CSC 577 Final Report

LAW AND ORDER: SVD - Robert Kaszubski, Neelabh Pathak, and Christopher Wieneke

Presentation: https://youtu.be/hmSNyEULNqQ

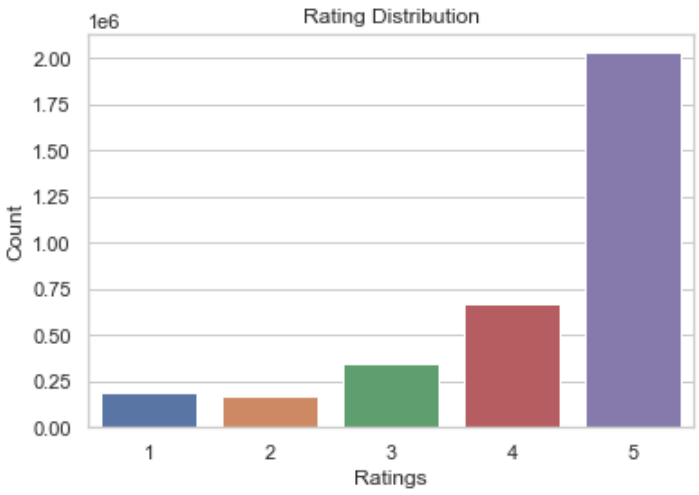
Dataset: https://nijianmo.github.io/amazon/index.html

Amazon TV & Movie Recommender Evaluation

The goal of this project is to provide TV and Movie recommendations based on user reviews. We will accomplish this by developing models and methods to predict a score based on similar products or users. We will be testing multiple different models primarily from the Surprise Recommender package, but also developing other methods not found in Surprise such as content based recommendation, and evaluating performance to determine optimal parameters and to find the best model for this dataset. Models will be evaluated primarily using RMSE, with percision@k and recall@k calculated for the top models.

Data Description

We are using the 5-core Amazon Movies and TV dataset meaning that each user has reviewed at least 5 movies or tv shows and each movie or tv show has at least 5 reviews. This provides us with a base for each recommendation where it won't be skewed by single reviewed users or products. The dataset features 3,410,019 reviews, and 12 variables stored in a JSON format. The variables included are the reviewer ID, the product ID ("asin"), the reviewer name, the number of "helpful" upvotes on the review, metadata of the product (bluray, dvd, 4k, etc), the review text, the rating ("overall"), the summary of the review, the review time, and any images the reviewer may have attached. From initial data analysis, we found that there are 297,529 unique products and 60,175 users in the dataset. The average rating for the full dataset was 4.22 with about 60% of the total reviews being 5 of 5, so the distribution of ratings is skewed fairly high. Only 21% of the reviews had a 3 star or lower rating. After aggregating at the product level, the average product had a review of 4.13. On a user aggregated level, the average rating was a 4.25 with 44.5% of users only giving a 5 star rating.



Due to the size of the data, we had to limit the data used in modeling to a core15 subset. This dataset trims the reviews down to where there are at least 15 reviews per person and per product in the dataset. Previously, we utilized the first one million rows of the dataset to combat memory issues running the models. This was a better solution to this problem. Our core15 subset contained a total of 1,246,518 ratings, with 39686 unique users, and 18952 unique items. The subset's distribution of ratings was fairly similar to the distribution of the entire dataset, thus we found this to be an appropriate method of reducing the amount of data.

Approach

The overall process for the project follows the pattern of training models on a training dataset, optimizing the training models, determining the best parameters, testing models on test dataset, and finally determining the best model overall. Below is more details about each part of the process including model choice, optimization methods, and evaluation methods. A couple of the custom models, neural network and text based, were too memory and process intensive to run on the whole dataset. We narrowed the dataset further to 100,000 rows for those two dataset to comply with computer limitations. Both of those models however followed the same process as the rest of the models just with the limited data.

The core15 dataset was first shuffled using a set seed, as the data was originally sorted in order of item, then split using a 80/20 train test split prior to running any models. Each model was trained on the training data and was evaluated on RMSE, precision@k, and recall@k. Using the three metrics we were able to tell how close the model performed to the actual ratings were to the predictions, the relevance of the predictions, and the accuracy of those relevant predictions.

For this project we chose several models from the surprise package and created a few custom models. We compared the similar models to determine the best model from each section before testing the final results. From the surprise package we looked at the following models.

- KNN Based: KNNBasic, KNNWithMeans, KNNWithZScore, KNNBaseline
- Matrix Factorization: SVD, SVD++, NMF
- Other: Slope One, CoClustering

The first of our custom models, the text based model, is based on creating similarities between the different products based on the text within the reviews. The similarities were used to determine the nearest neighbors for a given product. We then calculated the predicted value by taking a weighted average of the average rating for the neighboring items and the similarity scores.

The other custom model that we built used neural collaborative filtering (NCF) technique. Under this algorithm we adopted a multi layer representation to model a user-item interaction. The NCF model built used identity of user and an item as input feature vectors,can also be referred as embeddings.The user embedding and item embedding are then fed into a multi-layer neural architecture,which we term as neural collaborative filtering layers, to map the latent vectors to prediction scores.

For optimization, we primarily utilized grid search cross validation with 3 folds on the training data to go through many combinations of parameters for each model. Due to the size of the data, some models were optimized over more combinations at a time than others. The general process for optimization was including the default parameters and +/- some amount around the default. Once the RMSE stopped decreasing, that was considered the optimal parameters for that model. Certain models required some manual tuning as well, or multiple runs of grid search. For instance, the KNN Baseline model consisted of similarity options, baseline options, and K. Both similarity options and baseline options had 4 parameters each. Running grid search to find all of these parameters at once would take a long time especially when we were testing close to 10 different values for each parameter. In this case, similarity options were optimized first, then keeping those best parameters, a second grid search was run to optimize the baseline options. Additionally, for certain models we manually tuned the parameters found by grid search further to see if RMSE could be improved. The reason behind this is that while using grid search the difference between values was fairly large. For example, if we tested K = 5,10,15 using grid search , and it selected 10 as the best value, we would then manually try 9 and 11.

After optimization was completed, each model was evaluated using RMSE on the testing data. The models were then compared and the top three models were selected with the lowest testing RMSE. These models were further evaluated by calculating the precision@k and recall@k for k = 1 to k = 30. This was done to ensure models would be making relevant predictions (making recommendations of items with at least a 4 star rating) as RMSE alone is a general accuracy metric and does not evaluate this. The precision and recall values were plotted for each of the top three models, and an additional plot was made overlaying the precision-recall curve of each model for final comparison.

Results

Training Results

KNN Based

Model	RMSE
KNNBasic	0.2987
KNN Means	0.2844
KNN Zscore	0.2870
KNN Baseline	0.2850

Matrix Factorization

Model	RMSE
SVD	0.8691
SVD++	0.8674
NMF	0.6338

Other Models

Model	RMSE
Slope One	0.4435
Co Clustering	0.8749
Review Similarity	1.0698
Neural Network	12.27

Testing Results

KNN Based

Model	RMSE
KNNBasic	1.0633
KNN Means	0.9883
KNN Zscore	0.9880
KNN Baseline	0.9033

Matrix Factorization

Model	RMSE
SVD	0.9182
SVD++	0.9177
NMF	0.9434

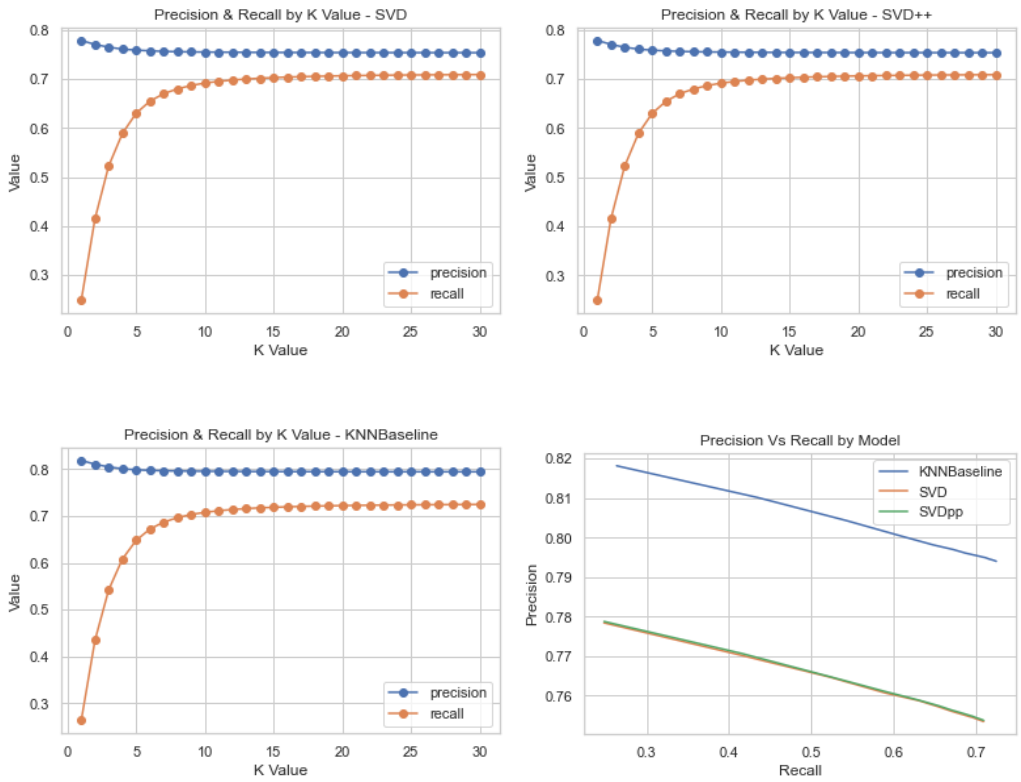
Other Models

Model	RMSE
Slope One	0.9499
Co Clustering	0.9519
Review Similarity	1.0901
Neural Network	12.26

Based on testing RMSE, we found the three best models to be KNNBaseline, SVD, and SVD++. Overall, many of the models performed quite well and had similar RMSE values. However, some models, primarily the KNN models and SlopeOne, had much lower training accuracies than testing accuracies, possibly indicating some overfitting. Although, this is somewhat expected as these are memory-based algorithms. The RMSE results of the content based review similarity model and neural network may be

slightly skewed due a limited dataset used in each. However, in the case of the neural network, the RMSE is so much higher than the other models that it's unlikely it would improve enough even with a larger dataset.

Precision@K - Recall@K - for best models



We can see that in terms of precision and recall, each model performed very similarly, with SVD and SVD++ being almost identical. However, KNNBaseline was the slightly superior model and evident by the final curves plot. Interestingly, the precision and recall never overlap as they typically would, so it's hard to identify the best K. The precision and recall remain high for each model at any K greater than 5.

Challenges:

The biggest challenge we encountered was the size of the dataset which causes memory and time issues while trying to train and optimize the models. As a result, we had to trim the dataset to the core15 as detailed above. While it makes for overall stronger recommendations, there are less products and users to train off of which make our models less scalable than if we were able to take advantage of all the reviews.

Conclusion

The best model overall was KNNBaseline with the lowest RMSE and highest precision/recall. However, SVD++ and SVD followed closely behind. The primary limitation of using KNNBaseline or any KNN model is that it is a memory based algorithm thus requiring far more memory and processing power. Likewise, for this data, it must be an item based recommender as there are far more users than items. In terms of speed alone, SVD models are far superior yet attain similar results. Other limitations included our neural network and content based models being trained and tested on a smaller subset of data, again due to memory and processing restrictions. Likewise, we believe that precision and recall may be skewed due to our dataset heavily favoring positive ratings. Thus, we primarily focused on RMSE.

As far as potential future work and improvements, we believe that with more time we could have also tested a hybrid recommender system. It may also have been possible to improve our neural network model further.