Unifying Qualitative and Quantitative Database Preferences to Enhance Query Personalization

Roxana Gheorghiu
University of Pittsburgh
roxana@cs.pitt.edu

Alexandros Labrinidis
University of Pittsburgh
labrinid@cs.pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
panos@cs.pitt.edu

ABSTRACT

Query personalization can be an effective technique in dealing with the data scalability challenge, primarily from the human point of view, i.e., making big data easier to use. In order to customize their query results, users need to express their preferences in a simple and user-friendly manner. In this paper, we present a graph-based theoretical framework and a prototype system that unify qualitative and quantitative preferences, while eliminating their disadvantages. Our integrated system allows for (1) the specification of database preferences and the creation of user preference profiles in a user-friendly manner, (2) the manipulation of preferences of individuals or groups of users and (3) total ordering of the tuples in the database, matching both qualitative and quantitative preferences, hence significantly increasing the number of tuples covered by the user preferences. We confirmed the latter experimentally by comparing our preference selection algorithm with Fagin's TA algorithm.

1. INTRODUCTION & MOTIVATION

The supply and demand of data is becoming commonplace in all aspects of our society; from everyday life (e.g., picking movies or restaurants), to business products, to medicine, and science in general. The term *"Big Data"* has been used to describe the challenges and opportunities from such a ubiquity of data, while also considering its volume, velocity, and variety characteristics [5].

We distinguish two types of scalability: (1) from a *systems point of view* – this refers to traditional challenges due to the volume of data (and the rate of increase) and limitations in network bandwidth, processing, and storage capacity; and (2) from a *human point of view* – given the volumes of data, new paradigms to aid in search are needed so that users do not get lost in a sea of data [7].

It is well-known that *query personalization* can be an effective technique in dealing with the scalability challenge, primarily from the human point of view. It is also well-known that when individual users within a group have the ability to personalize their work (i.e., see the content relevant to them) and share their experiences, the collaboration within the group is significantly improved (e.g.,[1]).

ExploreDB'15 May 31 - June 04 2015, Melbourne, VIC, Australia
ACM 978-1-4503-3740-3/15/05 ...$15.00.
http://dx.doi.org/10.1145/2795218.2795223

In order to personalize their query results, users need to provide their *preferences* in an effective manner (essentially letting the system form *user profiles*). These preferences are then used when users submit queries in order to only return the results that are most relevant to them. Cutting down the result set in this way improves both types of scalability.

There are two main types of user preferences defined in the literature [8]: *quantitative* and *qualitative*. *Quantitative preferences* are described by scores attached to each tuple that matches a preference. Using these scores we can define a total order over the database tuples, e.g., from the most preferred to the least preferred. *Qualitative preferences* are expressed as pairs of tuples. When put together, these pairs generally create only a partial order over database tuples. Each type of preferences – quantitative and qualitative – has its advantages over the other. There are examples when a user's preference can be conveniently expressed using one approach but not the other.

An important, but largely overlooked aspect of preferences, is that they usually have some notion of importance (or intensity) associated with them. In fact, we advocate that preferences should not be seen as a binary option. Instead, we should empower every user to express his/hers preferences along with the *intensity* of that particular preference, i.e., how "strongly" he/she feels about it. This intensity could be determined by the user selecting one of predefined intensity levels (e.g., very, some, little, none) or by the user specifying a numeric score for each tuple. It can easily be defined for a quantitative preference. However, in the case of a qualitative preference, the intensity suggests the strength between two different tuples and cannot be associated, individually, to any of the two tuples involved; it should instead be linked to the pair of tuples.

In this work, our hypothesis is that: *A hybrid model, which integrates qualitative and quantitative preferences by means of preference strength or intensity, is both user-friendly and creates a global view of preferences that can be effectively used to rank query results and support query exploration.*

2. OUR APPROACH: HYPRE

In our work [4, 3], we proposed a new hybrid model which is able to overcompensate the negative aspects of one preference model by using the solutions provided by the other model. The formal underpinning of our proposed hybrid model is a *Hybrid Preference Graph*, HYPRE =(PV,PE), a labeled directed and acyclic graph. Each node in the graph represents a query predicate. We express quantitative preferences using edges that have the same starting and ending point. Qualitative preferences are represented by edges between two different nodes. Each edge is labeled with a value that represents the preference's intensity. *Preference intensity* is a decimal value between -1 and 1 and is used to express a negative preference, a positive preference, or equality/indifference. Users submit both qualitative and quantitative preferences along with an intensity value. In this way, individual users and groups create their own profile by incrementally adding or removing preferences over the database tuples. When a query is submitted, the system effectively selects the best combination of preferences from the user/group's profile to filter and rank the query results.

Although providing intensity is not an easy task for users, we envision a system where intensities can be discrete values mapped to some attributes that a user can rely to. For the quantitative preferences, the model can easily be applied to a star-type of rating, where 5 stars represents the a preference with intensity value equal to one. For the qualitative preferences, we can have predefined values for *highly-preferred*, *indifference*, *not-preferred*, etc.

Representation Our solution for representing a HYPRE graph is based on a *graph database model* which is designed to provide efficient graph traversal and graph manipulation. With this implementation, we can create only one graph for all users and groups and select all the nodes for a particular user/group, as needed, using the *user_id* property of a node. In this implementation, a node with no connections represents a quantitative preference and contains four properties: *(node_id, user_id, predicate, intensity)*, where *intensity* refers to the quantitative preference intensity. Two connected nodes create a qualitative preference with the direction of the edge to define the preference order between predicates and store the qualitative preference intensity. This implies that, if intensity values of these nodes exists, then the value of the node that has an outgoing edge (refer to as the *left node*) must always have a greater or equal intensity value with respect to the node where the edge ends (refer to as the *right node*). Moreover, each edge has associated a label used to support graph traversals. The most common label is PREFERS, used to traverse the graph based on the partial order given by the qualitative preferences. Additionally, we use labels CYCLE and DISCARD to mark conflicts and inhibit traversal.

Intensity Value Computation To incorporate the nodes participating in a qualitative preference into the total order generated by the quantitative preferences we convert the qualitative preferences into quantitative preferences. We achieve this by deriving an intensity value for these nodes based on the existing qualitative preference intensity value and a quantitative preference intensity value (or a default value if this does not exist). For this purpose, we define the following functions:

$$\begin{aligned} \text{IntensityL (left, ql, qt)} &= \min(1, \text{qt}*2^{\text{sign(qt)*ql}}) \\ \text{IntensityR (right, ql, qt)} &= \max(-1, \text{qt}*2^{\text{-sign(qt)*ql}}) \end{aligned} \tag{1}$$

where: *left/right* is the position of the node for which we compute the intensity, *ql* is the intensity of the qualitative preference, and *qt* is the intensity of the quantitative preference.

Preference Aware Query Enhancement Once created, the preference graph can be used to identify and combine the relevant preferences to customize queries. In our model, we adopted the inflationary and reserved functions from [6]: $f_\wedge$ to calculate the combined intensity for conjunctive predicate combinations and $f_\vee$ to compute the combined intensity for disjunctive predicate combinations with the form given in Eq. 2 and Eq. 3. $f_\wedge$ behaves inflationary whereas $f_\vee$ has a reserved behavior.

$$f_\wedge(p_1, p_2) = 1 - (1 - p_1)(1 - p_2) \tag{2}$$

$$f_\vee(p_1, p_2) = \frac{p_1 + p_2}{2} \tag{3}$$

where p_1, p_2 are the preference intensity values.

3. EXPERIMENTAL EVALUATION

We implemented our preference graph in a real system, using real data in order to evaluate its practicality and usefulness under realistic conditions. We designed our experiments to show first, the benefits of having both qualitative and quantitative preferences in a unified model and second, the key role of the intensity value.

Experimental Testbed We stored the preference graph using the Neo4j 2.0 engine and we used Java 1.7 to query both the graph database and the MySQL database.

We tested our system using data extracted from an extended version of the DBLP dataset [9], that contains both the DBLP dataset (2011 version) and information about citations. Moreover, we created two new tables and populated them with preferences extracted from the data, such that most of the different type of preferences are covered [4, 3]. For qualitative preferences with missing intensity value, we computed a default value for each user by selecting the average positive value of the intensity values provided by the user. In this way, the default value represents better the range of intensity values provided by a user, instead of using, for example, a global default value for all users.

Evaluation Metrics We used three metrics in our evaluation:

- *Coverage* – the total possible number of tuples "touched" when all preferences are used independently,
- *Similarity* – given two lists of tuples, the similarity metric returns the percentage of tuples that are common in the two lists over all the tuples (i.e., their *Jaccard similarity*), and
- *Overlap* – given two lists with the same tuples, the overlap metric returns the percentage of tuples that are in the same relative order in both lists (i.e., similar to the *Kendall-Tau coefficient*).

Benefits of a Unified Model of Preferences The benefit of our unified model of preferences is measured in term of *coverage*. By using intensity values to combine the two preference types, our model generates significantly more quantitative preferences (Fig. 1). For one particular user (uid=2, chosen at random), the graph shows that initially there are 36 quantitative preferences, but after inserting all qualitative ones, the preference graph will contain 172 nodes.

With more quantitative preferences we can cover more tuples. Fig. 2 shows the number of distinct tuples returned for two users if we run: (1) only quantitative preferences (QT), (2) only qualitative preferences (QL), (3) both qualitative and quantitative preferences (QL+QT), or (4) all preferences extracted from our HYPER Graph. For both users, our model can cover significantly more tuples due to our mechanism that transforms a qualitative preference into two different quantitative preferences. This improvement is from 120% compared to both quantitative and qualitative (uid=388437) up to 336% compared to just quantitative preferences (uid=2). Of course, more results in this case means better results because we are able to order them according to the user's preferences.

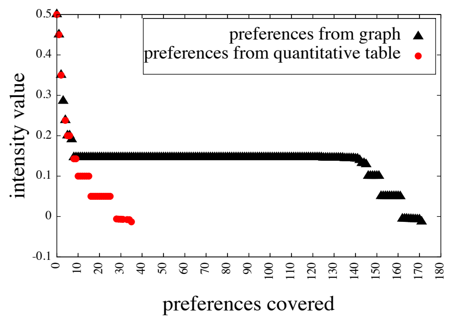

Figure 1: QT for uid=2

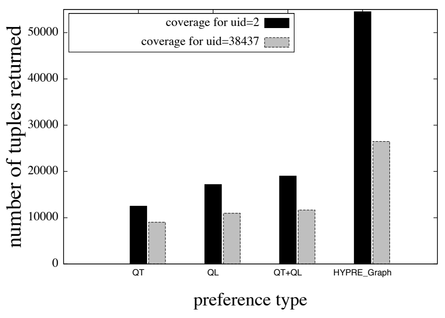

Figure 2: Coverage for uid=2 and uid=38437

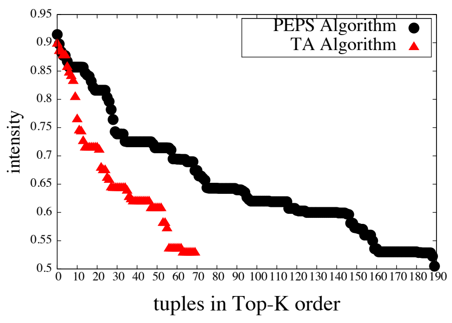

Figure 3: Coverage -PEPS vs. Fagin's TA algorithm

Top-K Queries The PEPS (*Practical and Efficient Preference Selection*) algorithm, is our Top-K algorithm implementation over HYPRE. To evaluate our PEPS algorithm's correctness, we implemented the well known TA algorithm [2] by generating, for different users, the combined intensity value of each paper. The TA algorithm assigns different scores to each tuple in the database based on the attributes used in preference's predicates. In our dataset, there are two types of predicates - on the *venue* and on the *author*. Because of that, we created two different tables *intensity_author* and *intensity_venue* with three attributes: (user_id, paper_id, combined_intensity). The combined intensity values, in both tables, were computed using Eq. 2. Finally, we combined the intensity values from the two tables to return a final score for each tuple. The final ranking given by the TA algorithm was used to evaluate the efficiency of our algorithm.

Since Top-K algorithms work only for quantitative preferences, we first created a HYPRE graph that stores only the quantitative preferences. We ran PEPS over this graph and we compared our results against those of the TA algorithm. The results show 100% similarity and 100% overlap.

To assess the advantages of PEPS when qualitative preferences are considered, we ran PEPS over the large HYPRE Graph, containing both qualitative and quantitative preferences. To make a comparison between the two algorithms more insightful, instead of specifying a value for K we looked at the ranking of tuples with combined intensity value at least as high as the maximum preference intensity value for user with uid=2 (i.e., 0.5). The results depicted in Fig. 3 show the two major advantages of our Top-K algorithm:

1. PEPS offers better coverage, i.e., finds more tuples than the TA algorithm with intensity value higher or equal to 0.5.
2. Overall, PEPS returns tuples with higher intensity value than the TA algorithm.

When looking at the similarity between the two returned Top-K lists, we found that there was 37% of matching tuples. To measure the overlap between the two lists, we first extracted the matching tuples from the two lists and then we verified that their order is preserved across the two lists. Again, there was a 100% match between the two lists as in our first experiment. Furthermore, it does not incur any performance penalty. For example, for Top-800, PEPS takes on average 2 sec to run for this workload.

Discussion The experiments above show that our solution does not only perform as good as the TA algorithm – we have a perfect match when only quantitative preferences are used – but it also performs better overall, because it has the advantage of using the qualitative preferences in addition to the quantitative ones.

4. CONCLUSIONS

In this paper we discussed our hybrid preference model that combines *quantitative* and *qualitative preferences* into a unified model using an acyclic graph, called HYPRE Graph. We implemented our framework using the Neo4j graph database system and experimentally evaluated it using real data extracted from DBLP. Our experimental results show the power of our solution in utilizing both types of preferences to retrieve more relevant results.

Acknowledgments We thank Christos Faloutsos, Wolfgang Gatterbauer, Andy Pavlo, and the reviewers for their insightful feedback. This work was funded in part by NSF awards OIA-1028162 and IIS-0746696.

5. REFERENCES

[1] C. E. Evangelou, M. Tzagarakis, N. Karousos, G. Gkotsis, and D. Nousia. Augmenting collaboration with personalization services. *IJWLTT*, 2(3), 2007.

[2] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[3] R. Gheorghiu. *Unifying Qualitative and Quantitative Database Preferences to Enhance Query Personalization*. PhD thesis, Sep 2014.

[4] R. Gheorghiu, A. Labrinidis, and P. Chrysanthis. A user-friendly framework for database preferences. In *CollaborateCom*, 2014.

[5] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[6] G. Koutrika and Y. Ioannidis. Personalization of queries in database systems. In *ICDE*, 2004.

[7] A. Labrinidis. The big data - same humans problem. In *Proc. of Conference of Innovative Data Systems Research*, 2015.

[8] K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM TODS*, 36(3), 2011.

[9] J. Tang, J. Zhang, R. Jin, Z. Yang, K. Cai, L. Zhang, and Z. Su. Topic level expertise search over heterogeneous networks. *Machine Learning J.*, 82(2), 2011.