Revisiting Neighbourhood-Based Recommenders for Temporal Scenarios

Alejandro Bellogín, Pablo Sánchez
Universidad Autónoma de Madrid
Spain

RecTemp @ RecSys, August 2017
Preliminaries

- Classical nearest neighbourhood-based approach
  - Rating aggregation from the $k$ most similar users:
    \[
    \hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} r_{vi} w_{uv}}{\sum_{v \in N_i(u)} |w_{uv}|}
    \]
  - A similarity function is used to weight the rating and to select the closest users
  - Different rating normalisations can be applied
Main idea

- How can we incorporate time in kNN recommenders?

- Several options in the literature:
  - Contextual filtering: pre and post [Baltrunas & Amatriain 2009] [Adomavicius & Tuzhilin 2015]
  - Adaptive heuristics: using a function to penalise older preferences
    - For rating prediction [Ding & Li 2005]
    - For similarity computation [Hermann 2010]
  - Selecting $k$ dynamically [Lathia et al 2009]
Proposal

- Reformulate the kNN problem so the temporal dimension can be exploited intuitively
  - Each neighbour provides a list of suggestions for each user
  - These suggestions are later combined considering rank aggregation techniques from Information Retrieval
  - The temporal aspect can be considered at different stages
- This approach provides an intuitive rationale about what is being recommended and why
Background: Rank aggregation

- Each algorithm (*judge*, e.g., a search engine in IR) generates a document ranking
- A final ranking has to be returned
- The process is usually divided in
  - Normalisation: scores or ranks from each judge to a document are normalised in a common scale
  - Combination: a fused score is computed for every document
kNN as rank aggregation

- The kNN problem can be seen as “ask each neighbour to provide a list of candidate items”

\[
\hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} r_{vi} w_{uv}}{\sum_{v \in N_i(u)} |w_{uv}|}
\]
Incorporating time in kNN

- Each neighbour will only provide items around the last item interacted with the target user (in yellow)
Incorporating time in kNN

- Each neighbour will only provide items **around** the last item interacted with the target user
  - Most recent $m$ items **after** the interaction: Forward (F)
  - Most recent $m$ items **before** the interaction: Backward (B)
  - A combination: Backward-Forward (BF)

- Time is considered twice:
  - Involving the target user (last common interaction)
  - Exploiting how the neighbour interacted with the items (temporal order)
Experiments

- Dataset: Epinions (from [He & McAuley 2016]), very sparse (0.004%), unbiased sample

- Evaluation methodologies (temporal split)
  - **CC**: same timestamp for everyone (more realistic), 80% of data as training
  - Fix: last 2 actions of each user (with at least 4 actions) are included in the test split
Experiments

- Baselines
  - ItemPop
  - KNN: kNN for ranking (no normalisation) using Jaccard coefficient
  - TD: exponential time decay weight
  - FMC: factorised Markov chains
  - FPMC: factorised personalised Markov chains
  - Fossil: factorised sequential prediction with item similarity models
- The first 3 baselines were implemented in RankSys
- We use the implementation provided by the authors for the rest
Results: CC split – Baselines

- KNN is one of the best baselines
- TD does not improve unless many items are considered
- Fossil is the best performing one among the sequential-based baselines

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision@5</th>
<th>nDCG@5</th>
<th>Recall@5</th>
<th>nDCG@10</th>
<th>Precision@50</th>
<th>Recall@50</th>
<th>cvg</th>
<th>Δ wrt KNN</th>
<th>Δ wrt Fossil</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemPop</td>
<td>1.81E-04</td>
<td>8.89E-04</td>
<td>2.25E-03</td>
<td>1.21E-03</td>
<td>3.80E-04</td>
<td>4.69E-02</td>
<td>100%</td>
<td>-144.08%</td>
<td>-36.88%</td>
</tr>
<tr>
<td>KNN</td>
<td>2.29E-04</td>
<td>2.17E-03</td>
<td>2.17E-03</td>
<td>2.94E-03</td>
<td>4.59E-05</td>
<td>4.34E-03</td>
<td>100%</td>
<td>–</td>
<td>43.92%</td>
</tr>
<tr>
<td>TD</td>
<td>2.29E-04</td>
<td>2.17E-03</td>
<td>2.17E-03</td>
<td>2.17E-03</td>
<td>6.88E-05</td>
<td>6.51E-03</td>
<td>100%</td>
<td>0.00%</td>
<td>43.92%</td>
</tr>
<tr>
<td>FMC</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>4.49E-04</td>
<td>2.69E-05</td>
<td>1.22E-03</td>
<td>85.21%</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>FPMC</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>2.69E-05</td>
<td>1.22E-03</td>
<td>85.21%</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Fossil</td>
<td>2.69E-04</td>
<td>1.22E-03</td>
<td>2.43E-03</td>
<td>2.43E-03</td>
<td>2.69E-05</td>
<td>2.43E-03</td>
<td>85.21%</td>
<td>-78.31%</td>
<td>–</td>
</tr>
<tr>
<td>BFuCF</td>
<td>2.29E-04</td>
<td>2.17E-03</td>
<td>2.17E-03</td>
<td>2.17E-03</td>
<td>6.88E-05</td>
<td>4.49E-03</td>
<td>100%</td>
<td>0.00%</td>
<td>43.92%</td>
</tr>
<tr>
<td>BFwCF</td>
<td>4.59E-04</td>
<td>3.10E-03</td>
<td>4.34E-03</td>
<td>3.10E-03</td>
<td>9.17E-05</td>
<td>6.66E-03</td>
<td>100%</td>
<td>30.10%</td>
<td>60.80%</td>
</tr>
</tbody>
</table>

Alejandro Bellogín – RecTemp @ RecSys, August 2017
Results: CC split – Backward-Forward

- BF performs better than F or B alone (not shown)
- BF coverage is the same as KNN (same similarity)
- Better performance than KNN in all metrics
- In this split, BFwCF (where each neighbour is weighted by the similarity) outperforms BFuCF
Conclusions

- A new formulation for neighbourhood-based recommenders is presented
  - Bitbucket repo: PabloSanchezP/bfrecommendation

- This formulation allows to integrate the temporal information in different parts of the algorithm

- Large performance improvements are obtained with respect to classical kNN methods and sequential-based baselines
  - These results depend on the splitting strategy
  - Results are more positive for the more realistic strategy (CC)
Future work

- Explore more aggregation (normalisation and combination) functions
- Analyse effect in other datasets
- Compare against other baselines (SVD++, BPR, ...)
- Study sensitivity to the number $m$ of items each neighbour includes in the ranking
- Explore sequence-aware similarity metrics
  - The temporal dimension could be also considered when selecting the neighbours
  - We are working on applying Longest Common Subsequence to recommendation [Bellogín & Sánchez 2017]
Thank you

Revisiting Neighbourhood-Based Recommenders for Temporal Scenarios

Alejandro Bellogín, Pablo Sánchez
Universidad Autónoma de Madrid
Spain

RecTemp @ RecSys, August 2017
References

▪ [Ding & Li 2005] Time weight collaborative filtering. CIKM.
▪ [He & McAuley 2016] Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. ICDM
▪ [Hermann 2010] Time-based recommendations for lecture materials. EMHT
Results: Fix split – Baselines

- KNN is one of the best baselines
- TD does not improve the performance
- ItemPop is the best one when several items are considered
- Fossil is not the best performing one among the sequential-based baselines

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision@5</th>
<th>nDCG@5</th>
<th>Recall@5</th>
<th>nDCG@10</th>
<th>Precision@50</th>
<th>Recall@50</th>
<th>cvg</th>
<th>Δ wrt KNN</th>
<th>Δ wrt Fossil</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemPop</td>
<td>3.32E-04</td>
<td>1.28E-03</td>
<td>1.74E-03</td>
<td>2.12E-03</td>
<td>4.06E-04</td>
<td>2.19E-02</td>
<td>100.00%</td>
<td>-200.02%</td>
<td>-5.48%</td>
</tr>
<tr>
<td>KNN</td>
<td><strong>1.05E-03</strong></td>
<td>3.84E-03</td>
<td>5.56E-03</td>
<td>4.94E-03</td>
<td>3.83E-04</td>
<td>2.05E-02</td>
<td>97.20%</td>
<td>–</td>
<td>64.84%</td>
</tr>
<tr>
<td>TD</td>
<td>3.15E-04</td>
<td>1.09E-03</td>
<td>1.99E-03</td>
<td>1.62E-03</td>
<td>2.97E-04</td>
<td>1.68E-02</td>
<td>97.20%</td>
<td>-252.10%</td>
<td>-23.79%</td>
</tr>
<tr>
<td>FMC</td>
<td>4.34E-04</td>
<td>1.39E-03</td>
<td>2.42E-03</td>
<td>2.27E-03</td>
<td>2.91E-04</td>
<td>1.57E-02</td>
<td>100.00%</td>
<td>-176.32%</td>
<td>2.85%</td>
</tr>
<tr>
<td>FPMC</td>
<td>4.08E-04</td>
<td>1.08E-03</td>
<td>1.93E-03</td>
<td>1.67E-03</td>
<td>2.20E-04</td>
<td>1.10E-02</td>
<td>100.00%</td>
<td>-255.14%</td>
<td>-24.86%</td>
</tr>
<tr>
<td>Fossil</td>
<td>3.32E-04</td>
<td>1.35E-03</td>
<td>1.64E-03</td>
<td>2.28E-03</td>
<td>2.60E-04</td>
<td>1.38E-02</td>
<td>100.00%</td>
<td>-184.43%</td>
<td>–</td>
</tr>
<tr>
<td>BFuCF</td>
<td><strong>1.05E-03</strong></td>
<td>3.89E-03</td>
<td>5.56E-03</td>
<td>4.75E-03</td>
<td>3.62E-04</td>
<td>1.96E-02</td>
<td>97.20%</td>
<td>1.39%</td>
<td>65.33%</td>
</tr>
<tr>
<td>BFwCF</td>
<td>9.46E-04</td>
<td>3.48E-03</td>
<td>4.96E-03</td>
<td>4.65E-03</td>
<td>3.55E-04</td>
<td>1.91E-02</td>
<td>97.20%</td>
<td>-10.50%</td>
<td>61.15%</td>
</tr>
</tbody>
</table>
Results: Fix split – Backward-Forward

- BF performs better than F or B alone (not shown)
- BF coverage is the same as KNN (same similarity)
- Better performance than KNN in most metrics for BFuCF
- In this split, BFuCF outperforms BFwCF (the opposite of what we observed in CC)