Feature-based recommendation framework on OLAP

Yang Yang, Jinli Cao
Department of Computer Science and Computer Engineering
La Trobe University, VIC, 3086
Australia

y14yang@students.latrobe.edu.au

J.Cao@latrobe.edu.au

Abstract

The queries in Online Analytical Processing (OLAP) are user-guided. OLAP is based on a multidimensional data model for complex analytical and ad-hoc queries with a rapid execution time. Those queries are either routed or on-demand revolved around the OLAP task. Most such queries are reusable and optimized in the system. Therefore, the queries recorded in the query logs for completing various OLAP tasks may be reusable. The query logs usually contain a sequence of SQL queries that show the action flows of users for their preference, their interests, and their behaviours during the action.

This research investigates the feature extraction to identify query patterns and user behaviours from historical query logs. The expected results will be used to recommend forthcoming queries to help decision makers with data analysis. The purpose of this work is to improve the efficiency and effectiveness of OLAP in terms of computation cost and response time. Furthermore, the proposed OLAP system will be able to adjust some parameters for finding common behaviours from different users that make the recommendation system flexible and user-adaptive.

Keywords: OLAP, session feature, query recommendation.

1 Introduction

The OLAP analysis system, a major part of business intelligence, is an effective approach to processing complex queries in multi-dimensions so that the user acquires a multi-dimensional view from a data warehouse (Dinter, Sapia, Hofling and Blaschka 1998). The typical applications of OLAP are in business reporting for sales, marketing, budgeting and forecasting, and similar areas. OLAP is interactive and is involved in a series of queries for either driven-down or roll-up queries to support specific tasks of decision making. During the analytical processing, a large amount of multidimensional data is required to be accessed and based on the online queries. Since OLAP is driven by the user query that the user needs a clear purpose and knowledge about areas of interest before operating OLAP processing (Sapia 1999). This approach is classified as query-driven centric as opposed to user centric techniques. However, when exploring a series of tasks, the user may not have clear ideas of what a next query should be made. This might cause analysis latency or lead users into an irrelative area, and thus reduce the benefits of using the OLAP system. To overcome this problem, we present a framework for query recommendation embedded in the OLAP system, which helps users predict forthcoming queries.

In business processing, users always utilize a sequence of queries to interact with the OLAP system to finish a task. The operation is often monotonous and interrupted since users could lose their orientation to the goal (Sarawagi 2000). Most frequently-used queries are recorded in the system as query logs. Those recorded queries may be reusable for completing similar OLAP tasks. The query logs usually contain a sequence of SQL queries that show the action flows of users for their preference, their interests, and their behaviours during the action. We will propose a new framework to provide navigation and forecast forthcoming queries which maximizes efficiency of using OLAP systems. A sequence of queries/requirements is called an analysis session (Giacometti Marcel and Negre 2008). The new framework is based on a collaborative filtering method (Herlocker, Konstan, Terveen and Riedl 2004). This method assumes if user A has the same interested area to user B, then they may be interested in the same data. The queries of user B may be reusable for user A. The proposed framework utilizes all users' query logs and summarises these query records to capture different users' query behaviour. The featured information is used to forecast what the forthcoming query could be.

It is important for query recommendations to identify the current user's purpose in order to make an accurate recommendation. However, current users cannot supply much information at the beginning so that previous approaches could not match a session to current users. This problem can be considered as unbalanced data. Previous sessions may include too many features that could not provide the central themes of the sessions, while the current session has little information. To overcome this drawback, our framework introduces a new approach that balances the features between the previous session and the current session. Our approach can satisfy the user's requirement and provides appropriate query recommendations. The proposed framework is flexible and user-adaptive for recommender systems.

This paper is organized as follows. Section 2 reviews some existing work about query recommendations. Section 3 introduces our framework and its instantiation. The experimental results are shown in section 4. Conclusions are presented in section 5.

2 Related work

Query recommendation is an important technique for business search engines. The majority of work on query recommendation focuses on measuring the similarity between the current query and the previous query (Fonseca, Golgher, Moura and Ziviani 2003) in order to expand the query or cluster of queries (Baeza-Yates, Hurtado and Mendoza 2004, Wen, Nie and Zhang 2001). The basic model of recommendation systems relies on two methods which are content-based methods and collaborative filtering methods (Adomavicius and Tuzhilin 2005).

2.1 Content-based methods

Content-based methods suggest items similar to the ones that users have previously shown interest in. They mainly extract information features, such as the different sales of cars each year, and decide which recommendations are appropriate. These methods compare various candidate items, and then the best matching items are presented to the user. Generally, they use a distance function to rate or order each candidate item. The cosine similarity measure (Adomavicius and Tuzhilin 2005) is one of the best vector measures that represents weight best-matching.

In the early data source, the significant information existed randomly. Content-based methods apply information retrieval on data sources (Baeza-Yates and Ribeiro-Neto 1999) and information filtering on recommendations (Belkin and Croft 1992). The methods extract useful profiles that contain information about users' preferences and search behaviour. The profiles can be elicited from users' queries.

The limitation of this method is obvious. The users need to explicitly describe their objects, so the system must first learn the user's purposes. However, the user's purpose is much harder to indicate when they get a new job task. Moreover, content-based methods cannot suggest forthcoming queries.

2.2 Collaborative filtering methods

Collaborative filtering methods recommend the items which have similar interest between the user and the other users. They can be categorized into memory-based and model-based methods by algorithm (Breese, Heckerman and Kadie 1998).

The memory-based algorithm exploits the weight of all previous items to compute the weight of the current session. The formula is below:

$$r_{c,s} = k \sum_{c' \in \hat{C}} sim(c, c') \times r_{c',s}$$

$$k = \frac{1}{\sum_{c' \in \hat{C}} |sim(c, c')|}$$

where $r_{c,s}$ denotes a weight of the user c and item s. $r_{c,s}$ multiplier k serves as a normalizing factor, and sim(c,c') is a distance measure between user c and user c', which can be considered as a weight. Sim(c,c') is introduced in order to be able to differentiate between levels of user similarity. This formula means that the high weight depends on the comparability of both c and c'. Different recommendation applications can use their own user similarity measure, as long as the calculations are normalized using the normalizing factor k.

Model-based algorithms (Billsus and Pazzani 1998) use known weights to build a model, and then recommend a query by this model. Cluster models and Bayesian networks are techniques in this algorithm that rate probability expression on the purpose of interest. The model learns all features from the data. For example, in a car recommendation system, a user might prefer a kind of car with job purposes like 'energy saving car' and completely different type for sport like 'four wheel drive'.

To improve the performance of the collaborative filtering method, several techniques that exclude noise, redundancy and exploit the sparsity of data should be integrated (Yu, Xu, Tao, Ester and Kriegel 2002). As in the case of the content-based method, the main improvement of the collaborative filtering method is that it uses other users' information to make a recommendation even though the information never appeared before by current user.

Collaborative filtering methods also have their own limitations. They have the same problem as content-based methods. In order to make accurate recommendations, the user must offer abundant information to describe the job purposes. Another significant drawback is that other users' information could include some irrelevant information that would affect the accuracy of recommendations.

2.3 Other methods

The graph model based similarity method describes the two consecutive queries (neighbouring queries) in the same query session. These have more similar weight than the queries which are not neighbours in the same session (Zhang and Nasraoui 2006). Instead, Fonseca et al. detects similar queries based on association rules. Each query in the query log is considered as a part of session where a single user inputs a sequence of related queries in a time interval (Fonseca, Golgher, Moura and Ziviani 2003). We will use a similar notion in this paper.

Many recommendation systems are applied in web searching, and some of those techniques can be adapted to the OLAP recommending system. Wen et al. (Wen, Nie and Zhang 2001) present four notions about clustering methods for query recommendation to measure query distance: the first notion is based on keywords or phrases; the second is on a string matching of keywords; the third is on common clicked URLs; and the fourth on the distance of the clicked documents in some pre-defined hierarchies. The first two notions will be adopted in our work.

The QueRIE (Chatzopoulou, Eirinaki and Polyzotis 2009) framework developed by Chatzopoulou et al. generates a predicted summary (S_0^{pred}) to capture the predicted degree between current user and the other users who have a similar query requirement. Where S_i represents the session summary of user i, when i= 0 denotes the current user. S_0^{pred} is described as a weighted vector, which represents the number of given tuples τ appeared in database. This framework builds a predicted model which also returns recommendations to the current user. There are two different schemes for calculating S_i (Chatzopoulou, Eirinaki and Polyzotis 2009).

$$S_Q[\tau] = \begin{cases} 1 & if\ tuple\ \tau\ appears \\ 0 & if\ tuple\ \tau\ does\ not\ appear \end{cases}$$

This is a binary weighting scheme referring to matching tuple τ or not.

$$S_Q[\tau] = \begin{cases} 1/|ans(Q)| & if\ tuple\ \tau\ appears \\ 0 & if\ tuple\ \tau\ does\ not\ appear \end{cases}$$

This is the result of the weighting scheme, where ans(Q) is the result-set of query Q. It implies that S_Q could have a small outcome if many queries return results since this query is unfocused. In other words, if S_Q is large, then the query is very specific. The session summary is defined below:

$$S_i = \sum_{Q \in Q_i} S_Q$$

This approach adopted the method of collaborative filtering. The predicted summary is defined as a function of the current user's summary S_0 and the normalized weighted sum of the existing summaries (Chatzopoulou, Eirinaki and Polyzotis 2009):

$$S_0^{pred} = \alpha \times S_0 + (1-\alpha) \times \frac{\sum_{1 \le i \le h} sim(S_0, S_i) \times S_i}{\sum_{1 \le i \le h} sim(S_0, S_i)}$$

The value of the mixing factor $\alpha \in [0,1]$ specifies which users' requirement will be considered. For example, if $\alpha = 0$, the result trends to past users' requirements. S_0^{pred} also tends into content-base filtering when $\alpha = 1$. The value of α can decide which approach is more favorable. This approach could exclude other uses' query requirement if the current session clearly declares the query's orientation. Moreover, it is desirable that current user can get some useful knowledge from past users to guide or adjust successor queries. Our approach is able to predict queries that combine the results already observed by the past users.

3 Query recommendation based on user behavior

By reviewing the existing methods above, we observed that query recommendation needs further improvement that a more feasible method is necessary to apply for OLAP systems. Therefore, we propose a new recommendation system to better the performance on query recommendation.

3.1 Improve existing query recommendation system

The existing query recommendation systems extend from collaborative filtering and content-based methods. They depend on distance measure to generate related sessions or queries. However, each previous session may include many queries, and thus it may involve much independent information. This fact shows that some sessions may be unfocused if the session contains many queries. The current session only contains few queries in the beginning so that the current session cannot clearly describe the main purpose of the user's requirements yet. Furthermore, if several previous sessions cover current session, the different focuses may offer different suggestions to current session.

This, therefore, motivates us to look for a novel method which can overcome the drawbacks of existing methods. Our proposed framework will satisfy the following requirements:

- It should filter unuseful information in the previous sessions to identify session patterns from historical query logs.
- It should summarize the main purpose of previous sessions. The expected results will be used to recommend forthcoming queries to help decision makers on data analysis.
- It should improve the efficiency and effectiveness of OLAP in terms of computation cost and response time.

3.2 Framework of feature-based recommendation

The feature-based recommendation framework can be formulated as follows: when interacting with a data warehouse, a user might have the same or similar requested task as a past user in the previous session. To extract the previous session, the system can use the log information to summarize the querying behavior from past users.

Our recommended framework (figure 1) works on the OLAP system using SQL queries. The user's queries are sent to OLAP and the recommendation engine. The OLAP system processes every query with the data warehouse and returns the results to the user. Every user's query is also recorded in the query log with the user ID to store its queries. The recommendation engine relies on current inputting queries and past user's queries which are stored in query log to generate a set of query recommendations using our novel approach, and then the recommendation engine returns the recommendations to the user.

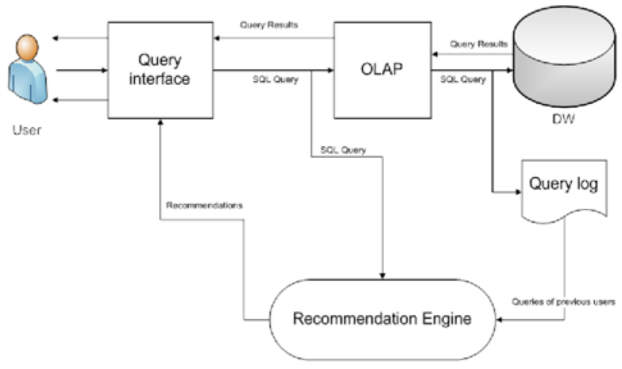

Figure 1: feature-based recommendation framework

3.3 Definition of notions and symbols

We now introduce some notions and symbols which will be used in the remainder of this paper.

3.3.1 Models of sessions and queries

The basic operations of OLAP include three methods that referred as slice and dice, pivot and drill down, and roll up to analyze a series of users' requirements by SQL queries.

These requirements (queries) of the same user can be considered as one task that may contain one purpose during the search job. Semantically, a series of requirements or queries will be reusable for other users who have similar tasks on their search job. These SQL queries are recorded into query logs in order to reuse them. The query logs are denoted as L and one purpose of a task is called a session as denoted as S_i. Each session includes a series of queries that are entered by the same user, and each query denoted as q_i^j for the i^{th} query in the j^{th} session. This relationship is represented as follows:

$$q_i^j \in S_j \text{ , where } 1 \leq i,j \leq n \text{ and } S_j \in L.$$

3.3.2 Session Feature

In each search task, the user usually inputs a sequence of queries to accomplish one task. They may ask the sale results of various products in different locations or income of branch offices in different countries. It means that every session has some kinds of characteristics shown as the keywords in queries. Many keywords in one session may produce the characteristics of the session. Our work focuses on extracting the particular characters as the guide to make recommendation. Each particular character is called a session feature denoted as F_i. Session features can express the main purpose of the session. Different sessions could have similar search tasks (session feature) that means existing a probability of similarity between session *j* and session *i* which is denoted as R_i^j. The research of this paper is to mine the query logs and extract the session features from query logs that will be used to recommend for current users. The symbols will be used in this paper shown in table 1.

L	Query logs
S_i	A sequence of queries entered by user *i*
t	A tuple of keywords
q_i^j	i^{th} query of the j^{th} session
R_i^j	similarity between session *j* and session *i*
F_i	Session feature

Table 1: Symbol summary

3.4 Feature-based recommendation approach

Query logs are important data sources for query recommendations. There are two steps in our approach. First, summarize session features by utilizing past users' search behaviour. Second, locate similar sessions to match the current session.

The Query log can be divided into sessions by many ways such as user id, timestamp and performance in a period. We combine user id and timeout threshold as a splitting sessions rule. The timeout threshold is set less than 12 hours. Thus, the query log is divided firstly into several parts by user ids, and then compares the timestamps of two consecutive queries. If timeout of these two queries exceeds 12 hours, the smaller timestamp of the query is a splitting point for a new session. Since each session could include many queries, the theme of a session is difficult to detect. Hence, we need extract keywords from each session to represent the features of this session.

Intuitively, for a session S_i and current session S_c, the more common keywords exist between S_i and S_c the more S_i is similar to S_c. Therefore, the problem of measuring similarity between S_i and S_c becomes the calculation of common features of S_i and S_c.

Let us look at an example: suppose that F_i and F_j are two feature sets of previous sessions S_i and S_j respectively, and S_i and S_j include the same number of keywords with current session S_c.

Figure 2 shows the graphical example of feature sets F_c, F_i and F_j. The shadow square box represents current session F_c. F_j and F_i contain F_c, that is, $F_c = |F_i| \cap |F_j|$. Since $|F_i| > |F_j|$, the proportion of similarity between F_c and F_i (F_c / F_i) is smaller than F_c / F_j. It is easily observed that current session S_c is more similar to S_j than S_i.

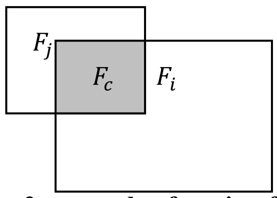

Figure 2: example of session feature

Measure session feature. Due to find suitable session to match current session, we need measure session features. Every session is composed of many SQL queries. Each query has a *select* statement as beginning, which includes several attributes from different tables in database. We consider these attributes as the keywords of the query. If a session S consists of a sequence of queries, the features of session S is the most frequently appeared keywords in those queries.

A tuple is one of the combinations from keywords in a query. We assume that the probability $P_i(t)$ represents the importance of a tuple with keywords, so we have:

$$P_i(t) = \alpha \times \frac{n_l}{N_l} + (1 - \alpha) \times \frac{n_s}{N_s}$$

Where *i* denotes the number of keywords in tuple t; n_l and n_s stand for the number of tuple *t* appearing in the query logs *L* and session *S* respectively; N_l and N_s denote all tuples in *L* and *S*. The value of the moderator $\alpha \in [0,1]$ determines which approach is more favourable when computing the session features. If $\alpha = 0$, only user session information will be taken into account when measuring session features, That is, the content-based filtering method is used. When $\alpha = 1$, the collaborative filtering approach is applied to measure probability of tuple *t*. Any value of α in an acceptable range allows us to adjust more importance to either side. This adjustment can be useful for two reasons. First, $\frac{n_l}{N_l}$ is able to find the important tuple that most users are interested in, but it cannot represent if a tuple has high frequency in the session instead of the query log. We do not want to exclude from unimportance tuples that seem to explain the purpose of session. Second, a particular search task in each session usually requests some typical tuples to expand on the following search task.

In other words, the tuple with high hits may have ability to predict what kinds of purposes with search task in this session. High hits have different means when they appeared in the query log and sessions, which represent a common view and an individual view respectively. The action of α is used to balance these two views.

The number of tuples generated from a query is the all combinations of keywords in the query. For example, a query q contains three keywords $q = \{k_1, k_2, k_3\}$, all combinations of k_1, k_2, k_3 make the set of tuples of q as follows:

$$tuples = \{(k_1), (k_2), (k_3), (k_1, k_2), (k_1, k_3), \\ (k_2, k_3), (k_1, k_2, k_3)\}$$

Where each element in the above combination set is called a tuple. $P_i(t)$ is a threshold used for determining the important tuple in a session. If $P_i(t)$ exceeds the threshold, the tuple is important in the session. It can be represented as one of session features, such as $F_i = \{(k_1), (k_1, k_3)\}$. From the result of our experiments, our approach can extract session features effectively which match the purpose of each session.

Weight of tuple. As a set of session features, each feature can be modelled as a weight in the session.

Definition 1: Let the tuple frequency be the number of occurrences of tuple in the session with the notation $freq(t_r^i)$. The weight of tuple frequency $TF(F,t)$ measures the relationship of a tuple *t* with regard to the given session feature *F*.

There are many ways to measure the weight of each tuple. For example, we can simply define that as the tuple frequency against the total number of occurrences of all the tuples in the session. In this paper, we utilize a formula from the Cornell SMART system to normalize the tuple frequency. The formula is described as follows:

$$TF(F,t) = 1 + log\left(1 + log\left(freq(t_r^i)\right)\right)$$

Computing R_i^j. The next step in framework is computing the similarity and recommendation value. Similarity of two sessions is computed by session feature. In this work, we adopt the method of cosine similarity that measures the similarity between two sessions and takes values in [0,1].

Definition 2: Given two clustering session features $F_i = \{t_{1,}^i t_2^i, \ldots, t_{r,}^i\}$ and $F_j = \{t_{1,}^j t_2^j, \ldots, t_m^j\}$, the overlapping tuples between F_i and F_j can be represented as $F_i \cap F_j = \{t_1^o, t_2^o, \ldots, t_k^o\}$, where t_k^o denotes the common tuple in F_i and F_j. If the session is the current session, the session feature F_c includes all tuples by combinations from keywords. Here, we have:

$$R_i^j = \frac{\sum_k TF(F_i, t_k^o) \cdot TF(F_j, t_k^o)}{\sqrt{\sum_r TF(F_i, t_{r,}^i)}^2 \cdot \sqrt{\sum_m TF(F_j, t_m^j)}^2}$$

If there is a highest score of R_i^c for current session S_c, it implies that S_i has the similar search task with S_c. Therefore, S_i can be recommended to S_c for forthcoming queries.

The users usually input a series of queries when they finish one job task. The purpose of their task will be clearer when more queries are performed. During the processing current user's queries, the system keeps the update of suitable recommendations. However, recalculating the recommendations takes much time because the query log may have large number of sessions. Therefore, our system generates top-k recommendations to provide multiple choices for the current user. When the user inputs successional queries, the system generates new recommendation from multiple choices of top-k recommendations rather than recalculation from query log. This top-k recommendation set is computed by the R_i^j, and is defined as follows:

$$rank(R_i^c) \geq threshold$$

The R_i^c has high rank if it is larger than the threshold. The top ranked sessions are returned as the recommendation set.

4 Experimental evaluation

In this section, we developed a virtual query interface system which realizes our framework and analyse actual data, as well as present experimental results of using our system.

4.1 Experimental data and methodology

The experiment was carried on a Core 2 Duo 2.4GHz computer with Window 7 Ultimate and 2 GB of main memory. All system was implemented in Java.

The system adopted the data of query log from sky server database. The query log recorded queries from year 2007 to 2010 and its size is 1.3 TB. We separated the query log into 4000 sessions by different IP addresses. We chose 55 sessions to construct current sessions and the number of queries in each session are no more than 6. For the rest of sessions, firstly, they were divided into 10 equally sized subsets as previous session logs. Secondly, we partitioned the rest of sessions into 10 different sized subsets, and the number of session in next subset is larger than the preceding subset. In order to analyse the performance of our framework, we use a smaller set of queries in each current session. For example, for each current session with *n* queries, we extracted *n-1* queries in each current session to build test set.

We evaluate the accuracy to analyse the effectiveness of each recommended sessions, using the following metrics:

$$accuracy = \frac{|k_c \cap k_i|}{|k_c|}$$

Where k_c and k_i represent all queries of keywords in a session. In our test, we have analysed all subsets and reported the best result in the experiments. Unless otherwise noted, we set $\alpha = 0.6$.

4.2 Result

Our first experiment evaluates the efficiency of the proposed approach to make the recommendations. Figure

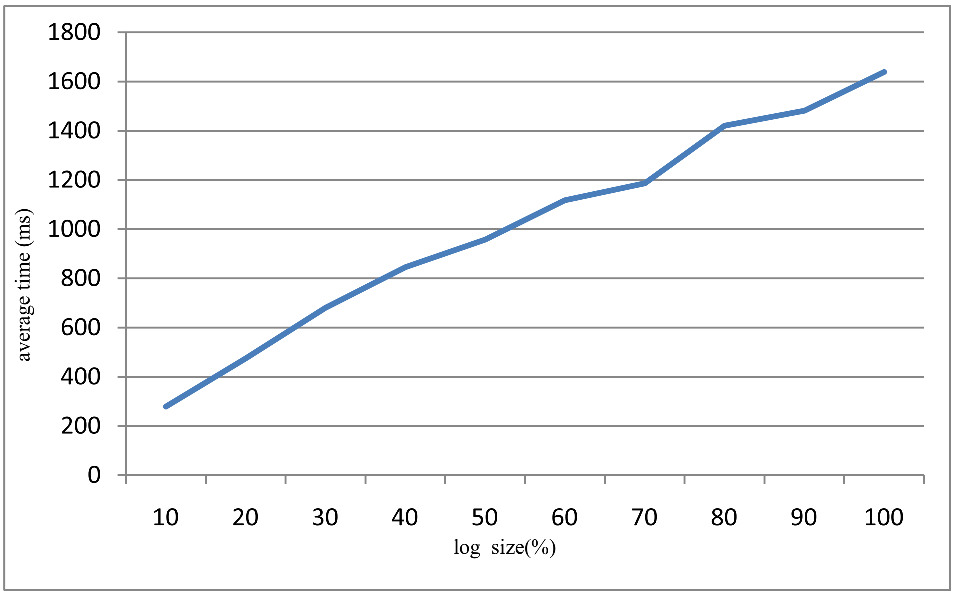

Figure 3: efficiency analysis

3 shows the performance of test sessions according to the size of query log. The measure of this experiment includes computation of similarity with current session and ranking the candidates. The features of sessions are processed on off-duty time so that the extraction time does not count in efficiency analysis.

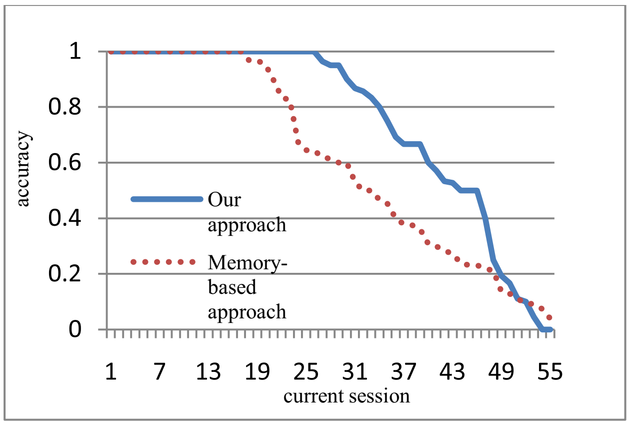

Figure 4: accuracy distribution

As can be seen from Figure 3, it is obvious that the trend of execution time is upwards with the log size. The execution time is acceptable with the size of 4000 sessions in query log.

In the next experiment, we evaluate the effectiveness of our approach. Figure 4 indicates the comparison of best performance between our approach and the memory-based approach (Breese, Heckerman and Kadie 1998). 55 current sessions have been tested among 10 equally sized subsets, which ranked by accuracy in decreasing order. It can be seen from Figure 4, 70% current sessions achieved high performance by our approach (around 40 current sessions have above 0.6 accuracy among total 55 current sessions tested). The system can provide valuable recommendations for most current sessions. There are two zero accuracy results among all tests, that is, the system may have 3.6% failure rate to predicate intention under some circumstances. We have also manually reviewed the generated data for the validation. No recommendation sessions have been obtained if low similarity R_i^j has been calculated between the current session and past sessions. For memory-based approach, only 55% current sessions can get valuable recommendations, and most of accuracy is lower than our approach (refer to the dot line in Figure 4).

Figure 5 describes an average accuracy of our approach and the memory-based approach from total 55 current sessions tested. We can find that the accuracy of our approach is always higher than the memory-based approach. The figure shows that the accuracy slightly increases with the log size. The Figures 4 and 5 demonstrate that most current sessions can obtain a successful recommendation by our approach with accuracy above 0.6 from the query log.

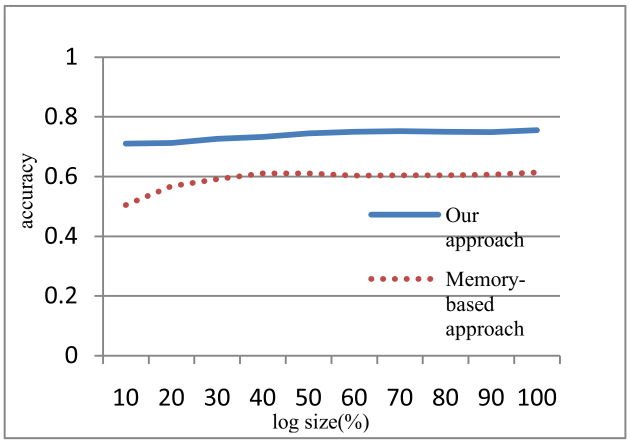

Figure 5: average accuracy

5 Conclusion

To sum up, our feature-based recommendation framework highlights on personal intention. It can summarize the important information and filter out unuseful data to build session features from previous sessions. The experimental evaluation proves our framework is efficient and effective.

Our proposed approach is promising and has great potential to apply in real-life.

6 References

Dinter, B., Sapia, C., Hofling, G. and Blaschka, M. (1998): The OLAP market: state of the art and research issue. *Proc. DOLAP'98, ACM First International Workshop on Data warehousing and OLAP*, Bethesda, Maryland, USA.

Sapia, C.(1999): On modelling and predicting query behaviour in OLAP systems. *Proc. Int'l workshop on design and management of data warehouses DMDW'99*, Swiss life.

Giacometti, A., Marcel, P. and Negre, E. (2008): A framework for recommending OLAP queries. *Proc. DOLAP'08, ACM 11th international workshop on data warehousing and OLAP*, New York, NY, USA.

Sarawagi, S. (2000): User-adaptive exploration of multidimensional data. *Proc. VLDB*, 307-316.

Baeza-Yates, R., Hurtado, C. and Mendoza, M. (2004): Query Recommendation Using Query Logs in Search Engines. *Proc. Springer international workshop on clustering information over the web (clustweb, in conjunction with EDBT)*: 588-596, Creete.

Giacometti, A., Marcel, P., Negre, E. and Soulet, A. (2009): Query recommendations for OLAP discovery driven analysis. *Proc. ACM twelfth international workshop on data warehousing and OLAP DOLAP'09*, New York, NY, USA.

Adomavicius, G. and Tuzhilin, A. (2005): Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Proc. IEEE Transactions on Knowledge and Data Engineering* 17(6): 734-749.

Breese, J., Heckerman, D. and Kadie, C. (1998): Empirical analysis of predictive algorithms for collaborative filtering. *Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, 461(8): 43-52.

Billsus, D. and Pazzani, M. (1998): Learning collaborative information filters. *Proc. International Conference on Machine Learning*, Morgan Kaufmann, 54:48.

Chatzopoulou, G., Eirinaki, M. and Polyzotis, N. (2009): Query recommendations for interactive database exploration. *Proc. Springer-verlag*: 3-18.

Herlocker, J., Konstan, J., Terveen, L. and Riedl, J. (2004): Evaluating collaborative filtering recommender systems. *Proc. ACM Transactions on Information Systems*, ACM, 22(1): 5-53.

Fonseca, B., Golgher, P., Moura, E. and Ziviani, N. (2003): Using Association Rules to discover Search Engines Related Queries. *Proc. First latin American web congress*, IEEE Computer Society: 66-71.

Zhang, Z. and Nasraoui, O. (2006): Mining Search Engine Query Logs for Query Recommendation. *Proc. 15th international conference on World Wide Web*, ACM: 1039-1040.

Wen, J., Nie, J. and Zhang, H. (2001): Clustering user queries of a search engine**.** *Proc. 10th international conference on World Wide Web*, ACM: 162-168.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999): Modern Information Retrieval. *Proc. Addison Wesley*, 463(1): 513.

Belkin, N. and Croft, B. (1992): Information filtering and information retrieval: two sides of the same coin. *Proc. Communications of ACM*, 35(12): 29-38.

Yu, K., Xu, X., Tao, J., Ester, M. and Kriegel, H. (2002): Instance selection techniques for memory-based collaborative filtering. *Proc. second SLAM international conference on data mining*.