. 2b. The user inspects the 6 concepts revealed and decides that she is not interested in any of them. Hence, she expands the "Amino Acids..." node one more time in Fig. 2c, revealing 4 additional concepts. Note that "Nucleoproteins" is an example of a descendant node being revealed, since its parent node "Proteins" is not revealed in Fig. 2c. In Fig. 2d, the user expands the "Nucleoproteins" node and reveals "Histones", one of the three key concepts for the query. To reach "Histones" using the BioNav navigation method, only 23 concepts are revealed, after 4 node expansions, compared to 152 concepts, also after 4 expansions, with the static navigation method of Fig. 1.

Fig. 2 BioNav Dynamic Navigation

For each expansion, the displayed descendent concepts are chosen in a way that the expected navigation cost is minimized, based on an intuitive navigation cost model we present in Section III. For example, the reason that "Proteins" is not displayed in Fig. 2 is that it is too general given the query results and the original distribution of citations in the PubMed database and hence displaying it would lead to an increase in the (expected) user navigation cost.

In addition to the static hierarchy navigation works mentioned above, there are works (e.g., the Clusty search

engine [8], or [2], [3]) on dynamic categorization of query results, which create unsupervised query-dependent results clusters. BioNav is distinct since it offers dynamic navigation on a predefined hierarchy, as is the MeSH concept hierarchy. Another difference is that BioNav uses a navigation cost model to minimize the navigation cost.

II. Framework and BioNav Overview

The MeSH concept hierarchy is the starting point of the framework and is defined as follows.

Definition 1 (Concept Hierarchy): A *Concept Hierarchy* $H(V, E, r)$ is a labeled tree consisting of a set V of concept nodes, a set E of edges and is rooted at node r. Each node $n \in V$ has a label l and a unique identifier *id*. □

According to the semantics of the MeSH concept hierarchy [13], the label of a child concept node is more specific than the one of its parent. This also holds for most concept hierarchies.

Once the user issues a keyword query, PubMed–BioNav uses the Entrez Programming Utilities (eUtils) [4]–returns a list of citations, each associated with several MeSH concepts. BioNav constructs an *Initial Navigation Tree* by attaching to each concept node of the MeSH concept hierarchy a list of its associated citations. Formally, an Initial Navigation Tree $T_I(V_I, E_I, r)$ is a concept hierarchy, where every node (concept) $n \in V_I$ is additionally labelled with a *results (citations) list* $L(n)$.

In a given initial navigation tree, several concept nodes might have an empty results list. Since MeSH is a rather large concept hierarchy, BioNav reduces the size of the initial navigation tree by removing the nodes with empty results lists, while preserving the ancestor/descendant relationships. Formally, the resulting structure is defined as follows.

Definition 2 (Navigation Tree): A *Navigation Tree* $T(V, E, r)$ is the maximum embedding of an initial navigation tree $T_I(V_I, E_I, r)$ such that no node $n \in V$ is labeled with an empty results list $L(n)$, excluding the root (in order to maintain the tree structure and avoid the creation of a forest).□

In principal, an *embedding* $T(V, E, r)$ of a tree $T_I(V_I, E_I, r)$ is an injection from V to V_I such that every edge in E corresponds to a path (disjoint from all other such paths) in T_I. An embedding T of a tree T_I, where both trees are rooted at node r, is *maximum* if no other node n with a nonempty results list $L(n)$ can be added to V and T still be an embedding. The maximum embedding of the initial navigation tree is recursively computed in a single depth-first left-to-right traversal. If a node n has an empty results list $L(n)$, then replace n with its children. If n is a leaf, then simply remove it. Fig. 3 shows part of the navigation tree for the "prothymosin" query, where the results lists are omitted for clarity.

The above procedure reduces the size of the initial navigation tree, but the structure is still too big (3,940 nodes for query "prothymosin") to simply display it to the user or let her navigate it, especially if her query is of exploratory nature. BioNav minimizes the user's effort to reach the desired citations in the navigation tree by expanding in a way that minimizes the expected overall user navigation cost. Moreover, BioNav avoids information clutter by hiding unimportant concept nodes leading to interesting ones. This is achieved through a series of *expand* actions that reveal only a few descendants (not necessarily children) of the user selected node for further navigation.

We model a node expansion at a given navigation step as an *EdgeCut* in the navigation tree. In graph theory, an EdgeCut in a graph $G(V, E)$ is a set of edges $E_C \subseteq E$ such that the graph $G'(V, E \backslash E_C)$ has more components than G. For trees, any subset of the edges constitutes an EdgeCut, since the removal of any single edge creates a new component.

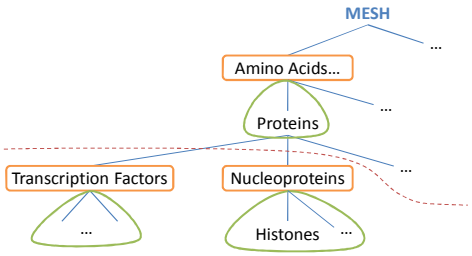

Fig. 3 Navigation Tree, EdgeCut and Component Subtrees

In Fig. 3, the dashed line illustrates the EdgeCut corresponding to the expansion of the node "Amino Acids…". This expansion reveals the highlighted concepts of Fig. 3, which include a subset of the highlighted concepts in Fig. 2c. The EdgeCut consists of the edges ("Proteins", "Transcription Factors") and ("Proteins", "Nucleoproteins"). Intuitively, an EdgeCut allows us to "skip" child nodes ("Proteins") and navigate directly to descendent nodes located deeper in the tree and show them as children of the node being expanded. Moreover, an EdgeCut can selectively reveal only a subset of a potentially large set of descendent nodes, as is the case in Fig. 2b where only 6 out of the 52 descendents of "Amino Acids..." are revealed.

Definition 3 (Valid EdgeCut): A valid EdgeCut of a tree $T(V, E, r)$ is an EdgeCut $C \subseteq E$ such that no two edges in C appear in a path from the root to a leaf node. □

We only consider valid EdgeCuts in the rest of the paper, because invalid EdgeCuts lead to unintuitive navigations.

Component Subtrees An EdgeCut causes the creation of two types of *component subtrees*, *upper* and *lower*. Given an EdgeCut C of a tree $T(V, E, r)$, a *lower* component subtree $I(y_i)$ rooted at y_i is created by each node $y_i \in V$, such that $(x, y_i) \in C$ for some node x. In Fig. 2c, at the expansion of node "Amino Acids...", four lower component subtrees are created, two of which are shown in Fig. 3, rooted at "Transcription Factors" and "Nucleoproteins". Moreover, for a given EdgeCut C, a single *upper* component subtree is created and comprises of the nodes not in any *lower* component subtree, and is always rooted at the root of the tree being expanded. In Fig. 3, the upper component subtree comprises of the nodes "Amino Acids..." (root) and "Proteins".

III. Navigation Cost Model

The navigation cost model of BioNav is formally defined in this section. After the user issues a keyword query, BioNav

initiates navigation by constructing the initial results tree and displaying its root to the user. Subsequently, the user navigates the tree by performing one of the following actions on a given component subtree $I(n)$ rooted at concept node n:

1. **EXPAND $I(n)$**: The user clicks on the ">>>" hyperlink next to node n and causes an EdgeCut($I(n)$) operation to be performed on it, thus revealing a new set of concept nodes from the set $I(n)$.
2. **SHOWRESULTS $I(n)$**: By performing this action, the user sees the results list $L(I(n))$ of citations attached to the component subtree $I(n)$.
3. **IGNORE $I(n)$**: The user examines the label of concept node n, ignores it as unimportant and moves on to the next revealed concept.

This navigation process continues until the user finds *all* the citations she is interested in. The cost of a navigation is computed similarly to [2] as follows: We assign (i) cost of 1 to each newly revealed concept node that the user examines after an EXPAND action, (ii) a cost of B (to be determined empirically) to each EXPAND action the user executes, and (iii) cost of 1 to each citation displayed after a SHOWRESULTS action. An open issue is the estimation of the EXPLORE and SHOWRESULTS probabilities. The EXPLORE probability could be proportional to the number of unique results in the corresponding component subtree, whereas *normalized* entropy of the component subtree [3] can be used as the SHOWRESULTS probability.

IV. Algorithms for Best EdgeCut

Given the cost equation described in Section III and [2], we can compute the optimal cost by recursively enumerating all possible sequences of valid EdgeCuts, starting from the root and reaching every concept in the navigation tree, computing the cost for each step and taking the minimum. However, this algorithm is prohibitively expensive. Instead we propose an alternative algorithm *Opt-EdgeCut* that makes use of the *dynamic programming* technique to reduce the computation cost. *Opt-EdgeCut* is still exponential.

The *Opt-EdgeCut* algorithm to compute the minimum expected navigation cost (and the EdgeCut that achieves it) traverses the navigation tree in post-order and computes the navigation cost bottom-up starting from the leaves. For each node n, the algorithm enumerates and stores the list $\mathbb{C}(n)$ of all possible EdgeCuts for the subtree rooted at n, and the list $\mathbb{I}(n)$ of all possible $I(n)$ sets that node n can be annotated with. The algorithm then computes the minimum cost for each subtree in $\mathbb{I}(n)$ given the EdgeCuts in $\mathbb{C}(n)$ and the already computed minimum costs for the descendants of n.

V. Experimental Evaluation

We present some preliminary experimental results comparing the navigation cost of *Opt-EdgeCut* to that of the Static navigation approach, where all children of the expanded node are displayed at each step. We used a small result tree of ~20 nodes, given the high execution cost of *Opt-EdgeCut*. Heuristic algorithms to approximate *Opt-EdgeCut* will be developed in our future work. Fig. 4 shows that our approach leads to much smaller navigation costs compared to the Static approach, for 10 queries selected by expert users in the biomedical domain. The average improvement is 36% even for the small sample trees used in the experiments.

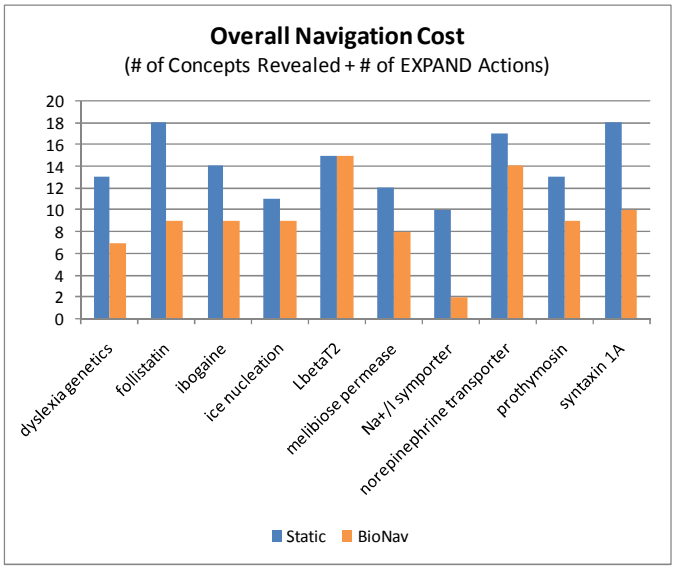

Fig. 4 Navigation Cost of BioNav vs. Static Navigation

VI. Conclusions and Future Work

Information overload is a major problem when searching biomedical databases like PubMed, where typically a large number of citations are returned, of which only a small subset is relevant to the user. In this paper, we presented the BioNav system to address this problem. Our solution is to organize the query results according to their associations to concepts of the MeSH concept hierarchy, and provide a dynamic navigation method that minimizes the information overload observed by the user. In the future, we will develop heuristic algorithms to improve the response time for expand operations in BioNav and work with domain experts to refine the navigation cost model.

VII. Acknowledgments

Vagelis Hristidis was partially supported by NSF grants IIS-0811922 and IIS-0534530.

References

[1] S. Agrawal, S. Chaudhuri, G. Das and A. Gionis: *Automated Ranking of Database Query Results*. In Proceedings of First Biennial Conference on Innovative Data Systems Research (CIDR), 2003.

[2] K. Chakrabarti, S. Chaudhuri and S.W. Hwang: *Automatic Categorization of Query Results*. SIGMOD Conference 2004: 755-766

[3] Z. Chen and T. Li: *Addressing Diverse User Preferences in SQL-Query-Result Navigation*. SIGMOD Conference 2007: 641-652

[4] (2008) Entrez Programming Utilities. [Online]. Available: http://www.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html

[5] V. Hristidis and Y. Papakonstantinou: *DISCOVER: Keyword Search in Relational Databases*. In Proc. of VLDB Conference, 2002

[6] Medical Subject Headings (MeSH®). http://www.nlm.nih.gov/mesh/

[7] J.A. Mitchell, A.R. Aronson and J.G. Mork: *Gene Indexing: Characterization and Analysis of NLM's GeneRIFs*. In Proceedings of the AMIA Symposium, 8th–12th November, Washington, DC, pp. 460–464

[8] (2008) Vivísimo, Inc. – Clusty. http://clusty.com/

[9] (2008) Transinsight GmbH – GoPubMed. http://www.gopubmed.org