Movie Data Mining and Rating Predictions

Zhehui Chen
Statistics, Department of Statistics
zhehuic2@illinois.edu

Ziang Zhao
Statistics, Department of Statistics
ziang3@illinois.edu

Hui Yang
Statistics, Department of Statistics
huiy4@illinois.edu

1 INTRODUCTION

Data mining has become an extremely important and popular task in a lot of territories in our modern lives, which could help people or companies make decisions as well as extracting knowledge from existing dataset. This project consists of several data mining applications (including clustering analysis, recommendation system and prediction) to get some insights of the data and its final goal is to construct models with some novel deep learning architectures to predict the movie ratings. To begin with, we will first introduce our data and then make a summarization of all the tasks we will complete together with some detailed and referenced methods and algorithms which are included in the models.

2 DATASET

The project uses the Movie Dataset [5] (an ensemble of data collected from TMDB and GroupLens) which consists of basic movie features and ratings for more than 45,000 movies listed in the Full MovieLens Dataset before July 2017. The full dataset consists of seven separate data files. Five of them record more than 20 movie features for all 45,436 movies, including cast, crew budget, revenue, posters and release dates. The other two files contain 26 million ratings from 270,000 users for all over 45,000 movies. Ratings are on a scale of 0-10 and have been obtained from the official GroupLens website. The dataset is a mix of categorical and numerical data, as well as text data to reflect the synopsis for each movie.

3 FRAMEWORK EXPLANATION

According to the introduction part, we will complete three major tasks in this project, which are clustering analysis, movie ratings predictions and finally movie recommendation system. The main task is the movie ratings prediction part and we will compare the performances of our predictive models to the two existing baseline approaches that are conducted by other groups in Kaggle. For this part, we will briefly introduce the framework of our models with some formulas and figures for further explanation.

The first task is to find the underlying groups among all movies and characteristics of top-rated movies which gives us some insights of the underlying patterns of the movie dataset we are using. Clustering algorithm [8] aims at finding similar observations in the whole dataset and group them into subgroups, in which objects in the same cluster are similar while objects in different clusters are as different as possible from each other. The inputs of the task are some numeric features of the movies with their corresponding averaged ratings. Thus, we will not take specific users' ratings into account. First, we will use a dimension-reduction method (t-SNE) to visualize underlying groups among all movies. Second, we will perform the K-means clustering algorithm to get the "optimal" number of clusters of movies and compare attributes of movies in different clusters to see any existing differences. The K-means algorithm uses Euclidean distance to measure the distance between objects and centers and an elbow method is used to determine the best K.

Table 1: Data Splitting

Train size	Validation size	Test size
20428(65%)	3142(10%)	7859(25%)

The second task is to construct three primary movie recommendation systems. The inputs of the three algorithms are the combinations of movie information together with users' rating for the movies they have watched. The simple recommendation engine is designed for the new users and it simply recommends the movies that have the highest average rating scores. For the content-based recommendation system, we calculate the Term Frequency Inverse Document Frequency (TF-IDF) vectors for the description features of movies and give recommendations based on the cosine similarities [2]. For the Model-based collaborative filtering, a well-known matrix factorization method, SVD, will be used first and then an ANN will be applied to show the improvement.

The third task (main task) is to predict the average ratings given features of new movies based on neural networks with NLP technique. For this part, we will use the *movies_metadata.csv* file which contains about over ten movie features with their average ratings. We split data into training, validation and test sets and the number of objects in each subset are shown in Table 1.

We train our neural network regressors based on the training data, compute loss and update parameters based on the validation data and evaluate the final performance based on the test set.

The Figure 1 is the overall framework of the predictive model, with details including the specific models, input/output dimensions. Parameter settings and optimization functions will be mentioned later. We split the data into batches with a batch size of 32. Unlike the two baselines which we will mention in the next section, we combine both text data and numeric features into our model. We split all movies features into three parts which are overview (a sentence that describes the main plot of the movie), keyword (a word list that contains the theme of the movie) and numeric features (a vector that includes numeric movie features). Figure 2 shows the architecture of our FC layers, in which we apply batch normalization for the input data, followed by FC layers, apply ReLU transformation together with dropout layers to avoid the problem of overfitting.

For overview data, we first find the maximum length of overview of all movies. We use the technique called word embedding which not only converts the words into a vector of real values with a length of embedding dimension (hyperparameter), but also identifies the semantics and syntaxes of the words with some embedding features. To be specific, we use a pre-trained word embedding model from FastText model [4] with embedding dimension of 128 and our

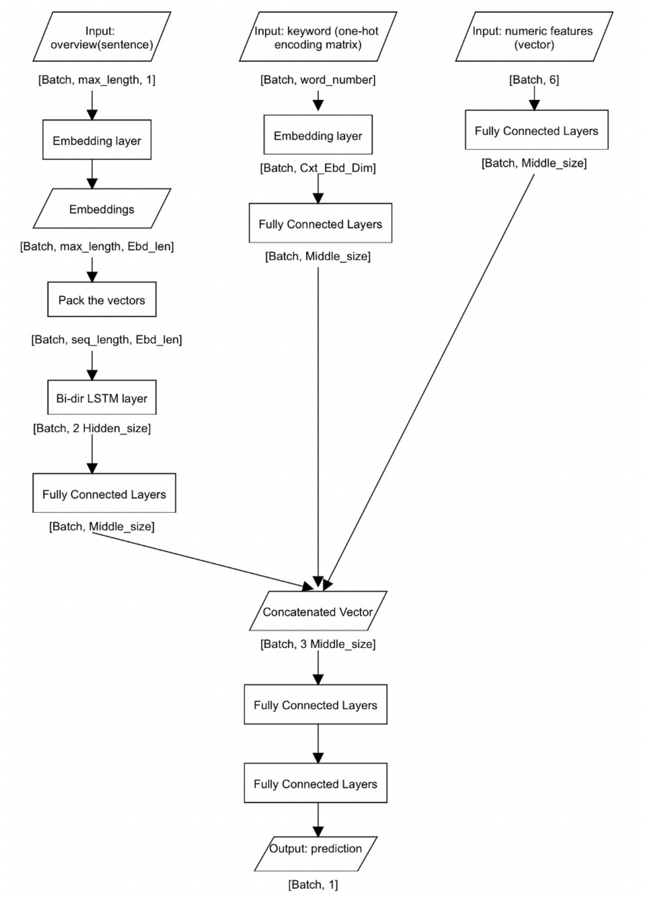


Figure 1: Prediction Model Framework

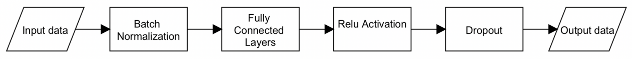


Figure 2: Architecture of FC Layers

embedding for overview data has dimension (18354, 128). Since the sentences have different length, we should pack the sentence vectors to their original length and then feed our word embedding into the most popular NLP model called LSTM [?] which is good at processing sequential data. We choose to use a Bidirectional LSTM model to improve the performance of our model and hidden layers with a size of 128.

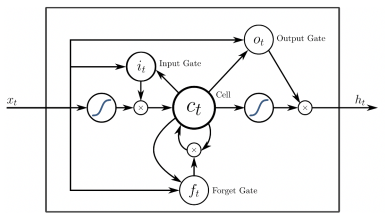


Figure 3: LSTM Cell

For the LSTM cell (Figure 3), we feed a list of words of a sentence and get the output of those input words. For the output, we only extract the output from the last layer for each word and concatenate all the outputs into a vector with size 256 (since the two directions double the number of hidden sizes). Then, we feed the output of our LSTM model into fully connected layers (with size 64) to get our vector representation for the overview text feature.

For keyword data which are separate keywords for each movie, unlike sentences data (overview), we do not need to consider the semantic relationships and contexts between each word. We create a python dictionary for each word with its corresponding frequency in the dataset. To build the vocabulary, we only select the words with frequency greater than 50 since too rare words are not useful for processing the model. Then, we get a vocabulary length of 515. For each data batch, we create the word one-hot encoding matrix [7] (with batch dimension of [32, 515]) using the $MultiLabelBinarizer$ function under python's sklearn module which converts movie keywords in a binary form. Then, we feed the one-hot encoding matrix to a embedding layer which we create using the Word2Vec model [10] with an embedding size of 128. The word embedding matrix is def into fully connected layers with size 64, and finally get our vector representation for the keyword text feature.

For the six numeric features, since they are vectors already, there is no need to do any transformations on them. We only need to feed the feature vectors into several fully connected layers (with an output size of 64) since these hidden layers could learn more complex and wise patterns of the feature vectors that are useful for making predictions. Finally, we get our vector representation for the numeric feature.

Having getting the three feature vectors of three kinds of input data, we concatenate them into a long vector (with a dimension of 192) and feed it into several fully connected layers. We train the model with 20 epochs. Since the outputs are numeric values, we use MSE criteria to compute loss and optimize our parameters using Adam optimizer. For gradient descent, we use a learning rate of 0.001 with learning decay of 0.95. Our final output is the predictions of each movie in the test set.

4 PARAMETER SETTING

Throughout our prediction model, the hyperparameters we used in the model include: batch size, embedding dimension, LSTM hidden layer size, output size of the three feature vectors, learning rate, number of epochs.

5 EVALUATION METRIC

Among the three tasks, we will only evaluate the performance of rating prediction models (clustering is an unsupervised learning technique and we do not have any labels for the accuracy of our clustering result; we could hardly evaluate the performance of our recommendation system since we do not know the users' future ratings for movies). The metrics we use are the Root Mean Squared Error (RMSE) and Mean Squared Error (MAE) with formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{1}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \quad (2)$$

where y_i is the true rating of the i-th movie and $\hat{y}_i$ is out predicted rating of the i-th movie. The less RMSE, the better is our model.

6 BASELINE APPROACHES

To perform the ratings prediction task, there are two existing methods performed by other groups in Kaggle competition.

The first baseline model we will compare is called Movie Rating Prediction by Movie Keywords which is an NLP-based method [1]. The goal is trying to predict the average ratings of movies given their corresponding plot keywords. Predictions are generated with a tensorflow artificial neural network. Since the original notebook uses users' ratings (in the scale between 0 and 5), we apply the same idea to the movie dataset (in the scale between 0 and 10).

We get the word dictionary for storing all the 19956 words. For the input of the model, we only keep the keywords with frequency greater than 100. Thus, the training features contain the keywords of each movie. The model is fitted using a 3-layer fully connected neural network optimized with Adam and mean square error. The distributions of predicted ratings and true ratings are displayed in Figure 4.

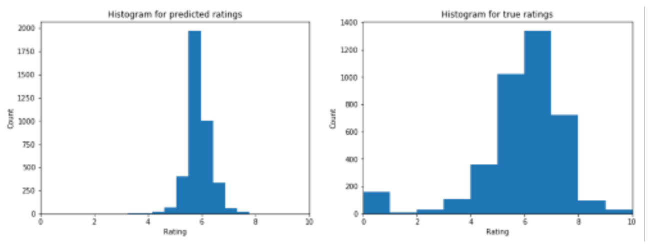
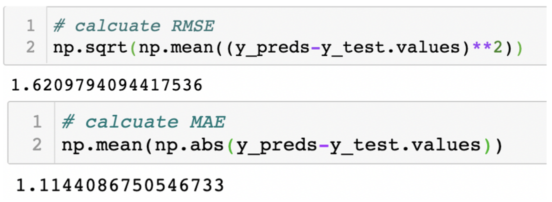


Figure 4: Baseline Model 1: Predicted Ratings

The RMSE is 1.621 and the MAE is 1.114 (Figure 5).



```
# calcuate RMSE
np.sqrt(np.mean((y_preds-y_test.values)**2))
```

```
1.6209794094417536
```

```
# calcuate MAE
np.mean(np.abs(y_preds-y_test.values))
```

```
1.1144086750546733
```

Figure 5: RMSE and MAE for Baseline Model 1

The second baseline model is called Movie Rating Prediction with Xgboost [3]. The method is based on the idea of XgBoost, which implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solves many data science problems in a fast and accurate way. The input explanatory variables include numeric features (budget, runtime, popularity, revenue and released year) and multi-hot features (genre). The method uses cross validation on the training dataset to tune best parameters (using xgb.cv) and the distributions of predicted ratings and true ratings are displayed in Figure 6.

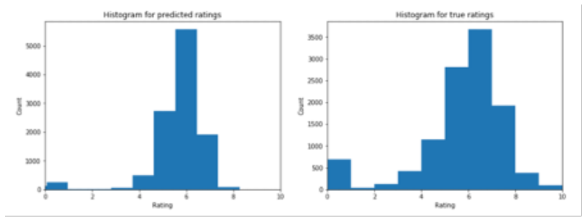
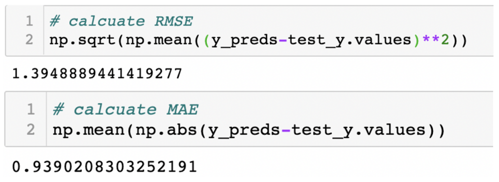


Figure 6: Baseline Model 2: Predicted Ratings

The RMSE is 1.395 and the MAE is 0.939 (Figure 7).



```
# calcuate RMSE
np.sqrt(np.mean((y_preds-test_y.values)**2))
```

```
1.3948889441419277
```

```
# calcuate MAE
np.mean(np.abs(y_preds-test_y.values))
```

```
0.9390208303252191
```

Figure 7: RMSE and MAE for Baseline Model 2

Here we should mention that we could hardly compare the goodness of these two baseline methods since the two methods use different sample sizes. Among all the 45463 movies in the whole dataset, there are 14822 movies (takes up one third of the whole dataset) that do not have any keywords (empty square brackets). Thus, there are far less training and testing samples for the keyword method compared to our method and the XGBoost baseline method.

7 RESULTS

7.1 Cluster Analysis

For this dataset, we will only use numeric features (since the categorical features has a lot of levels and it makes the data matrix too sparse). To visualize the distribution of this dataset, we create t-SNE plot [9] which models each high-dimensional object by a two- or three-dimensional point by applying dimension-reduction methods. In this way, the dimension-reduced dataset could be visualized in a 2-D plot.

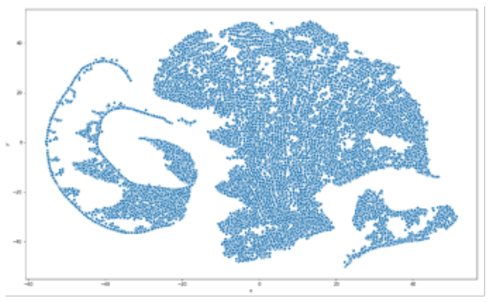


Figure 8: t-SNE Plot

The t-SNE plot captures structure in the sense that neighboring points in the input space will tend to be neighbors in the low dimensional space. In the Figure 8, we could see that there does exist some separations among movie objects since there are roughly some groups in the plot. To further verify the clusterability of our dataset, we introduce the Hopkins statistic. The null hypothesis of the test is that no meaningful clusters exist and it happens when the Hopkins statistic is around 0.5. For the movie's dataset, the Hopkins test is around 0.003 which is very close to 0, indicating that there are meaningful clusters in the space. Next, we select the best k (number of clusters) for the K-means algorithm using elbow method with average inertia (the sum of squared error for each cluster) metric. The relationship plot between average inertia and k is shown in the Figure 9.

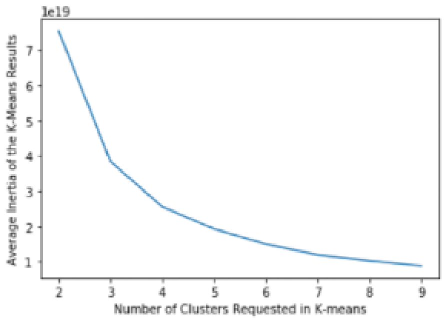


Figure 9: Elbow Method

To determine the optimal number of clusters (k) for this dataset, we have to select the value of k at the "elbow" point after which the inertia starts decreasing in a high magnitude. In this case, there is no significant "elbow" in the plot and we choose k=5 because the inertia becomes flat after k=5. We conduct the algorithm and the movies are grouped into five clusters. Same as before, we create the same t-SNE plot (Figure 10) and colored the points with their assigned labels.

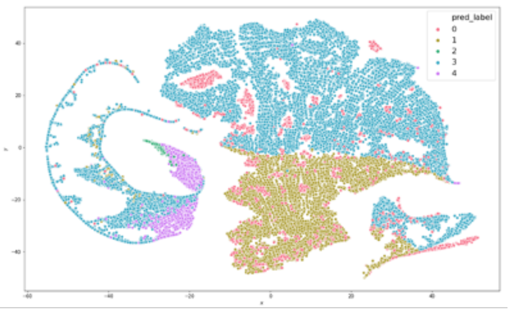
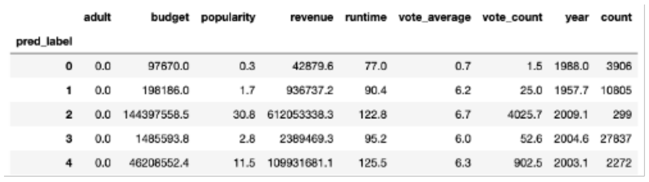


Figure 10: t-SNE Plot with Cluster Labels

We create a table to calculate the mean of features to compare different clusters and the result is shown in the Figure 11.

In the table, we could see that there are significant differences among the five clusters. The "count" column shows that cluster 1 and cluster 3 contain the largest number of objects and cluster 2 has the least number of objects. Movies in cluster 2 have the highest budgets, the highest popularity, the highest revenues and the highest average ratings, including movies like Star Wars, Forrest Gump, The Lion King and so on, which are all super popular movies throughout history. Movies in cluster 0 have the least budgets, the least revenues and the least average ratings. Movies in cluster 4 are less successful than movies in cluster 3, but still create great revenue. Movies in cluster 1 and cluster 3 occupy the majority movie market with no outstanding characteristics compared to other movies.



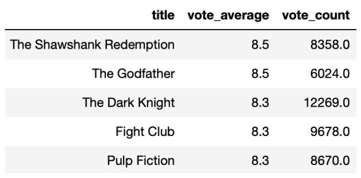
pred_label	adult	budget	popularity	revenue	runtime	vote_average	vote_count	year	count
0	0.0	97670.0	0.3	42879.6	77.0	0.7	1.5	1988.0	3906
1	0.0	198186.0	1.7	936737.2	90.4	6.2	25.0	1957.7	10805
2	0.0	144397558.5	30.8	612053338.3	122.8	6.7	4025.7	2009.1	299
3	0.0	1485593.8	2.8	2389469.3	95.2	6.0	52.6	2004.6	27837
4	0.0	46208552.4	11.5	109931681.1	125.5	6.3	902.5	2003.1	2272

Figure 11: Clusters Results

7.2 Recommendation System

The following part introduces a primary recommendation system with three different recommender engines to deal with the movie recommendation for both new and old users.

7.2.1 The simple recommendation system. The simple recommendation engine, which is for new user, is based on the highest average rating scores and regardless of users' preferences. By setting the threshold for minimum votes requirements, the recommended movies can be output directly (Figure 12). Other methods like weighted score used by IMDb [6] have similar performance.



title	vote_average	vote_count
The Shawshank Redemption	8.5	8358.0
The Godfather	8.5	6024.0
The Dark Knight	8.3	12269.0
Fight Club	8.3	9678.0
Pulp Fiction	8.3	8670.0

Figure 12: Top 5 movies

7.2.2 The Content-based recommendation system. The Content-based recommendation system focus on the attributes of the items and give the recommendations based on the similarity between them. The main idea is to apply NLP to the description part of each movie, including the title, the director, main actors, genres and keywords. Generally, the goal is to find Term Frequency Inverse Document Frequency (TF-IDF) vector for each movie and calculate the cosine similarity between them. The cosine similarity will act as score to help recommend the most similar movies based on the movie descriptions. Based on this recommender, we would recommend "Police Story" for the users who like "The Godfather", and "Blood of Redemption" for those who like "The Shawshank Redemption".

7.2.3 The Model-based collaborative filtering. The Model-based collaborative filtering is based on matrix factorization (MF) which has received greater exposure, mainly as an unsupervised learning method for latent variable decomposition and dimensionality reduction.

Notice that the user IDs are ordered sequentially with no missing numbers, but the movie IDs are not. Hence, we need to change them to a sequential set and then perform SVD matrix factorization model. The latent dimensionality k is chosen as 8 and the RMSE for this model is about 2.95, which is quite limited. This result has also been verified by Rounak Banik [2].

MF extracts the features automatically using only ratings and we can feed them into and neural network. Keras is used to train the ANN and the general idea is shown in the figure 13. After embedding the user vector and the movie vector, input them into ANN and tune the parameters.

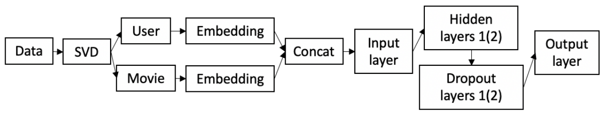


Figure 13: ANN-based recommender

Two hidden layers are used with hidden dimension equals to 200 and 10 respectively. The activation function is ReLU and dropout regulation are applied after each layer. The optimizer could be Adam with 0.001 learning rate or SGD with 0.08. The training and testing results are shown in the Figure 14.

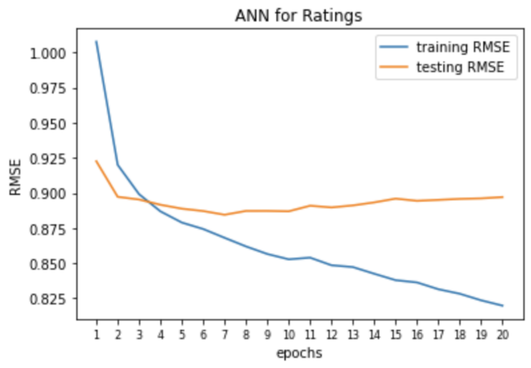


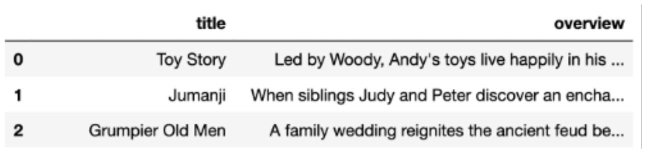
Figure 14: RMSE for ANN-based Recommender

Comparing with the SVD model, the RMSE for neural network are lower. For some more complex networks, the hyperparameters need to be chosen more precisely to avoid overfitting. Another direction is to combine the MF and NN together since MF methods work well in linear parts and NN can find nonlinear patterns.

7.3 Movie Rating Prediction

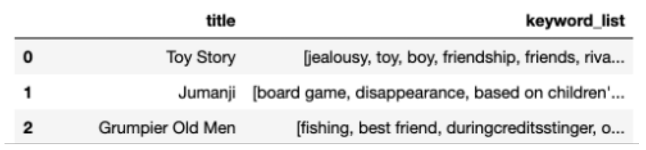
According to the framework in the explanation part, our proposed model contains three main parts of features (overview sentence (Figure 15), keyword lists (Figure 16) and numeric features (Figure 17)) and we would like to show how these data looks like.

We train our model with 20 epochs and the training loss and testing loss of each epoch is shown in Figure 18.



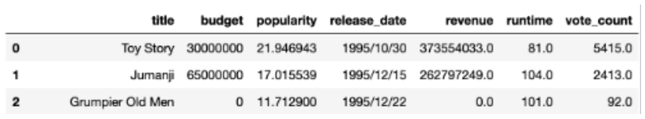
	title	overview
0	Toy Story	Led by Woody, Andy's toys live happily in his ...
1	Jumanji	When siblings Judy and Peter discover an encha...
2	Grumpier Old Men	A family wedding reignites the ancient feud be...

Figure 15: Overview



	title	keyword_list
0	Toy Story	[jealousy, toy, boy, friendship, friends, riva...
1	Jumanji	[board game, disappearance, based on children'...
2	Grumpier Old Men	[fishing, best friend, duringcreditsstinger, o...

Figure 16: Keywords



	title	budget	popularity	release_date	revenue	runtime	vote_count
0	Toy Story	30000000	21.946943	1995/10/30	373554033.0	81.0	5415.0
1	Jumanji	65000000	17.015539	1995/12/15	262797249.0	104.0	2413.0
2	Grumpier Old Men	0	11.712900	1995/12/22	0.0	101.0	92.0

Figure 17: Numeric features

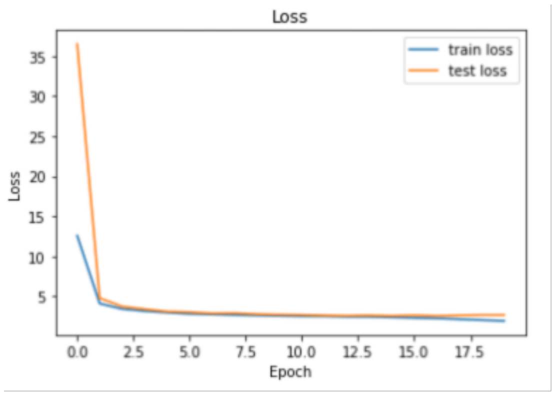


Figure 18: Loss for Rating Prediction Model

Then, we get the predicted ratings for each sample in the test set. The distributions of our predicted values and the true values are shown in Figure 19.

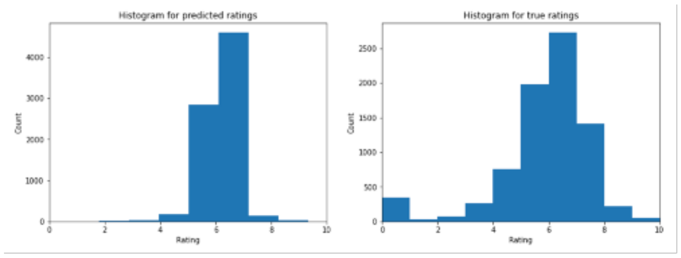
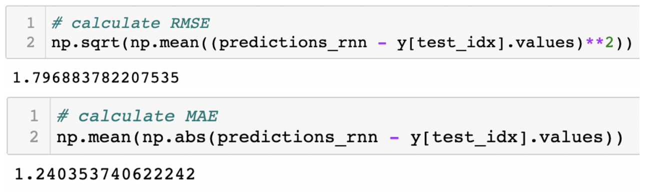


Figure 19: Predicted Value v.s. True Value

The histogram is kind of a bell shape which is very similar to the true distribution. However, more predicted values are around 6 and the model could hardly predict ratings with very low (below 2) or very high (above 7) values. To compare the performance of our model, we calculate the test RMSE as well as MAE and results are shown in Figure 20.



```
# calculate RMSE
np.sqrt(np.mean((predictions_rnn - y[test_idx].values)**2))
```

```
1.796883782207535
```

```
# calculate MAE
np.mean(np.abs(predictions_rnn - y[test_idx].values))
```

```
1.240353740622242
```

Figure 20: Test RMSE for Rating Prediction

8 IMPLICATION

In the results parts, we have specified our framework and develop a elaborated explanation for each task. We first explore the inner connection and patterns between the movies in the MovieLens dataset based on cluster analysis. The significant differences among different movie clusters bring us a thought to create recommendation systems based on movie contents and users' preference. Generally, to construct a better recommender, we need to improve the accuracy of our predictions, which leads to our main task of the whole project. By constructing the architecture shown in former parts, we could roughly match the distribution of the test dataset and perform rating predictions. In the following part, we will discuss the stability and disadvantages of our current framework.

9 ABLATION STUDY

For the rating prediction part, our "optimal" architecture contains three input parts which is an improvement of some lower-level architectures we tried earlier. If we delete some part of the model, there might be some problem which leads to a worse performance on the testing set. For example, we tried to only use the overview data to build the model, however, the result is awful.

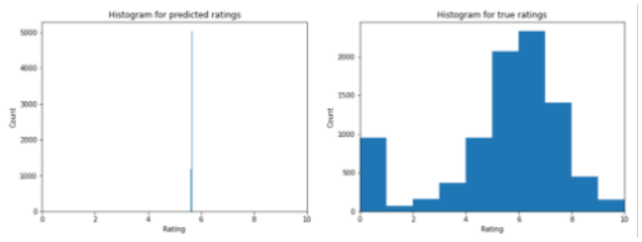


Figure 21: Predicted rating

We could see from Figure 21 that almost all the predictions are about 5.6 which is very close to the mean of the whole test data. This result indicates that our network might potentially not learning anything instead of using the average value.

For parameter analysis, some parameters play important roles in this model. Based on our tuning experience for our model, we find several rules. First, if we set the embedding dimension too large (i.e. over 256), the prediction distribution becomes tall and thin which is similar to the model with only overview data. Second, the middle size should not be too large (i.e. over 128). Third, if the learning rate is too large (i.e. 0.01), the loss could hardly monotone decrease which results in large error.

10 FUTURE WORK

By analyzing the result of our prediction model, we could see that even though the result could roughly match the distribution of the test dataset, the model could hardly predict zero ratings which is just the same as the baseline using keywords. In fact, zero ratings take up a somewhat large proportion of the dataset which results in a larger RMSE and MAE compared to the XGBoost baseline model. The reason behind this might be that text data is not that capable for capturing movie features which are important for regression. To solve this drawback, in future work, we consider introducing a binary classification model before our regression model. To be specific, we will first split the whole dataset into two classes, one is zero rating movies and the other one is non-zero-rating movies. By changing the response variable (rating) into categorical values and using the cross-entropy loss for updating the parameters, we could use the same architecture as in this project to first find the movies in the test set with zero ratings. Then, for the rest movies, we continue using our regression model to get non-zero ratings. By using this improvement, the error might be significantly decreases due to the offset of zero ratings (instead of using the majority value).

11 CONTRIBUTION OF MEMBERS

- Determine the main content of the project (Zhehui, Hui)
- Perform clustering analysis and complete task 1. (Zhehui)
- Find some referenced paper of natural language processing especially for the LSTM model that deals with sequential (word) dataset. (Ziang)
- Perform neural network regressors and LSTM to complete task 2. (Ziang, Zhehui)
- Construct movie recommendation systems and test their performance. (Hui)
- Evaluate the whole framework and the models of the project; provide suggestions to the future work.

REFERENCES

[1] Gabriel Atkin. 2020. Kaggle: Using Movie Keywords to Predict Ratings. https://www.kaggle.com/gcdatkin/using-movie-keywords-to-predict-ratings
[2] Rounak Banik. 2017. Kaggle: Movie Recommender Systems. https://www.kaggle.com/rounakbanik/movie-recommender-systems
[3] Fortimoi and Regina Yan. 2019. Kaggle: Movie Visualization, Recommendation, Prediction. https://www.kaggle.com/sjj118/movie-visualization-recommendation-prediction
[4] GENSIM. 2009. models.fasttext - FastText model. https://radimrehurek.com/gensim/models/fasttext.html
[5] Kaggle. 2017. The Movies Dataset. https://www.kaggle.com/rounakbanik/the-movies-dataset
[6] Manish Sahay. 2016. Quoting from IMDb. https://www.quora.com/How-does-IMDbs-rating-system-work
[7] Athif Shaffy. 2017. Vector Representations of Text for Machine Learning. https://athif-shaffy.medium.com/one-hot-encoding-of-text-b69124bef0a7
[8] Wikipedia. 2020. Cluster analysis. https://en.wikipedia.org/wiki/Cluster_analysis
[9] Wikipedia. 2020. t-distributed stochastic neighbor embedding. https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
[10] Wikipedia. 2020. Word2vec. https://en.wikipedia.org/wiki/Word2vec