CMPE 593 - FALL 2017

Report - Assignment II

# MOVIE SELECTOR

Boğaziçi University
2016800003 & 2016800021
Gönül AYCI & Yavuz KÖROĞLU

**Instructor**: Prof. Dr. Pınar YOLUM

November 21, 2017

# 1 INTRODUCTION

We propose Movie Selector, an agent which suggests the five best-unwatched movies for the user. Movie Selector ranks movies based on ratings given by other users as well as trust relationships of the user. We propose a fast scoring function, *Gonul Score*, which takes only two seconds to compute. We implement and show that Gonul Score selects the most reasonable movies for all five case studies in our work. We compare Gonul Score with *Naive Score* and *Yavuz Score*. We show that other scoring functions fail in some case studies.

We use the *FilmTrust* dataset which contains movie ratings and trust values for a set of users. We pick five users where each user has a different scenario. We describe these five scenarios as follows.

1. **User 6:** This user has only one friend and that friend only has one friend. We put this case to show how much the Movie Selector gives relevance to few opinions of friends over many opinions of unknown people.

2. **User 16:** This user has only two friends, but those friends have more than 30 friends. We put this case to show when the friends opinions are too few, we give importance to the friends' friends opinions.

3. **User 104:** This user has no friends. In this case, we show that both Gonul Score and Yavuz Score still works as reasonable as Naive Score.

4. **User 546:** This user has 23 friends. We show that in this case, Naive Score selects totally different movies than Yavuz and Gonul Score.

5. **User 1187:** This user has already watched some of the movies in the dataset. We deduce this information by checking if the user already gave a rating to the movie or not. We show that in this case we do not even consider already watched movies.

We also consider scenarios where the dataset is not available at the beginning but builds over time. We do this by taking only the first N rows of either the movie rating data or the trust data.

We use R language to implement our agent. R is very convenient for obtaining statistics from data. We obtain statistics from the data to calculate the *Gonul Score* and the *Yavuz Score*. You can find our implementation in `https://www.cmpe.boun.edu.tr/~yavuz.koroglu/cmpe593/CMPE593_Project2_GonulAYCI_YavuzKOROGLU.zip`.

In Section 2, we discuss the specifics of the dataset and the relationships we use to calculate scores for movies. In Section 3, we explain the details of our proposed scoring fuctions, Gonul Score and Yavuz Score. In Section 4, we discuss the implementation details. We present our case studies in Section 5. In Section 6, we discuss some potential fallacies in this study. We describe the future work in Section 7 and conclude with Section 8.

# 2 BACKGROUND

## 2.1 DATASET

The FilmTrust dataset contains two files, a *ratings* and a *trust* file. The ratings file contains triples in the form of (user, movie, rating). Here the user and the movie are represented by unique IDs. The rating is a fractional value between 0-4 where four being the highest rating for a movie. The trust file contains triples in the form of (user1, user2, trust). Trust is a boolean value indicating that user1 trusts user2. The file contains only the rows where trust value is one. The ratings file contains 35496 triples and the trust file has 1852 triples.

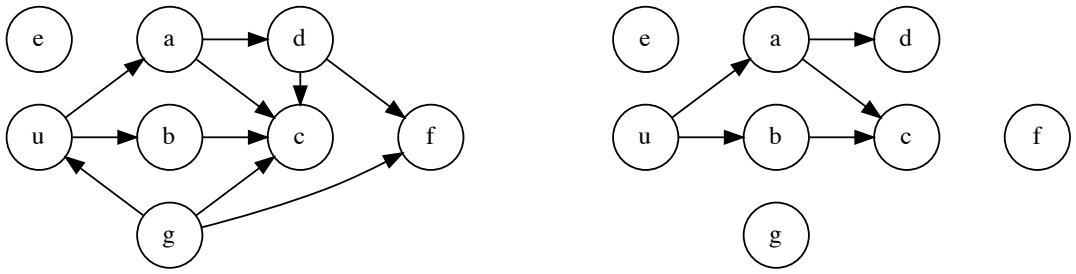


Figure 2.1: An example Trust Graph (on the left) for an arbitrary movie $m$ and its bounded version (on the right) w.r.t. the user $u$

### 2.2 Naive Score and Trust Graphs

We define the Naive Score as the average rating of all other users for an unwatched movie $m$. We will later show that this scoring function is inadequate in terms of selecting good enough movies. To build a better scoring function, we build Trust Graphs for each movie using the trust file. Trust Graph is a directed graph where each node is a user and each edge represents that there is a trust between users. We give an example Trust Graph in Figure 2.1. We argue that ratings of other users become more relevant as they get close to the user in the Trust Graph.

We define the length of the shortest path from a user $u$ to another user $o$ in the Trust Graph as $N_u(o|m)$ where $o$ is given to have made a review to movie $m$. For example, in Figure 2.1 $N_u(a|m) = 1$ and $N_u(e|m) = \infty$.

We denote the average rating for an unwatched movie $m$ given by users with the same shortest distance $n$ to the user $u$ as $s_{n,m}(u)$. It is hard to calculate $s$ for all $n$ values in a trust graph, therefore we construct Bounded Trust Graphs for each user $u$ as we show in Figure 2.1. A Bounded Trust Graph contains all one and two length paths outgoing from the user $u$ and all other edges are removed from the original Trust Graph.

## 3 Methodology

$$Y_u(m) = s_{1,m}(u) + \lambda s_{2,m}(u) + \lambda^2 \sum_{n=3}^{\infty} s_{n,m}(u) \tag{3.1}$$

$$G_u(m) = \begin{cases} s_{1,m}(u) + \lambda s_{2,m}(u) & \left|\{k : N_{u,m}(k) = 1\}\right| > 0 \\ \lambda^2 s_{\infty,m}(u) & o/w \end{cases} \tag{3.2}$$

Now, we define our scoring functions, Yavuz Score (Equation 3.1) and Gonul Score (Equation 3.2). In these scoring functions, $\lambda$ is the decay factor. The decay factor $\lambda$ determines the relative importance of the opinions of friends' friends w.r.t. the opinions of friends. We take $\lambda = .5$ for this study.

Yavuz Score considers all ratings given by all users. Yavuz Score assumes that opinions of friends are the most important, opinions of friends' friends are less important, and all other opinions are the least important with the same decay factor $\lambda^2$.

Gonul Score categorizes users into two, users who have at least one friend with a review to movie $m$ and users who have no friends with a review to movie $m$. In the first case, Gonul Score gives importance to only the opinions of friends and opinions of friends' friends (with the same decay

factors as in Yavuz Score). In the second case, the average rating of all users is multiplied by $\lambda^2$ and returned as the Gonul Score.

# 4 Implementation

## 4.1 How to Execute

We implemented the Movie Selector agent along with Yavuz Score and Gonul Score in *R* language. We did not use any third party libraries. Our program is just one script file, *main.R*.

To execute our program, first open RStudio. Open the directory which contains *main.R* script and *filmtrust* dataset directory. Click *more* and then click *Set as Working Directory* to correctly set the R environment. Then, open the *main.R* script and change the UserID variable to whoever you want to select movies for. Then, execute the whole script.

## 4.2 Output Format

When we execute the script, it outputs the following:

1. **Total Execution Time**,
2. **UserID**,
3. **Number of Friends**,
4. **Total number of unwatched movies**, and
5. **IDs of the best five ranking movies** according to the Gonul Score.

The script also stores the following tables:

1. **SelectedFilms1 :** The best five unwatched movies according to the Gonul Score.
2. **SelectedFilms2 :** The best five unwatched movies according to the Yavuz Score.
3. **SelectedFilms3 :** The best five unwatched movies according to the Naive Score.
4. **Sorted1 :** All unwatched movies sorted by their Gonul Score in descending order.
5. **Sorted2 :** All unwatched movies sorted by their Yavuz Score in descending order.
6. **Sorted3 :** All unwatched movies sorted by their Naive Score in descending order.

Each row of these tables represents a movie with a unique movie ID. There are 9 columns in all tables. These columns are:

1. Movie ID,
2. S_all as the Naive Score,
3. S_1 as $s_{1,m}(u)$,
4. S_2 as $s_{2,m}(u)$,
5. Gonul Score,

6. Yavuz Score,
7. **n1** as the total number of reviews made by friends,
8. **n2** as the total number of reviews made by friends' friends,
9. **nAll** as the total number of all reviews made by other users.

We review all these outputs carefully to derive our conclusions in case studies.

### 4.3 Main Algorithm

**Algorithm 1:** Main Algorithm

**Inputs:**
UserID $\in \mathbb{N}$,
RatingsData $\in (\mathbb{N},\mathbb{N},[0,4])^M$ where $M \in \mathbb{N}$ is the number of triples in the rating data.
$S_u : \mathbb{N} \to \mathbb{R}$ Scoring Function

**Output:**
SelectedMovies $\in (\mathbb{N},\mathbb{N},\mathbb{N},\mathbb{N},\mathbb{N})$

*// Get all triples NOT watched by the user*
1: moviesNotWatchedByUser $\leftarrow$ **Empty List of** $Movie \in \mathbb{N}$
2: **for all** (User, Movie, Rating) $\in$ RatingsData **s.t.** User $\neq$ UserID **do**
3: **Append** Movie **to** moviesNotWatchedByUser
4: **end for**

*// Sort all movies by the Scoring Function*
5: **Sort** $\forall$Movie $\in$ moviesNotWatchedByUser **by descending** $S_{\text{UserID}}$(Movie)

6: SelectedMovies $\leftarrow$ The first five movies **in** moviesNotWatchedByUser

We describe our main procedure in Algorithm 1. This algorithm takes three inputs. First, it takes the ID of the user. Second, it takes the rating triples coming from the FilmTrust dataset. Third, it takes the Scoring Function which can be either Naive, Yavuz, or Gonul Score. Our algorithm mainly outputs the first five movie IDs as SelectedMovies. From lines 1−4 we eliminate movies already watched by the user. Then, we sort all remaining movies by the scoring function in descending order in line 5. Finally, we take the first five movies in line 6.

## 5 Case Studies

We present five users as case studies. These users' IDs are 6, 16, 104, 546, and 1187. We show selected films by Naive, Yavuz, and Gonul Score for users 6, 16, 104, and 546 in Tables 5.1−5.4.

In Table 5.1, we show that Yavuz Score and Gonul Score are highly sensitive to the opinions of few friends over the opinions of unknown people. Naive Score selects five films that have no reviews from any friends or friends' friends. Furthermore, Naive Score is indifferent between all of its selected movies. On the other hand, Yavuz Score and Gonul Score select movies with good reviews from friends even though these movies have low Naive Scores. Note that Yavuz Score and Gonul Score select identical movies in this case.

Table 5.1: Movie Selector Results for User 6

| Sorted by Naive Score | | | | | | | Sorted by Yavuz Score | | | | | | | Sorted by Gonul Score | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** |
| **446** | 4 | 1 | 1 | 0 | 0 | 5 | **211** | 3.20 | 5.80 | 5 | 1 | 1 | 607 | **211** | 3.20 | 5.80 | 5 | 1 | 1 | 607 |
| **770** | 4 | 1 | 1 | 0 | 0 | 5 | **207** | 2.86 | 4.46 | 3.75 | 1 | 1 | 883 | **207** | 2.86 | 4.46 | 3.75 | 1 | 1 | 883 |
| **834** | 4 | 1 | 1 | 0 | 0 | 4 | **12** | 2.80 | 4.45 | 3.75 | 1 | 1 | 756 | **12** | 2.80 | 4.45 | 3.75 | 1 | 1 | 756 |
| **415** | 4 | 1 | 1 | 0 | 0 | 3 | **239** | 2.94 | 3.73 | 3 | 1 | 0 | 590 | **239** | 2.94 | 3.73 | 3 | 1 | 0 | 590 |
| **456** | 4 | 1 | 1 | 0 | 0 | 3 | **247** | 2.88 | 3.22 | 2.5 | 1 | 0 | 372 | **247** | 2.88 | 3.22 | 2.5 | 1 | 0 | 372 |

Table 5.2: Movie Selector Results for User 16

| Sorted by Naive Score | | | | | | | Sorted by Yavuz Score | | | | | | | Sorted by Gonul Score | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** |
| **1317** | 4 | 4 | 4 | 1 | 0 | 1 | **286** | 3.76 | 6.79 | 5.85 | 1 | 5 | 23 | **286** | 3.76 | 6.79 | 5.85 | 1 | 5 | 23 |
| **1322** | 4 | 4 | 4 | 1 | 0 | 1 | **511** | 3.44 | 6.59 | 5.75 | 1 | 1 | 9 | **69** | 3.43 | 6.56 | 5.75 | 1 | 2 | 7 |
| **1323** | 4 | 4 | 4 | 1 | 0 | 1 | **69** | 3.43 | 6.56 | 5.75 | 1 | 2 | 7 | **511** | 3.44 | 6.59 | 5.75 | 1 | 1 | 9 |
| **770** | 4 | 1.67 | 1 | 0 | 2 | 5 | **251** | 3.02 | 6.32 | 5.57 | 1 | 15 | 476 | **251** | 3.02 | 6.32 | 5.57 | 1 | 15 | 476 |
| **834** | 4 | 2 | 1 | 0 | 2 | 4 | **6** | 2.99 | 6.29 | 5.54 | 1 | 6 | 442 | **6** | 2.99 | 6.29 | 5.54 | 1 | 6 | 442 |

Table 5.3: Movie Selector Results for User 104

| Sorted by Naive Score | | | | | | | Sorted by Yavuz Score | | | | | | | Sorted by Gonul Score | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** |
| **446** | 4 | 1 | 1 | 0 | 0 | 5 | **446** | 4 | 1 | 1 | 0 | 0 | 5 | **446** | 4 | 1 | 1 | 0 | 0 | 5 |
| **770** | 4 | 1 | 1 | 0 | 0 | 5 | **770** | 4 | 1 | 1 | 0 | 0 | 5 | **770** | 4 | 1 | 1 | 0 | 0 | 5 |
| **834** | 4 | 1 | 1 | 0 | 0 | 4 | **834** | 4 | 1 | 1 | 0 | 0 | 4 | **834** | 4 | 1 | 1 | 0 | 0 | 4 |
| **415** | 4 | 1 | 1 | 0 | 0 | 3 | **415** | 4 | 1 | 1 | 0 | 0 | 3 | **415** | 4 | 1 | 1 | 0 | 0 | 3 |
| **456** | 4 | 1 | 1 | 0 | 0 | 3 | **456** | 4 | 1 | 1 | 0 | 0 | 3 | **456** | 4 | 1 | 1 | 0 | 0 | 3 |

Table 5.4: Movie Selector Results for User 546

| Sorted by Naive Score | | | | | | | Sorted by Yavuz Score | | | | | | | Sorted by Gonul Score | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** | **mID** | **NS** | **YS** | **GS** | **n1** | **n2** | **nAll** |
| **770** | 4 | 5 | 4 | 3 | 0 | 5 | **834** | 4 | 7 | 6 | 1 | 2 | 4 | **839** | 3.2 | 6.5 | 6 | 2 | 1 | 5 |
| **827** | 4 | 4 | 4 | 2 | 0 | 2 | **1442** | 4 | 7 | 6 | 1 | 1 | 3 | **834** | 4 | 7 | 6 | 1 | 2 | 4 |
| **834** | 4 | 7 | 6 | 1 | 2 | 4 | **689** | 3.86 | 6.95 | 6 | 1 | 1 | 7 | **1123** | 3.46 | 6.84 | 6 | 1 | 1 | 13 |
| **1442** | 4 | 7 | 6 | 1 | 1 | 3 | **1694** | 3.75 | 6.88 | 6 | 1 | 1 | 4 | **689** | 3.86 | 6.95 | 6 | 1 | 1 | 7 |
| **826** | 4 | 6 | 6 | 1 | 1 | 2 | **1123** | 3.46 | 6.84 | 6 | 1 | 1 | 13 | **429** | 2.83 | 6.56 | 6 | 1 | 1 | 6 |

For user 16, opinions of friends are not enough to differentiate between selected movies. Therefore, Yavuz Score and Gonul Score depend on the opinions of friends' friends. Note that Yavuz Score and

Gonul Score still select identical movies in this case.

Table 5.3 shows that for user 104 all scoring functions select identical movies. This is because user 104 has no friends. Hence, Yavuz Score and Gonul Score select movies as reasonably as Naive Score in case of no friends. Therefore, the user will never need to consult Naive Score.

Table 5.4 shows a case that Yavuz Score and Gonul Score disagree on selected movies. Although user 546 has many friends, Yavuz Score still gives some importance to all ratings whereas Gonul Score gives no importance to the opinions of unknown people.

We also consider cases where only a portion of the dataset is available, and the dataset builds over time. We simulate this by temporarily removing some of the rows in either the ratings file or the trust file. We examine results when the ratings file hasn't got 35496 triples but only 5000 triples and the trusts file hasn't got 1852 triples but only 500 triples. In all these results, our systems behave as expected.

## 6 Discussion

We implement the Movie Selector agent separately from our previous project, Conference Helper BDI Agent. We assume that the user already decided to watch a movie, Conference Helper BDI Agent already planned the user's schedule, and there will be nothing to interrupt this plan. If anything happens to violate these assumptions, Conference Helper BDI Agent handles it and there is no need for Movie Selector. Although implemented separately, Conference Helper BDI Agent can easily take the output of Movie Selector and fill the movie parameter in a "Watch <movie>" intention.

We argue that ratings from unknown people are misleading and the user should always prioritize opinions of people who are known to the user. In this aspect, selected movies for User 546 are more reasonable when Gonul Score is used rather than Yavuz Score. Still, we report selected films for every scoring function since relative importance of the opinions w.r.t. trust relationships can be a matter of taste and culture.

A user $a$ may be a friend of another user $b$ and also a friends' friend of that user $b$ at the same time. In this case, we always count $a$ to be a friend of $b$ and not a friends' friend of $b$ in the Trust Graph. In other words, we remove all paths that are not the shortest path between each node.

Our implementation recomputes everything from scratch whenever the dataset is changed. We justify this redundancy by arguing that our implementation only takes two seconds to compute the five best movies in all cases.

## 7 Future Work

In the absence of friends, we still can obtain a Trust Graph. Specifically, if there exists a strong correlation between User1's ratings and User2's ratings, we can assume that these users should trust each other. This assumption will not just allow us to build Trust Graphs in the absence of friends, but also allow us to rank friends by their rating correlation with the user. As a future work, we will use this assumption in our scoring functions.

We did not use the CiaoDVD dataset, which also contains film genres compare to the FilmTrust dataset. As a future work, we intend to integrate the user's genre choices into our scoring functions.

## 8 Conclusion

In this study, we propose a fast scoring system, Gonul Score, which comes up with the five best-unwatched movies as suggestions to the user. Gonul Score gives the most importance to the opinions

of friends, less importance to the opinions of friends' friends and the least importance to all other opinions. We show that in all case studies, Gonul Score comes up with the most reasonable movie selections.