ExRank: An Exploratory Ranking Interface

Ramon Bespinyowong
School of Computing
National University of Singapore
Singapore
ramonb@u.nus.edu

Wei Chen
State Key Lab of CAD&CG
Zhejiang University
China
chenwei@cad.zju.edu.cn

H. V. Jagadish
Department of Computer Science and Engineering
University of Michigan
Ann Arbor, MI, USA
jag@umich.edu

Yuxin Ma
State Key Lab of CAD&CG
Zhejiang University
China
mayuxin@zju.edu.cn

ABSTRACT

Even with simple everyday tasks like online shopping or choosing a restaurant, users are easily overwhelmed with the large number of choices available today, each with a large number of inter-related attributes. We present ExRank, an interactive interface for exploring data that helps users understand the relationship between attribute values and find interesting items in the dataset. Based on a kNN graph and a PageRank algorithm, ExRank suggests which attributes the user should look at, and how expressed choices in particular attributes affect the distribution of values in other attributes for candidate objects. It solves the problem of empty result by showing similar items and when there are too many results, it ranks the data for the user. This demo consists of 1) the description of the software architecture and the user interface 2) the logic and reason behind our solution and 3) a list of demonstration scenarios for showing to the audience.

1. INTRODUCTION

Users are often overwhelmed with many choices offered by databases today. This is particularly when there are many attributes of interest, when users do not have technical knowledge, and when they are unfamiliar with the data. Faceted navigation is a popular tool to help users explore the data (See [9] for a survey). It has become increasingly popular and is widely used in several applications such as hotel booking [1], finding restaurants [2], and on-line shopping [3]. However, in the exploration stage a user might get an empty result set and does not know how to adjust the query. In addition, when the result set is too large to handle, the user may not know which items the user should first look at and which additional attribute values the user should specify in the search interface.

Following is a summary of some of the problems that arise from the existing methods and how we solve them:

- **A user has no knowledge about the dataset.** We add the interactive visualization using a kNN graph with a force-directed layout to help a user gain insights into the data. In addition, we rank and suggest the attribute values that a user might be interested in. The user can start exploring the dataset from those suggestions.
- **The result set is empty or too small.** Instead of showing an empty result, we show similar items to the user's query.
- **The result set is too large.** We rank the items so that it is possible for a user to see only the first few items.

To give an overview of the data, several visualization methods have been adopted to improve user experience in exploring the data. Some of them support multivariate data and allow the interactive exploration in the dataset [7, 8]. Dust& Magnet [8] metaphors the data where attribute values and items are magnet and dust respectively. The magnet strength depends on how large the attribute value of an item is. When a user is interested in a particular attribute, the user can select items closer to the magnet of the attribute. However, these systems do not show how attribute values are related.

Example 1: In the multivariate dataset of restaurants, a user would like to find a Thai restaurant near the house. However, there is no matched result. In the attribute "cuisine", "Thai" is highly similar to "Vietnamese" because other attribute values of Thai restaurants are similar to those of Vietnamese restaurants. Therefore, based on the **intra-attribute similarity**, the system should suggest the Vietnamese restaurants in the neighbourhood.

Example 2: A user needs to decide whether to go the west or the east region of the city for a sightseeing trip. Therefore, the user wants to know what are the attribute values with high **inter-attribute similarity** to the west

[1] http://www.agoda.com
[2] http://www.openrice.com
[3] http://www.amazon.com

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/4.0/. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org.
Proceedings of the VLDB Endowment, Vol. 9, No. 13
Copyright 2016 VLDB Endowment 2150-8097/16/09.

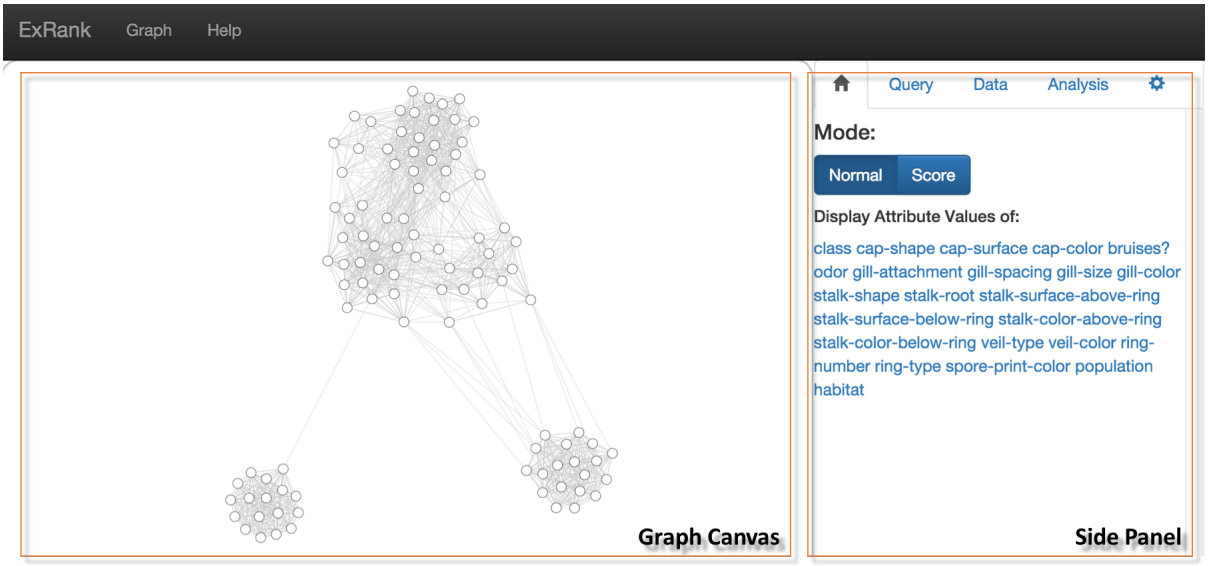

Figure 1: The overview of the user interface.

and to the east. The user can see what each region is good at and finally decide where to go.

We propose **ExRank**, an exploratory ranking interface based on a kNN graph and the PageRank algorithm to help a user explore the database and locate interesting objects through ranking. To address examples such as those above, we use a kNN graph as items which interest the user should have high similarities. Hence, these items are fairly close in the kNN graph. To the best of our knowledge, it is the first data exploration and visualization tool based on the kNN graph.

When there is a query from the user, we display the items sorted by our scoring function. Ranking provides users options either to browse the whole list or to see the top few items. It is especially useful when there is no exact match.

In addition, we introduce concepts called **inter-attribute similarity**, which is the similarity between attribute values from different attributes, and **intra-attribute similarity** which is defined as the similarity between attribute values from the same attribute.

Section 2 describes the architecture of the system and the interactions with the user. Section 3.1 shows how and why we use the PageRank algorithm to rank items in the dataset. Section 3.2 explains how attribute values are ranked and how we imply inter-attribute similarities and intra-attribute similarities. Then, we describe all the demonstration plans in Section 4.

2. SYSTEM OVERVIEW

A user interacts with the system through a web interface which sends and receives REST requests/responses to/from the back-end server. The back-end connects to the database that has the precomputed kNN graph and information of all the points. Next, we will describe the user interface and its functionalities.

Figure 1 shows the user interface. The system provides several functionalities that assist users in understanding data and finding interesting items. In our system three main views are presented: 1) a kNN graph view, 2) an attribute value query panel for specifying interesting attribute values for further exploration, and 3) a data table. During

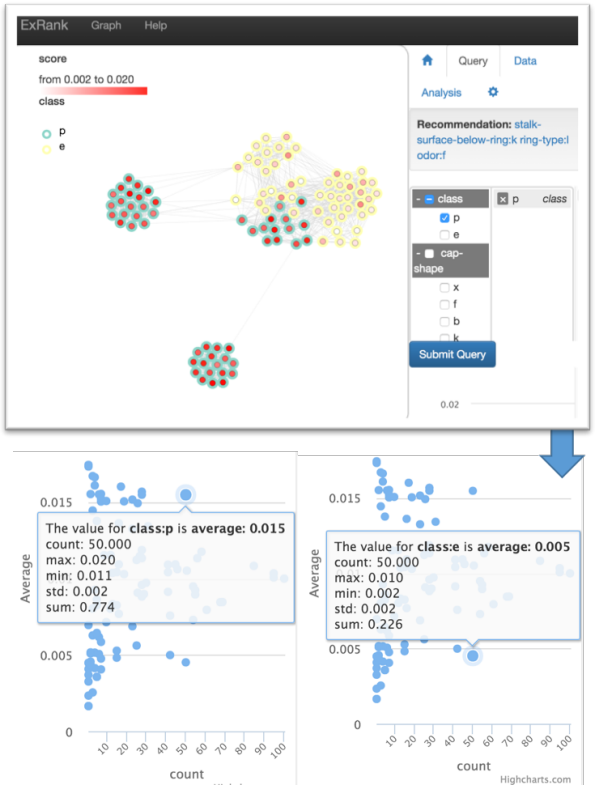

Figure 2: The interface update after making a *class:p* query.

the explanation of the user interface, we use the mushroom dataset [5] and *attribute:attributeValue* indicates *attributeValue* in *attribute*.

There are two main GUI components. The **Graph Canvas** displays the kNN graph. A user can get the full information of an item by a mouse hover. The **Side Panel** is for creating a query to the system and for setting the system. It consists of five pages: home, query, data, analysis, and setting.

We show a kNN graph in the Graph Canvas. The arrow heads for marking the direction of edges are omitted for

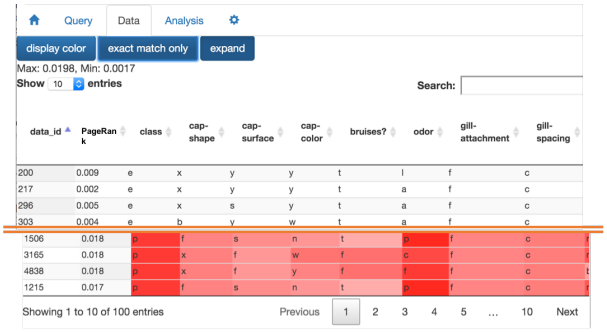

Figure 3: The data page before(top) and after(bottom) clicking display color.

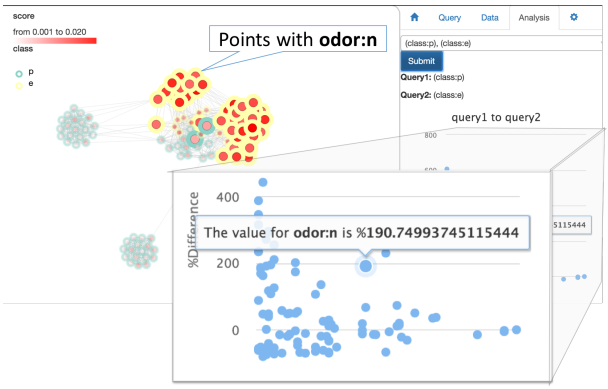

Figure 4: the Analysis Page when a user specifies two queries, (*class:p*) and (*class:e*).

reducing visual clutter. The details of each node can be shown by a mouse hover. For layouting the graph nodes, we adopt a force-directed layout [2] so that users can observe the clustering patterns of the graph if they exist.

In the **home** page, a user can switch the color mapping of the graph between **normal** and **score** mode. The score mode colors the nodes based on the PageRank score computed in Section 3.1. The redder a node is, the more PageRank score it has. Borders of the nodes represent attribute values and can be triggered by clicking one of the attribute names in this page. Figure 2 shows the Graph Canvas with the score mode enabled and node borders are used to show attribute *class*.

When a user wants to learn about inter-/intra- attribute similarities, a user can specify a list of attribute values in the **query** page (see Figure 2). The system will update the PageRank scores of all items and compute the average PageRank of each attribute value. The scatter plot between the average PageRank and the count is shown. In the figure, the plot shows that *class:p* and *class:e* have low intra-attribute similarity as *class:p* has high average score but *class:e* has low average score. If the user does not know what the first query should be, we recommend the possible attribute values at the top (see Section 3.2 for the computation).

Figure 3 shows the **data** page where a user can browse and sort items based on the PageRank score, id, or attribute values. When the user has a hard constraint on the query, the user can click on the **exact match** button to show only the items that match the query. Additionally, the user can see the average PageRank of each attribute value encoded in color by clicking **display color**.

The **analysis** page is for a user to compare between queries. After the user selects two queries, the system computes the percentage change of the average PageRank of each attribute value. The result is shown as a scatter plot between the percentage change and the count. For example, a user specifies two queries; *class:p* and *class:e*. Attribute value *odor:n* which is more related to *class:e* than *class:p* shows *190%* increase in average PageRank as shown in Figure 4.

The **setting** page allows a user to change k, $dampingfactor$ in the PageRank algorithm, and adjust the display.

3. RANKING COMPUTATION

3.1 Item ranking

We model each object as a d-dimensional point $p \in P$. $p_i[j]$ denotes a value of the j-th dimension of p_i. There are totally n objects. Let $A_j = \{a_{j1}, a_{j2}, ..., a_{j|A_j|}\}$ be a set of possible attribute values of the j-th attribute. For simplicity, we use a_j for an arbitrary attribute value. We assume that each attribute is categorical (and if an attribute is numerical, we discretize it first.) A user preference can be described as a set of attribute values, $A^u \subset \bigcup_{j=1}^{d} A_j$.

We rank the items so that when the exact result set is empty, the user can see the alternatives and when there are too many results, the user only has to consider the first few items. The ranking information is also used for finding attribute value similarities.

Problem Statement: Given a user preference A^u, rank all the items in P.

Ranking based on Topic-Specific PageRank. PageRank [6] is a well-known algorithm for finding important nodes in a graph. Nodes are important if there are incoming links from other important nodes. In a kNN graph, having a high PageRank implies that a node has many incoming links or the node is highly similar to many other nodes in the graph. The original PageRank scores can be computed as followed:

$$Rank = \alpha M \times Rank + (1 - \alpha) \times v \quad (1)$$

where M is a transition matrix of the graph, $0 \leq \alpha \leq 1$ be a damping factor and $v = [1/n]_{n \times 1}$. However, PageRank is not query-specific and cannot gather to different users' preferences. There are several techniques to improve the performance of PageRank such as assigning more weights to popular pages [4]. Here, we want to assign more weights to nodes matching the query. Topic-Sensitive PageRank [3] introduces bias towards a group of nodes, T, by using v as follows:

$$v[i] = 1/|T| \text{ if } p_i \in T, 0 \text{ otherwise} \quad (2)$$

We can classify nodes as matching or non-matching nodes. However, sometimes it is impossible to find nodes matching all values specified in the query and we should consider nodes partially matching the query. The random walk should visit items with more matches to the query more often. Therefore, v becomes:

$$v[i] = |\{j | p_i[j] \in A^u\}| \quad (3)$$

and we divide it by $\sum_{i=1}^{n} v[i]$ to normalize v.

3.2 Attribute Value Similarities

We are interested in finding inter-attribute value and intra-attribute value similarity as mentioned earlier. ExRank visualizes attribute value similarities by an assumption that two attribute values a_1 and a_2 are similar if when items containing a_1 rank high, those items are likely to contain a_2.

We allow users to select multiple attribute values in their study.

Problem Statement Given a user preference A^u, how to find attribute values with high inter-/intra-attribute value similarity to the preference?

Using Jaccard's similarity. We can compute the similarity as follows:

$$sim(A_u, a_j) = \frac{|contains(A^u) \cap contains(a_j)|}{|contains(A^u) \cup contains(a_j)|} \quad (4)$$

where $contains(\cdot)$ returns a set of items containing a set of specified attribute values. Sometimes there are no items matching the user's query; hence, the similarity of the user's query and all other attribute values become zero.

Using the Average Ranking. We sort each attribute value based on the average PageRank of all the items containing it. High average PageRank attribute values are those with a lot of incoming links from (or very similar to) items matching the user's query. Therefore, they are highly similar to the query.

However, not all high average PageRank values are interesting. Consider when there are 100 item, the attribute value a_{j1} has high average since there is only one item having it and that item has a high PageRank score. While items having a_{j2} are 50 high PageRank items and 3 low PageRank items, giving a_{j2} a slightly lower average but making it more interesting. On the other hand, an attribute value with low average but high count is not interesting either as it implies that this attribute value is different from the query. Here, from the user's perspective, we present another problem.

Problem statement: Among all the attribute values similar to A^u, which are the attribute values which are more likely to attract the user?

Skyline of count and average. There is trade-off between the number of items containing attribute values and their average PageRank. We display the scatter plot between the average value and the count. When a user wants to be more specific, the user can select attribute value with low count but high average.

Skyline [1] is widely used for data with trade-off. We recommend attribute values in the skyline. Points are ranked based on Equation 5. We use $0 \leq \beta \leq 1$ to assign more weight to the average PageRank. The attribute values in the recommendation are in the skyline (see Theorem 1.)

$$score(a_j) = \beta \frac{average(a_j)}{max_l(average(a_l))} + (1-\beta)\frac{count(a_j)}{n} \quad (5)$$

4. DEMONSTRATION

A demonstration video can be found at `https://youtu.be/oS3CGk4mti0`. In the demonstration, we have two available datasets. The first dataset is **Mushroom** dataset from the UCI machine learning repository. There are 8124 instances with 22 categorical attribute values [5]. The second dataset is **Singapore Restaurants**, crawled from `sg.openrice.com`. We select 5,000 restaurants with 12 attributes.

Scenario 1: Without any knowledge, a user does not know what first attribute value the user should look into.

(1.1) Add attribute values in the recommendation bar to the query.

(1.2) See the top few items of the data table. If the user likes them, continue following the recommendation. If no, remove the added values and select other values instead.

Scenario 2: There is no item exactly matching the query.

(2.1) Toggle **Exact Match** to display all the items. Sort the items based on the PageRank score.

(2.2) Find the interesting items from the top of the list.

Scenario 3: The user wants to study about the trade-off of selecting an attribute value a_1 over an attribute value a_2.

(3.1) Make a query consisting of a_1 and another query consisting of a_2.

(3.2) Go to the Analysis Page, select these two queries and click submit. The scatter plot between % changes and count will be shown.

Scenario 4: The user is more strictly interested in items that match the query than the similar items.

(4.1) Go to the setting tab and lower the damping factor, making a random walk goes to matching items with higher probability.

5. ACKNOWLEDGEMENTS

This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centre in Singapore Funding Initiative.

6. REFERENCES

[1] S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430. IEEE, 2001.

[2] T. Dwyer. Scalable, versatile and simple constrained graph layout. In *Computer Graphics Forum*, volume 28, pages 991–998. Wiley Online Library, 2009.

[3] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM, 2002.

[4] K. Kumar and F. D. M. Abhaya. Pagerank algorithm and its variations: A survey report. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 14(1):38–45, 2013.

[5] M. Lichman. UCI machine learning repository, 2013.

[6] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[7] J. M. Rzeszotarski and A. Kittur. Touchviz:(multi) touching multivariate data. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1779–1784. ACM, 2013.

[8] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust & magnet: multivariate information visualization using a magnet metaphor. *Information Visualization*, 4(4):239–256, 2005.

[9] B. ZHENG, W. ZHANG, and X. F. B. FENG. A survey of faceted search. *Journal of Web engineering*, 12(1&2):041–064, 2013.

APPENDIX

THEOREM 1. *All the attribute values displayed in the recommendation are skyline points.*

PROOF. *For the sake of contradiction, assume that there exists an attribute value a_{j1} in the recommendation box not in the skyline. Then, there is at least one attribute value a_{j2} dominating a_{j1}. Hence, both count and the average PageRank of a_{j2} are greater than or equal to those of a_{j1}. Either count or the average PageRank of a_{j2} is greater than a_{j2}. Thus, $score(a_{j2}) > score(a_{j1})$, which is a contradiction.* □