A Flexible Recommendation System for Cable TV

Diogo Gonçalves, Miguel Costa, Francisco Couto
LaSIGE @ Faculty of Sciences, University of Lisbon

RecSysTV 2016, Boston, USA
September 15, 2016
Problem

• Information overflow
  • too many video contents from which to choose
  • too much time exploring video contents
  (thousands of programs broadcast in hundreds of TV channels, plus thousands of movies & series on VOD)

If a typical subscriber doesn’t find something to watch in about 60 to 90 seconds, they could lose interest and move on to something else.


• Impact
  • dissatisfaction
  • change to other systems with recommendations (e.g. Netflix, Youtube)
  • less visualization time
  • less revenue
  • churn
Learning-to-rank (L2R) Approach

• Characterization of user behaviour in cable TV (previous study).
  • showed that Live TV and Catch-up TV have more users and views

• Improvement of state-of-the-art algorithms using the learning-to-rank (L2R) framework with contextual information and implicit feedback.

  1. Extraction of implicit feedback for each pair <user, program>
  2. Feature engineering on the contextual information
  3. Creation of a large-scale dataset for learning & evaluation
  4. Creation of a L2R model (cast recommendations as a supervised learning problem)
  5. Evaluation against state-of-the-art algorithms
12 weeks, between October and December of 2015

10K clients randomly chosen that watched at least 10 programs per week

21K programs available to the users during that period

4.5M of user visualizations (users saw more than 50%)
  • we assumed users liked these ones

78.5M of user visualizations inferior to 50% (most are 0%)

In average, each user watched 454 programs and each program was watched by 216 users
Feature Engineering

• Users tend to see the same programs & channels
  • Rankings of programs and channels & the relative visualization in each one (per user and for all users)

• Users tend to see the same categories & subcategories of contents
  • Rankings of categories and subcategories & the relative visualization in each one (per user and for all users)

• Time influences user preferences
  • Weekend information, broadcast period, time passed since broadcast, last days information

• Users tend to see programs with similar textual description of their content
  • Textual similarity functions (e.g. Jaccard, TFxIDF) between metadata (e.g. title, description)

• Users tend to see programs with similar characteristics
  • Number of episodes, content age & duration, category & subcategory

• Different types of contents require different strategies
  • Information about who saw the content and whether is new for the user or for all

• Similar users watch the same programs
  • Collaborative-filtering algorithms, such as WRMF (*Collaborative Filtering for Implicit Feedback Datasets, ICDM 2008*)

(60 features created based on the insights of previous characterization)
Learning to Rank Framework (LambdaMART)

![Diagram of Learning to Rank Framework]

- Users: $q^{(1)}$, $q^{(2)}$, ..., $q^{(i)}$, ..., $q^{(m)}$
- Preference scores: $y_1^{(i)}$, $y_2^{(i)}$, ..., $y_j^{(i)}$, ..., $y_{n(i)}^{(i)}$
- Programs: $d_1^{(i)}$, $d_2^{(i)}$, ..., $d_j^{(i)}$, ..., $d_{n(i)}^{(i)}$
- Feature vectors: $x_1^{(i)}$, $x_2^{(i)}$, ..., $x_j^{(i)}$, ..., $x_{n(i)}^{(i)}$
- Model-generated scores: $z_1^{(i)}$, $z_2^{(i)}$, ..., $z_j^{(i)}$, ..., $z_{n(i)}^{(i)}$

Listwise loss function:

$$\sum_{i=1}^{m} L(y^{(i)}, z^{(i)})$$

LambdaRank [Burges et al. NIPS’06]
LambdaMART [Wu et al. MSR-TR-2008-109]

Algorithms work differently for different types of programs. For instance, new programs tend to work better with content-based approaches, while the programs never seen by a user tend to work well with collaborative-filtering approaches. How to complement them?
Algorithms

• **Random**: returns random recommendations.
• **Popular**: recommends the most popular programs watched by everyone.
• **UserPopular**: recommends the programs most watched by the user.
• **WRMF**: a matrix factorization technique for collaborative filtering where user and item latent vectors are inferred from implicit feedback.
• **Content-based**: recommends programs based on similar content of programs watched by the user (e.g. category, actors, directors).

• **Learning to Rank**: LambdaMART is a type of Gradient Boosted Regression Trees algorithm that was used to combine recommendation features and algorithm outputs.
**Evaluation Methodology**

1. Learning System \( \rightarrow \) model \( h_1 \) \( \rightarrow \) Predicting System \( \rightarrow \) prediction
2. Client visualizations (training data)
3. Client visualizations (test data)
4. Learning System \( \rightarrow \) model \( h_2 \) \( \rightarrow \) Predicting System \( \rightarrow \) prediction

Time-series 5-fold cross validation
Evaluation Metrics

• Accuracy – How well the items meet the users’ information need?
  NDCG@k - match between recommendations and what the client watched after

• Diversity – How different are the items with respect to each other?
  Intra-list diversity@k (ILD@k)

• Novelty – How novel is the item for all users?
  Mean self-information (MSI@k)

• Serendipitious – How different is the item with respect to the user history?
  Unexpecteness@k

Only the top-k items in the recommendation lists are considered.
We used a k of 5 and 10, because these are the typical sizes of recommendation lists shown in Cable TV.
1. Compute a recommendation list optimized for accuracy
2. Rerank recommendations, usually trading-off accuracy for other metrics (e.g. diversity and novelty)

GreedyRecommendations(programs,objective_function) {
    ranking = Ø
    repeat
        programsᵢ = argmaxᵢ objective_function(ranking ∪ programsₖ), for all k ∈ [1..#programs]
        ranking = ranking ∪ programsᵢ
        programs = programs \ programsᵢ
    until programs = Ø
    return ranking
}

objective_function() calculates a score for a ranking list
    e.g. 50% NDCG@5 + 25% ILD@5 + 25% MSI@5
## Results Catch-up TV

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
<th>nDCG@5</th>
<th>nDCG@10</th>
<th>nDCG@5</th>
<th>nDCG@10</th>
<th>diversity</th>
<th>nDCG@5</th>
<th>nDCG@10</th>
<th>novelty</th>
<th>nDCG@5</th>
<th>nDCG@10</th>
<th>serendipitous</th>
<th>nDCG@5</th>
<th>nDCG@10</th>
<th>global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td></td>
<td>0.002</td>
<td>0.002</td>
<td>0.001</td>
<td>0.002</td>
<td><strong>0.919</strong></td>
<td><strong>0.920</strong></td>
<td><strong>0.677</strong></td>
<td><strong>0.679</strong></td>
<td><strong>0.935</strong></td>
<td><strong>0.935</strong></td>
<td>0.400</td>
<td>0.401</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Popular</td>
<td></td>
<td>0.293</td>
<td>0.247</td>
<td>0.021</td>
<td>0.018</td>
<td>0.600</td>
<td>0.541</td>
<td>0.058</td>
<td>0.080</td>
<td>0.888</td>
<td>0.887</td>
<td>0.311</td>
<td>0.279</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserPopular</td>
<td></td>
<td>0.694</td>
<td>0.600</td>
<td>0.000</td>
<td>0.000</td>
<td>0.708</td>
<td>0.727</td>
<td>0.200</td>
<td>0.213</td>
<td>0.872</td>
<td>0.872</td>
<td>0.574</td>
<td>0.535</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRMF</td>
<td></td>
<td>0.306</td>
<td>0.277</td>
<td>0.034</td>
<td>0.030</td>
<td>0.523</td>
<td>0.588</td>
<td>0.153</td>
<td>0.162</td>
<td>0.856</td>
<td>0.858</td>
<td>0.322</td>
<td>0.326</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content-based</td>
<td></td>
<td>0.345</td>
<td>0.316</td>
<td>0.059</td>
<td>0.052</td>
<td>0.503</td>
<td>0.526</td>
<td>0.346</td>
<td>0.354</td>
<td>0.837</td>
<td>0.831</td>
<td>0.385</td>
<td>0.378</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2R</td>
<td><strong>0.726</strong></td>
<td><strong>0.630</strong></td>
<td><strong>0.110</strong></td>
<td><strong>0.093</strong></td>
<td></td>
<td>0.683</td>
<td>0.692</td>
<td>0.199</td>
<td>0.214</td>
<td>0.870</td>
<td>0.868</td>
<td>0.583</td>
<td>0.541</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greedy + L2R</td>
<td>0.556</td>
<td>0.440</td>
<td>0.027</td>
<td>0.010</td>
<td></td>
<td>0.753</td>
<td>0.784</td>
<td>0.606</td>
<td>0.654</td>
<td>0.850</td>
<td>0.857</td>
<td><strong>0.618</strong></td>
<td><strong>0.580</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**L2R provides the best accuracy.**

**Random provides the best results in these 3 metrics.**

Used as example: objective = 50% nDCG + 25% ILD + 25% MSI

Greedy trade-off accuracy for diversity and novelty
## Results Live TV

<table>
<thead>
<tr>
<th>Method</th>
<th>accuracy</th>
<th>accuracy for new</th>
<th>diversity</th>
<th>novelty</th>
<th>serendipitious</th>
<th>global</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nDCG@5</td>
<td>nDCG@10</td>
<td>nDCG@5</td>
<td>nDCG@10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>0.002</td>
<td>0.002</td>
<td>0.003</td>
<td>0.003</td>
<td>0.919</td>
<td>0.920</td>
</tr>
<tr>
<td>UserPopular</td>
<td>0.295</td>
<td>0.252</td>
<td>0.007</td>
<td>0.006</td>
<td>0.600</td>
<td>0.550</td>
</tr>
<tr>
<td>WRMF</td>
<td>0.699</td>
<td>0.603</td>
<td><strong>0.000</strong></td>
<td><strong>0.000</strong></td>
<td>0.708</td>
<td>0.727</td>
</tr>
<tr>
<td>Content-based</td>
<td>0.318</td>
<td>0.289</td>
<td>0.020</td>
<td>0.017</td>
<td>0.534</td>
<td>0.598</td>
</tr>
<tr>
<td>L2R</td>
<td>0.435</td>
<td>0.396</td>
<td>0.047</td>
<td>0.038</td>
<td>0.525</td>
<td>0.550</td>
</tr>
<tr>
<td>Greedy + L2R</td>
<td>0.726</td>
<td><strong>0.631</strong></td>
<td><strong>0.115</strong></td>
<td><strong>0.081</strong></td>
<td><strong>0.683</strong></td>
<td><strong>0.692</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.524</td>
<td>0.434</td>
<td>0.030</td>
<td>0.011</td>
<td>0.735</td>
<td>0.793</td>
</tr>
</tbody>
</table>

**Note:** The table shows the performance metrics for different methods across various scenarios.

- **accuracy** measures the accuracy of the model.
- **accuracy for new** focuses on the accuracy for new items.
- **diversity** and **novelty** assess the diversity and novelty of recommendations.
- **serendipitious** evaluates the serendipitous aspect of recommendations.
- **global** provides a comprehensive overview of the model's performance.
Conclusions

• Typical approaches used in VOD, such as collaborative-filtering and content-based filtering, are not effective for Live TV and Catch-up TV.

• Our learning to rank approach with contextual information and implicit feedback is superior in accuracy and accuracy for programs never seen before, while maintaining high scores of diversity and serendipity.
  • still, recommending programs never seen before requires further investigation.

• A solution to optimize recommendations for multiple metrics were proposed, which enables to easily adapt recommendations to user preferences.
  • still, it is necessary to understand what clients value (e.g. accuracy vs. diversity).
Thank you.

migcosta@gmail.com
Evaluation Metrics (formulas)

**Accuracy**

\[
\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}
\]

**DCG**

\[
\text{DCG}_p = \text{rel}_1 + \sum_{i=2}^{p} \frac{\text{rel}_i}{\log_2(i)}
\]

IDCG is the value for the perfect ranking

rel\(_i\) is 1 if the user watched the program at rank \(_i\) or 0 otherwise

**Diversity**

\[
d(i, j) = \frac{1}{3} \text{ category}(i=j) + \frac{1}{3} \text{ subcategory}(i=j) + \frac{1}{3} \text{ channel}(i=j)
\]

\[
\text{ILD}(u) = \frac{1}{|R|(|R|-1)} \sum_{i \in R} \sum_{j \in R} d(i, j)
\]

\[
\text{ILD} = \frac{1}{|U|} \sum_{u \in U} \text{ILD}(u)
\]

**Novelty**

\[
\text{MSI}(u) = \frac{1}{|R_u|} \sum_{v \in U} \frac{|\{v \in U \mid i \in W_v\}|}{|U|}
\]

\[
\text{MSI} = \frac{1}{|U|} \sum_{u \in U} \text{MSI}(u)
\]

**Serendipity**

\[
\text{Unexp}(u) = \frac{1}{|R||W_u|} \sum_{i \in R} \sum_{j \in W_u} d(i, j)
\]

\[
\text{Unexp} = \frac{1}{|U|} \sum_{u \in U} \text{Unexp}(u)
\]

\(R_u = \text{list of recommended items for user } u\)

\(W_u = \text{watching history for user } u\)

\(U = \text{list of users}\)