A Study on Novel Recommendation based on Personal Popularity Tendency

Implementation

Presented by XUE, Lanqing
November 23, 2016
ItemRank
A Random-Walk Based Scoring Algorithm for Recommender System

Tangent
A Novel, “Surprise-me”, Recommendation Algorithm

PPTM
Novel Recommendation based on Personal Popularity Tendency
1. **ItemRank**
   A Random-Walk Based Scoring Algorithm for Recommender System

2. **Tangent**
   A Novel, “Surprise-me”, Recommendation Algorithm

3. **PPTM**
   Novel Recommendation based on Personal Popularity Tendency
**ItemRank**

<table>
<thead>
<tr>
<th>User ($u_i$)</th>
<th>Movie ($m_j$)</th>
<th>Rate ($r_{ij}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

$\mathcal{U}_{ij} = \{ u_k : u_k \text{ is the user who watched both movie } m_i \text{ and } m_j \}$.  
$\mathcal{U}_{ij} = \emptyset$, if $i = j$.  
i.e.: $\mathcal{U}_{1,2} = \{ a, b \}$.  

$\hat{\mathcal{C}}_{ij} = |\mathcal{U}_{ij}|$, i.e.: $\hat{\mathcal{C}}_{1,2} = 2$.  

$\omega_j$ = sum of entries in $j$ - th column of $\hat{\mathcal{C}}$.

**Correlation Graph:** $\mathcal{C}_{ij} = \hat{\mathcal{C}}_{ij} / \omega_j$.

![Note](https://example.com/note.png)

\[
\hat{\mathcal{C}} = \begin{pmatrix}
0 & 2 & 1 \\
2 & 0 & 1 \\
1 & 1 & 0
\end{pmatrix}
\]

\[
\mathcal{C} = \begin{pmatrix}
0 & \frac{2}{3} & \frac{1}{3} \\
\frac{2}{3} & 0 & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & 0
\end{pmatrix}
\]
\( d_{ui} \): denotes user \( u_i \)'s preferences.
\( d'_{ui} = r_{ij} \): user \( u_i \)'s rating of movie \( j \).

**Individual Preferences:** \( d_{ui} = d'_{ui} / \mid d'_{ui} \mid \).

### Calculate the Individual preferences

<table>
<thead>
<tr>
<th>User ((u_i))</th>
<th>Movie ((m_j))</th>
<th>Rate ((r_{ij}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

i.e.: \( d'_a = (4 \ 5 \ 0) \)  \( d'_b = (3 \ 4 \ 0) \)  \( d_a = (4/7 \ 5/7 \ 0) \)
### ItemRank

<table>
<thead>
<tr>
<th>User ($u_i$)</th>
<th>Movie ($m_j$)</th>
<th>Rate ($r_{ij}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

**ItemRank Score:** \( IR_{ui} = \alpha \cdot C \cdot IR_{ui} + (1-\alpha) \cdot d_{ui} \)

\[
IR_{ui}(0) = |M|^{-1} \cdot 1_{|M|} \\
IR_{ui}(t+1) = \alpha \cdot C \cdot IR_{ui}(t) + (1-\alpha) \cdot d_{ui}
\]

\(|M|\) denotes the number of movies in dataset.

ItemRank Score represent the possibility of recommending a movie to a certain user.
ItemRank
A Random-Walk Based Scoring Algorithm for Recommender System

Tangent
A Novel, “Surprise-me”, Recommendation Algorithm

PPTM
Novel Recommendation based on Personal Popularity Tendency
**2. Tangent**

Based on graph mining.

Recommend movies which connect two groups — **Bridge nodes**.

![Diagram](image)

- **preferences score** × **bridge score**

**Tangent score**
ItemRank
A Random-Walk Based Scoring Algorithm for Recommender System

Tangent
A Novel, “Surprise-me”, Recommendation Algorithm

PPTM
Novel Recommendation based on Personal Popularity Tendency
**Daily Music Recommendation**

- **Korean**: 66.7%
- **Other**: 33.3%

---

**Ineffectiveness of a system**

1. The system mainly recommends popular movies;

2. The recommendation results are highly concentrated on a specific item set.
• Assumption
  The total earnings/gross of a movie reflects the popularity of the movie.

• Popularity
  log-scaled popularity. ($\log_{10}$)

  eg.  | Roman Holiday | $8,000,000$ | 6.9 |

• Dataset: MoveLens
  - 943 users
  - 100000 ratings
  - 1682 movies

  Positive ratings
  - 942 users
  - 48014 ratings
  - 1306 movies

![Popularity distribution of users' rating](image)

- $x$: degree of popularity
- $y$: number of ratings in each degree of popularity
Popularity distribution of users' rating

- $x$: degree of popularity
- $y$: number of ratings in each degree of popularity

Diversity distribution of users' rating

- Movie at Top: rarely rated by users
- Movie at Bottom: Frequently rated by users
• Idea of PPTM
  
  ⭐ Different users have different interest pattern.
  ⭐ Recommending movies according to an user’s personal popularity tendency.
• Personal Popularity Tendency of each User

\[ P = \{(p_1, w_{p1}), (p_2, w_{p2}), \ldots, (p_m, w_{pm})\} \]
1. Recommend items following the PPT of each individual;
   
2. Reflect a user’s preferences as much as possible.

\[
\begin{align*}
\text{max} & \quad \sum_i p_i \cdot z_i - c \cdot D_{EMD}(R, P) \\
\text{Subject to} & \quad z_i \in \{0, 1\} \text{ for all } i \\
& \quad \sum_i z_i = k
\end{align*}
\]

Item = \(I_1, I_2, \ldots\)

\(p_i\): a user’s preference scores on a particular item \(I_i\).

\(z_i\) = \[
\begin{cases} 
1 & \text{if item } I_i \text{ include in recommendation,} \\
0 & \text{otherwise.}
\end{cases}
\]

\(R\): the PPT of the recommendation result for user \(u\).

\(P\): the PPT of an user \(u\).

\(c\): weight parameter.

\(k\): top \(k\) recommendation.

\(D_{EMD}(R, P)\): Earth Mover’s Distance between distribution \(R\) and \(P\)

Using ItemRank Score here
• Earth Mover’s Distance (EMD)

\[ P = \{(p_1, w_{p_1}), (p_2, w_{p_2}), \ldots, (p_m, w_{p_m})\}\]
\[ Q = \{(q_1, w_{q_1}), (q_2, w_{q_2}), \ldots, (q_n, w_{q_n})\}\]
\[ \text{WORK}(P, Q, F) = (6 - 3) \cdot w = 3w \]

**EMD measures the minimum workload:**
\[ D_{EMD}(P, Q) = \min_F \text{WORK}(P, Q, F) \]
• Distribution of Popularity (Top 10 Recommendation result)

PPTM achieves the highest diversity among 3 algorithms
• Coverage

How many movies in dataset have been recommended

\[
\text{coverage} = \frac{\text{movie recommended}}{\text{total number of movies}}
\]
• Accuracy (1)

Relavant Accuracy

- compared to ItemRank

\[
\text{relavant Acc.} = \frac{\text{Acc}}{\text{Acc}_{\text{ItemRank}}}
\]

- 3-fold cross validation 10 times
  - 2 folds for recommendation
  - 1 fold for test

\[
\text{Acc.} = \frac{|\text{recommend itemset} \cap \text{test itemset}|}{|\text{test itemset}|}
\]
Experiment and Evaluation

• Accuracy (1)

![Graph showing relevant accuracy comparison](image)

**Tricky**

- Relevant Accuracy

\[
\text{relevant Acc.} = \frac{Acc}{Acc_{\text{ItemRank}}}
\]

- 3-fold cross validation 10 times
  - 2 folds for recommendation
  - 1 fold for test

\[
Acc. = \frac{|\text{recommend itemset} \cap \text{test itemset}|}{|\text{test itemset}|}
\]
Experiment and Evaluation

• Accuracy (2)

Actual Accuracy

$$Acc. = \frac{|\text{recommend itemset} \cap \text{test itemset}|}{|\text{test itemset}|}$$
### Time and Memory

- **Programming Language:** Java
- **Running environment:** 2.9 GHz Intel Core i5

<table>
<thead>
<tr>
<th>Top 10</th>
<th>Time needed for each person</th>
<th>Time needed for whole dataset</th>
<th>Memory used for whole dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemRank</td>
<td>0.7317s</td>
<td>689.2683s</td>
<td>97.4MB</td>
</tr>
<tr>
<td>TANGENT</td>
<td>0.8203s</td>
<td>772.7226s</td>
<td>72.7MB</td>
</tr>
<tr>
<td>PPTM0.01</td>
<td>0.7279s</td>
<td>658.6818s</td>
<td>79.3MB</td>
</tr>
<tr>
<td>PPTM0.001</td>
<td>0.7344s</td>
<td>691.7993s</td>
<td>69.8MB</td>
</tr>
</tbody>
</table>
### Experiment and Evaluation

**Time and Memory**

- Programming Language: Java
- Running environment: 2.9 GHz Intel Core i5

<table>
<thead>
<tr>
<th>PPTM0.01</th>
<th>Time needed for each person</th>
<th>Time needed for whole dataset</th>
<th>Memory used for whole dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 10</td>
<td>0.7279s</td>
<td>658.6818s</td>
<td>58.5MB</td>
</tr>
<tr>
<td>Top 20</td>
<td>3.9144s</td>
<td>3,687.4011s</td>
<td>58.7MB</td>
</tr>
<tr>
<td>Top 30</td>
<td>6.9305s</td>
<td>6,528.5134s</td>
<td>59.9MB</td>
</tr>
<tr>
<td>Top 50</td>
<td>26.0897s</td>
<td>24,576.5385s</td>
<td>78.79MB</td>
</tr>
<tr>
<td>Top 100</td>
<td>55.6642s</td>
<td>52,491.3283s</td>
<td>79.3MB</td>
</tr>
</tbody>
</table>