Netflix Recommendation System
CS 484 Project Report

Niha Imam

May 11th, 2020

Netflix is all about connecting people to the movies they love. They frame themselves as lovers of movies and people who use technology and data to do a certain sort of matchmaking. Netflix accomplishes this by using recommendation systems to personalize as much of Netflix as possible. Recommendation systems have always been the black box or the known unknown. Tech companies such as Google, Amazon, Facebook, and many more have their version of. But these recommendation systems have often been kept a secret. We live in a world where algorithmic information processing affects the meaning of the word culture, so it is easy to understand why such recommendation systems are kept locked behind doors, laws, patents, and trademarks.

On their blog, Netflix has revealed only a small part of their best-kept secret and its most valued asset, their recommendation system. In 2006 Netflix announced the Netflix Prize competition. It was a machine learning and data science competition for movie rating prediction. They offered a million dollars to anyone who could improve the accuracy of their existing system, Cinematch. All they wanted was an algorithm that could beat their RMSE of 0.9525 and reduce it to 0.8572 or less.

As Netflix's competition happened in 2006, I don't have access to the original data, but I found a dataset on Kaggle. The dataset contains 100 million ratings by 50000 people. the data set is huge. It is way bigger than the world has ever seen or maybe will see. Before anything, I had to get started on understanding the data.

For a given user we had to predict the rating they would give to another movie. With a dataset, this big most of the time I had spent was on preprocessing. I started this project like any other, trying to write a bit of code and running all of it and checking but that wasted a lot of time. for this project, I switched over to the jupyter notebook. It's much better than running all the code over and over again and it allowed me to visualize the data.

Like I said that the competition was in 2006, so there was no way of testing my algorithm on the qualifying dataset and submitting to get results. So, I had to work with the 100 million ratings provided by 50000 users and a list of movie titles. My first step was to accumulate my data and divide it into useful chunks. The data files included a movie ID followed by multiple user ID's, their rating, and the date. So, a data point would look like

76:3455593,4,2005-05-02

76 – represents the movie id

3455593 – represents the user-id

4 – represents the rating given by user 3455593

2005-05-02 – represents the date

Now was time to preprocess the data and clean it up. My first step was to create a data frame out of the data files. I combined all four data files and saved it as a .csv file for later uses. The next step was to visualize the data and some exploratory data analysis on the combined data. The data means nothing to me if I can't extract any viable information out of it. I then used the combined data to create a data frame and arranged the data according to the date provided. Then moved on to the usual preprocessing part of removing any null values and checking for duplicates. After that, I was left with 100480507 ratings of 17770 movies from 480189 users.

Since the data was too big, I had to split the data into the train set and test set. Since the data was already sorted by date, I used the first 80% for the training data and the last 20% as the test data. The train data had 80384405 ratings of 17424 movies from 405041 users while the test data has 20096102 ratings of 17757 movies from 349312 users. It is important to note that the data was split on ratings so the same movies or users could in both sets.

After reading multiple articles, I learned that when working with big data it is good to gather some information on it. This preliminary information gathering is called Exploratory Data Analysis or EDA. EDA is used to answer questions, test assumptions, generate a hypothesis, or even prep the data for modeling. For EDA I used the training set to find the Gaussian distribution of the data, and the average ratings per user and a few more. The distribution of the data using ratings was as expected, people giving the rate of 3 or 4 more rather than 1,2 or 5. INSERT HERE. The average rating per user showed that some users have rated >17k movies while most have only rated a few. This shows us that the data is very skewed. If we delve deeper into the average ratings per user, we can see that 50% of the users rated around 90 movies. I also decided to look into the average ratings

given by a user. The results were very skewed again. The popular movies have been rated by multiple people while the more niche movies have very few ratings. The 80-20 rule applies to data as well, the top 20% have been rated a lot while the bottom 80% hasn't.

After doing EDA I had to move on to prepping the data to be used in models. The EDA had proven that my data was extremely sparse, so I created a sparse matrix. This would help me in using dimensionality reduction and other operations. After creating a sparse matrix, I also found the average of movie ratings, the average rating per user, and the average rating per movie.

Recommendation systems work based on similarity. In my case, I needed to know the similarity between movies and the similarity between users. After a lot of googling and going over the notes, I found that the similarity matrix works best for this type of data. to start I needed to compute the similarity between two users. Calculating user similarity would not be easy especially on my MacBook Air with only 8gb of ram. But since the data was extremely sparse, I was able to create a similarity matrix that computes the top 50 similarities for each user. This process took 30 mins. Then I also had to create a similarity matrix between the movies. This process only took 12 minutes because we only have 17770 movies. we are calculating the top 100 similar movies and storing them in a hash table or a dictionary? So, if in the future we want the top 10 similar movies, we can easily access it.

Now it was time to start modeling the data. since the data set, I was using was huge I wanted to sample it. I randomly sampled data with 10000 unique users. I created a sample train sparse matrix from train sparse matrix and a sample test sparse matrix from the test sparse matrix.

After reading Netflix's blog, they had shed some light on to what models were used by the winning team. Netflix had published an incomplete list of models a person who is working in machine learning for personalization should know about. The blog discussed an incomplete list of models that one should know if they are working in machine learning for personalization. A few that stood out to me were linear regression, singular value decomposition, association rules, gradient boosted decision trees, K-means clustering, and matrix factorization. I intended to use all of them to conduct experiments to see if I could lower my RMSE and MAPE scores.

I started by following the blog post. I used the surprise baseline model. I used the svd matrix, factorization model. I also used the knn baseline model, but none had a good RMSE. The least RMSE I had gotten was from svd, but I was nowhere as low as Netflix's RMSE. I started researching again on how to raise my RMSE. It was that week we had started learning about boosting in class.

After a little more research, I came across XGBoost. XGBoost provides gradient boosting. Gradient Boosting is a machine learning technique for classification and regression problems that produces a prediction from an ensemble of weak classifiers. With XGBoost I can use multiple models on the data and use them as features.

To start I wanted to use XGBoost on regular data and then keep adding models to check RMSE of each. First, I had to featurize the train and the test data. what this meant was to create a data frame from the sparse matrices and add multiple features such as the top 5 most similar users and the top 5 most similar movies and the average movie rating and the average user rating and the average rating of the movie. The featurizing of the data took 12 hours even though we were only working with a small sample of the original data.

After I had featurized the data. I ran multiple models on it. I ran XGBoost with only the original features. Then I used the knn baseline classifier with user similarities and a knn baseline classifier with movie similarities. I used the results from both and ran another XGBoost. I then ran the matrix factorization SVD. I used the results as another feature and ran another XGBoost. Finally, I ran a surprise baseline model and ran yet another XGBoost.

Even after multiple attempts, my RMSE wasn't as low as I would have liked but that might be because we did not train on our entire dataset. In the future I would like to try again with the data set but try to use more than just a sample.

Resources

Blog, Netflix Technology. "Netflix Recommendations: Beyond the 5 Stars (Part 1)." *Medium*, 18 Apr. 2017, https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429.

Blog, Netflix Technology. "Netflix Recommendations: Beyond the 5 Stars (Part 2)." *Medium*, 18 Apr. 2017, https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5.

Deng, Houtao. "Recommender Systems in Practice." *Medium*, 5 Dec. 2019, https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a.

Luo, Shuyu. "Intro to Recommender System: Collaborative Filtering." *Medium*, 6 Feb. 2019, https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26.

Winning the Netflix Prize: A Summary. https://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/. Accessed 6 Apr. 2020.

"Recommendation Systems". http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

How to Build a Recommender System(RS) - Data Driven Investor - Medium. https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs-616c988d64b2. Accessed 6 Apr. 2020.

Prabhu, Tanu N. "Exploratory Data Analysis in Python." Medium, Towards Data Science, 10 Aug. 2019, towardsdatascience.com/exploratory-data-analysis-in-python-c9a77dfa39ce.

"Python Exploratory Data Analysis Tutorial." DataCamp Community, www.datacamp.com/community/tutorials/exploratory-data-analysis-python.

Mwiti, Derrick. "Boosting Your Machine Learning Models Using XGBoost." Medium, Heartbeat, 11 Feb. 2020, heartbeat.fritz.ai/boosting-your-machine-learning-models-using-xgboost-d2cabb3e948f?gi=106662e9468e.

"Source Code for Surprise.evaluate." Surprise.evaluate - Surprise 1 Documentation, surprise.readthedocs.io/en/v1.0.0/_modules/surprise/evaluate.html.