ISSN:0975-9646

Effective Navigation of Query Results Using Apriori Algorithm

Tagore Kumar Tummapudi, Uma M

Department of Software Engineering, S.R.M. University
Chennai, India

***Abstract*- Search queries on biomedical databases, often return a large number of results, only some part of data is relevant to the user. Results categorization and ranking for biomedical databases is the focus of this work. The general way to organize biomedical data results is according to the MeSH annotations. MeSH is used by medical database for comprehensive searching of the query results. To alleviate the information overload problem Ranking and Categorization can be combined. 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. The efficiency of the results fetched by the user can be improved by using Data Mining Algorithms. Presently, few algorithms are being considered to provide the data to the fetched queries. A new approach for evaluating the relevant information for the query is done by Apriori Algorithm. Apriori gives the relevant information to the user by which the user can have easy way to access the information needed.**

Index Terms – MeSH, BioNav, efficient results , effectiveness

I. INTRODUCTION

Data mining is extraction of data which is useful and it is extracted from different types of databases like biomedical, web, domain specific databases, etc. Ranking and Categorization on extracted query results helps the user to fetch quality data from the database, helps in increasing efficiency of the database. A user might consider the effectiveness for a piece of information appropriate for one task but not sufficient for another task. Accuracy is subjective, as a second less quality concerned user might consider the quality of the same piece of information appropriate for both tasks. Which quality dimensions are relevant and which levels of quality are required for each dimension is determined by the specific task .The biomedical database, on which the search engine operates, contains over 18 million citations.

The user submits an initially broad keyword- based query that typically returns a large number of results with concept hierarchies associated with MeSH concept as hierarchy. Biologists, chemists, medical and health scientists and researchers will search their data from their domain literature which need to be efficient and accurate. For keyword search system will use citation id which will be annotated with concept hierarchy [1].

The efficiency of the search results from the search engines varies as information providers have different levels of knowledge and different intentions. Users of query based systems are therefore confronted with the increasingly difficult task of selecting the efficient results from the vast amount of web information the can b accepted. The web search helps in the resulting of the efficient and accurate result.

In order to optimize, the system uses concept hierarchies for navigation of query results and opt edge cut algorithm to minimize the cost and heuristic edge cut algorithm to increase the efficiency of query navigation in the biomedical database. The combination of both ranking and categorization will improve the efficiency and effectiveness of the query results. By using Apriori algorithm the efficiency will be increased resulting in the accurate result. It increases the performance in retrieving efficient query results. Quality is provided information and ratings of the website [9]. This will increase efficiency and provide effectiveness, time saving and reduce cost of search and provides to get expected trusted results.

A. Optimization of Search query Results:

Optimizing of the query results fetched by the user when a query is placed on the database is done by different techniques using ranking and categorization. Ranking is done based on citation count, citation relevance, and by date. Categorization is done by using concept hierarchies, navigation tree. Partitioning is used for categorization and the k partitioning is the linear partitioning method that is done based on the weight of the node in the tree i.e., navigation tree and the active tree is generated where the clustering is done and the dynamic pruning is also done to save the time and the cost of query processing. Based on k value the partitioning is done with the less number of k sub groups that is less than the more weight of the tree.

The clustering is the technique which is used in dividing the data into subsets i.e., categorizing the results. The data can be supervised or unsupervised data i.e., training set are already defined or training sets are not defined previously, for clustering or grouping of data the k value or the number of the sets should be known at the starting of the clustering or grouping is to be done .

The k means clustering is done based on mean distance for partitioning medians clustering is based on median distance and k medics clustering are four different techniques used for clustering the data.

II. FRAMEWORK AND BIONAV OVERVIEW

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

Definition 1 (Concept Hierarchy): A *Concept Hierarchy* is a labeled tree consisting of a set of concept nodes, a set of edges and is rooted at node . Each node has a label and a unique identifier *id*. According to the semantics of the

MeSH concept hierarchy, 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) −returns a list of citations, each associated with several MeSH concepts.

BioNav reduces the size of the initial navigation tree by removing the nodes with empty results lists, ancestor/descendant relationships are being preserved. The resulting structure is defined as follows.

Definition 2 (Navigation Tree): A *Navigation Tree* is the maximum embedding of an initial navigation tree such that no node is labeled with an empty results list, excluding the root (in order to maintain the tree structure and avoid the creation of a duplication. The maximum embedding of the initial navigation tree is recursively computed in a single depth-first left-to-right traversal. If a node has an empty results list , then replace with its children node.

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.

III. System Architecture And Implementation

The BioNav system architecture is shown in Fig. 7. There are more than 48,000 concept nodes in the hierarchy. The architecture results in the efficient and accurate results based on the query processing and query optimization. From this architecture, the query is processed from the BioNav web interface.

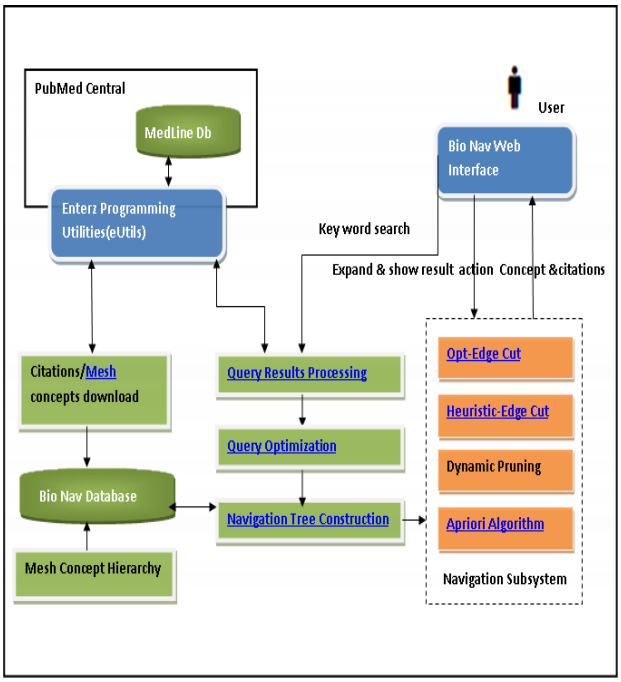

Fig. 7 BioNav System Architecture

Then, the BioNav database is populated with the associations of the MEDLINE citations to MeSH concepts. These associations are not directly provided by the Entrez Programming Utilities (eUtils), so we had to implement the following method to infer these associations. For each concept in the MeSH hierarchy, we issued a query on PubMed using the concept as the keyword. For each citation ID in the query result, we added a tuple concept, citationID to a table in the BioNav database. Alternatively, we could determine the associations by using the MeSH concepts that each citation is annotated with in the MEDLINE database. This information is available through eUtils. In this case though, the navigation trees of BioNav would not be very informative, since each citation is annotated with 20 concepts on average in MEDLINE, while the PubMed indexing associates each citation with approximately 90 concepts on average (and include the 20 from MEDLINE.) restrictions on the number of queries that can be executed within a certain period of time, it took almost 20 days to collect all the concept, citationID tuples. In the end, there were almost 740 million such tuples. To improve the selection queries on this table, we de-normalized it by concatenating all concepts associated with each citation into a comma-separated list, that is:

Citation Id, Conpcet1, Concept 2

On-Line Operation Upon receiving a keyword query from the user, BioNav executes the same query against the MEDLINE database and retrieves only the IDs (PubMed Identifiers) of the citations in the query result. This is done using the ESearch utility of the Entrez Programming Utilities (eUtils).

IV. Apriori Algorithm:

We can compute the optimal cost by recursively enumerating all possible sequences of valid Edge Cuts, 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 also prohibitively expensive. Instead we propose an alternative algorithm *Apriori Algorithm* that makes use of the *dynamic programming* technique to reduce the computation cost. Apriori Algorithm is mainly used for the providing the efficient and accurate result.

Algorithm Used

Apriori Algorithm Pseudo code

Procedure **Apriori** (T, *minSupport*)
{
//T is the database and *minSupport* is the minimum support
L1= {frequent items};
for (k= 2; Lk-1!=Φ; k++)
{
Ck= candidates generated from Lk-1
//that is Cartesian product Lk-1 x Lk-1 and eliminating any k-1 size itemset that is not
//frequent
for each transaction **t** in database **do{**

```
#increment the count of all candidates in Ck that
are contained in t
Lk = candidates in Ck with minSupport
}//end for each
}//end for
return UkLk;
}
```

Explanation

Support of an Item Set

Let S be an item set and T the multiset of all transactions under consideration. Then the *absolute support* (or simply the *support*) of the item set S is the number of transactions in T that contain S. Likewise, the *relative support*of S is the fraction (or percentage) of the transactions in T which contain S.

More formally, let S be an item set and $U = \{ X \in T \mid S \subseteq t \}$ the bag/multiset of all transactions in T that have S as a subset (i.e. contain all of the items in S and possibly some others). Then

$supp_{abs}(S) = |U| = |\{ X \in T \mid S \subseteq t \}|$

is the absolute support of S and

$supp_{rel}(S) = (|U| / |T|) * 100\%$

is the relative support of S. Here |U| and |T| are the number of elements in U and T, respectively.

The goal of frequent item set mining is to find all item sets (that is, all subsets of the item base) that occur in the given bag/multiset of transactions with at least a user-specified *minimum support* $supp_{min}$. Such item sets are called*frequent item sets*.

The default value for the minimum support in my Apriori program is 10% (the percentage indicates implicitly that it refers to relative support). This value can be changed with the option -s. Note that the argument to this option is interpreted as a percentage if it is positive, but if it is negative, it is interpreted as an absolute number (number of transactions) rather than a percentage. That is, -s20 means a minimum *relative* support of 20%, while -s-20means a minimum *absolute* support of 20 transactions.

Confidence of an Association Rule

If we search for association rules, we do not want just any association rules, but "good" association rules. To measure the quality of association rules, the inventors of the Apriori algorithm, introduced the *confidence* of a rule. The confidence of an association rule $R = "X \rightarrow Y"$ (with item sets X and Y) is the support of the set of all items that appear in the rule (here: the support of $S = X \cup Y$) divided by the support of the antecedent (also called "if-part" or "body") of the rule (here X). That is,

$$conf(R) = \frac{supp(X \cup Y)}{supp(X)}$$

(Note that it does not matter whether the confidence is computed from the absolute or the relative support of an item set, as long as the same support type is used in both the numerator and the denominator of the fraction.)

More intuitively, the confidence of a rule is the number of cases in which the rule is correct relative to the number of cases in which it is applicable. If the rule is applicable, it says that the customer can be expected to buy cheese. But he/she may or may not buy cheese, that is, the rule may or may not be correct (for this customer). Naturally, we are interested in how good the prediction of the rule is, that is, how often its prediction that the customer buys cheese is correct. The rule confidence measures this: it states the percentage of cases in which the rule is correct. It states this percentage relative to the number of cases in which the antecedent holds, since these are the cases in which the rule makes a prediction that can be true or false. If the antecedent does not hold, then the rule does not make any prediction, so these cases are excluded.

Rules are reported as association rules if their confidence reaches or exceeds a given lower limit (minimum confidence; to be specified by a user). That is, we look for rules that have a high probability of being true: we look for "good" rules, which make correct (or very often correct) predictions. My Apriori program always uses a minimum confidence to select association rules. The default value for the minimum confidence is 80%. This value can be changed with the option -c. (Note that for the minimum confidence, the argument is always interpreted as a percentage. Negative values cause an error message, because there is no "absolute confidence".)

In addition to the rule confidence, my Apriori program lets you select from several other (additional) rule evaluation measures, which are explained below, but it will also use rule confidence. If you want to rely entirely on some other measure, you can do so by setting the minimal rule confidence to zero. (Attention: If you have a large number of items, setting the minimal rule confidence to zero can result in *very* high memory consumption. Therefore: use this possibility with a lot of care, if at all.)

Support of an Association Rule

The support of association rules may cause some confusion, because I use this term in a different way do. For them, the support of an association rule "A and B → C" is the support of the set $S = \{ A, B, C \}$. This may be fine if rule confidence is the only evaluation measure, but it causes problems if some other measure is used. For these other measures it is often much more appropriate to call the support of the antecedent of the association rule, that is, the support of $X = \{ A, B \}$ in the example above, the support of the association rule.

The rule support can be used to filter association rules by stating a lower bound for the support of a rule (minimum support). This is equivalent to saying that you are interested only in such rules that have a large enough statistical basis (since my Apriori program uses the term "support" in my interpretation and not in the one used by The default value for this support limit is 10%. It can be changed with the option -s. Note that the argument, if positive, is interpreted as a percentage. If, however, the given argument is negative, it is interpreted as an absolute number (number of transactions) rather than a percentage.

The minimum support is combined with the minimum confidence to filter association rules. That is, my Apriori program generates only association rules, the confidence of which is greater than or equal to the minimum confidence*and* the support of which is greater than or equal to the minimum support.

V. NAVIGATION AND COST MODEL

The navigation model of BioNav is formally defined in this section. Then the navigation cost model is presented, which is used to devise and evaluate our algorithms in later sections. Navigation Model After the user issues a keyword query, BioNav initiates a navigation by constructing the initial active tree (which has a single component tree rooted at the Mesh root) 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 rooted at concept node:

1. EXPAND: The user clicks on the">>>" hyperlink next to node and causes an Edge Cut operation to be performed on it, thus revealing a new set of concept nodes.
2. SHOWRESULTS: By performing this action, the user sees the results list of citations attached to the component subtree.
3. IGNORE: The user examines the label of concept node, ignores it as unimportant and moves on to the next revealed concept.
4. BACKTRACK: The user decides to undo the last Edge Cut operation. This navigation process continues until the user finds *all* the citations she is interested in. In order to define a cost model, we focus on a simplification of the general navigation model, which we call TOPDOWN, where only EXPAND, SHOWRESULTS.

VI. COMPLEXITY RESULTS

To prove that the problem of selecting the optimal valid Edge Cut for a given tree is NP-Hard, where "optimal" means minimize the user navigation cost according to the navigation model of Section, we prove that the problem is Incomplete for a simplified navigation model, which we refer to as TOPDOWN-EXHAUSTIVE and is a special case of the TOPDOWN model.

Hence, the duplicates are the reason that the problem is NP-complete for TOPDOWN-EXHAUSTIVE, because we need to maximize the number of duplicates within the created sub trees, and at the same time create a relatively small number of component sub trees. Note that even for a given ?, the problem of selecting the best Edge Cut is NP-hard

	Algorithm	Processing Time (Minute)	Accuracy (%)	Unused Fields	Area Under Curve
1	K-Means	<1	80	6	0.93
2	Clustering	<2	85	3	0.75
3	Opt Edge Cut	<3	75	5	0.76
4	K-Partition	<2	80	4	0.85
5	Apriori	<1	92	8	0.95
6	SVM	<1	90	7	0.90

Table 1 Comparative study of Algorithms

Presently the algorithms which are using in the medical database searching have been providing large amount of the data. By using the Apriori Algorithm we can provide the accuracy and efficiency for the query results. The Apriori uses small data sets for the information retrieval for the query results which user searches.

VIII. CONCLUSIONS AND FUTURE WORK

The search results from the search engine which gives the Efficiency and accuracy for the query user searches. The search results provide the relevant data for the user. Performance of retrieving trustworthy data is also improved.. The future work will be providing better optimization by using other techniques in data mining which will provide high quality to the data in database.

REFERENCES

[1]. Abhijith Kashyap, Vagelis Hristidis, Michalis Petropoulos, and Sotiria Tavoulari, "*Effective Navigation of Query Results Based on Concept Hierarchies*", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 23, NO. 4, APRIL 2011.

[2]. Fie Wang, Noah Lee, Jianying Hu, Jimeng Sun, Shahram Ebadollahi, Andrew F.Laine ,"*A Framework For Mining Signatures From Event Sequences And its Applications In Health Care System*". IEEE Transactions on Knowledge and Data Engineering, 2013.

[3]. J.S. Agarwal, S. Chaudhuri, G. Das, and A.Gionis, "Automaed Ranking of Database Query Results", Proc. First Biennial Conf. Innovative Data Systems Research, 2003.

[4]. K.Chakrabarti, S.Chaudhuri and S.W.Hwang, "Automatic Categorization of Query Results", Proc.ACM SIGMOD, pp.755-766, 2004.

[5]. Karthikeyan, Saravanan, Vanitha, *High Dimensional Data Clustering Using Fast Cluster Based Feature Selection*, Int. Journal of Engineering Research and Applications, ISSN: 2248-9622, Vo.4, Issue 3, March 2014, pg.65-71.

[6]. T.Zhang, R. Ramakrishnan and M Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", Proc. ACM SIGMOD, pp.103-114, 1996.

[7]. V.Hristidis and Y.Papakonstantinou, "DISCOVER: Keyword Search in Relational Databases", Proc.Int`l Cong.Very Large Data Bases (VLDB), 2002.

[8]. Zheng Lu, Hongyuan Zha, Xiaokang Yang, Weiyao Lin, Member, and Zhaohui Zheng, *A New Algorithm for Inferring User Search Goals with Feedback Sessions*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,2013.

[9]. Zhixian Zhang, Kenny Q. Zhu , Haixun Wang, Hongsong Li , *Automatic Extraction of Top-k Lists from the Web*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING YEAR 2013.