Netflix Movie Rating Prediction using Enriched Movie Metadata

Wei Tu, Chung Wu, Michael Yu

1. Introduction

The Netflix Prize is a competition hosted by Netflix, Inc. to find a better algorithm for predicting how a user would rate a new movie. Netflix provides a test set consisting of over 100 million ratings generated by over 480 thousand users on over 17 thousand movies to use for training. Because of its one million dollar prize, the Netflix Prize has drawn attention from scientists and researchers across the world. BellKor in BigChaos, the team currently in the lead, has published a series of papers documenting their approach. According to their paper, "Improved Neighborhood-based Collaborative Filtering" [1], recommender systems use either of two strategies: content-based approach and collaborative filtering. Their team has spent over two years perfecting their collaborative filtering algorithm, so we decided to explore a content-based approach.

Since the Netflix data set provides very little data for each movie -- only its title, the ratings from the users and the date of the ratings -- we turned to the Internet Movie Database [2] for richer metadata. We also experimented with clustering sparser metadata like actors and actresses. We then ran experiments on predicting ratings with and without the richer metadata. We found that enriching that enriching our baseline collaborative filtering approach with movie metadata only made a small improvement of 0.1% in the root mean squared error (RMSE) of our predictions.

2. Baseline setup

In order to predict user ratings, we viewed each user as a separate regression problem. This allowed us to narrow down the data to a manageable size and to personalize our predictions for each user. To learn the hypothesis for a single user u, we trained a support vector regression machine. We created one training example for each movie m that u has rated. The score / target value for a training example is u's rating of m. The features of the training examples the ratings of movie m from each of the top 10,000 users (as ranked by total number of ratings per user). This feature set, which does not include any movie metadata, forms our baseline model.

2.1 Normalizing ratings

Some users give higher or lower ratings on average than other users. The variance of a user's ratings can also differ from user to user. To compensate for these differences, we normalized every rating by first centering the mean of each user's ratings to 0 and then dividing each rating by the standard deviation of that user's ratings. In other words, a normalized rating is the number of standard deviations the rating is from user's mean. After training a model and making predictions, we then transform the normalized rating predictions back to a 1 to 5 scale, capping any ratings that fall outside this range.

Using a radial basis function (RBF) kernel, we found that normalizing the ratings improved the RMSE of the baseline model by 2.94%. Using a linear kernel, normalization had almost no effect; it increased the RMSE of the baseline by 0.06%.

3. Enriching Netflix data with IMDb metadata

We used a combination of tools [3, 4, 5] plus a lot of custom code to load the IMDb data into our MySQL database. One challenge we faced was in mapping Netflix DVD titles to IMDb entity names. The two databases use slightly different punctuations, abbreviations and capitalizations. For foreign films, Netflix also used translated English titles, while IMDb used original-language titles transliterated into English. And since the basic unit of entity in the Netflix database is a DVD while the basic unit of entity in IMDb is more abstract (e.g. a film or a TV episode), there are often no logical mapping for Netflix entities like "Friends: Season 2". We employed various heuristics to work around these issues, but the results are still far from perfect.

3.1 Turning metadata into features

To turn the meta data into features, we took the "bag of words" approach, where our vocabulary is a dictionary of all the genres, countries, languages, director/writer/actor names that appeared at least 3 times among the movies the particular user has rated. Each word is a binary feature for a movie. For each movie that the user has rated, we a feature to 1 if the corresponding word appears in the movie's metadata.

4. Clustering people in the IMDb social graph

We were concerned about mapping each actor to a unique feature in the input vectors. Since the actor features are very sparsely populated, it can be rare for one actor to show up more than once for a user's movie set. For example, if a user has seen and liked many Brad Pitt movies, then the Brad Pitt feature is a good predictor of ratings. But if he has only seen but also liked Jackie Chan and Jet Li only once each, we should still capture some fondness for Asian martial arts actors, even though there is no direct actor feature overlap. Similarly, we might predict that he will enjoy a movie with Bruce Lee, even though he has never seen a Bruce Lee film. Intuitively, then, if we could cluster Jackie Chan, Jet Li and Bruce Lee together into the same feature, we can make better predictions for other films with actors in the same cluster.

We found in the IMDb data set a massive social graph of directors, writers, actors and actresses working with each other over the span of more than a hundred years. If we build a graph where the nodes are people, and draw an edge each time a pair of people worked together on a movie, then this graph would have almost two million nodes with more than 300 million edges! We decided to leverage this social graph to cluster people into groups, whose members have all worked together with similar people. Because there are so many people, we first divided the people into four non-exclusive groups -- those who have performed in the role of director, writer, actor or actress. Note that even when we are clustering only directors, we still leverage the entire graph and make use of their work history with writers, actors and actresses.

4.1 Clustering people using K-means

We used K-means to cluster people. Each entity to cluster is a person and his work history. Conceptually, it is a vector in n-space, where n is the number of people in the dataset. The i-th element of the vector is the number of edges from this person to the i-th person. We normalize each vector to a norm of 1.

The "centroid" of a group of people is also a vector in n-space, where the i-th element

stores the number of people in this group that have worked with the i-th person. Intuitively, the i-th element stores a "popularity" measure of the i-th person in this centroid. The centroid vector is also normalized to a norm of 1.

The difference function between a person and a centroid is simply the Euclidean distance between the two corresponding vectors. This distance measure awards those who have worked with "popular" people in the centroid, and punishes those who have not, and those who have worked with a lot with people that are not well-represented in the centroid. It is very effective at keeping the cluster sizes relatively even so that everyone does not simply gravitate towards the same cluster.

There were about 8k directors, 15k writers, 75k actresses and 145k actors who have worked on the Netflix movies. For actors and actresses, we decided to only include people who have worked on five movies or more, thus skipping one-time actors who likely have no predictive power in a user's rating. This reduced our number of actors and actresses down to about 34k and 67k, respectively. We clustered each group into 50 clusters using 10 iterations.

The cluster results tended to be intuitive. The director clusters were the most insightful. For example, the Pixar directors -- John Lasseter, Brad Bird and Andrew Stanton -- fell into one cluster, and popular entertainment machines like Steven Spielberg and George Lucas fell into another. Clustering for actors and actresses, though, put most of the popular, modern Hollywood people into the same clusters. To more effectively separate that critical group of people, we ran k-means again on just the "famous" actors and actresses, and found finer-grained clusters. We used results from all six clusterings to build our feature vectors.

5. Experiment setup

To measure the utility of movie metadata for predicting user ratings, we ran experiments using three different models that differ only in their feature sets:

Baseline Model:

- The ratings of movie `m` from each of the top 10,000 users (as ranked by total number of ratings per user).

Unclustered Metadata Model:

- All features from the Baseline Model.
- The genre, language, countries, directors, actors, actresses, and writers for movie `m`.

Clustered Metadata Model:

- All features from the Baseline Model.
- The genre, language, and countries features from Unclustered Metadata Model
- Director clusters, actor clusters, actress clusters, Hollywood actor clusters, Hollywood actress clusters, and writer clusters for movie `m`.

We used libsvm to train our SVM [6]. Libsvm is a widely used SVM library that provides many useful features, such as support for common kernels, cross validation, and automatic parameter selection. After some initial experimentation, we found the linear kernel yielded the best performance. So, we used a linear kernel and a regularization parameter of C = 1.

Due to time and machine resource constraints, we trained hypotheses for 1000 randomly

selected users out of 480 thousand users total. For each user, we split the data into two equally sized sets, one for training and one for testing. We trained hypotheses using each of the three models mentioned above and computed the average RMSE of each model over the 1000 users. We also computed the RMSE of a simplistic hypothesis function that always outputs the average rating that all users have given to a movie.

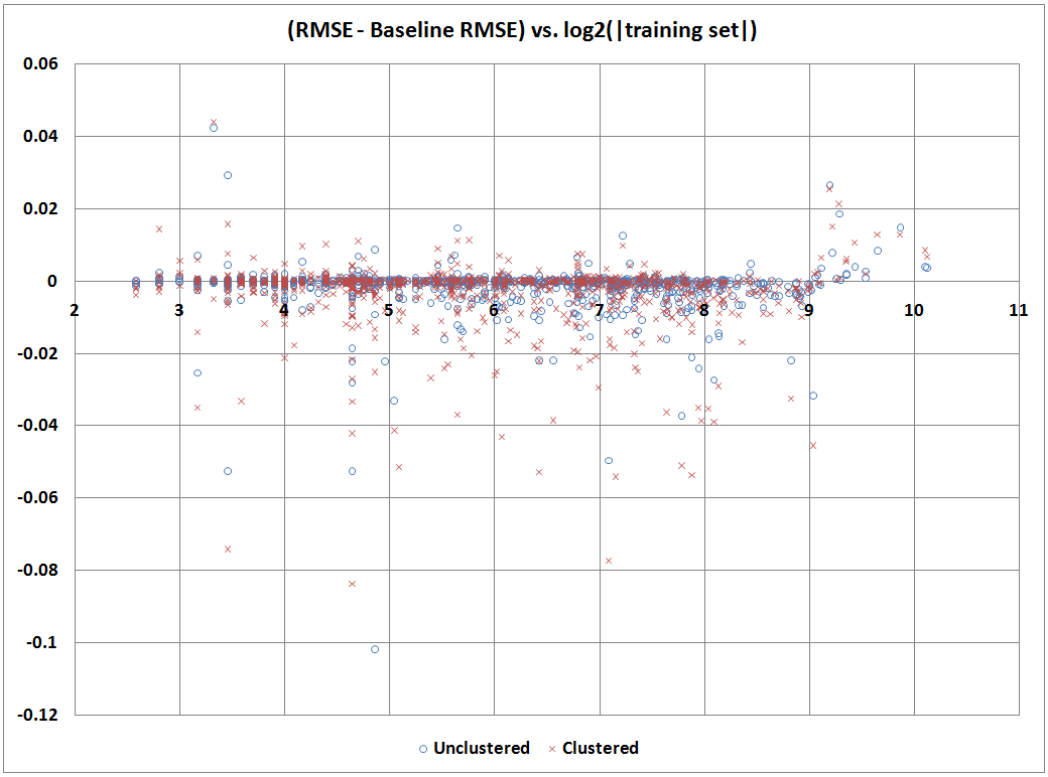
6. Experimental results

The following table contains the observed RMSE on our test sets for the 1000 random users:

Model	Always Avg. Rating	Baseline	Unclustered	Clustered
RMSE	0.99542	0.95957	0.95881	0.95797

Our baseline model achieved an RMSE that was within 1% of the RMSE achieved by Netflix's Cinematch system, despite using only half of the available training data. Adding movie metadata to the model yielded very little improvement in the RMSE, however. The unclustered metadata model decreased the RMSE by less than 0.08% over our baseline. Clustering the metadata did appear to be somewhat effective, but the overall improvement was only 0.16% over our baseline.

The following chart shows the absolute change in RMSE from the baseline using the clustered and unclustered metadata models vs. the size of the training set:



The clustered metadata model tends to have a significant effect on the RMSE (both positive and negative) for a larger number of users compared to the unclustered metadata model. This makes sense, as the unclustered features are more sparse. Both metadata models

appear to significantly increase the RMSE for users with a very high number of ratings. For these users, the test error was significantly higher than the training error, which suggests overfitting.

7. Future Work

There are several potential improvements we could make to our recommender system that do not depend on how movie metadata is used. We found that the RMSE was higher on average for users that had very few movie ratings (under 20). For these users, we could try to cluster them with more prolific users and use the movies these prolific users rated as training examples, effectively increasing our training set. This will also help our algorithm scale better; we would only need to train a model for each cluster of users that have similar taste instead of a model for each individual user, making it more feasible to cover all 480k users in the test set using a reasonable amount of resources.

Along the same lines, if we can calculate the distance between users based on taste, we could use the ratings from the users that are most similar to the user we are training the model for as features. The ratings from these users may be more predictive than the ratings from the top 10k most prolific users, which we are currently using.

We also feel there are many improvements possible for our clustering distance function. For example, one goal would be to try to end up with small-sized clusters of highly-influential people and large-sized clusters of less-significant people. The clusters could be leveraged to fill in the gaps in similarity in less popular people while still allowing very popular people that are in most users' data sets to have maximal influence.

Finally, even though that our experiment showed that the metadata we've collected does not improve the quality of the recommendation engine significantly, we are still optimistic that some metadata will. One interesting experiment to try is to build another vocabulary of movie's plot keywords and add those as features.

8. References

[1] R. M. Bell and Y. Koren, "Improved Neighborhood-based Collaborative Filtering", *Proc. KDD- Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.
[2] Internet Movie Database. At http://www.imdb.com
[3] Internet Movie Database plain data files. Retrieved on November 15th, 2008, at http://www.imdb.com/interfaces#plain
[4] Col Needham's Command Line Search Tool for the Internet Movie Database. Retrieved on November 15th, 2008, at http://www.imdb.com/interfaces#unix
[5] IMDbPY 3.8, a python interface to IMDb data. Retrieved on November 15th, 2008, at http://imdbpy.sourceforge.net/
[6] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm