BROAD: Diversified Keyword Search in Databases

Feng Zhao [#], Xiaolong Zhang [*], Anthony K. H. Tung [#], Gang Chen [*]
[#]School of Computing, National University of Singapore, Singapore
{zhaofeng,atung}@comp.nus.edu.sg
[*]College of Computer Science, Zhejiang University, China
{xiaolongzhang,cg}@cs.zju.edu.cn

ABSTRACT

Keyword search in databases has received a lot of attention in the database community as it is an effective approach for querying a database without knowing its underlying schema. However, keyword search queries often return too many results. One standard solution is to rank results such that the "best" results appear first. Still, this approach can suffer from redundancy problem where many high ranking results are in fact coming from the same part of the database and results in other parts of the database are missed completely.

In this demo, we propose the BROAD system which allows users to perform diverse, hierarchical browsing on keyword search results. Our system partitions the answer trees in the keyword search results by selecting k diverse representatives from the trees, separating the answer trees into k groups based on their similarity to the representatives and then recursively applying the partitioning for each group. By constructing summarized result for the answer trees in each of the k groups, we provide a way for users to quickly locate the results that they desire.

1. INTRODUCTION

With increasing amount of textual data being stored in relational databases, keyword search is well recognized as a convenient and effective approach to retrieve results without knowing the underlying schema or learning a query language [4]. The result of keyword query is often modeled as a compact substructure, such as a tree or a graph, which connects keyword tuples to include all the keywords. Potentially, a user could discover underlying relationships and the semantics based on structural answers.

However, keyword search queries can often return too many answers, because the tuples that keywords are located in might come from different tables and connect with each other in many ways. Exploring and understanding keyword search results can be time consuming and not user-friendly. To illustrate, we describe an example on CiteSeerX[1] dataset.

[1]http://citeseerx.ist.psu.edu/

Proceedings of the VLDB Endowment, Vol. 4, No. 12

EXAMPLE 1. *Consider a keyword query on "skyline" and "rank" over the CiteSeerX dataset. There are* 78 *tuples containing the keyword "skyline", and* 729 *tuples containing the keyword "rank". A snapshot of keyword tuples are presented in Table 1, and part of the answers related to these tuples are shown in Figure 1. For clear illustration, we use "a" to denote an author and "p" to denote a paper. It can be seen that the relationship between them varies a lot even for fixed keyword tuples. Presenting and exploring the results of this keyword query will be difficult.*

Table 1: The Snapshot of Keyword Tuples.

ID	Content Excerpt
kn_1	The [Skyline] Operator
kn_2	[Skyline] with Presorting
kn_3	An Optimal and Progressive Algorithm for [Skyline] Queries
kn_4	Merging [Ranks] from Heterogeneous Internet Sources
kn_5	Why [Rank]-Based Allocation of Reproductive Trials is Best
kn_6	The PageRank Citation [Rank]ing

A typical solution to large number of keyword search results is to return top-k answers according to relevant score [4]. Without knowing the schema, however, it is hard for a user to explicitly express the preference. For instance, the query {skyline, rank} aims to discover the relationship between them, but it is difficult to indicate which keyword is more important or what types of path connections are meaningful before a user realizes what can be found in the dataset. Even if it is possible to estimate users' preference, the top-k results usually include many overlapped answers that are redundant to present. This can be seen in Example 1 with an extreme case that T_2 and T_4 share two keyword nodes and even an identical answer structure.

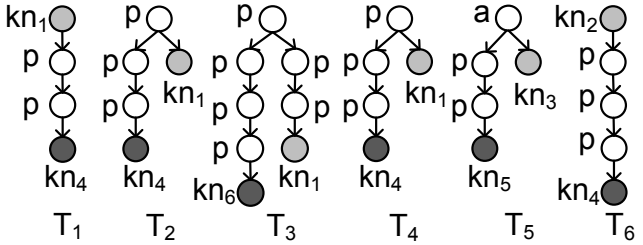

Figure 1: Search Result Examples.

To tackle this problem, result diversification has been well studied in information retrieval community [3]. They try to put documents with broad information and different semantics in the first page of search interface. We aim to adapt this idea in database area and propose the **keyword search diversification** problem: discover a set of k answer trees $\mathcal{S}$ from the whole answer set $\mathcal{T}$ which maximize

$\min\{dist(T_a, T_b)\}, T_a, T_b \in \mathcal{S}$. To make this possible, three new challenges must be overcome. The first is to define a meaningful distance measure $dist(\cdot,\cdot)$ between answer tree results tailored for keyword search in databases. Second, due to the NP hardness of result diversification [3], it is thus necessary to develop an efficient scheme to produce diversified results. Third, to support user friendly result browsing, we need to effectively summarize distinct features from rich structures and contents in diversified results.

In this demo, we develop a novel system for browsing and diversified keyword searching in databases, i.e. BROAD (BROAD is an acronym for BROwsing And Diversified keyword searching). Our contributions towards diversified keyword search in databases are as follows:

- We have devised an effective kernel distance to measure the diversity of keyword search result. This metric integrates both the textual difference and the structural distinction in the answer trees.
- We have developed an efficient algorithm to find diverse answers based on cover tree index. Our solution can separate answer trees into groups and allow us to browse results hirarchically.
- By coupling our solution with summarization techniques, we enable users to efficiently locate desired results by drilling down the hierarchy incrementally.

2. SYSTEM ARCHITECTURE

The BROAD system architecture is presented in Figure 2. We try to use a pipelined framework to overcome the challenges we discussed earlier. When a user inputs one l-keyword query in the browsing interface, it will be sent to keyword search engine generating candidate answer tree set $\mathcal{T}$. Here we rely on the standard keyword search engine in graph databases, which discovers answer trees from the data graph of a relational database [4]. BROAD system builds on top of the keyword search engine. Based on the Cover Tree Indexer, we find diverse results $\mathcal{S}$ and let users browse and refine them hierarchically in the Result Browsing Interface. The detailed descriptions will be shown in the following sections.

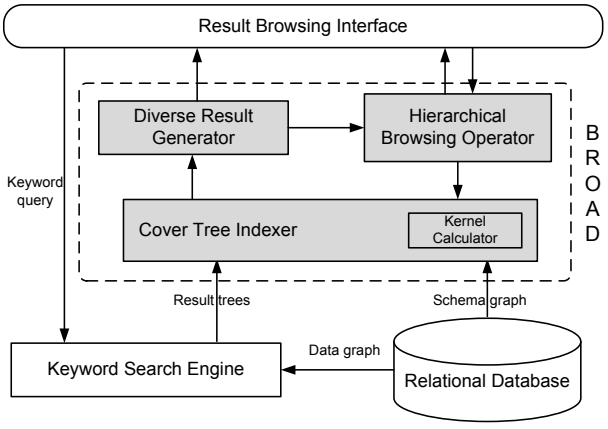

Figure 2: BROAD System Architecture.

3. METHODOLOGY

In this section, we propose a novel diversity measure with a cover tree based algorithm to effectively solve the keyword search diversification problem. Moreover, we provide summarization techniques to visualize the diverse results.

3.1 Kernel Based Diversity Measure

The core of diversity problem is the need to measure the pairwise dissimilarity between answer trees, i.e. $dist(\cdot,\cdot)$. We propose the **Answer Tree Kernel** to capture both the structural and the textual information, which is adapted from the subtree kernel [5]. The basic idea is to express a kernel on a discrete object by a sum of kernels of its constituent parts, i.e. proper subtrees of the input tree T. A proper subtree f_i comprises node n_i along with all of its descendants. Two proper subtrees are identical if and only if they have the same tree structure and the corresponding nodes are from the same table. Figure 3 depicts all of T_1's and T_5's proper subtrees respectively. Both answer trees contain four different proper subtrees and they share three of them, namely, f_1, f_2, f_3.

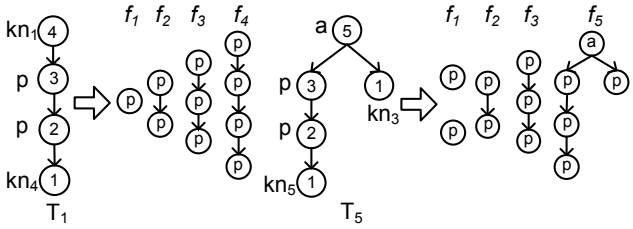

Figure 3: Kernel Example.

Originally, the subtree kernel is designed to compare only tree structures without taking node contents into consideration. In our case, the comparison between answer trees needs to consider node contents as well. The idea is to take a fuzzy match between proper subtrees. Since answer trees contain textual information, we could compare the content similarity of two proper subtrees from two answer trees that have the same structure. Let f_i^a and f_i^b be proper subtrees in T_a and T_b respectively that share the proper subtree f_i. We merge the textual content in the nodes of f_i^a and f_i^b into d_i^a and d_i^b and refer to them as documents. Next, we represent each document as $v = (w_1, w_2, \ldots, w_t)$ with each dimension corresponding to a separate term. Applying TF-IDF weighting, we obtain the document kernel $\kappa_D(d_i^a, d_i^b) = \langle v_i^a, v_i^b \rangle$ where v_i^a and v_i^b are the weighted term vectors of d_i^a and d_i^b respectively. Furthermore, the keyword query q provides another source of semantic information. Intuitively, f_i^a and f_i^b contribute more to the overall kernel if they share more keywords. Thus, we introduce a weight setting $w_{ab} = \sqrt{s/l}$ where s indicates the number of shared keywords and l represents the number of input keywords, yielding:

DEFINITION 3.1. *(Answer Tree Kernel)*
Given two trees T_a and T_b, the Answer Tree Kernel is:

$$\kappa_A(T_a, T_b) = \sum_{n_a \in N(T_a)} \sum_{n_b \in N(T_b)} w_{ab} \Delta(n_a, n_b)$$

where $\Delta(n_a, n_b) = \sum_{i=1}^{|\mathcal{F}|} I_i(n_a) I_i(n_b) \kappa_D(d_i^a, d_i^b)$, and where $I_i(n)$ is an indicator function which determines whether the proper subtree f_i is rooted at node n.

The answer tree kernel serves as an effective method to map original answer trees to a kernel space. However, larger

trees have higher chances to share many common features with any small tree in the answer tree kernel. To overcome this drawback, we compute the normalized kernel, i.e.

$$\kappa(T_a, T_b) = \kappa_A(T_a, T_b)/\sqrt{\kappa_A(T_a, T_a) \cdot \kappa_A(T_b, T_b)} \quad (1)$$

Based on the normalized kernel, we define the answer tree distance as $dist(T_a, T_b) = \sqrt{1 - \kappa(T_a, T_b)}$. This diversity measure is a metric and can be efficiently computed in linear time. Differently, one alternative solution [2] decomposes answer trees into a set of nodes and utilizes Jaccards distance to measure answer tree difference. This method is efficient but sacrifices the result quality. First, it only considers the exact match of nodes, but ignores the textual similarity between them. Second, it fails to measure the structural connections due to decomposition. Readers are referred to our technical report [6] for a comprehensive introduction on kernel based measure.

3.2 Cover Tree Based Diversification

We next describe a cover tree approach to solve the keyword search diversification. Cover tree [1] is a metric tree to index data and perform nearest neighbor search in metric spaces. It is a leveled tree where each level is a "cover" for the level beneath it. Assume that we use the cover tree CT to index answer set $\mathcal{T}$ based on the answer tree distance. For instance, a cover tree in Figure 4 indexes fifteen results of Example 1. Cover tree index has one important property, i.e., for all trees $T_a, T_b \in C_i$, the distance from T_a to T_b is greater than $1/2^i$. C_i indicates answer trees in $\mathcal{T}$ associated with the nodes at level i.

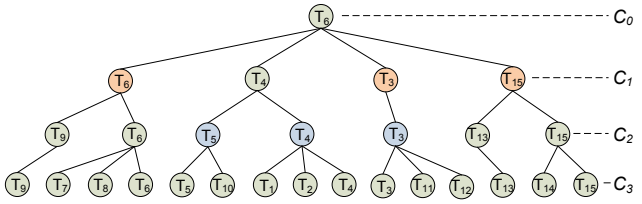

Figure 4: Cover Tree Example.

The separation property of cover tree suggests that nodes at higher level are more diverse. Therefore, instead of discovering k diverse results from the whole answer set, we choose them from the highest possible level in cover tree to reduce the number of distance computations. The algorithm accesses cover tree from top to bottom, and stops at the first level including at least k nodes, denoted as the working level C_i. If $|C_i| = k$, all the answer trees in this level are returned as diverse results. Otherwise, C_{i-1} is selected as the partial results due to the separation property. Next, we heuristically expand the farthest node in the working level until $|\mathcal{S}| = k$. Let $k = 3$ for example in Figure 4. Level 0 of cover tree only contains the root node, so the algorithm continues to the next level with four nodes. The number of nodes is larger than k, so this level becomes the working level and T_6 from the above level is selected as the partial result. Next, T_3 and T_{15} are included by means of farthest expanding. Finally, we discover $\{T_6, T_3, T_{15}\}$ as diverse results. The detail algorithm and experiments can be found in our technical report [6].

Due to the tree-like structure, cover tree index can elegantly support hierarchical browsing. We proceed as follows. First, we separate answer trees in the working level into k answer tree groups based on their kernel similarities to the k diverse results. A user then selects a subset $\mathcal{H}$ of interest. Second, we fetch all nodes in next level covered by $\mathcal{H}$, and treat them as nodes in a new working level. Thus, we can perform cover tree based diversification again to obtain a new set of diverse results. This procedure iteratively proceeds until reaches the intended answer tree/s. For instance, T_6, T_3 and T_{15} in Figure 4 are diverse results found previously. We first assign T_4 to T_3 due to the kernel similarity and these four answer trees form three groups. Assume that users are interested in the group $\{T_3, T_4\}$, so we drill down to next level with answer tree set $\{T_3, T_4, T_5\}$. They are directly selected as new diverse answers because the size of this level equals to three.

3.3 Result Summarization

To support hierarchical browsing, we develop a circular view to summarize both structures and contents of a group of answer tree. As such, users can quickly browse and select preferred answer trees from the whole answer set. The basic idea is derived from the Circos project[2] and we adapt it for the answer trees' summarization. In the following, we take answer tree T_5 to show the approach of mapping one answer tree into a circle. Figure 5a depicts T_5 and it is transformed to the red part in Figure 5b. The root node and keyword nodes are mapped to segments, and pathes between nodes are mapped to ribbons. The detailed connections between the root node and keyword nodes are illustrated in focused view, which will be discussed later in Section 4. For a group of answer trees, the shared nodes among answer trees are presented just once to save space. For instance, Figure 5b only contains two "skyline" nodes and three "rank" nodes for eight answer trees. As a result, the circular view for a group of answer trees salvages large spaces compared to the original layout. Furthermore, we utilize different colors to distinguish answer trees so that users can quickly capture how many answer trees are covered in a group. In general, this view is suitable for keyword search, because k is usually much less than one hundred in real keyword search use cases.

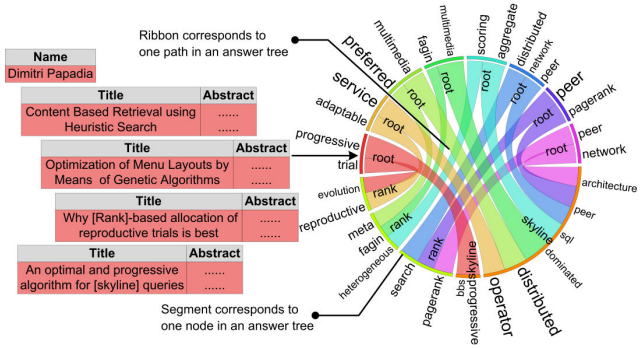

Figure 5: Answer Tree Mapping.

Aside from structural summarization, representative words are attached to the related segments in order to distinguish the circles that are on the same level of the hierarchy. Given any segment s, let the node it represents be n. The ribbons that connect s to other segments in the circle actually represent paths in the answer trees that connect n to other nodes in the answer trees. For each segment, candidate words are selected from these pathes. For each candidate word, we compute a TF-IDF like score, and select top candidates as representative words. The detailed algorithm can

[2]http://mkweb.bcgsc.ca/circos/

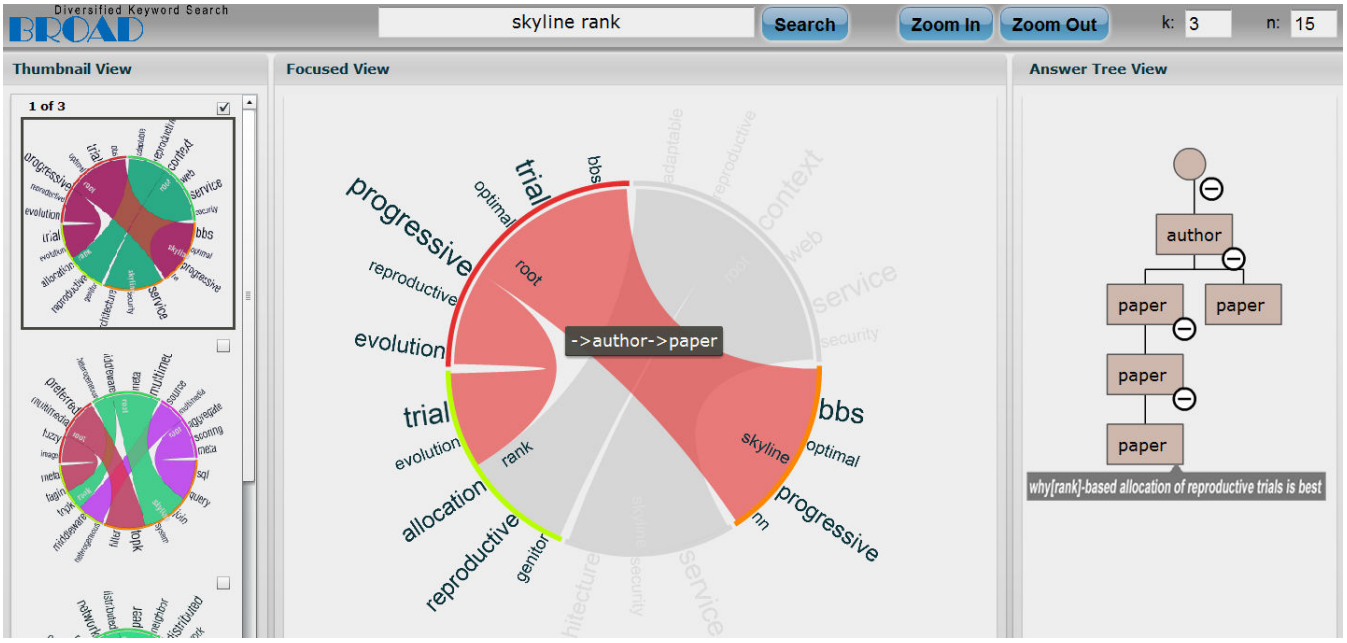

Figure 6: BROAD Interface.

be found in our technical report [6]. As such, we sketch out distinct contents of answer trees. The number of representative words selected depends on the width of a segment. Take the highlighted root segment in the focused view of Figure 6 as an example. We select six representative words attached to the segment because their scores are the highest. To further emphasize the word distinctions within a segment, we present the selected words in different font sizes, according to their term frequencies in one segment. The words "progressive" and "trial" are highlighted with the biggest font size since their term frequencies are the largest in the segment.

4. DEMONSTRATION

In demonstration, we develop a web based browsing interface[3] to support interactive diversified keyword search. As shown in Figure 6, the interface consists of a search input area and a result display area. Search input area on the top of the interface contains a keyword input field, zoom in/out buttons and setting fields for user-specified parameters (k and n). Therefore, we enable users to search by keyword query as well as perform hierarchical browsing using zoom in/out buttons.

Result display area on the bottom is composed of three views from left to right: thumbnail view, focused view and answer tree view. The thumbnail view displays k circles to summarize k answer tree groups as a list of thumbnail images. Consequently, users can click the desired circle and enlarge it in the focused view, which allows users to focus on certain segments or ribbons. The chosen element is highlighted with color, while other elements become transparent. We also utilize a tooltip above a ribbon to describe the structure of related path. Furthermore, the corresponding answer tree is represented as a tree layout in the answer tree view. Node labels in this view correspond to the answer tree structure and tooltips correspond to the answer tree content.

[3]http://db128gb-b.ddns.comp.nus.edu.sg:8080/broad/

Take the keyword query {skyline,rank} as an example in Figure 6. A user sets $k = 3$ and $n = 15$ to discover three diverse answers from fifteen candidates. The thumbnail view shows a preview of three circles, so the user can browse the overview through a scroll bar and select the desired one to display in the focused view. For example, the top circle in the thumbnail view is selected with the detailed information in the center. Besides, T_5 is highlighted with red color because the user clicks on the root segment of T_5. Moreover, when mouse hovers over on the ribbon between the root node and the "skyline" keyword node, a tooltip "author→paper" shows the structure information. Correspondingly, the tree layout in the answer tree view visualizes the structure and the content of T_5. If the user ticks the top checkbox in the thumbnail view and then presses the zoom in button, the system can drill down to next level and present a set of new circles.

In summary, our BROAD system provides a user friendly interface that helps users search and explore diversified keyword search results. To the best of our knowledge, our work is the first attempt to support interactive hierarchical browsing on keyword search in databases.

5. REFERENCES

[1] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *ICML*, pages 97–104, 2006.

[2] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl. *DivQ*: diversification for keyword search over structured databases. In *SIGIR*, pages 331–338, 2010.

[3] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.

[4] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *VLDB*, pages 505–516, 2005.

[5] S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. *Kernel methods in computational biology*, pages 113–130, 2004.

[6] F. Zhao, X. Zhang, Anthony K.H. Tung, and G. Chen. BROAD:Diversified Keyword Search in Databases. Technical report, TRD3/11, School of Computing, NUS.