Diversity in Recommender Systems: Extensions

Toni Mikkola, Andy Valjakka, Heng Gui, Wilson Poon
RECAP: Boim et al 2011 - Priority Cover Tree

- **PCT**: data structure specialized on nearest neighbour search
  - **Ordered Insertion** - Algorithm first sorts items with respect to ratings, then inserts in descending order
  - **Tight Insertion** - During item insertion into tree, nodes with smallest distance measure are preferred as parents
  - Each node has a rating higher or equal to that of any of its children

- Why tight insertion for diversity?
  - Smaller distances measured = more similar items
  - Highly rated items with more diversity will be omitted in the item set

- Proposed solution: Tag Usage
  - Each tag is given a weight from 0 to 1 based on relevance and preference
  - Ideally, would be possible to combine multiple tag in queries
Priority Cover Tree using Tags

- **PCT**: Modification of tight insertion, applying weights from tags
  - Instead, prefer items that are more diverse or has tag weight of 0
  - Example: Book is selected
    - Tags in the same genre would be given a weight of 1, but same author a 0
    - Returns an item set with the highest ratings in the same genre by different authors
    - Rationale: 1) Author is known to user; 2) Genre is good indicator for preference

- **Drawbacks:**
  - Tags are not used in many frequented sites: Amazon, Youtube
  - Non-standardized tag usage: Playstation, PS4, Playstation 4, Sony PS
  - Tagging weights may need to be tweaked by user preference
    - User may want complete diversity, tags with the same genre given 0 weight
    - Subjective tags for items: Shirt, Female, **stylish, love, beautiful, fun**
      - May be easiest to just ignore these for most queries, i.e set weight to 0
simplify recommender systems in real circumstances

- Better result of recommendation always requires longer processing time and larger storage space, especially with large amount of users. Simplification or in other way, concession, is needed in real circumstances. For instance, by simplifying CF strategy.
- In CF strategies computing, only go through those representative and valuable users.
  - Only 15% of users contribute to the collaborative filtering. So it is reasonable to ignore some users.
  - Representative users represent the taste of a group of people.
Users cluster

0. Assume a database have 10 millions of users

1. Filter the valuable users.
   a. Only the users have lots of related items contribute to the recommendation. And only them are valuable to be candidate.
   b. Assuming 1/10 of the users are selected.

2. Cluster the users
   a. Cluster these 1 millions of users into 1000 groups. Each group represent a taste.
   b. I guess 1000 types of taste is enough to describe all the users.

3. Choose a representative users from each cluster.
   a. We get 1000 representative users

4. Manually adjust
Diversity is not just dissimilarity
Diversity is not just dissimilarity

- If something is *similar*, it’s still not the *same*
- Why discard information?
  - Different usage instead
- Is there a way to do this efficiently
  - Collect as much stuff as possible in one go ("user-first")
  - ..or utilize different approaches in parallel
  - ..or?
• Especially a “matured” music taste is very diverse on its own
  ○ Take advantage of this
• Structure similar music on their own clusters
  ○ Links user taste to them
  ○ (not user-specific)
How to Diversify with low number of recommendations?

We could improve baseline recommender and we don’t have to think about over specialization because Diversification Algorithm takes care of it.

How to improve recommender model?

- Add contextual or background information to user x item matrix
  - E.g. in film recommender we could add genre, director
  - Restaurant recommender time of the day (lunch time, dinner time),
  - hotel recommender time period (summer, winter), trip reason( holiday, business trip) etc.
  - All kind information which is easy to collect and valuable for recommender in specific area
Normally contextual aware recommender systems (CARS) work with multidimensional matrix (User X Item X Context...)

Dimensions as Virtual Items (DaVI) [Domingues et al., 2011]

- The idea behind DaVI is to treat additional dimensions as virtual items, using them together with the regular items in a recommender system.
- From results virtual items are filtered out.
- DaVI has great potential to improve the predictive ability of top-N recommender systems.

If get more recommendations to Diversification Algorithm also users and items which don’t have so wide network could benefit from diversified recommendations.
### User x Item Matrix

<table>
<thead>
<tr>
<th>user</th>
<th>item</th>
<th>genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>i1</td>
<td>g1</td>
</tr>
<tr>
<td>u1</td>
<td>i2</td>
<td>g1</td>
</tr>
<tr>
<td>u1</td>
<td>i4</td>
<td>g2</td>
</tr>
<tr>
<td>u2</td>
<td>i2</td>
<td>g1</td>
</tr>
<tr>
<td>u2</td>
<td>i4</td>
<td>g2</td>
</tr>
<tr>
<td>u3</td>
<td>i3</td>
<td>g2</td>
</tr>
<tr>
<td>u3</td>
<td>i5</td>
<td>g2</td>
</tr>
</tbody>
</table>

### Item x Item Similarity Matrix (Jaccard)

<table>
<thead>
<tr>
<th></th>
<th>i1</th>
<th>i2</th>
<th>i3</th>
<th>i4</th>
<th>i5</th>
<th>g1</th>
<th>g2</th>
</tr>
</thead>
<tbody>
<tr>
<td>i1</td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i2</td>
<td>0.5</td>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i3</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g1</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g2</td>
<td>0.33</td>
<td>0.67</td>
<td>0.33</td>
<td>0.67</td>
<td>0.33</td>
<td>0.67</td>
<td></td>
</tr>
</tbody>
</table>
References

