Query Recommendations for Interactive Database Exploration

Work by
Gloria Chatzopoulou*, UC Riverside
Magdalini Eirinaki, San Jose State Univ
Neoklis Polyzotis, UC Santa Cruz

Presented by
Wuzhou Zhang

*work performed while author affiliated with UC Santa Cruz
Motivation

- Scientific community rely increasingly on *relational databases*

- Users, with *diverse* information needs, employ a web-based client to issue *SQL queries* for data analysis

- Users may find it hard to write *interesting* queries:
  - They are not SQL experts
  - They are not aware of all parts of the database

**Goal**: Assist users in finding useful information
Proposed Solution

- Recommend queries to users based on the queries of other similar users
- Inspiration: Collaborative Filtering
- Example: Movie Recommendations

If Alice and Bob both like movie X and Alice likes movie Y
If Alice and Bob both query data X and Alice queries data Y
then
then
Bob is likely to be interested in seeing movie Y
Bob is likely to be interested in querying data Y
System Architecture

How to generate meaningful queries?

How to define similar users?

Which parts of the database are interesting to the user?
Conceptual Framework

Current User

Session Representation

Session Summaries

Similarities

sim(S0,S1)

sim(S0,S2)

... sim(S0,Sn)

Prediction

Past Users

Session Representation

Top-N List of Recommendations

Recommendations Generator

Query Log

Predicted Summary
## Session Representation

### Binary Weighting Scheme
- **q1** = <1,1,0,0,0,1,1,1,0>
- **q2** = <0,1,0,0,0,1,1,1,0>
- **s0** = <1,2,0,0,1,2,2,0>

### Result Weighting Scheme
- **q1** = <0.33,0.33,0.33,0.33,0.33,0.33,0.33,0.33,0.33>
- **q2** = <0.50,0.50,0.50,0.50,0.50,0.50,0.50,0.50,0.50>
- **s0** = <0.33,0.83,0.33,0.83,0.33,0.83,0.83,0.83,0.83>

### Queries
- **q1**: $R \cap_{R.a=L.a} L$
- **q2**: $\sigma_{R.b=4}(F \cap_{R.a=L.a} L)$
Similarity Function

- Vector-space similarity functions can be used
  - Cosine Similarity

\[
\text{sim}(S_i, S_j) = \frac{S_i S_j}{||S_i||_2 ||S_j||_2}
\]

- High similarity means that users are most likely interested in the same parts of the database
Predicted Summary

\[ S_0^{\text{pred}} = \alpha \cdot S_0 + (1 - \alpha) \cdot \frac{\sum_{1 \leq i \leq h} \text{sim}(S_0, S_i) \cdot S_i}{\sum_{1 \leq i \leq h} \text{sim}(S_0, S_i)} \]

\( \alpha \): the mixing factor
Generating Recommendations

Using queries of past users

Query Log Data

\[ q_1 = <1,0,0,...,0> \]
\[ q_2 = <0,1,0,...,0> \]
\[ \ldots \]
\[ q_N = <1,0,1,...,1> \]

\[ u^{\text{pred}} = <1,0,0,...,0> \]

**Similarity Function**

\[ \text{rank}(q_1) = \text{sim}(u^{\text{pred}}, q_1) \]
\[ \text{rank}(q_2) = \text{sim}(u^{\text{pred}}, q_2) \]
\[ \ldots \]
\[ \text{rank}(q_N) = \text{sim}(u^{\text{pred}}, q_N) \]

Return Top-K Queries
Experimental Setup

- **SkyServer Dataset**

- **Evaluation Metrics: Precision and Recall**
  - **High precision**: most witnesses of the recommended query are witnesses in the actual query.
  - **High Recall**: most witnesses of the actual query are witnesses in the recommended query.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database size</td>
<td>2.6TB</td>
</tr>
<tr>
<td>#Sessions</td>
<td>720</td>
</tr>
<tr>
<td>#Queries</td>
<td>6713</td>
</tr>
<tr>
<td>#Distinct queries</td>
<td>4037</td>
</tr>
<tr>
<td>#Distinct witnesses</td>
<td>13,602,430</td>
</tr>
<tr>
<td>Avg. number of queries per session</td>
<td>9.3</td>
</tr>
<tr>
<td>Min. number of queries per session</td>
<td>3</td>
</tr>
</tbody>
</table>
Binary vs Result Weighting Schemes

Binary outperforms Result Weighting Scheme
Effect of mixing factor $\alpha$

Hybrid Collaborative Filtering yields better results
Top-3 vs Top-5 Binary Weights

The bigger recommendation set the higher accuracy
Discussion

- **Performance** improvement
- Though we can return the same tuple, queries might be different
- Query **structure** instead of tuples retrieved
- Correlation between **sequence** of queries (causality, incremental)
- Extension: automatically import **other relations**

- Relation to our project?

**Recent work: fragment-based** (attribute ref, tables ref, join and selection predicates)
Thanks!

Reference