Facilitating User Interaction With Data

Zainab Zolaktaf
Supervised by Rachel Pottinger
University of British Columbia
Vancouver, B.C, Canada
{zolaktaf, rap}@cs.ubc.ca

ABSTRACT

In many domains, such as scientific computing, users can directly access and query data that is stored in large, and often structured, data sources. Discovering interesting patterns and efficiently locating relevant information, however, can be challenging. Users must be aware of the data content and its structure, before they can query it. Furthermore, they have to interpret the retrieved results and possibly refine their query. Essentially, to find information, the user has to engage in a repeated cycle of data exploration, query composition, and query answer analysis. The focus of my PhD research is on designing techniques that facilitate this interaction. Specifically, I examine the utility of recommender systems for the data exploration and query composition phases, and propose techniques that assist users in the query answer analysis phase. Overall, the solutions developed in my thesis aim to increase the efficiency and decision quality of users.

1. INTRODUCTION

With the advent of technology and the web, large volumes of data are generated and stored in data sources that evolve and grow over time. Often, these sources are structured as relational databases that users can directly query and explore. For instance, astronomical measurements are stored in a large relational database, called the Sloan Digital Sky Survey (SDSS) [19, 21]. Climate data collected from various sources is integrated in relational databases and offered for analysis by users [5].

At a high level, user interaction with data involves two phases: a *query composition* phase, where the user composes and submits a query, and a *query answer analysis* phase, where the user analyses query answers produced by the system. During both phases, however, users can face problems in understanding the data.

Consider, for example, the scientific computing domain. The SDSS schema has over 88 tables, 51 views, 204 user-defined functions, and 3440 columns [14]. A variety of users, ranging from high school students to professional astronomers, with varied levels of skills and knowledge, interact with this database. Furthermore, scientific databases are typically used for Interactive Data Exploration (IDE), where users pose exploratory queries to understand the content and find patterns [13]. Efficiently composing queries over this data to discover interesting patterns, is one of their main challenges.

After successfully composing the query, the next step is to interpret query answers. However, the retrieved results can often be difficult to understand. For example, consider an aggregate query `SELECT AVG(TEMPERATURE)` over climate data. In the weather domain, observational data regarding atmospheric conditions is collected by several weather stations, satellites, and ships. For the same data point, e.g., temperature on a given day, there can be conflicting and duplicate values. Consequently, the aggregate query can have an overwhelming number of correct and conflicting answers. Here, mechanisms that aid the user in understanding the query answers are required.

In my thesis, I develop techniques that assist user interaction with data. I consider the data exploration and query composition phase, and examine the utility of recommendation systems for this phase. Furthermore, I consider the query answer analysis phase and devise efficient techniques that provide insights about query answers. More precisely, I study three problems: 1. how do classical recommendation systems perform with regards to exploration tasks in standard recommendation domains, and how can we modify them to facilitate data exploration more rigorously (Section 2)? 2. what are the challenges of recommendation in the relational database context and which algorithms are appropriate for helping users explore data and compose queries (Section 3)? 3. how can we assist users in the query answer analysis phase (Section 4)? Overall, I aim to develop techniques that help users explore data and increase their decision quality.

2. FACILITATING DATA EXPLORATION WITH RECOMMENDER SYSTEMS

One way to facilitate data navigation and exploration is to find and suggest items of interest to users by deploying a *recommendation system* [6, 16, 29]. Classical recommendation systems are categorized into content-based and collaborative filtering methods.

Content-based methods use descriptive features such as genre of movies, or user demographics, to construct informative user and item profiles, and measure similarity between

© Proceedings of the VLDB 2017 PhD Workshop, August 28, 2017. Munich, Germany.

them. But descriptive features might not be available. Collaborative filtering methods instead infer user interests from user interaction data. The main intuition is that users with similar interaction patterns have similar interests.

The interaction data may include *explicit* user feedback on items, such as user ratings on movies, or *implicit* feedback, such as purchasing history, browsing and click logs, or query logs [11]. An important property of the interaction data is that the majority of items (users) receive (provide) little feedback and are infrequent, while a few receive (provide) lots of feedback and are frequent. But many models only work well when there is a lot of data available, i.e., they make good recommendations for frequent users, and are biased toward recommending frequent items [6, 15, 17].

However, recommending popular items is not sufficient for exploratory tasks. Users are likely already aware of popular items or can find them on their own. Concentrating on popular items also means the system has low overall coverage of the item space in its recommendations. It is essential to develop methods that help users discover new items that may be less common but more interesting. Therefore, we investigate the following research question:

> How do existing recommendation models perform with regard to data exploration tasks in standard recommendation domains, and how can they be modified to facilitate data exploration more rigorously?

To answer this question, we focus on top-N item recommendation, where the goal is to recommend the most appealing set of N items to each user [6]. Informally, the problem setting is as follows: we are given a log of explicit user feedback, e.g., ratings, for different items. We want to assign a set of N unseen items to each user.

2.1 Solution

In our solution [20], we focused on promoting less frequent items, or *long-tail* items, in top-N sets to facilitate exploration. Recommending these items introduces novelty and serendipity into top-N sets, and allows users to discover new items. It also increases the item-space coverage, which increases profits for providers of the items [3, 6, 26, 22]. Our main challenge was in promoting long-tail items in a targeted manner, and in designing responsive and scalable models. We used historical rating data to learn user preference for discovering new items. The main intuition was that the long-tail preference of user u, captured by θ^*_u, depends on the types of long-tail items she rates. Moreover, the long-tail type or weight of item i, captured by w_i, depends on the long-tail preference of users who rate that item. Based on this, we formulated a joint optimization objective for learning both unknown variables, $\boldsymbol{\theta}^*$ and $\mathbf{w}$.

Next, we integrated the learned user preference estimates, $\boldsymbol{\theta}^*$, into a generic re-ranking framework to provide customized balance between accuracy and coverage. Specifically, we defined a re-ranking framework that required three components: 1. an *accuracy recommender* that was responsible for recommending accurate top-N sets. 2. a *coverage recommender* that was responsible for suggesting top-N sets that maximized coverage across the item space, and consequently promoted long-tail items. 3. the user long-tail preference.

In contrast to prior related work [1, 10, 27], our framework learned the personalization rather than optimizing using cross-validation or parameter tuning; in other words, our personalization method was independent of the underlying recommendation model.

	Algorithm	P@5	R@5	L@5	C@5
MT-200K	Random	0.000	0.000	0.871	**0.873**
	Pop [6]	**0.051**	**0.080**	0.000	0.002
	MF [28]	0.000	0.000	**1.000**	0.001
	5D_ACC [10]	0.000	0.000	0.995	0.157
	Cofi^R [24]	0.025	0.046	0.066	0.020
	PureSVD [6]	0.018	0.022	0.001	0.067
	$\theta^{*\text{Dyn900}}_{\text{Pop}}$ [20]	0.027	0.050	0.416	0.171

Table 1: Top-5 recommendation performance.

We evaluated our framework on several standard datasets from the movie domain. Table 1 shows the top-5 recommendation performance for the MovieTweetings 200K (MT-200K) dataset [9] which contains voluntary movie rating tweets from users. For accuracy, we computed precision (P@5) and recall (R@5) [6] wrt the test items of users. Long-tail accuracy (L@5) [10], is the normalized number of long-tail items in top-5 sets per user. Long-tail items are those that generate the lower 20% of the total ratings in the train set, based on the Pareto principle or the 80/20 rule [26]. Coverage (C@5) [10] is the ratio of the number of distinct items recommended to all users, to the number of items.

We compared with non-personalized baselines: Random that has high coverage but low accuracy, and most popular recommendation (Pop) [6], that provides accurate top-N sets but has low coverage and long-tail accuracy. We also compared with personalized algorithms: matrix factorization (MF) with 40 factors, L2-regularization, and stochastic gradient descent optimization [28], a resource allocation approach that re-ranks MF (5D_ACC) [10], CofiRank with regression loss (Cofi^R) [24], and PureSVD with 300 factors [6]. On MT-200K, we chose the non-personalized Pop algorithm as our accuracy recommender, and combined it with a dynamic coverage recommender (Dyn900) introduced in [20]. Our personalized algorithm is denoted $\theta^{*\text{Dyn900}}_{\text{Pop}}$. Table 1 shows that while most baselines achieve best performance in either coverage or accuracy metrics, $\theta^{*\text{Dyn900}}_{\text{Pop}}$ has high coverage, while maintaining reasonable accuracy levels. Furthermore, it outperforms the personalized algorithms, PureSVD and Cofi^R, in both accuracy and coverage metrics.

3. FACILITATING DATA EXPLORATION AND QUERY COMPOSITION

Getting information out of database systems is a major challenge [12]. Users must be familiar with the schema to be able to compose queries. Some relational database systems, e.g, SkyServer, provide a sample of example queries to aid users with this task. However, compared to the size of the database and complexity of potential queries, this sample set is small and static. The problem is exacerbated as the volume of data increases, particularly for IDE. A mechanism that helps users navigate the schema and data space, and exposes relevant data regions based on their query context, is required. We consider using recommendation systems in this setting and focus on the following research question:

> What are the challenges of recommendation in the database context, and which algorithms are suitable for facilitating interactive exploration and navigation of relational databases?

To answer this question, we address top-N *aspect* recommendation, where the goal is to suggest a set of N aspects to the user that facilitate query composition and database exploration. Similar to the collaborative filtering setting in Section 2, we analyse user interaction data, available in a *query log*. Informally, the problem setting is as follows: we are given a query log that is partitioned into sessions, sets of queries submitted by the same user. Furthermore, we also have a relational database *synopsis* with information about the schema of the database (#relations, #attributes, and foreign key constraints) and the range of numerical attributes. Given a new partial session, the objective is to recommend potential query extensions, or aspects.

3.1 Proposed Work

To formulate an adequate solution, the following challenges must be addressed:

1. Aspect Definition. There is no clear notion of "item" or aspect in this setting. Instead, we need to find an adequate set of aspects that can be used to to capture user intent and characterize queries. Given the exploratory nature of queries in the scientific domains, the aspects should enable both schema navigation and data space exploration.

2. Sequential Aspects and Domain-Specific Constraints. Individual elements in a SQL query are sequential and there is dependency between them. For instance, in `SELECT T.A FROM T WHERE X > 10`, the domain of variable X is attributes in table T. Thus, given partial query, only a subset of the aspects are syntactically valid. Queries in the same session, are also submitted sequentially.

3. Session and Aspect Sparsity. In SDSS, the typical session has six SQL queries and lasts thirty minutes [21] which indicates aspect sparsity in queries and sessions.

The relational database setting exhibits some similarities to standard recommendation domains (e.g., movie): Some aspects, e.g., tables, attributes, data regions, are popular while the majority of them are unpopular. Some sessions are frequent, i.e., many queries are submitted, while the majority are infrequent. Scalability and responsiveness is important in both domains.

Analogous to our work in Section 2, our main hypothesis is that merely recommending popular aspects is not sufficient for exploratory tasks. Although popular aspects can help familiarize novice users with concepts like the important tables and attributes, given the exploratory nature of queries in IDE, recommendations are deemed more useful if they can help users narrow down their queries and expose relevant data regions. For example, recommending a specific interval like $b_1 <$ `BRIGHTNESS` $< b_2$ is more useful than just suggesting the attribute `BRIGHTNESS`.

Based on these intuitions, we will focus on recommending interesting aspects that enable data exploration and schema navigation for users of a relational database, and in particular, in IDE settings. Using the query log and the database synopsis, we will devise a set of aspects that include not just the relations, attributes, and user-defined functions, but also *intervals* of numeric attributes, e.g., $b_1 <$ `BRIGHTNESS` $< b_2$. Subsequently, we can use a vector-based query representation model where each element denotes the presence of a certain aspect. Alternatively, a graph-based representation [23] might be more suitable. After formulating similarity measures between queries (or sessions) [2], we can use a nearest neighbour model to suggest relevant aspects to the user.

In contrast to prior work that focuses on supervised learning and query rewriting [7], we focus on aspect definition and extraction. In contrast to [4, 7, 8], we rely on the database synopsis only. Accessing a large scientific database like SDSS to retrieve the entire set of tuples is expensive. In contrast to [14] our recommendations include intervals not just tables and attributes. The intermediate query format in [18] is complementary to our work.

4. FACILITATING QUERY ANSWER ANALYSIS

After users have successfully submitted a query, their next challenge is to analyse and understand the query answers. When the answer set is small, this task is attainable. The challenge is in examining and interpreting large, or even conflicting, answer sets.

To illustrate the problem, consider again climate data that is reported by various sources and integrated in relational databases. Because the sources were independently created and maintained, a given data point can have multiple, inconsistent values across the sources. For example, one source may have the high temperature for Vancouver on 06/11/2006 as 17C, while another may list it as 19C. As a result of this *value-level* heterogeneity, an aggregate query such as `SELECT AVG(TEMPERATURE)` does not have a single *true* answer. Instead, depending on the choice of data source combinations that are used to answer the query, different answers can be generated. Reporting the entire set of answers can overwhelm the user. Here, mechanisms that summarize the results and help the user understand query answers are required. Therefore, we study the following research question:

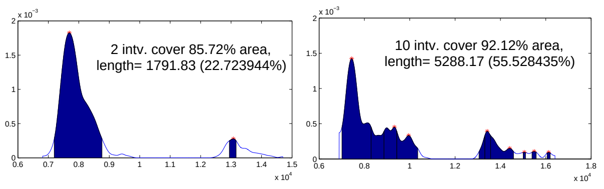
> After a query has been submitted to the system, how can we help the user understand and interpret the query answers?

Specifically, we address the problem of helping users understand aggregate query answers in integration contexts where data is segmented across several sources. We assume meta-information that describes the mappings and bindings between data sources is available [25].Our main concern is how to handle the value-level heterogeneity that exists in the data, to enable the user to better understand the range of possible query answers.

4.1 Solution

In our solution [30], we represented the answer to the aggregate query as an *answer distribution* instead of a single scalar value. We then proposed a suite of methods for extracting statistics that convey meaningful information about the query answers. We focused on the following challenges 1. determining which statistics best represent and answer's distribution 2. efficiently computing the desired statistics. In deriving our algorithms, we assumed prior knowledge regarding the sources is unavailable and all sources are equal.

A *high coverage interval* is one of the statistics we extract to convey the shape of the answer distribution and



(a) S_1 (b) S_4

Figure 1: High coverage intervals tell where the majority of answers can be found.

the intervals where the majority of viable answers can be found. Figure 1 shows the multi-modal answer distributions of the aggregate query `AVG(TEMP)`, on Canadian climate data (S_1) [5] and synthetic data (S_4) [30], and their corresponding high coverage intervals.

5. SUMMARY AND OUTLOOK

The goal of my thesis is to devise techniques that facilitate user interaction with data. I address three aspects:

- (Accomplished) Facilitating data exploration with recommender systems in standard domains (Section 2).
- (In progress) Facilitating data exploration and query composition in the relational database context (Section 3). I am currently working on extracting a dataset, and narrowing down the problem statement.
- (Accomplished) Facilitating query answer analysis by extracting statistics and semantics about the range of query answers (Section 4).

6. REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *TKDE*, 24(5):896–911, 2012.

[2] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Similarity measures for olap sessions. *Knowledge and information systems*, 39(2):463–489, 2014.

[3] Pablo Castells, Neil J. Hurley, and Saul Vargas. *Recommender Systems Handbook*, chapter Novelty and Diversity in Recommender Systems. Springer US, 2015.

[4] Gloria Chatzopoulou, Magdalini Eirinaki, and Neoklis Polyzotis. Query recommendations for interactive database exploration. In *International Conference on Scientific and Statistical Database Management*, pages 3–18. Springer, 2009.

[5] Climate Canada. Canada climate data. `http://climate.weatheroffice.gc.ca/climateData/canada_e.html`, 2010.

[6] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, 2010.

[7] Julien Cumin, Jean-Marc Petit, Vasile-Marian Scuturici, and Sabina Surdu. Data exploration with sql using machine learning techniques. In *EDBT*, 2017.

[8] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 517–528. ACM, 2014.

[9] Simon Dooms, Toon De Pessemier, and Luc Martens. Movietweetings: a movie rating dataset collected from twitter. In *CrowdRec at RecSys*, 2013.

[10] Yu-Chieh Ho, Yi-Ting Chiang, and Jane Yung-Jen Hsu. Who likes it more?: mining worth-recommending items from long tails by modeling relative preference. In *WSDM*, pages 253–262, 2014.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. IEEE, 2008.

[12] HV Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. Making database systems usable. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 13–24. ACM, 2007.

[13] Martin L Kersten, Stratos Idreos, Stefan Manegold, Erietta Liarou, et al. The researchers guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB Challenges and Visions*, 3, 2011.

[14] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. Snipsuggest: context-aware autocompletion for sql. *Proceedings of the VLDB Endowment*, 4(1):22–33, 2010.

[15] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *WWW*, pages 85–96, 2014.

[16] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *arXiv preprint arXiv:1205.3193*, 2012.

[17] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[18] Hoang Vu Nguyen, Klemens Böhm, Florian Becker, Bertrand Goldman, Georg Hinkel, and Emmanuel Müller. Identifying user interests within the data space-a case study with skyserver. In *EDBT*, pages 641–652, 2015.

[19] M Jordan Raddick, Ani R Thakar, Alexander S Szalay, and Rafael DC Santos. Ten years of skyserver i: Tracking web and sql e-science usage. *Computing in Science & Engineering*, 16(4):22–31, 2014.

[20] Information removed for double-blind review. Submitted paper, 2017.

[21] Vik Singh, Jim Gray, Ani Thakar, Alexander S Szalay, Jordan Raddick, Bill Boroski, Svetlana Lebedeva, and Brian Yanny. Skyserver traffic report-the first five years. *arXiv preprint cs/0701173*, 2007.

[22] Saúl Vargas and Pablo Castells. Improving sales diversity by recommending users to items. In *RecSys*, 2014.

[23] Roy Villafane, Kien A Hua, Duc Tran, and Basab Maulik. Mining interval time series. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 318–330. Springer, 1999.

[24] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. Maximum margin matrix factorization for collaborative ranking. *Advances in neural information processing systems*, pages 1–8, 2007.

[25] Jian Xu and Rachel Pottinger. Integrating domain heterogeneous data sources using decomposition aggregation queries. *Information Systems*, 39(0), 2014.

[26] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *PVLDB*, 5(9):896–907, 2012.

[27] Weinan Zhang, Jun Wang, Bowei Chen, and Xiaoxue Zhao. To personalize or not: a risk management perspective. In *RecSys*, pages 229–236, 2013.

[28] Yong Zhuang, Wei-Sheng Chin, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel sgd for matrix factorization in shared memory systems. In *RecSys*, pages 249–256, 2013.

[29] Sedigheh Zolaktaf and Gail C Murphy. What to learn next: recommending commands in a feature-rich environment. In *ICMLA*, pages 1038–1044. IEEE, 2015.

[30] Zainab Zolaktaf, Jian Xu, and Rachel Pottinger. Extracting aggregate answer statistics for integration. *EDBT*, 2015.