"You May Also Like" Results in Relational Databases

Kostas Stefanidis
Dept. of Computer Science
University of Ioannina, Greece
kstef@cs.uoi.gr

Marina Drosou
Dept. of Computer Science
University of Ioannina, Greece
mdrosou@cs.uoi.gr

Evaggelia Pitoura
Dept. of Computer Science
University of Ioannina, Greece
pitoura@cs.uoi.gr

ABSTRACT

In this position paper, we consider extending relational database systems with a recommendation functionality. In particular, we propose that, along with the results of each query, the user gets additional recommended results that we call *"You May Also Like"* or YMAL results. We discuss a suite of different approaches to computing YMAL results and exploit one that uses only the database content and the query results. Some preliminary evaluation results are also provided.

1. INTRODUCTION

The typical interaction of a user with a database system is by formulating queries. This interaction mode assumes that users are to some extent familiar with the content of the database and also have a clear understanding of their information needs. However, as databases get larger and accessible to a more diverse and less technically-oriented audience, a new "recommendation"-oriented form of interaction seems attractive and useful.

In this paper, motivated by the way recommenders work, we consider "recommending" to the users tuples not in the results of their queries but of potential interest. For instance, when asking for a *Woody Allen* movie, we could recommend a *Woody Allen* biography. When looking for drama movies produced in *England* with *Oscar* nominations, we could also recommend similar movies with *BAFTA* awards. Further, we may recommend what similar users have asked for in the past.

We call such results *"You May Also Like"* or YMAL results for short. YMAL results are useful because they let users see other tuples in the database that they may be unaware of.

We consider three fundamentally different approaches to computing YMAL results. The first one, termed *current-state*, uses the results of the current query and the database content. The second one, termed *history-based*, is similar to traditional recommendation systems. It uses the past history of user queries to suggest tuples that are results of either similar past queries or results of queries posed by similar users. The last one, termed *external sources*, considers using information from resources external to the database, such as the web.

We focus on the *current-state* approach and present a novel method for computing YMAL results. The method explores both the database schema by expanding the original query through joins with appropriate other relations and the database content through value correlations. We also report some preliminary evaluation results of this method.

Extending database queries with recommendations has also been suggested in two very recent works, namely [12] and [7]. FlexRecs [12] proposes a framework and a related engine for the declarative specification of the recommendation process. Here, we address a specific recommendation process, that of suggesting results relevant to a given user query and propose methods for generating them. Recommendations in [7] are more restricted than our YMAL results in that they are solely based on the past behavior of similar users. There is also some relation to query relaxation (e.g. [11]). Query relaxation addresses a different problem: extending the original query when there are not enough matching results.

The remainder of this paper is structured as follows. In Section 2, we provide a taxonomy of methods for producing YMAL results. In Section 3, we focus on a method that uses the current database instance and query results, while in Section 4, we present some preliminary results of this method. Section 5 summarizes related work and Section 6 concludes the paper.

2. YMAL RESULTS

Assume a database system $\mathcal{D}$ and a set of users $\mathcal{U}$ interacting with it by posing traditional select-project-join (SPJ) queries. Given a user $u \in \mathcal{U}$ and a query q, a typical database system returns a set of results $\mathcal{R}(q)$ in the form of tuples, possibly produced by joining several relations of $\mathcal{D}$. Besides $\mathcal{R}(q)$, we would like to locate and recommend to u a set of tuples that may also be of interest to the user. We call this set of tuples *"You May Also Like"* tuples or YMAL results for short. We denote this set as $\text{YMAL}(q, u)$. As a running example, we shall use the movies database shown in Figure 1.

In this paper, we propose a number of approaches to compute YMAL results. These approaches fall into three main categories:

1. *Current-state approaches*, that exploit the content and schema of the current query result and database instance.

PersDB 2009

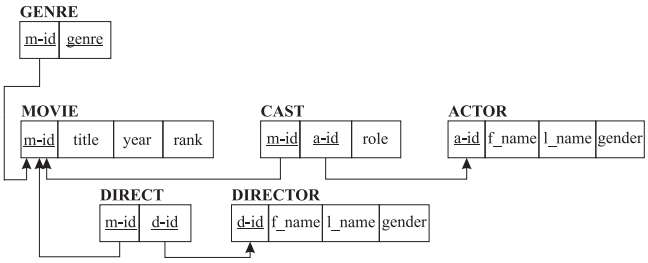

Figure 1: Movies database schema example.

2. *History-based approaches*, that exploit the history of previously submitted queries to the database system, e.g. by using query logs.

3. *External sources approaches*, that exploit resources external to the database, such as related published results and reports, relevant web pages, thesaurus or ontologies.

Figure 2 depicts a taxonomy of YMAL computation techniques. Next, we focus on each of these categories separately and present challenges and strategies towards the computation of YMAL results.

Current-state Approaches: We assume first, that there is no other information available other than a query q posed by a user u and its result $\mathcal{R}(q)$. Then, YMAL results can be computed based on either (i) *local analysis* of the intrinsic properties of the result $\mathcal{R}(q)$ or (ii) *global analysis* of the properties of the database $\mathcal{D}$. In both cases, we can exploit (i) the *content* and/or (ii) the *schema* of $\mathcal{R}(q)$ or $\mathcal{D}$ respectively.

There are many directions for computing YMAL results along these axes. For instance, in the local analysis approach, after $\mathcal{R}(q)$ has been computed, we examine the content of its tuples to locate common information patterns appearing in many of them. We then employ such information to retrieve and recommend tuples of the database that do not belong in $\mathcal{R}(q)$ but exhibit similar behavior. For example, assume that u poses a query to retrieve movie titles in which *Morgan Freeman* stars. Since *Morgan Freeman* often acts in detective roles, the relative attribute value *detective* appears many times in the result. Therefore, we recommend to u a number of movie titles in which other actors play the role of detective. Another option, in a schema-based approach, is to expand the tuples of the result through joins. Intuitively, in this way, we add extra, possibly useful information to the result and search for common patterns in the expanded result tuples. In our example, instead of presenting to u only the titles of *Morgan Freeman*'s movies, we may enhance the result with information about the corresponding genres. Based on the most frequent genres appearing in these expanded tuples, we recommend to u other movies of those genres.

In the global analysis case, we base YMAL computation on properties of $\mathcal{D}$. In the content approach, we rely on the correlation of specific attribute values as well as their selectivities. For example, when querying for *Walter Matthau* movies, we may also recommend a number of *Jack Lemmon* movies, since these two actors often star together. Correlation among relations can be used to direct the expansion of tuples in $\mathcal{R}(q)$ in a schema-based view of the problem.

Hybrid methods can also be applied by combining local and global analysis or content and schema information when processing a query result. Table 1 shows a taxonomy of the current-state approaches. We will examine further details of current-state approaches in Section 3.

History-based Approaches: History-based approaches assume that there is available information about the previous interactions of the users with the database, similar to traditional recommenders. In this respect, there are two alternatives: one could either log the results of the queries or the queries themselves. Technically, the two approaches are not equivalent, since the result of each query depends on the database instance, thus, the result of executing a logged query in the current database instance may differ significantly from the original result. Since logging results imposes significant overheads, for simplicity, in this position paper, we opt for logging queries.

Given a set of queries $\mathcal{Q}$ and a set of users $\mathcal{U}$, the utility function f: $\mathcal{Q} \times \mathcal{U} \rightarrow \mathcal{N}$, where $\mathcal{N}$ is a totally ordered set, measures the usefulness of a query $q \in \mathcal{Q}$ to a user $u \in \mathcal{U}$. We assume that the utility $f(q, u)$ is equal to the number of times user u has posed the query q.

Following the usual classification of recommendation systems, we distinguish between two different approaches: (i) *query-based* YMAL results (similar to *content-based* recommendations) and (ii) *user-based* YMAL results (similar to *collaborative* recommendations).

In the query-based approach, when a user u poses a query q, YMAL(q, u) includes results of the logged queries $q_i \in \mathcal{Q}$, that are the most similar to q, according to some similarity function $sim_q(q_i, q_j)$ between queries. For example, we may use:

$$sim_q(q_i, q_j) = \left|\mathcal{R}(q_i) \bigcap \mathcal{R}(q_j)\right|.$$

Using the utility function, we can represent each query q as a vector $(f(q, u_1), f(q, u_2), \ldots f(q, u_{|U|}))$. Then, for example, we can use as similarity, the inner product:

$$sim_q(q_i, q_j) = \sum_{k=1}^{|U|} f(q_i, u_k) f(q_j, u_k)$$

In the user-based approach, when a user u poses a query q, YMAL(q, u) includes results of queries posed by those users $u_j \in \mathcal{U}$ that exhibit the most similar behavior to u. Similar users are located via a similarity function $sim_u(u_i, u_j)$ between users, such as:

$$sim_u(u_i, u_j) = \left|\mathcal{Q}(u_i) \bigcap \mathcal{Q}(u_j)\right|$$

where $\mathcal{Q}(u_i)$ is the set of queries posed by u_i. Analogously to the queries, using the utility function, we can represent each user u as a vector $(f(q_1, u), f(q_2, u), \ldots f(q_{|Q|}, u))$. Then, for example, we can use as similarity, the inner product:

$$sim_u(u_i, u_j) = \sum_{k=1}^{|Q|} f(q_k, u_i) f(q_k, u_j)$$

In a hybrid approach, we present to u the results of the most similar queries to q out of those that were posed by similar users. Table 2 synopsizes history-based approaches.

Finally, we note that there is an important temporal dimension that needs to be considered. It is often the case that recent queries reflect better the current trends and interests

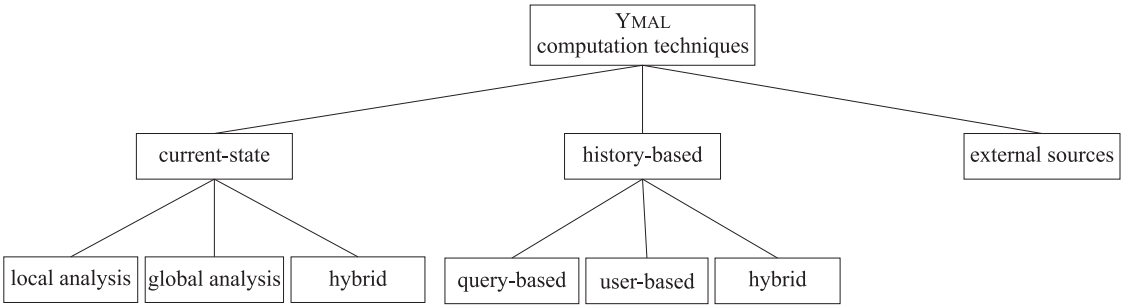

Figure 2: A taxonomy of YMAL computation techniques.

Table 1: A taxonomy of current-state approaches.

	Local analysis	**Global analysis**	**Hybrid analysis**
Content-based	Most frequent values in $\mathcal{R}(q)$	Most correlated values in $\mathcal{D}$	Combine frequent and correlated values
Schema-based	Direct joins through frequencies of values in expanded $\mathcal{R}(q)$	Direct joins through correlations among relations in $\mathcal{D}$	Direct joins through frequencies in expanded $\mathcal{R}(q)$ and correlations among relations in $\mathcal{D}$

of users, thus an aging scheme that gradually degrades the importance of queries in the log is into place.

External Sources Approaches: Up to now, we have discuss how we can locate and recommend YMAL results by exploiting intrinsic information of the database, such as correlation among attribute values and relations themselves. However, there are cases where relationships among data items are not captured in the database, even if present. Nowadays, a plethora of useful, well-organized information is available over the Web in the form of articles, reports and reviews in collectively maintained knowledge repositories, such as Wikipedia[1] and LibraryThing[2]. Information retrieved from such external sources can also be used for the computation of YMAL results.

For example, assume the database schema of Figure 1 and a query about *Sofia Coppola* movies. Using external information, we could recommend a number of *Francis Ford Coppola*'s movies, since he is the father of *Sofia Coppola*, a relationship that is not reflected in the schema. As another example, consider that a user poses a query that retrieves movies of various directors. Using an external source, interesting information may be inferred, e.g. that most of these directors are Asian. In this case, we could recommend other movies by Asian directors. Note that, the origin of directors cannot be found in the schema, so this correlation can be found only through external sources.

3. CURRENT-STATE APPROACHES

Current-state approaches explore the results $\mathcal{R}(q)$ of an SPJ query q to direct the computation of YMAL results for a user u. In this section, we will further examine such approaches and their application. First, we will focus on local analysis methods and then on global analysis ones.

[1]http://www.wikipedia.org
[2]http://www.librarything.com

Local Analysis: During local analysis of a query result $\mathcal{R}(q)$, we aim at discovering interesting patterns that we will later exploit to recommend YMAL results. Such patterns can be either found in the tuples of $\mathcal{R}(q)$ (content-based approach) or in the extended tuples produced by joining the tuples of the result with other tuples of the database (schema-based approach).

In this work, we view interesting patterns as frequently appearing attribute values, or combinations thereof. To quantify attribute values appearances in $\mathcal{R}(q)$, we define the *value-frequencies* matrix $M_{\mathcal{R}(q)}$. There is one row in $M_{\mathcal{R}(q)}$ for each attribute $A_1, \ldots, A_m$ of $\mathcal{R}(q)$ and one column for each distinct attribute value $v_1, \ldots, v_n$ appearing in its tuples. $M_{\mathcal{R}(q)}(i, j)$ contains the number of occurrences of v_j for A_i in $\mathcal{R}(q)$. As an example, consider a user that is interested in movie titles staring *Lee Phelps*. In Figure 3, we see a part of the related value-frequencies matrix. For ease of presentation, we depict only the five most frequent values for the attribute *Role*.

	Policeman	Detective	Cop	Bartender	Guard
Role	36	24	23	22	13

Figure 3: Value-frequencies matrix example.

Given a query q and the corresponding value-frequencies matrix $M_{\mathcal{R}(q)}$, our goal is to present to the user a set of YMAL results with cardinality p. Such results are computed with regards to the most frequent attribute values in $\mathcal{R}(q)$ as specified by $M_{\mathcal{R}(q)}$. In particular, we locate the k elements in $M_{\mathcal{R}(q)}$ with the highest values and, for each such element, we construct an appropriate SPJ query to retrieve interesting results. For clarity in notation, we also consider the matrix $M'_{\mathcal{R}(q)}$ for which $M'_{\mathcal{R}(q)}(i, j) = M_{\mathcal{R}(q)}(i, j)$ if $M_{\mathcal{R}(q)}(i, j)$ belongs to the k, $k > 0$, most frequent attribute values and $M'_{\mathcal{R}(q)}(i, j) = 0$ otherwise. Each element contributes a num-

Table 2: A taxonomy of history-based approaches.

Query-based approaches	**User-based approaches**	**Hybrid approaches**
Similarities among queries	Similarities among users	Similarities among both users and queries

ber of YMAL results based on the function F:

$$F(i,j) = \frac{M'_{\mathcal{R}(q)}(i,j)}{\sum_i \sum_j M'_{\mathcal{R}(q)}(i,j)} \cdot p$$

For the above example, let $p = 10$ and $k = 2$. Then, based on F, we will recommend six movies containing the role *Policeman* and four ones containing the role *Detective*.

When a schema-based approach is followed, we expand $\mathcal{R}(q)$ prior to constructing M, so that, additional interesting common patterns can be discovered. In our example, if we expand $\mathcal{R}(q)$ towards the *GENRE* relation, we discover other interesting information, such as, this actor mainly stars in *Drama* movies. For a specific query q and its result $\mathcal{R}(q)$, we expand $\mathcal{R}(q)$ towards all possible directions through the same number of join operations and construct the corresponding value-frequency matrices. Then, we select the matrix containing the most frequent value appearances and proceed with the YMAL computation as before, based on this matrix alone. We consider that patterns discovered after one join operation are more relevant than patterns discovered after two join operations and so on. For this reason, when selecting which matrix to use, we also take into account the corresponding number of needed join operations for each matrix and favor matrices with fewer joins.

Global Analysis: Global analysis aims at taking advantage of database properties during the recommendation of YMAL results. In this work, we consider that YMAL computation is guided by certain statistics maintained for our database $\mathcal{D}$. Such statistics include the correlation among the various attribute values of the database and the correlation among the various relations in $\mathcal{D}$.

We define the *value-correlation* matrix V that captures the correlation between pairs of distinct attribute values in each database relation. V is a $(x \times y \times y)$ matrix, where x is the number of attributes in $\mathcal{D}$ and y is the number of distinct attribute values. Given a query q, we consult the matrix V to locate attribute values correlated to the selection predicates of q. We select the k attribute values that are the most correlated to q. The more correlated such a value is, the more YMAL results it will contribute. This can be calibrated via the use of a function similar to F that is defined based on V instead of M.

The correlation among the relations of $\mathcal{D}$ can be used to direct the joining operations in a schema-based approach. Such correlations are captured in the $(z \times z)$ *relation-correlation* matrix A, where z is the number of relations in $\mathcal{D}$. For example, assuming that the relation *CAST* is strongly correlated with the relation *ACTOR*, then, when querying for specific actor names we could present roles that these actors have portrayed.

Hybrid Analysis: Each of the methods described above exploits different properties: local analysis is based on the actual results of a query, while global analysis depends on the whole database. As a next step, we can combine the advantages of both approaches in a hybrid method that exploits both local and global properties. A hybrid content-based approach is to use attribute values that are both frequent and strongly correlated. To calibrate the importance of each factor, we rely on a weighted function. Similarly, in a schema-based approach, the joining of $\mathcal{R}(q)$ with other database relations is directed using both the correlations among the relations of the database, as well as, frequent appearances of attribute values in those relations.

Table 3: Relation cardinalities.

Relation name	**Cardinality**
GENRE	109.261
MOVIE	70.266
CAST	1.266.462
ACTOR	322.467
DIRECT	109.226
DIRECTOR	152.533

4. EVALUATION

Our evaluation objective is to demonstrate the effectiveness of our approach for current-state methods. In particular, to show the usefulness of YMAL results, we will present for representative queries both their $\mathcal{R}(q)$ and YMAL(q, u) results. For our experiments, we use a real movie dataset [1]. The schema of the database is depicted in Figure 1, while Table 3 shows the cardinalities of the relations.

Local analysis: To experiment with local analysis YMAL computation, we use the following query:

```
q1 :  select  *
      from    MOVIE, CAST, ACTOR
      where   MOVIE.m-id = CAST.m-id and
              CAST.a-id = ACTOR.a-id and
              ACTOR.f_name = 'Lee' and
              ACTOR.l_name = 'Phelps';
```

A subset of $\mathcal{R}(q_1)$ is shown in Figure 4. The part (*372839 - Lee - Phelps - M*) is common in all tuples of $\mathcal{R}(q_1)$ due to the query selection conditions, therefore, we exclude its attribute values from the construction of $M_{\mathcal{R}(q_1)}$. For $k = 2$, the two most common values in the 394 tuples of $\mathcal{R}(q_1)$ are the values *Policeman* and *Detective* of the attribute *Role* (Figure 3). For $p = 3$, *Policeman* and *Detective* contribute two and one YMAL results respectively, when the content-based approach is followed (Figure 4).

$\mathcal{R}(q_1)$ **results**										
m-id	**title**	**year**	**rank**	**a-id**	**m-id**	**role**	**a-id**	**f_name**	**l_name**	**gender**
4619	Abbott and Costello in Hollywood	1945	5.6	372839	4619	Detective	372839	Lee	Phelps	M
218015	Money to Loan	1939	6.3	372839	218015	Policeman	372839	Lee	Phelps	M
46730	Bride Came C.O.D., The	1941	6.8	372839	46730	Policeman	372839	Lee	Phelps	M
330384	Thin Man Goes Home, The	1945	7	372839	330384	Policeman	372839	Lee	Phelps	M
31821	Beast From 20,000 Fathoms, The	1953	6.3	372839	31821	Cop	372839	Lee	Phelps	M
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ymal(q_1, u) **results: Content-based Approach**										
m-id	**title**	**year**	**rank**	**a-id**	**m-id**	**role**	**a-id**	**f_name**	**l_name**	**gender**
323813	Talented Mr. Ripley, The	1999	7	411152	323813	**Policeman**	411152	Manuel	Ruffini	M
195807	Love Letter	1998	8.2	420874	195807	**Policeman**	420874	Bsaku	Satoh	M
155070	I, Robot	2004	6.9	297823	155070	**Detective**	297823	Craig	March	M

Figure 4: Content-based local analysis for q_1.

When the schema-based approach is employed, the possible directions for expansion are the relations *GENRE* and *DIRECT*. The most common patterns are observed when $\mathcal{R}(q_1)$ is expanded towards the *GENRE* relation and are the values *Drama* and *Comedy* that appear 216 and 120 times respectively. The expanded tuples of $\mathcal{R}(q_1)$ and the recommended YMAL results in this case, for $k = 2$ and $p = 3$, are the ones shown in Figure 5.

Global analysis: To experiment with content-based, global analysis YMAL computation, we use the following query that retrieves romance movies:

```
q2 :  select  *
      from    MOVIE, GENRE
      where   MOVIE.m-id = GENRE.m-id and
              GENRE.genre = 'Romance';
```

In our dataset, the attribute value *Romance* appears along with other genres for the same movies as many times as shown below:

Drama (2801)	Comedy (2398)	Musical (538)
Action (351)	Adventure (323)	Thriller (267)
Fantasy (263)	Crime (263)	Family (234)
War (199)	Short (162)	Mystery (131)

We use again $k = 2$ and $p = 3$. As we can see above, the two mostly correlated values to *Romance* are *Drama* and *Comedy*. Therefore, two drama movies and a comedy one will be recommended. We omit the relative figure due to space limitations.

Consider now the query:

```
q3 :  select  *
      from    MOVIE, DIRECT, DIRECTOR
      where   MOVIE.m-id = DIRECT.m-id and
              DIRECT.d-id = DIRECTOR.d-id and
              DIRECTOR.f_name = 'Steven' and
              DIRECTOR.l_name = 'Spielberg';
```

In our database, the relation that is most correlated to *MOVIE* is *GENRE*. Therefore, when computing YMAL results using the schema-based approach, we enhance $\mathcal{R}(q_3)$ with information about the genres of Steven Spielberg's movies.

5. RELATED WORK

In this paper, we have proposed extending relational database systems with recommendation functionality in the form of YMAL results. In general, recommendation methods are categorized into: (i) *content-based*, that recommend items similar to those the user has preferred in the past (e.g. [16, 14]), (ii) *collaborative*, that recommend items that similar users have liked in the past (e.g. [10, 6]) and (iii) *hybrid*, that combine content-based and collaborative ones (e.g. [4, 5]). [3] provides a comprehensive survey of the current generation of recommendation systems. Several extensions have also been proposed, such as employing multi-criteria ratings [2] and extending the typical recommendation systems beyond the two dimensions of users and items to include further contextual information [15].

In terms of relating recommendations and databases, there are two very recent works [12, 7]. [12] provides a general framework and an engine for the declarative specification of the recommendation process over structured data. In this paper, we focus on the specific recommendation process of computing YMAL results related to a specific user query. The recommendation process in [12] is specified through a series of interconnected operators, which apart from the traditional relational operators, includes also a number of specific to the recommendation process operators, such as the *recommend* operator, that recommends a set of tuples of a specific relation with regards to their relationship with the tuples of another relation. In our approach, we rely on typical relational algebra operators. In [7], the focus is on recommending SQL queries to the users of a database. The proposed method is based on "session summaries", i.e. the set of tuples that contributed to some result for queries imposed by the user in the current session. Given a session summary for a user u, the purpose is to compute a prediction summary of tuples that may be of interest to the user and then locate a number of queries able to retrieve the tuples in it. The prediction summary is computed based on the session summary of u and other users similar to u. The suggested queries are retrieved from a pool of past queries submitted by the users. Recommendations in [7] are more restricted than our YMAL results, since they are solely based on the past behavior of similar users.

There is also some relation with *query reformulation*, where a query is relaxed or restricted when the number of results of the original query are too few or too many respectively, and

Expanded $\mathcal{R}(q_1)$ results												
m-id	**title**	**year**	**rank**	**a-id**	**m-id**	**role**	**a-id**	**f_name**	**l_name**	**gender**	**m-id**	**genre**
4619	Abbott and Costello in Hollywood	1945	5.6	372839	4619	Detective	372839	Lee	Phelps	M	4619	Comedy
218015	Money to Loan	1939	6.3	372839	218015	Policeman	372839	Lee	Phelps	M	218015	Drama
46730	Bride Came C.O.D., The	1941	6.8	372839	46730	Policeman	372839	Lee	Phelps	M	46730	Comedy
330384	Thin Man Goes Home, The	1945	7	372839	330384	Policeman	372839	Lee	Phelps	M	330384	Drama
31821	Beast From 20,000 Fathoms, The	1953	6.3	372839	31821	Cop	372839	Lee	Phelps	M	31821	Sci-Fi
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ymal(q_1, u) results: Schema-based Approach												
256348	Pinocchio	2002	4	36868	256348	Pinocchio	36868	Roberto	Benigni	M	256348	**Comedy**
34306	Berlin Berlin	1998	9.7	426546	34306	Sammy	426546	Asad	Schwarz	M	34306	**Drama**
218345	Monster	200	7.4	91661	218345	Newscaster	91661	Jim R.	Coleman	M	218345	**Drama**

Figure 5: Schema-based local analysis for q_1.

with *automatic result ranking*, where the results of a query are restricted to the top-ranked ones, when the number of results of the original query is very large. Such approaches usually use the frequency of values of specific attributes in the database to restrict or expand the result set, which is also the basic idea of our *current-state* approaches in computing YMAL results.

A framework for relaxing queries involving numerical conditions in selection and join predicates is proposed in [11], while the relaxation algorithm proposed in [9] produces a relaxed query for a given query range and a desired cardinality of the result set. To estimate the result size, the algorithm uses multi-dimensional histograms. [13] studies the problem of query refinement through transformations of the selection query predicates. Transformations aim at either relaxing the query predicates in order to increase the result cardinality or contracting the query predicates in order to decrease the result cardinality. A systematic approach for the automatic ranking of query results is proposed in [8]. To estimate the rank of a result tuple, they use both workload and data analysis; this is similar to the *history-based* and *current-state* approaches respectively. The main difference is that we consider recommending tuples not in the result set, whereas [8] consider ranking the tuples in the result set. The operator *frequent co-occurring term* [17] can also be used to direct query refinement in relational keyword search. Given a keyword query q, this operator returns a set of terms that appear frequently in the result of q and none of them is contained in q. These are the terms that can be employed by users to refine their queries.

6. CONCLUSIONS

In this paper, we presented a first approach to computing YMAL results and organized the various alternatives into categories. There is a number of open issues for research. In terms of current-state YMAL computation, we could exploit information about the importance of each relation attribute. For history-based YMAL computation, we could explore techniques based on the logging of query results or statistics about query results instead of logging queries. Also, in this work, we looked into selecting YMAL results based on similarity. We could also consider other criteria, such as presenting to the users novel, fresh or diverse information [18].

Acknowledgment

We would like to thank Yufei Tao for providing the dataset used in this work.

7. REFERENCES

[1] *Internet Movies Database*. Available at www.imdb.com.
[2] G. Adomavicius and Y. Kwon. New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55, 2007.
[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
[4] M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
[5] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.
[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.
[7] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, 2009.
[8] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. Database Syst.*, 31(3):1134–1168, 2006.
[9] A. Kadlag, A. V. Wanjari, J. Freire, and J. R. Haritsa. Supporting exploratory queries in databases. In *DASFAA*, pages 594–605, 2004.
[10] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.
[11] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *VLDB*, pages 199–210, 2006.
[12] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Flexrecs: Expressing and combining flexible recommendations. In *SIGMOD*, 2009.
[13] C. Mishra and N. Koudas. Interactive query refinement. In *EDBT*, pages 862–873, 2009.
[14] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. *CoRR*, cs.DL/9902011, 1999.
[15] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Trans. Knowl. Data Eng.*, 20(11), 2008.
[16] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.
[17] Y. Tao and J. X. Yu. Finding frequent co-occurring terms in relational keyword search. In *EDBT*, pages 839–850, 2009.
[18] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, pages 368–378, 2009.