Recommender Systems and SPARQL: More than a Shotgun Wedding?

8th Alberto Mendelzon International Workshop
Cartagena, Colombia
(AMW 2014)
Overview

1. Motivation
2. RecSPARQL
3. Experiments
4. Summary
Motivation

- RDF’s flexibility
  - Recommendation domain
    - Users, Items, Ratings
    - Consumption relationship
Motivation

- How to get recommendations from RDF-graphs: SPARQL?
  - Retrieval of explicit data from RDF-graphs
  - Graph pattern matching
  - Flexible
  - No possible to express fuzzy queries
    - Similarity

SPARQL \{ RDF

Source: http://courses.ischool.berkeley.edu/i253/f11/
Motivation

- How to get recommendations from RDF-graphs: Recommender Systems
  - Predicts a degree of preference for a user towards a set of non-consumed items
  - Based on Information Retrieval techniques

Source: http://courses.ischool.berkeley.edu/i253/f11/
Motivation

- A straightforward approach

- Classic recommender
  - **Input**: users, items, ratings
  - **Output**: users, recommended items, predicted rating

Recommendations
Motivation

- Example: lack of flexibility
  - Collaborative filtering approach
    - Recommendations from similar users

Extraction / Pre-processing

Recommendation System (CF)

Users

Items

<table>
<thead>
<tr>
<th>User</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>...</th>
<th>Item m</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>User 2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>User 5</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User n</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Motivation

- **Example: lack of flexibility**
  - Collaborative filtering approach
    - Recommendations from similar users

### Items

<table>
<thead>
<tr>
<th>User</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>...</th>
<th>Item m</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>User 2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>User 5</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User n</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

(B) Recommendation System (CF)
Motivation

- **Example: lack of flexibility**
  - Collaborative approach
    - Recommendations from similar users
  - Customization
    - Neighbors geographically close
    - Age of neighbors differ by $\delta$
    - Speak same languages
    - etc....
Motivation

- **Example: lack of flexibility**
  - Collaborative approach
    - Recommendations from similar users
  - **Customization**
    - Neighbors geographically close
    - Age of neighbors differ by $\delta$
    - Speak same languages
    - etc...
  - **Where is the problem?**
    - Fixed recommender model
1. Motivation
2. RecSPARQL
3. Experiments
4. Summary
Our Approach

**RecSPARQL: Recommendations + SPARQL**

- Extension of SPARQL 1.1
  - Consistent mechanism to select parts of the graph
  - Flexibility

**RecSesame**

- Platform for the evaluation of RecSPARQL queries
  - Based on Sesame’s framework
  - Cache System
RecSPARQL in a nutshell

RECOMMEND [Projected Variables]
USING [Recommendation Algorithm]
WHERE { [Basic Graph Pattern] }
BASED ON { [RecSPARQL Type Pattern]
   [RecSPARQL Model Building Pattern] }
RecSPARQL in a nutshell

RECOMMEND [Projected Variables]
USING [Recommendation Algorithm]
WHERE { [Basic Graph Pattern] }
BASED ON { [RecSPARQL Type Pattern]
  [RecSPARQL Model Building Pattern] }

- Must contain recommendation entities
RecSPARQL in a nutshell

RECOMMEND [Projected Variables]
USING [Recommendation Algorithm]
WHERE { [Basic Graph Pattern] }
BASED ON { [RecSPARQL Type Pattern]
 [RecSPARQL Model Building Pattern] }

- **Algorithm used**

- **Algorithms**
  - Content-based (CB)
  - Collaborative Filtering (CF)
  - Hybrid (H)
RecSPARQL in a nutshell

RECOMMEND [Projected Variables]
USING [Recommendation Algorithm]
WHERE { [Basic Graph Pattern] }
BASED ON { [RecSPARQL Type Pattern]
  [RecSPARQL Model Building Pattern] }

- Content-based
  - Genre
  - Cast
  - Director

- Collaborative filtering
  - Watched movies and ratings
  - Age
  - Geographical location
RecSPARQL in a nutshell

- RecSPARQL in a nutshell

RECOMMEND [Projected Variables]
USING [Recommendation Algorithm]
WHERE { [Basic Graph Pattern] }
BASED ON { [RecSPARQL Type Pattern]
    [RecSPARQL Model Building Pattern] }

- Similar to projection
- Makes it possible to project recommendations
- ?movie.REC
RecSPARQL’s flexibility

Filters
- Recommend only action movies

FILTER ( ?genre.REC = “action” )
RecSPARQL’s flexibility

- Filters
  - Recommend only from users whose age differ by at most 5 years

```
FILTER (abs(xsd:integer(?age) - xsd:integer(?age.REC)) <= 5)
```
RecSPARQL’s flexibility

- Much more...
  - Recommend movies watched under a certain context

```sparql
FILTER (?watchTime.REC = “weekend” && ?company.REC = “partner”) .
```

- Recommend movies whose directors have the same citizenship of the user for which we want the recommendations

- Recommend ....
Experiments

1. Motivation
2. RecSPARQL
3. Experiments
4. Summary
Experiments

- **Example: lack of flexibility**
  - Collaborative approach
    - Recommendations from similar users

- **Gradual restriction of Neighborhood:**
  - Neighbors geographically close
  - Age of neighbors differ by \( \delta \)
Experiments

- Example: lack of flexibility
  - Collaborative approach
    - Recommendations from similar users

```
FILTER (abs(xsd:integer(?age) - xsd:integer(?age.REC)) <= %K%).
FILTER (abs(xsd:integer(?zip) - xsd:integer(?zip.REC)) <= %L%)
```
Experiments

- **Beneficial**
  - Backed by our experiments

- **Customization**
  - Neighbors geographically close
  - Age of neighbors differ by $\delta$

- **Similarity among users in the neighborhood**
  - Increases
Experiments

- **Beneficial**
  - Backed by our experiments

- **Customization**
  - Neighbors geographically close
  - Age of neighbors differ by $\delta$

- **Similarity among users in the neighborhood**
  - Increases

**Users**

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>...</th>
<th>Item m</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>User 2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User 3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>User 4</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>User 5</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>User 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User n</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Items**

- MovieLens
  - Helping you find the right movies
Summary

- Tight integration of recommender systems with SPARQL
- Customizable recommendations on arbitrary RDF graphs

Future Work
- Enhance the integration of both paradigms
- Support more recommendations techniques
- Increase the expressiveness of RecSPARQL
  - Binding variables
  - Sub-queries
  - Property paths

Thanks for your attention!