Movie recommendation system

Ilija Malinović, Marina Jerković

1. Motivation

Searching through tons and tons of movies and choosing the right one isn't the greatest option, so we wanted to be able to quickly pick a movie which has more chance to sooth our taste. There are different approaches to this problem, so we wanted to compare them and see which one works the best and why. The idea was to take relevant data about movies and users' likings and create recommendation engine which recommends top 10 movies to specific user. Gathering and processing data would be improved if we would get feedback from the user and his real rate. Of course, we must be aware that user's taste is changing with time, so it is not possible to make a perfect recommendation engine.

2. Research questions

Movie recommendation systems fall into two categories: collaborative filtering and content based filtering. We experiment with both of these approaches in our project. Both of these ways use the same preprocessed data about users, movies and ratings. Content based filtering compares movie similarities and recommends those which have the most similar content. Collaborative based filtering relays on user similarities and recommends movies which were liked by a similar user.
The dataset we used was found on Kaggle and consists of 2079 movies which were rated by 260 users, total of 10220 ratings.
One of the big problems was to choose relevant information about movies. There are 3 files we took as dataset: movie metadata, credits in those movies and users' rating of those movies. Preprocessing consisted of transforming json values in those files and grouping them into bag of words for a specific movie. We had to distinct attributes which we thought define movie's likability by a specific user. In the end, those attributes were: adult, genres, title, spoken languages, production companies, keywords and credits (most relevant cast, writers and directors). Values of those attributes were put in the right bag of words. The file with ratings had information about users' ratings of different movies. Rates were ranging from 1 to 5, step is 0.5, so the rates >= 3.5 were good.

3. Related work

Content-based filtering makes recommendation based on similarity in item features. Popular techniques in content-based filtering include the term-frequency/inverse-document-frequency (tf-idf), Bayesian classifier, decision tree, neural networks... This type of filtering has the advantage of being able to solve the cold start problem when there hasn't been enough users or when the contents haven't been rated.

Collaborative filtering recommends items that similar users like. There are two major approaches in collaborative filtering: the neighborhood model and latent factor models. The neighborhood model recommends the closest items or the closest user's top rated items. The latent factor model such as matrix factorization examines the latent space of movie and user features.

Many work has been done to combine the two techniques. One such approach is realized by incorporating content information in collaborative filtering, in which the content of a rated item is used to estimate the user’s preference for other items. This is helpful when users rated only a small portion of the items population, as the information of the unrated items can be inferred from content-based filtering, instead of having just a missing value.

4. Methodology

Because the dataset is too big, we made and saved the preprocessed data so the data load wouldn't take too long. Preprocessing data - creating bag of words for each movie was explaind in 2.

Content based filtering - this type of recommendation is mainly based on searching movies whose content is most similar to the content of movies user has already liked. Using the dataset, user's profile and movies' profile are created. User's profile model consists of bag of words which are found in bag of words of movies that user liked before (user rated that movie >= 3.5). The second thing we needed were movies that user hasn't rated already. Movie profile consists of bag of words for that movie. We fitted TfidfVectorizer on bag of words for all of the movies and then transformed it into values we needed: user profile model and model of movies user didn't watch. This vectorizer works like this: the importance of a term t is positively correlated with the number of times it appears in document d, but the frequency of term t among all documents is inversely related its ability to distinguish between documents. Thus we calculate the frequency of word t in document d, weighted by the inverse of frequency of t in all documents: tf-idf(t) = tf(t, d)∗idf(d) = tf(t, d)∗log |D|/(1+|d:t∈d|) where |D| is the length of the document, and |d : t ∈ d| is the number of documents where t appears.Now we were able to calculate similarities between user's preferences and potential movies. Two types of comparison were implemented: cosine similarity and jaccard similarity. In both cases the result were movies that have the most similar content to those user has already liked.

Collaborative filtering - this type of recommendation is based on finding users who liked similar movies and then recommending movies which are liked by the similar user. For this we needed matrix of users' ratings. Not every user rated every movie, so we needed to predict some ratings. We did this on 2 different ways. The first one was predicting specific rating by specific user by finding the most similar movie that user rated already to the one that hasn't been rated and then its rating was the missing rating. The second one included clustering. All of the movies were clustered into 30 clusters and then the missing rating was calculated by finding the cluster to which specific movie belongs and then finding the mean value of all the ratings of movies that belong to that cluster and are rated by that specific user. Now that we have all of the ratings, we can calculate user similarity. After this, we want to calculate predicted rating for a movie by specific user. If we want to predict user A's rating, we will calculate user B's rating multiplied by similarity of A and B, sum that with calculation of user C's rating multiplied by similarity of A and C and then that sum is divided by sum of similaritied (A and B) and (A and C).

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

5. Discussion

The good part of the content based filtering is that it will recommend movies that haven't been rated yet. The bad thing is that it does not distinguish between the quality of items. For example, a well celebrated movie and a badly received one are equally likely to be recommended if they share similar characteristics such as common phrases in movie descriptions.

Collaborative filtering addresses some of issues of content-based filtering – it recommends items that similar users like, and avoids the need to collect data on each item. The algorithm was faster when we used clustering for missing values.

In terms of time, content based filtering worked faster. We had idea to test content-based filtering by using precision and recall but we stuck with empirical evaluation.

6. References

[1] Getting Started with a Movie Recommendation System, Kaggle

[2] How to build a content-based movie recommender system with Natural Language Processing, Emma Grimaldi

[3] Recommender system, Wikipedia