A Movie Recommender System from Tweets Data

Mengyi Gao
m1gao@stanford.edu

Xiang Zhang
x265zhan@stanford.edu

1. Introduction

Nowadays, we are living in an age of recommendation. Amazon stays ahead of the curve in the eCommence 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; TripAdvisor 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 project, we want to explore 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 However, these datasets are becoming outdated and can hardly incorporate new items and ratings.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], we collected rating data by similar methods. We extracted more information about the movies by referencing to the movies' IMDb pages. With such data stored in database and updated automatically, we want to build a movie recommendation engine with the objective to minimize Root Mean Square Error (RMSE) of user ratings. We explore various models such as collaborative filtering, content-based model and SVD. Eventually we want 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, we will create a data entry interface where users can rate a few movies from a given list. The web interface will be connected to our recommendation engine by using Python Flask/Django library. We will recommend 5 movies that the user might like.

2. 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]. We can also compute ROC instead. Decision-support accuracy measures how effective the recommendation engine is at recommending users high-quality items [7]. We 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]. We used the first approach in our project. Because this independent combination allows us to improve CF and CB separately, and interpret individual contribution from both models.

3. Dataset and Features

We scraped data through Twitter API similar to the method in the paper [1]. 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. We 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.

We created a mysql database to store the user information, the movie features and the rating. To facilitate data sharing and storage, we connected the local database to AWS RDS. Those two steps were also realized by Python. We 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, we 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.

4. Methods

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

4.1 Baseline Model

As our baseline model, we used $\hat{r}_{xi}^{base} = \mu + b_x + b_i$ to estimate the rating for item i by user x. μ is the global mean rating; $b_x = (avg.rating.of.user.x) - \mu$ is the rating deviation of user x ; and $b_i = (avg.rating.of.item.i) - \mu$ is the rating deviation for item i. The baseline model has a test RMSE of 1.7. For all of our later models, we will reference back to this baseline for comparison.

4.2 Item-based Collaborative Filtering

Item-item collaborative filtering is a memory-based algorithm. First we computed the similarity between every pair of items i and j, and stored the similarity information in a table. When making a prediction for user x on item i, we referenced back to the table for the similarity between item i and other items the user has rated. We use the rating and similarity information to make the prediction. Here we defined the Pearson correlation as the similarity measure. Specifically, $sim(i,j) = S_{ij} = \frac{\sum_{x \in U_{ij}} (r_{xi} - \overline{r}_i)(r_{xj} - \overline{r}_j)}{\sqrt{\sum_{x \in U_{ij}} (r_{xi} - \overline{r}_i)^2} \sqrt{\sum_{x \in U_{ij}} (r_{xj} - \overline{r}_j)^2}}$, where S_{ij} is the set of users who rated by both item i and item j.

The estimated rating is $\hat{r}_{xi}^{CF} = \mu + b_x + b_i + \frac{\sum_{i,j} S_{ij}(r_{xj} - b_{xj})}{\sum_{i,j} S_{ij}}$. Note that b_x and b_i have the same definition as our baseline model in 3.1. The Item-based collaborative filtering has a test RMSE of 1.65.

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

For our first content-based model, we constructed item profiles, which consisted of movie features i.e. actors/year/genre. We defined binary variables of 1's and 0's for each feature in our training set. This gives each movie an array of binary features. Next, we built user profiles. For each user, the user profile is defined by the profiles of the movies he has rated and what score he gave. Movies with higher ratings contribute more to the user profile. Then we computed cosine similarity for each user-movie pair. We fit a linear regression model on the user-movie similarity S_{xi} and global mean μ: $\hat{r}_{xi}^{CB1} = \beta_0 + \beta_1 \mu + \beta_2 S_{xi}$.

4.4 Content-based model by using movie description (NLP)

For our second content-based model, we used some NLP techniques. For each movie, we selected 10 words with highest TF.IDF scores as the movie features. Then we combined the selected words from each movie in the training set as a long vector. If a movie contains a certain word, the component for that word is 1 and 0 otherwise. Next we built user profiles, computed user-movie similarity, and fit the linear regression model as part 3.3 in the above. This gives us: $\hat{r}_{xi}^{CB2} = \beta_0 + \beta_1 \mu + \beta_2 S_{xi}^{NLP}$.

4.5 Content-based model Combined

We combined the user-movie similarity measures from 3.3 and 3.4 into one regression model. Our model now looks like: $\hat{r}_{xi}^{CB} = \beta_0 + \beta_1 \mu + \beta_2 S_{xi} + \beta_3 S_{xi}^{NLP}$.

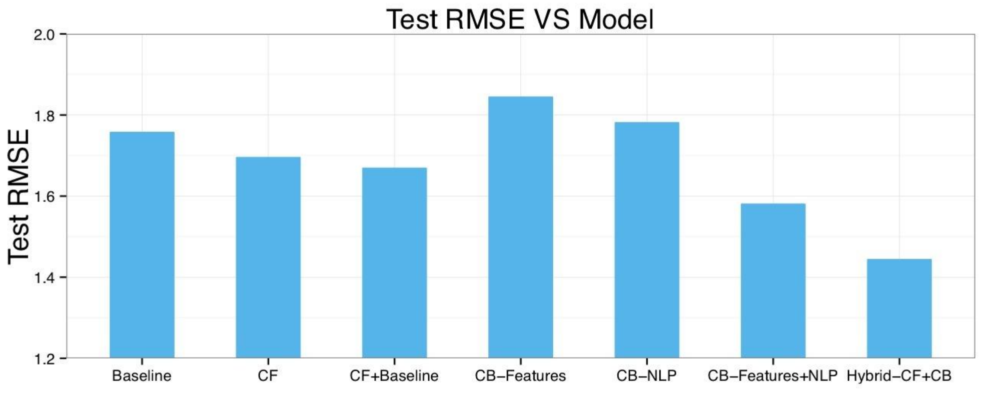
4.6 Hybrid model using Collaborative Filtering and Content-based model

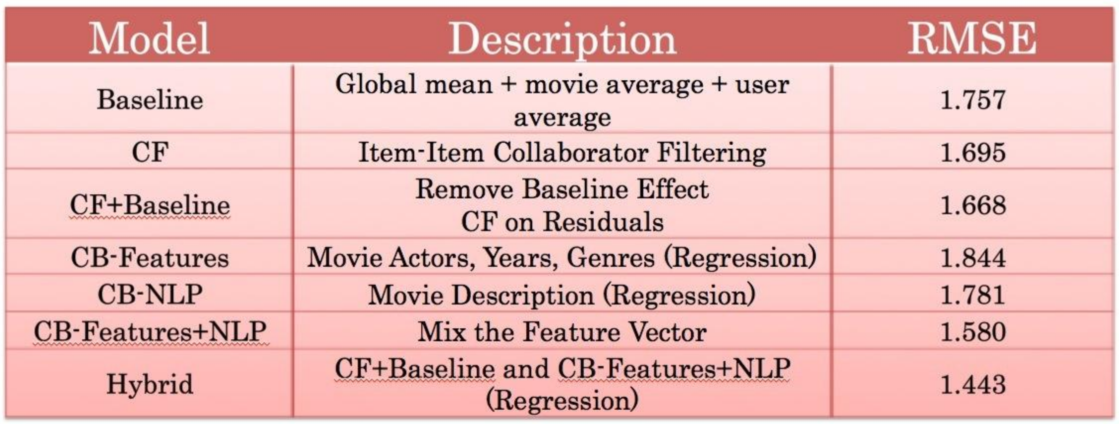
We treated the predicted ratings from 3.2 and 3.5 as input variables, and fit a linear regression model: $\hat{r}_{xi}^{Hybrid} = \beta_0 + \beta_1 \hat{r}_{xi}^{CF} + \beta_2 \hat{r}_{xi}^{CB}$.

5. Results

For our content model in 4.4, we chose the top 10 words with highest TF.IDF score as the movie features. We tried a range of values from 1 to 20, and eventually selected 10 based on their test RMSE.

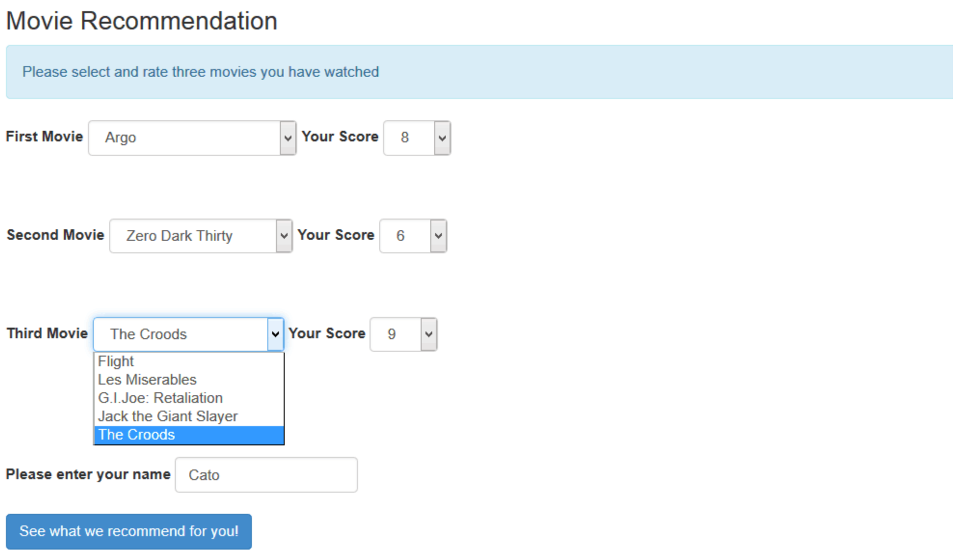
We evaluated each model by their test RMSE. The result is shown in the plot. CF (4.2) improves over baseline model (4.1). Two content-based models (4.3, 4.4) individually don't perform as well as the baseline. The content-features are solely based on objective information about the movie, which doesn't 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.





Model	Description	RMSE
Baseline	Global mean + movie average + user average	1.757
CF	Item-Item Collaborator Filtering	1.695
CF+Baseline	Remove Baseline Effect CF on Residuals	1.668
CB-Features	Movie Actors, Years, Genres (Regression)	1.844
CB-NLP	Movie Description (Regression)	1.781
CB-Features+NLP	Mix the Feature Vector	1.580
Hybrid	CF+Baseline and CB-Features+NLP (Regression)	1.443

We also created a web user interface. Users can rate 3 movies, and we will give 3 recommendations based on their ratings.


Movie Recommendation

Please select and rate three movies you have watched

First Movie: Argo — Your Score: 8

Second Movie: Zero Dark Thirty — Your Score: 6

Third Movie: The Croods — Your Score: 9

Flight
Les Miserables
G.I.Joe: Retaliation
Jack the Giant Slayer
The Croods

Please enter your name: Cato

6. Future Work

As a next step, we plan to run bias and variance diagnostic and error analysis. We will explore more NLP methods and take advantages of existing packages like word2vec and NLTK. Improving the efficiency of content-based model is also an area to work on.

References

[1] Simon Dooms, Toon De Pessemier, Luc Martens. MovieTweetings: a Movie Rating Dataset Collected From Twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, (2013).

[2] Jure Leskovec (2015). *Recommender Systems: Content-based Systems & Collaborative Filtering* [PowerPoint slides]. Retrieved from http://web.stanford.edu/class/cs246/handouts.html.

[3] Jure Leskovec (2015). *Recommender Systems: Latent Factor Models* [PowerPoint slides]. Retrieved from http://web.stanford.edu/class/cs246/handouts.html.

[4] Xavier Amatriain (2014) *Recommender Systems: Collaborative Filtering and other approaches* [PowerPoint slides]. Retrieved from http://www.slideshare.net/xamat/recommender-systems-machine-learning-summer-school-2014-cmu.

[5] Toby Segaran. Programming Collective Intelligence: Building Smart Web 2.0 Applications. *O' Reilly Media*, (2007).

[6] C. Christakou, A. Stafylopatis, A hybrid movie recommender system based on neural networks, in *International Conference on Intelligent Systems Design and Application*s, 2005.

[7] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker and J. Riedl, “Combining Collaborative Filtering with Personal Agents for Better Recommendations”, In *Proceedings of AAAI, Vol. 35, 1999.*

[8] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.. 1999. Combining contentbased and collaborative filters in an online newspaper, In *Proc. ACM-SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*.

[9] L. Qing and M.K. Byeong, “An Approach for Combining Content-based and Collaborative Filters”, In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages, 2003.*

[10] S. Grant and G. McCalla, “A hybrid approach to making recommendations and its application to the movie domain”. In *2001 Canadian AI Conference, 2001.*