A Game Theoretic Evaluation Framework of Recommendation Algorithms
Chaitanya Palaka, Tanvi Patel
Department of Computer Science Arizona State University Tempe AZ 85287 USA
{ chaitanya.palaka | tmpatel } @ asu.edu
6th December, 2016

Abstract
While several approaches to evaluate Recommendation algorithms do exist, there is little research available to find best suitable approach for recommendation to a particular user. We consider using a novel game theoretic approach to find a way to evaluate the best suitable recommendation algorithm for a user over a given span of time. We formulate payoff values for the game players to find best algorithm for the user over a number of sequential, repeated games.

Introduction
Recommendation systems are subclass of information filtering system that seeks to predict preference that a user would give to an item. Recommendation system changed the way websites communicate with their users. In the current age of the internet and big data, recommender systems have become more and more prevalent. Amazon and Netflix are not the only websites that use recommendation systems, but most medium and large scale commercial and social websites have recommendation engines in place for them. Many different types of recommendation algorithms have been developed and thus a key research area currently in recommender systems is on how to evaluate them, or how to measure the ‘goodness’ of a given recommendation algorithm. In this project, we compare recommendation based algorithms using a game theoretic approach. Research has been done for the evaluation of various algorithms but often times it considers a customized form of data set and user group. Also, there exists a gap between linking algorithm evaluation to user feedback evaluation of recommendations being provided to them by those algorithms.

In other studies[1], the most common method is to allow the users themselves to judge the recommendations by giving them a survey or questionnaire. The questionnaire might ask a series of questions to
evaluate different aspects of the recommendations such as accuracy, novelty, diversity, satisfaction, etc. The only issue with this approach is that the users, more often than not, will not opt to take part in the survey. Indeed, in the aforementioned paper, it was found that less than 1/3rd of the participants actually completed the survey. This makes things difficult to truly assess the recommender system with complete accuracy.

Our approach makes use of game theory concepts to formulate a novel evaluation framework for different recommendation algorithms. This method can be used to analyze the ‘goodness’ of each algorithm without requiring any explicit participation from the user. We do this by devising a ‘game’ played by the recommender system and a unique user. The recommender system’s strategies will be the various algorithms that we are to judge. The user’s strategies are simple: given the recommendations pertaining to the chosen algorithm, he can choose to Accept or Ignore. We consider this as the ‘stage game’. Since the user will be recommended items throughout his usage of the application or website, we can repeat this stage game multiple times to improve the accuracy of the evaluation. This is known as a ‘repeated game’. Also, this game will be ‘sequential’ and will be a game with ‘incomplete information’. The recommender system will employ a form of ‘trigger strategy’. We will explain these terms in the next section.

**Background**

The algorithms we have chosen to evaluate are: Content-Based Filtering, Collaborative Filtering, and a Hybrid method.

*(i) Content Based Recommendations (CB)*:
This uses user’s historical browsing information. Content can be manually defined or automatically extracted based on other similarity methods.

<table>
<thead>
<tr>
<th></th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item1</td>
<td>11</td>
</tr>
<tr>
<td>Item2</td>
<td>0</td>
</tr>
<tr>
<td>Item3</td>
<td>7</td>
</tr>
</tbody>
</table>

In above example user’s ranking for items are considered to suggest from a ranked set of similar items. Similarities are ignored as those items are already rated by users. Differences are used to make recommendations. In ranked set say item 1 is ranked highest followed by item 3 and item 2. User will get item 2 as recommendation.
(ii) Collaborative Filtering (CF):
Collaborative filtering arrives at a recommendation that’s based on a model of prior user behavior. When it takes other users' behavior into account, collaborative filtering uses group knowledge to form a recommendation based on like users.

<table>
<thead>
<tr>
<th></th>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item1</td>
<td>10</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>Item2</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Item3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

For example, in above table, user 1 has rated 10 to item 1 and user 3 has rated 11 to item 1. Next recommendation for user 3 will be item 2 as user 1 has high rating for item 2 while user 3 has not rated item 2 yet. User 1 cannot be recommended Item 3 based on user 3's rating to item 3 because there is a small difference between their rating for item 3.

(iii) Hybrid recommender system:
Recent research has demonstrated that a hybrid approach, combining collaborative filtering and content-based filtering could be more effective in some cases. Netflix is a good example of the use of hybrid recommender systems.

As mentioned earlier, the evaluation framework involves a repeated game. A repeated game, or iterated game, is an extensive form game which consists in some number of repetitions of some base game (called a stage game). We will define the stage game for our framework in the next section. In repeated games, we can consider 2 types: finitely and infinitely repeating games. We will consider both for our framework.

Furthermore, our game will be sequential. This is because the recommender system will always have to play first, as his move will decide the items to recommend to the user. The user will then play his turn by choosing whether to Accept or Ignore the recommendation.

The recommender system will have knowledge of the users move from the previous round, and will use this information to play his next strategy. However, the user will not know any information about the recommender system - neither his strategies or his payoffs. This type of game is a game with incomplete information, as the user will not possess full information about their opponents.
The recommender system will also employ a form of a *trigger strategy*. In classical repeated games, the player using a trigger strategy initially cooperates but punishes the opponent if a certain level of defection (i.e., the trigger) is observed. In our scenario, notions of ‘cooperation’ and ‘defection’ can only be loosely defined. We can say that the user ‘cooperates’ when he accepts the recommendations provided to him, and ‘defects’ otherwise.

**Experiment Setup**

To evaluate recommendation algorithms, we had set up an experiment. We conducted a user study to build user feedback report. We tested prototype application. User used Stack Overflow based application that provided her recommendations for each question being browsed by user on discussion forum. Implicit feedback from user was recorded. User has been using Stack Overflow for programming related discussions since at least more 6 months. Also, “Java” was one of her active tags linked to her profile on Stack Overflow. Hence it was safe to assume that user knew how to interact with application. We recorded 3 rounds of user feedback with implementing Content based, collaborative and hybrid algorithm one by one to generate recommendations.

*Calculating payoff for CF strategy:*
We use item based collaborative filtering to make recommendations similar to Stack Overflow questions user browses. Weighted sum of questions asked and answered by user on discussion forum was calculated as:
\[
q'_{u,i} = \frac{\sum_j s_{ij} \cdot v_{ui}}{\sum_j s_{ij}}
\]

The sum is over a subset (neighbor) of all the similar questions (to the target i) that the user u has earlier answered and asked on Stack Overflow. \(v_{ui} - S_{ij}\) is the similarity of i and j.

*Calculating payoff for Hybrid recommender system:*
We implement hybrid algorithm to allow deriving similarity measure between questions of “Java” tag. Mobasher et. al.[4] developed a weighted hybrid algorithm in which item-to-item similarity values are computed as the linear combination between content-based and collaborative similarities:
\[
c_{ij} = \alpha \cdot s_{ij}^{CBF} + (1 - \alpha) \cdot s_{ij}^{CF}
\]
where \(s_{ij}^{CBF}\) and \(s_{ij}^{CF}\) are computed, respectively, using (1) and (2).
Using above approaches, we recommended questions from available set of 215986 questions under 4358 second level and 8047 third level of tag categories.

**Payoff Calculation**

User’s payoff for “accept” strategy is calculated based on results obtained from user study. Payoff for user’s “Ignore” strategy and user’s accept strategy sum up to 1.

**User Feedback Collection:**

<table>
<thead>
<tr>
<th>Feedback activity</th>
<th>Feedback value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on Question</td>
<td>1</td>
</tr>
<tr>
<td>Click on Load more</td>
<td>0.9</td>
</tr>
<tr>
<td>Click on Related</td>
<td>0.8</td>
</tr>
<tr>
<td>Click on Up vote</td>
<td>0.7</td>
</tr>
<tr>
<td>Click on Mark as favorite</td>
<td>0.6</td>
</tr>
<tr>
<td>Click on Print</td>
<td>0.5</td>
</tr>
<tr>
<td>Mail to A friend</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Every single day applications and websites that are out there collect tons of GBs of data of how users interact with their applications. Websites like Amazon and Spotify store user data and use relevant information to improve their recommendation algorithms. Similar to how we did in experiment, this kind of implicit user feedback can be used to calculated user payoff.

**Proposed Framework**

We begin with a formulation of an initial base game, or stage game.

<table>
<thead>
<tr>
<th>User/Recommender</th>
<th>CB</th>
<th>CF</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>0.64,1</td>
<td>0.67,1</td>
<td>0.72,1</td>
</tr>
<tr>
<td>Ignore</td>
<td>0.36,-1</td>
<td>0.33,-1</td>
<td>0.28,-1</td>
</tr>
</tbody>
</table>

*Table: The initial stage game in normal form*
Each time the recommender will recommend something to the user, we can form the next stage game. In each stage game, we increment or decrement the recommender’s specific strategy payoff by some amount, depending on whether it was accepted or not by the user in the previous stage game. For now, we have chosen to increment/decrement by 1, but a specific constant per algorithm can be chosen based on some internal information known to the recommender. The user’s payoffs will be calculated in the same way as the initial game (depending on the recommendations). In this way, each round of the game, certain strategies for the recommender will increase in payoff, depending on the user’s actions.

The recommender will also use a form of trigger strategy, as mentioned earlier. If the user ‘defects’ or ignores the recommendations of the previous round, then the recommender will not choose that strategy in the next round. If the user accepts the recommendation, then the recommender will play as usual in the next round, i.e. choosing the strategy which gives him the maximum payoff.

In this way, at the end of all the repeated games, we can select which recommendation algorithm has performed the best by seeing which of the strategies have the highest payoff. This strategy, or algorithm, will be the best algorithm for that particular user.
This formulation happens to be an ‘online’ method of evaluating recommendation algorithms. We can set up this framework so that each time the user accesses the site or application, he will implicitly accept or ignore the recommendations given to him. This will happen without the user even realizing that it is happening, and each game, the choice for best algorithm will be updated and refined. As we can see, this method does not require any interaction from the user except for the implicit acceptance or ignorance of the recommendation.

We can also choose to either make this game infinitely repeating or finitely repeating. For a finitely repeating case, we can allow this game to happen for N recommendations, and from then on, employ the winning algorithm for that user. Alternatively, we can choose to make the game infinite by continuously evaluating the algorithms each time a recommendation has been made for that user. This approach might not be as efficient, because we can assume that after a certain number of rounds, a clear winner amongst the algorithms will be reached and no further games will change that.

**Conclusion**

In this project, we have proposed a brand new framework for evaluating recommendation algorithms. We have used the Stack Overflow dataset and a small sample set of users to create our base game, but without actively observing these individuals, we cannot create the repeated game. Hence, this approach is purely theoretical as of now. In the future, we can test it on different datasets from different website or applications to prove the effectiveness of this approach.

**References**


[5] Energy-saving routing method based on repeated games, CN104540181 A