Project #2 Proposal

Group: Hotdog

Dongru Jia (dj465@georgetown.edu); Jianhao Ji (jj913@georgetown.edu)
Miao Wang (mw1219@georgetown.edu); Yuxuan Yao (yy560@georgetown.edu)

1. Motivation:

Our project is designed to solve a familiar dilemma. Consider the following situation. During a regular date night, both you and your partner have favorite movies to watch in mind. However, it is hard to come to a perfect solution. It is either someone is sorry or annoyed, or both. Hence, to prevent such a situation, and let everyone enjoy a date night, our group is motivated to come up with an optimized solution, a movie recommendation application that based on two input movies, gives movie suggestions that could accommodate both parties.

2. Relevant (scientific) literature:

2.1 *Yang, C. et al. "A Hybrid Movie Recommendation Method Based on Social Similarity and Item Attributes." Vol. 10942. Springer Verlag, 2018. 275–285. Web.*

In this article, the author put forward a hybrid recommendation method in combination with collaborative filtering and content-based recommendation. The system was implemented in two phases work. In the first stage, the author used BPR-MF model which includes Bayesian Personalized Ranking Model (BPR) and Matrix Factorization (MF) to estimate the target preference. BPR model is to establish a partial sequence pair for each user to represent users' preferences. The MF model is based on latent semantic analysis, which assumes that the attributes and user preferences can be represented in lower dimensional space with lower dimension. The BPR-MF model trains the training dataset and obtain a refined candidate set according to the rating. Each movie has a new tag. In the second stage, the author used TF-IDF model of Gensim to calculate the similarity of tags and genres and compared the similarities between the candidate movies and movies that users had rated in training dataset. Finally, the author applied Root Mean Square Error and Mean Absolute Error for model evaluation.

2.2 Colucci, L., et al. Evaluating Item-Item Similarity Algorithms for Movies. in Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems. 2016. ACM.

This article also discusses how to determine the similarity of movies, while the main focus is on similarity-algorithms evaluation/comparison. It evaluates algorithms of four similarities: TMBD, Collaborative Filtering-Cosine, Content Filtering, and Collaborative Filtering-Pearson. The TMBD similarity, according to the article, is an unknown algorithm from TMBD.com, so it was treated as a black box. Both similarities of Collaborative Filtering-Cosine and Collaborative Filtering-Pearson come from the idea of collaborative filtering, which considers users' opinions as input. For example, if movie A and movie B are often watched by the same audiences, then they are adjudged to be similar. However, this algorithm would require data as detailed as providing specific users' entire watch history. We currently do not capture such a dataset, but if we spot one, we may consider borrowing these similarity ideas. Lastly, the Content Filtering, which we will implement in our project. We found Content Filtering particularly interesting, because as stated in the paper, it is highly scalable and less resource intensive, since its ideas is to consider subject's attributes as input, such as genre, keywords, director, etc. This similarity is suitable for our project, because it can be used to recommend similar movie to anonymous users, as we do not have our users' previous watch history.

2.3 Choi, Sang-Min, Sang-Ki Ko, and Yo-Sub Han. "A movie recommendation algorithm based on genre correlations." Expert Systems with Applications 39.9 (2012): 8079-8085.

This paper introduces a calculating genre correlation method which is based on collaborative filtering by user preference. Pearson correlation coefficient is used in this genre similarity calculation and authors do some revision work to make it fit their program better. As for the further predicting and classification

work, the author first selects neighbors according to the threshold pearson correlation coefficient and then does the nearest neighbor classification/prediction.

3. Data:

Our dataset includes around 5000 movies generated from The Movie Database (TMDB) API. The attributes include movie name, tagline, spoken languages, genres, overview, keywords, cast, crew, audience ratings, popularity, release date, revenue, and etc. Among them, four important variables are listed as below:

- Genres: Each movie is represented as a few genre categories such as Action, Adventure, etc.
- Keyword: Keywords are short description relating to the main plot.
- Overview: It is 1-5 line brief plot summaries and does not include any analysis of the movie.
- Tagline: Tagline is catchphrase or slogan for advertising and trying to attract audience's attention.

For the purpose of natural language processing, we will mainly tokenize the tagline, genres, overview, cast and keywords in order to explore the features of each movie. Our analysis is based on content analysis and similarity calculation, and we aim to provide suggestions on movies which are most similar to the two input movies. Meanwhile, the popularity and ratings are also considered when we look for audience's preference to movies. During the process, we may also consider to scarp audience's comments of each movie from TMDB.

4. Plan:

We plan to apply NLP skills learnt from this course and machine-learning techniques to build an application to recommend preferred movies based on two specific movie inputs. We will consider movies as vectors and compute cosine similarity by using movie attributes such as genres, keywords, tagline, overviews, etc. Then, according to the two input movies, our application will generate recommendations that are close to both inputs, so that both parties can enjoy the show. We currently consider two situations, depending on how similar the two input movies are. To start with the intuitive one, if two input movies are similar to each other, we will certainly return suggestions lying right in-between these two inputs. However, if two input movies are completely different or have totally opposite cosine similarity, simply returning movies lying right in-between may result in unsatisfying results for both parties. Since they are so different, the right-in-middle movies may be completely different and unrelated to neither inputs. To come with a better solution, we decide to include our recommendations with both movies lying right in-between and similar movies to either input. To be specific, we will also contain movies that are similar to either party respectively in recommendations. When considering how similar two movies are, we will also use cosine similarity. Finally, to sort recommendations, we will consider movie's popularity and rating.

5. Expected results:

For movie recommendation, we separate the whole model into two phases. The first step is to match similar movies. As users input the movie names, we would gather the movie information like directors, casts, genres, and movie intros and then input this information into our trained model. The model would compare the information of input movies and sort out the similarity. In this part, we hope our model can return many results match this similarity.

Since our project is to recommend similar movies to the two input movie samples, one way to evaluate our work is to determine how similar between our recommendations and the two inputs, and then measure the precision. To evaluate similarity, we need to compare our results with a labeled dataset of similar movies. So far, we have spotted such a dataset also on TMBD.com. But we will keep searching for such samples to use as our evaluation tool. We expect our recommendation precision would be higher when the two inputs are similar to each other than when the two inputs are different. And our precision would be 50-60 percent, as all four algorithms from the third article in related work achieved precisions in such range, even did the commercially used TMBD algorithm.