Towards a Scalable Social Recommender Engine for Online Marketplaces

The Case of Apache Solr

Emanuel Lacic, Dominik Kowald, Denis Parra, Christoph Trattner
Many thanks to

Emanuel Lacic
elacic@know-center.at
Graz University of Technology
Austria

Denis Parra
dparra@ing.puc.cl
Pontificia Universidad Catolica
Chile

Christoph Trattner
ctrattner@know-center.at
Know-Center Graz
Austria
What will this talk be about?

• (Real-time) product recommendations for online marketplaces

• Scalability of recommender systems

• Utilizing social network data for the recommendations of products to people
How did this work start?

• Joint project with the Austrian start-up Blanc-Noir

• Personalized product recommender for online marketplaces based on
  – Actions in the marketplaces (e.g., Ebay, Amazon)
  – Product information
  – Social network data (e.g., Facebook, G+)
  – Filter criteria

• Provided at (near) real-time!
  ... especially if there is a lot of data
    ... together with many data updates
So now, how we have solved that issue?
What’s available out there?

- Frameworks/approaches for scalable recommendations
  - Distributed data processing
    - Apache Hadoop / Mahout (map/reduce paradigm)
  - Relational databases
    - MySQL, PostgreSQL (e.g., RecDB project)
  - Collaborative Filtering improvements
    - Matrix factorization

- Lack of a framework / approach that combines all things we need
Why Solr?

• „High-performance, full-featured text search engine library“

... but more precise ...


... which provides ....

- full-text searches (content-based)
- powerful queries (e.g., MoreLikeThis or Facets)
- (near) real-time data updates (no pre/re-calculations)
- easy schema updates (social data integration)

• Established open-source software (Apache license) with big community
Our framework
https://github.com/learning-layers/SocRec
How does the thing perform?

- Dataset of virtual world SecondLife
  - Marketplace and social data

http://Learning-Layers.eu – Scaling up Techno
What’s about the marketplace and social data features?

- Both types of data are important for the recommender quality and user coverage
What’s about the hybrids?

- The hybrid approach provides a good trade-off of recommender quality and user coverage

http://Learning-Layers-eu – Scaling up Technologies for Informal Learning in SME Clusters – layers@learning-layers.eu
What’s about the scalability?

- Recommendations can be provided in (near) real-time in both cases (with and without data update)
What we have shown!

• Apache Solr is more than a search engine!

• Actually it is a great framework to implement a **scalable recommender engine for online marketplaces**
  • Near real-time recommendations through build-in query-functions
  • Near real-time **data updates**
  • Easy integration of **social data**
    + a high-performance **full-text search engine** for free!

• Evaluation on dataset gathered from **SecondLife**
  • Different **marketplace** and **social** data features are important
  • **Hybrid approaches** produce more robust recommendations
  • It **scales**!
What do we want to do in the future?

• Online study together with BlancNoir with “real” data

• Impact of geo-spatial data

• Impact of temporal data (see WebScience track)

• Comparative study with other backend solutions (e.g., ElasticSearch)
Thank you for your attention!

Code and framework:
https://github.com/learning-layers/SocRec

Questions?

Dominik Kowald
dkowald@know-center.at
Know-Center
Graz University of Technology (Austria)
Backup
Short hands-on session

• Collaborative Filtering

// Find similar users based on purchased items using
// Solr’s facet queries
/select?q=id:("some_product_1")+OR+id:("some_product_2")&
   facet=true&facet.field=my_users_field
// Find items purchased by those similar users that are
// new to the target user
/select?q=my_users_field:("user_1"^5+OR+"user_2"^3)&
   fq:-id:("some_product_1")+OR+-id:("some_product_2")

• Content-Based

/select?q=id:("some_product_id")&mlt=true&
   mlt.fl=description
## SecondLife dataset

<table>
<thead>
<tr>
<th>Marketplace (Market)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
<td>72,822</td>
</tr>
<tr>
<td>Number of purchases</td>
<td>265,274</td>
</tr>
<tr>
<td>Mean number of purchases per user</td>
<td>3.64</td>
</tr>
<tr>
<td>Number of products</td>
<td>122,360</td>
</tr>
<tr>
<td>Mean number of purchases per products</td>
<td>2.17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Online Social Network (Social)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
<td>64,500</td>
</tr>
<tr>
<td>Number of likes</td>
<td>1,492,028</td>
</tr>
<tr>
<td>Number of comments</td>
<td>347,755</td>
</tr>
<tr>
<td>Mean number of likes per user</td>
<td>14.91</td>
</tr>
<tr>
<td>Mean number of comments per user</td>
<td>3.47</td>
</tr>
<tr>
<td>Number of groups</td>
<td>260,137</td>
</tr>
<tr>
<td>Mean number of groups per user</td>
<td>8.91</td>
</tr>
<tr>
<td>Number of interests</td>
<td>88,371</td>
</tr>
<tr>
<td>Mean number of interests per user</td>
<td>1.57</td>
</tr>
</tbody>
</table>
How to Use the Engine?

- Implement and run a new recommender

```java
// Implement the recommender strategy
public interface RecommendStrategy {
    public RecommendResponse recommend(RecommendQuery q,
                                        Integer maxResults, SolrServer SolrServer);
}

// Run the new recommender strategy
RecommendStrategy strategyToUse = new MyStrategyImpl();
Filter filter = new ContentFilter(); // optional
RecommendationService.getRecommendations("some_user",
                                         "some_product", 10, filter, strategyToUse);
```
Recommendation Algorithms implemented in the Engine

- **MostPopular (MP)**
  - Recommends for any user the most purchased items

- **Collaborative Filtering (CF)**
  - Find similar users ($k$ nearest neighbors) and recommend novel items of those users [Schafer et al., 2007]
  - In Solr: select queries and facet counts

- **Content-Based (C)**
  - Analyze item meta-data to find similar items [Pazzani et al., 2007]
  - In Solr: *MoreLikeThis* function

- **Hybrid (CCF)**
  - Combine different algorithms to overcome their individual limitations [Burke et al., 2002]
  - Each algorithm can be weighted / tuned according to its performance