MovRec

Hybrid Movie Recommendation Systems

Fanpan Zeng
School of Information
University of Michigan
Ann Arbor, Michigan, USA
zfp@umich.edu

Xinchun Li
School of Information
University of Michigan
Ann Arbor, Michigan, USA
xincli@umich.edu

ABSTRACT

In this paper, we outlined different methods to implement movie recommendation systems. The goal is to recommend users satisfactory movies that are tailored to their taste and of decent quality. We have used IMDB 5000 movies dataset and the part of movielens 20m dataset for such an experiment. The approaches we used are content-based recommendation, collaborative filtering recommendation and clustering-based recommendation, each of them based on different thinking processes. For the collaborative filtering system, the evaluation metrics we adopted is RMSE. We were able to achieve an average RMSE of 0.8 for the collaborative filtering recommender system. Further work on improving recommender system robustness and exploring new ways of implementation is to be done.

1. PROBLEM

In modern society, it's common that ordinary users feel overwhelmed with hundreds and thousands of movies to choose from on every major network channel. There are many recommender systems out there of varying quality and the problem of being not very personalized. And there are also cases in which the recommender systems are too complicated to be helpful. We want to build solid practical recommender systems that would actually help users. If users want to search for movies of similar topics or themes, or get recommendation from a reliable source, they can get reasonably satisfactory ones that meet their needs.

2. METHODS

In the recommender system, the essential question is how to ensure a user will like a item. Usually, there are two strategies: 1. Content-based filtering: to learn the content the user may like then find something match 2.Collaborative filtering: assume what user will like by referring to other similar users choices [1]. To optimize the recommendation, we will try to combine two approach and explore more.

2.1 Content-based approach

Content-based filtering methods focus on both the profile of the user's preferences and the item description in order to recommend items that are most similar to the items that are highly rated in the past [4]. The basic idea behind is that if one love something, he/she would like the similar ones. To define a reasonable item-based system, we first took a look at the features of our IMDB 5000 movie dataset with different features. We want to build a system based on the similarity of the content across different movies. Thus, we made a new column called "content" that combines 'plot_keywords', 'director_name', 'actor_1_name", 'actor_2_name" and "actor_three_name" together, as we believe these core elements are what defines movie content.

Then, we vectorized "content" using "TfidfVectorizer", "CountVectorizer" from [1]sklearn.feature_extraction package, and based on the values in the matrix, calculated the cosine similarity index, which is the similarity between different movies under this assumption. Cosine similarity is a measure that calculates the cosine of the angle between two vectors [2]. Mathematically, it is formulated as:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Our first try-out content-based recommender system was based on the cosine similarity index of the "content" matrix after vectorization and built with the input of movie titles, and output of the movie titles of ten most similar movies.

We further improved our content-based recommender system by counting in imdb score, the number of votes, the

[1]https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

number of user reviews and the number of critic reviews of each movie. The idea is that although our first priority is to recommend movies that cater to the taste of individual users, we also want to make sure that the movies are of decent quality, which is shown through "imdb_score", and not too unpopular, which is shown through "num_voted_users", "num_user_for_reviews" and "num_critic_for_reviews". We added an additional filtering "above_average" to make sure all these scores of our recommended movies are above the mean of these individual scores across different movies.

2.2 Collaborative Filtering

2.2.1 Matrix Factorization: Singular Value Decomposition (SVD)

Besides similar contents, it is proven that the similar crowd could be the other reliable source for recommendation. The method utilizing similar users' opinion and preference to make prediction to the individual is called Collaborative Filtering, and it has the assumptions: 1) Users with a common interest will have similar preferences 2) Users with similar preferences share the same interest [1]. There are two approach based on the assumption: one is memory-based, which utilizes entire user-item database to predict current users' preference by retrieving similar users' preference; the other is model-based which uses only a set of ratings to train the model and predict the unrated one. In our case, we choose to apply the model-based approach to the movielens dataset because it can overcome the problems of sparsity, cold-start as well as the scalability issues which the memory-based approach suffer from [3].

Matrix factorization (MF), as an unsupervised learning method for latent variable decomposition and dimensionality reduction, is well-known for its advantage in dealing with scalability and sparsity issue. Singular Value Decomposition (SVD) is one of the most important matrix factorization methods usually used for reducing the number of features of a set of data, which can improve the recommender systems [4]. At a high level, SVD is an algorithm that decomposes a matrix A into the best lower rank (i.e. smaller/simpler) approximation of the original matrix A. Mathematically, it decomposes A into a two unitary matrices and a diagonal matrix:

$$\mathbf{A}_{nxp} = \mathbf{U}_{nxn}\ \mathbf{S}_{nxp}\ \mathbf{V}^{\mathbf{T}}_{pxp}$$

Where the columns of U are the left singular vectors (gene coefficient vectors); S (the same dimensions as A) has singular values and is diagonal (mode amplitudes); and VT has rows that are the right singular vectors (expression level vectors). The SVD represents an expansion of the original data in a coordinate system where the covariance matrix is diagonal [6].

In the movie recommendation system, users' rating is the input data matrix A, user "features" matrix representing user's preference to the movie (ratings) can be taken as the left singular vectors U, the diagonal matrix of singular values (essentially weights/strengths of each concept) is S, and V^T is the right singular vectors (movie "features" matrix) representing how relevant each feature is to each movie [5]. Our system turns out to be a user-based collaborative filtering.

2.2.2 Combine clustering: k-means-SVD-based recommendation

To improve the accuracy of recommendation, we also tried the K-means clustering technique which theoretically group the users into the cluster with the fewer and closer users in comparison with the general population [4]. Our idea is: firstly, cluster users based on their ratings using K-means. Then for each cluster, we apply SVD to obtain the decomposition matrices.Finally output the recommendation model.

2.3 Solely Clustering-Based Approach

Inspired by the experiment in the k-means-SVD-based method, we tested the solely clustering-based approach. This approach is simple but intuitive. The first way is to cluster users based on their ratings using K-means. And we would recommend movies rated by other users in the same cluster that the user we recommend to hasn't watched before. The other way is to cluster movies based on the movies' average ratings using K-means. And based on the number of clusters the user's watched-movies are in, we would recommend a given number of movies to users that are in the cluster that has the most number of movies this user has watched in the past, in descending order of the movies' average ratings.

3. RESULTS

In the research and experiment above, we found collaborative filtering has the solid evaluation methods compared to other method. To evaluate the results of collaborative filtering approach, we use the measure called root-mean squared error (RMSE).

$$RMSE(P) = \sqrt{\frac{1}{|P|} \sum_{(\hat{r},r)\in P} (\hat{r} - r)^2}.$$

The metrics measure the accuracy of predict ratings r^ given by actual ratings r across all user-item pairs [1]. For our dataset, we use the Python scikit [2]Surprise library that provided various ready-to-use powerful prediction algorithms including (SVD) to evaluate its RMSE (Root Mean Squared Error), which is widely used in evaluating recommender system. We measure the RMSE of full data applied SVD, and the truncated data for demo (300k). The mean RMSE of demo data is little better than using full data, so we continue to use demo data for SVD plus clustering technique. In K-means-SVD, the RMSE performance is not improved as expected, but in some clusters, the Mean RMSE decreased much and could be as low as 0.24 (Table 1).

Table 1 RMSE of SVD, K-means-SVD

Data/ Method	Mean RMSE	Best RMSE
Full data (20m) by SVD	0.8251	0.8247
Demo data (300k) by SVD	0.8096	0.8090
Demo data (300k) by K-means-SVD	0.9201	0.2405

Overall, we combine content-based filtering method and collaborative filtering method in our system. It allows users to pick the movie that they like and the content-based system would recommend movies that similar in terms of the keywords. Alternatively, users can rate the movies that they watched, and the collaborative filtering system would recommend the movies rated the top 10 predicted by the model. The system we presented is deployed on Flask (https://github.com/onceuponapril/650final) which contains mainly the content-based and collaborative filtering without the improvement and evaluation. The whole experiment including improved content-based, K-means-SVD and evaluation could be found as "650demo.ipynb" in https://drive.google.com/drive/folders/1T0SwEIHTsknohBS9f37ofj32die25a0w

[2] https://surprise.readthedocs.io/en/stable/index.html

4. DISCUSSION

From this project, we developed a good understanding of commonly used methods in building recommender systems including content-based filtering and collaborative filtering, and how to implement them on real dataset.

For content-based filtering, we assume that, obtaining more information of the movies, such as description, user reviews could be helpful. In the future we would love to apply natural language processing techniques on these text-based information to dig the highly related movies. However, the most detrimental problem we met in building content-based filtering is the reasonable measurement about the accuracy of this kind of recommendation system. We believe it is an important topic for content-based recommendation system, and in need of the exploration.

For collaborative filtering, we adopt rating as the major input for prediction. And we tried methods to improve the performance including matrix factorization and clustering. In this system, we think the key is to find the right metrics and find the right group for recommendation. In our project, we choose movies ratings as metrics, it is feasible to use other metrics like user reviews' semantic meaning (negative/positive) in the future. And to find the right group, more clustering technique could be involved.

Overall, we tried the methods to improve the performance by enriching item profile, using clustering technique, etc., by which we have better idea in building recommender system. In the future, we hope to carry out more and detailed work in better combining different methods together and improving the precision/decreasing the error rate of our system.

5. ACKNOWLEDGEMENT

We sincerely thank all those who have helped our project at any stage, especially Professor Qiaozhu Mei and Graduate Student Instructor Yanshi Yan. We couldn't do it without the kind support from everyone.

6. REFERENCE

[1] ChengXiang Zhai and Sean Massung.2016. Text Data Management: A Practical Introduction to Information Retrieval and Text Mining. ACM and Morgan & Claypool Publishers.

[2] Christian S. Perone. 2013. Machine Learning : Cosine Similarity for Vector Space Models (Part III). Retrieved from http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

[3] Gilda Moradi D akhel, Mehregan Mahdavi. 2011. A new collaborative filtering algorithm using K-means clustering and neighbors' voting, 2011

11th International Conference on Hybrid Intelligent Systems (HIS), DOI: 10.1109/HIS.2011.6122101
[4] Hafed Zarzour, Ziad Al-Sharif, Mahmoud Al-Ayyoub, Yaser Jararweh. 2018. A New Collaborative Filtering Recommendation Algorithm Based on Dimensionality Reduction and Clustering Techniques.2018 9th International Conference on Information and Communication Systems (ICICS)DOI: 10.1007/978-3-319-89743-1_56
[5] James Le. 2018. The 4 Recommendation Engines That Can Predict Your Movie Tastes. Retrieved from: https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223
[6] MIT. Retrieved from http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm
[7] Rounak Banik. 2017. Movie Recommender Systems.Retrieved from https://www.kaggle.com/rounakbanik/movie-recommender-systems/notebook