Using Rank Aggregation in Continuously Answering SPARQL Queries on Streaming and Quasi-static Linked Data

Shima Zahmatkesh, Emanuele Della Valle, and Daniele Dell'Aglio
DEIB - Politecnico of Milano

DEBS 2017 – Barcelona, Spain
23 June 2017
Stream Processing in Nutshell

Stream data

Windows

Stream Processing Engine

Results

Register query once and execute it continuously
Web Stream Processing

- Web Streams
- Linked Data
- Join
- Web

- High Latency
- Rate Limits
- Loosing Reactiveness

Stream Processing Engine
RDF Stream Processing (RSP) Engine

- RDF Streams
- SPARQL endpoint

Join

Results
Motivation

The cloth brand ACME wants to persuade influential Social Networks users to post commercial endorsements.

Every minute give me the ID of the users that are mentioned on Social Network in the last 10 minutes whose number of followers is greater than 100,000.

```
REGISTER STREAM <:InfluencersToContact> AS
CONSTRUCT {?user a :influentialUser}
FROM NAMED WINDOW W ON S [RANGE 10m STEP 1m]
WHERE {
    WINDOW W {?user :hasMentions ?mentionsNumber}
    SERVICE BKG {?user :hasFollowers ?followerCount}
    FILTER (?followerCount > 100,000)
}
```
Problem Definition

- RDF Streams
- SPARQL endpoint
- Local Replica
- Web

- Define Refresh Budget to control reactivity
- Data become stale if not refreshed
- Correct vs approximate answer
- Correct vs approximate answer
- Windows
- Data become stale if not refreshed
- Data become stale if not refreshed

RSP engine

Join

Results
Problem Definition

- RDF Streams
- SPARQL endpoint
- Local Replica
- Maintenance Policy
- Join
- Results
- Web
- Windows
- RSP engine
- Replication
- Policy
ACQUA, ACQUA.F Frameworks

**WINDOW clause**
Stream data

**SERVICE clause**
ACQUA: without FILTER
ACQUA.F: with FILTER Clause

**JOIN**
Proposer

1
Local Replica

WSJ: Filter out mappings that are not involved in current evaluation

Candidate set

C

**Ranker**
Maintainer

2
E

Elected set: top $\gamma$ mappings of Candidate set

**Maintainer**

3

Soheila Dehghanzadeh, et al., Approximate Continuous Query Answering over Streams and Dynamic Linked Data Sets, ICWE 2015.
Shima Zahmatkesh, et al., When a FILTER Makes the Difference in Continuously Answering SPARQL Queries on Streaming and Quasi-Static Linked Data, ICWE 2016.
Rankers

- **LRU**
  - Use Least-Recently Used (LRU) cache replacement algorithm
  - The less recently a mapping have been refreshed in a query, the higher is its rank.

- **Filter Update Policy**
  - For each mapping in the replica:
    - Computes how close is the value associate to the variable of the mapping to the Filtering Threshold used in Filter clause.
    - Arrange mappings in ascending order.

<table>
<thead>
<tr>
<th>User</th>
<th>Last Update Time</th>
<th>LRU policy</th>
<th>#followers</th>
<th>Filter Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>8</td>
<td>1</td>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>2</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>Carol</td>
<td>14</td>
<td>3</td>
<td>95</td>
<td>1</td>
</tr>
</tbody>
</table>

Filtering Threshold = 100
Rank Aggregation

- Fairly take into account the opinions of different algorithms.
- Combine the ranking lists obtained from different algorithms by computing aggregated score

\[ score_{agg} = \alpha \cdot score_{list-1} + (1 - \alpha) \cdot score_{list-2} \]

<table>
<thead>
<tr>
<th>User</th>
<th>Score</th>
<th>User</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>0.8</td>
<td>Bob</td>
<td>0.9</td>
</tr>
<tr>
<td>Bob</td>
<td>0.7</td>
<td>David</td>
<td>0.8</td>
</tr>
<tr>
<td>Carol</td>
<td>0.5</td>
<td>Alice</td>
<td>0.7</td>
</tr>
<tr>
<td>David</td>
<td>0.4</td>
<td>Eve</td>
<td>0.4</td>
</tr>
<tr>
<td>Eve</td>
<td>0.1</td>
<td>Carol</td>
<td>0.1</td>
</tr>
</tbody>
</table>

\[ T = 0.5 \cdot 0.8 + 0.5 \cdot 0.9 = 0.85 \]

\[ \alpha = 0.5 \]
Rank Aggregation

- Fairly take into account the opinions of different algorithms.
- Combine the ranking lists obtained from different algorithms by computing aggregated score

\[ score_{agg} = \alpha \times score_{list-1} + (1 - \alpha) \times score_{list-2} \]

\[ T = 0.5 \times 0.7 + 0.5 \times 0.8 = 0.75 \]
Experimental Evaluation

• Data Sets
  • Streaming data, and realistic background data from real data of Twitter

• Query
  • Contains WINDOW, SERVICE, and FILTER clauses
  • Generate correct answer of the query by an Oracle

• KPIs
  • Measure diversity of the set generated by the query and correct answer
  • Compute cumulative errors over evaluations
Experimental Results

For high selectivity Filter Update Policy is better than WBM

Experiment Dimension
Experimental Results

For low selectivity WBM is better than Filter Update Policy.
Experimental Results

- **Filter Update**
- **ACQUA (WBM)**
- **Best Result**
- **Rank Aggregation**

Comparable to Best Result

- **Cumulative Jaccard Distance**
- **Selectivity of Filter Clause**

Experiment Dimension
Conclusion

• Problem of continuously evaluating queries over data stream and background data.

• The results of experiments show that proposed policies have the same accuracy of the best result achieved without using any assumption.

• They also show that the proposed policies are not sensitive to the value of alpha used in rank aggregation formula.
Future works

• Broaden the class of queries
  • Multiple filtering
  • Filtering condition formulated as a ranking clause

• Pushing the FILTER clause into the SERVICE clause and considering caching instead of local replica

• Study the effect of different trends in the data
Thank you!
Any Question?

Using Rank Aggregation in Continuously Answering SPARQL Queries on Streaming and Quasi-static Linked Data

Shima Zahmatkesh
shima.zahmatkesh@polimi.it
DEIB - Politecnico of Milano