IJASRM

International Journal of Advanced Scientific Research and Management, Vol. 1 Issue 12, Dec 2016.

www.ijasrm.com

ISSN 2455-6378

A Movie Recommender System using Tweets Data

P.Visalakshi[1],

[1] Asst.Professor(S.G).,Department of Computer Science and Engineering,
SRM University, Kattankulathur Campus,
Kancheepuram, Tamilnadu, India

Abstract

Nowadays, we are living in an age of recommendation. Amazon stays ahead of the curve in the e-Commerce industry by personalized recommendation of items shoppers might like based on past orders; Google news generates click through rates by showing relevant content to readers; Trip Advisor provides different hotel rankings for different users; Last.fm displays "Play your recommendations" button on the home page to attract users; Netflix achieves 2/3 of its movie views by recommendations. In this paper, it has been explored various methodologies of building a movie recommender system. MovieLens, Netflix and other companies distributed a couple of datasets for public research. However, these datasets are becoming outdated and can hardly incorporate new items and ratings. Analysis and Investigation based on these datasets would not have an up-to-date effect. Inspired by a paper which extracts rating information from twitter [1], I have collected rating data by similar methods, extracted more information about the movies by referencing to the movies' IMDb pages. With such data stored in database and updated automatically. It is required to build a movie recommendation engine with the objective to minimize Root Mean Square Error (RMSE) of user ratings. I explore various models such as collaborative filtering, content-based model and SVD. Eventually I would like to come up with a hybrid model which can take advantage of each of the models to cope with the cold-start problem, popularity bias problem, and sparsity problem. To facilitate the use of our recommendation system, I have created a data entry interface where users can rate a few movies from a given list. The web interface will be connected to my recommendation engine by using Python Flask/Django library. I will recommend 5 movies that the user might like.

Keywords: *AMAZON, MOVIELens, RMSE (Root Mean Square Error), Netflix*

1. Related Work

Researchers use different metrics to evaluate recommendation quality. Common metrics fall into two categories. One is ***statistical accuracy metric***, which evaluates the accuracy of the predicted ratings versus true ratings. Mean absolute error (MAE), root mean square error (RMSE), and correlation between predictions and ratings are representatives for this category. Another kind of ***metric (decision-support accuracy)*** involves transforming original numerical ratings into binary variables (high/low ratings) by defining a rating threshold. For example, movies with grades 4-5 are considered good-quality, while those with 1-3 are low-quality. Then misclassification rates are computed [6]. I can also compute ROC instead. Decision-support accuracy measures how effective the recommendation engine is at recommending users high-quality items [7]. I chose RMSE as the metric in our project, because choosing a threshold to convert numeric ratings into binary is a little arbitrary. Two main approaches to building a hybrid system combining collaborative filtering (CF) and content-based (CB) models have been proposed in literature. The first approach is a linear combination of both methods. For example, fitting a linear model on the predicted values from CF and CB. Claypool uses different weights of CF and CB for different users: the more items the user has rated, the larger the weight of the CF is [8]. Another way is a linear combination of individual content features and predicted ratings from CF [7]. The second approach is sequential combination of CF and CB. Item-item similarities are defined by content features. Then CF is applied to make the prediction [9] [10]. I used the first approach in our project. Because this independent combination

IJASRM International Journal of Advanced Scientific Research and Management, Vol. 1 Issue 12, Dec 2016.
www.ijasrm.com
ISSN 2455-6378

allows us to improve CF and CB separately, and interpret individual contribution from both models.

2. Dataset and Features

I scraped data through Twitter API similar to the method in the paper [1]. The original tweets come from rating widget of IMDb apps. Whenever a user rates a movie, a structured tweets in the form of "I rated [movie title] [rating]/10 [IMDb link] #IMDb" will be generated. Twitter API also returns a json object for each tweet which contains the user's Twitter ID and account information. I have referenced to the IMDb link in the tweet by OMDb API to collect the movie's information such as the release year, actors, genre and descriptions. This data parsing process occurred in Python. I have created a mysql database to store the user information, the movie features and the rating. To facilitate data sharing and storage, I connected the local database to AWS RDS. Those two steps were also realized by Python. I integrated the ETL process of tweets collection, data parsing, and database loading through Unix shell, which calls the Python function to update automatically and periodically. As a preprocessing of our dataset, we omitted those users who rate less than 5 movies and those movies with less than 5 ratings for better model quality. To split the data into training set and test set, I used a random number generator to choose 40% of users. For each of those 40% users, we selected half of the ratings into the test set.

3. Methods

In this section we describe the models we used in building our recommendation engine.

3.1 Baseline Model

In this baseline model, I have used $_{xi}r^{base} = \mu + b_x + b_y$ *to estimate the rating for item i by user x.* μ *is the global mean rating;* b_x *= (avg. rating for user_x)-* μ *is the rating deviation of user x; and* b_i *=(avg. rating of item i)-* μ *is the rating deviation from i.*

3.2 Item-based Collaborative Filtering

The main concept is item-item collaborative and memory based algorithm. First and foremost I have computed similarity between every pair of items i and j and stored the similarity information in a table. When making the prediction of the user x on item i, I have referenced back to the table for similarity between item i and other items the user has rated. In this computation I have used rating and similarity information to make the prediction. For this, I have used Pearson correlation as the similarity measure. **(i.e)**

$$Pearson\ (x,y) = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

3.3 Content-based model by using movie features actors/year/genre

When the process of pruning, I have pruned out 50% of predictions needed to be calculated in the original set, and then I randomly select 50,000 predictions for each run. In addition, we also prune out users whose average rating is either higher than 4.5 or lower than 1.5. The intuition is that if users have such high or low average ratings, it means that those users always rate movies either very high or very low. The expectation of those users will continue this rating behavior.

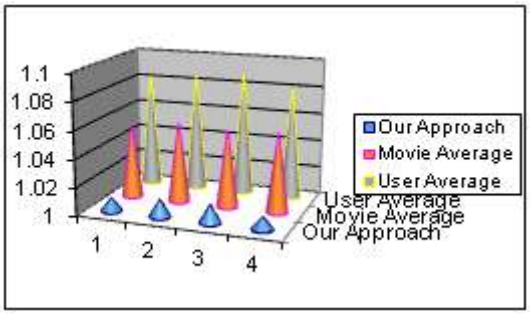


Fig 1 : RMSE Comparison

To evaluate the accuracy of our algorithm, I have compared the RMSE to the RMSEs obtained by using the average of the movie to predict, and the user's average rating. I have selected tp2 = 0.6 and tp = 4 because I find that using those values yield the best RMSE. In Figure (4), the x-axis show the number of run, and the y-axis show the RMSE. It is obvious that my approach is around 4% better than using the average of the movie I am trying to predict as prediction heuristic, and it is 7% better than using the user's average rating as prediction heuristic. The movie average prediction heuristic performs better than the user average prediction heuristic because even I consider the movies that are rented less than 40,000 times, most of the movies are rented more than 1,000 times. Therefore it is still possible for me to use the average rating for those movies, but those predictions are less accurate compared to those movies that are rented for more than 40,000 times. I

also want to compare the my algorithm to the RMSE obtained from using a wild guess approach, meaning that we give a rating 3 for all of the predictions we are trying to calculated. Figure (5) shows the result. In Figure (5), x-axis shows the number of run, and the y-axis shows the RMSE. The graph shows that my algorithm is almost 30% better than the result obtained from wild guess prediction heuristics. Moreover, it is very surprising to see that the averaging prediction heuristics are not very inaccurate. One of the reasons could be the size of our data set. In my data set, each user has rented 209 movies on average, and each movie is rented 5,655 times on average. Given such a huge data set, using averaging schemes should not be very inaccurate.

Even though our approach is able to achieve a RMSE that is at least 4% better than the averaging scheme, I expect our approach to perform much better. The algorithm does not perform as expected because in the data set, I encounter the major drawback of collaborative filtering technique, which is the new user and new item problems. In my data set, there are over 8,000 movies that are rented less than 500 times, and there are over 244,072 users who have rented less than 100 movies. Whenever we are trying to predict the ratings of those new movies for those new/inactive users, my Pearson Correlation Coefficient would not give much meaning info, and my weighted averaging scheme is error-prone.

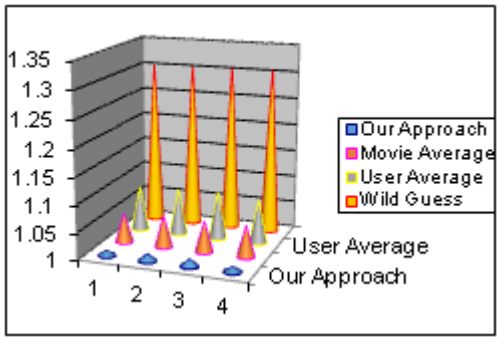
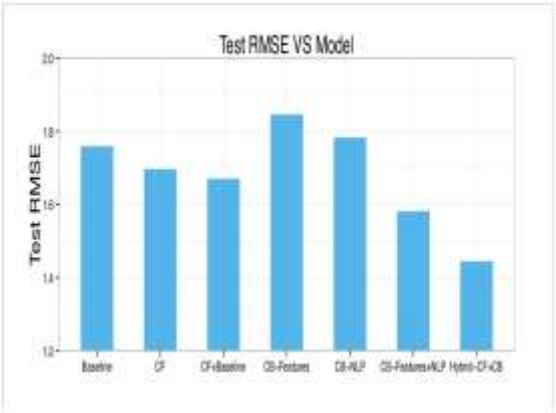


Fig 2 : RMSE Comparison with the Wild Guess Approach

4.Results

For my content model in 4.4, we chose the top 10 words with highest TF.IDF score as the movie features. I tried a range of values from 1 to 20, and eventually selected 10 based on their test RMSE.
I evaluated each model by their test RMSE. The result is shown in the plot. CF (4.2) improves over baseline model (4.1). Two content-based models (4.3, 4.4) individually don't perform as well as the baseline. The content-features are solely based on objective information about the movie, which doesn't ensure that a user will like it. The quality of recommendation will likely to improve if the user rates more movies. But a combination of the two CB models (4.5) outperforms baseline. The hybrid model (4.6) achieves the lowest RMSE among all other models. CF contains information of human opinions, while CB mainly incorporates information of the movie features. The two models are complementary. The hybrid model takes advantage of both CF and CB, and thus improves the prediction result



5.Conclusion

In this paper, I have presented my approach on building a movie recommendation system. I addressed the challenges of computing accurate rating by first using a straightforward scheme. It turns out that scheme does not work very well since the scheme used the information from only one neighbor to calculate predictions, and the algorithm of determining a neighbor was incorrect. Therefore, we decide to use Pearson Correlation Coefficient and weighted averaging as our prediction heuristic. I run the algorithm on 5 randomly generated sets and evaluate the result. The result is better than just using averaging scheme by at least 4%. In the data set, we are able to identify characteristics of the users and the movies, which help us to prune out some of the prediction calculation. Moreover, we encounter the new user and new item problem in our data set, so our algorithm did not perform as expected.

Reference

[1] Gediminas Adomavicius and Alexander Tuzhilin *"Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions"* IEEE Transactions on Knowledge and Data Engineering, Vol 17, No 6, June 2005.
[2] Dan R. Greening *"Building Consumer Trust with Accurate Product Recommendations"*, A

White Paper on LikeMinds Personalization Server.
[3] Jonathan L. Herlocker, Joseph Konstan, Al Borchers, and John Riedl *"An Algorithmic framework for Performing Collaborative Filtering"*.
[4] The argument regarding the importance of the long tail in on-line systems is from [2], which was expanded into a book [3].
[5] Discusses the use of computer games to extract tags for items.See [5] for a discussion of item-item similarity and how Amazon designedits collaborative-filtering algorithm for product recommendations. There are three papers describing the three algorithms that, in combination,won the NetFlix challenge. They are [4], [6], and [7].
[6] G. Adomavicius and A. Tuzhilin, "Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," IEEE Trans. on Data and Knowledge Engineering 17:6, pp. 734-749, 2005.
[7]C.Anderson,http://www.wired.com/wired/archive/12.10/tail.html
[8] C. Anderson, The Long Tail: Why the Future of Business is Selling Less of More, Hyperion Books, New York, 2006.
[9] Y.Koren, "The BellKor solution to the Netflix grand prize, "www.netflixprize.com/assets/GrandPrize2009 BPC BellKor.pdf 2009.
[10] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," Internet Computing 7:1, pp. 76–80, 2003.
[11] M. Piotte and M. Chabbert, "The Pragmatic Theory solution to the Net-flix grand prize, "www.netflixprize.com/assets/ GrandPrize2009, BPC Pragmatic Theory.pdf 2009.
[12] A. Toscher, M. Jahrer, and R. Bell, "The BigChaos solution to the Netflix grand prize,www.netflixprize.com/assets/GrandPrize2009 BPC BigChaos.pdf 2009.
[13] L. von Ahn, "Games with a purpose," IEEE Computer Magazine, pp. 96–98, June 2006.