**Predicting Movie Ratings with Plots Using NLP**

Group Members:

Jasmine Huang

MengChen Chung

**Introduction**

Movies have been an integral part of public entertainment, cultural discourse, political movement, and artistic expression since the public screening of the Lumière brothers' short films in Paris on 28 December 1895. Indeed, motion pictures have evolved drastically since the early years of cinematography, in terms of quality, length, audience, plot, composition, techniques, casts and so many other aspects. Movies provide social scientists with rich information on human society, psychology, and their boundless imaginations. Public tastes for movies, therefore, tell important stories of the wants and needs of individuals as well as the greater cultural social environments where they live.

In this project, we employ NLP techniques to analyze a large number of movie plots to predict public ratings of a movie, in hopes of better understanding the changes in the public taste for entertainment throughout time. Therefore, we will be exploring the following questions: *Can information from movie plots predict their average ratings? How do fine-tuned word embeddings perform compared to task-specific embeddings? Are different word embedding algorithms able to capture the nuance within the text and engender a better model?*

First, we created two baseline models (linear and neural network respectively) with relevant categorical and numerical features in the dataset. Then we perform basic sentiment analysis with the lexicon and rule-based Valence Aware Dictionary for Sentiment Reasoning (VADER) and include these metrics into the baseline model. Lastly, we move beyond sentiments of the plots and delve deep into their semantic meanings with word embeddings. We compare three different embedding methods- one trained on our plots dataset, a pre-trained GloVe (6B-d100) and a fine-tuned GloVe. We test all three embeddings on both linear and neural network models. We also fit an LSTM model with six layers and fine-tuned embeddings to see if it performs up to its theoretical superiority.

**Data**

Our dataset consists of over 20,000 movies between 2010-2019 in the United States from IMDB. This dataset is appropriate for our analysis for the following reasons: First, it covers one decade, allowing us to examine the movie trend changes over time. Second, it does not include the year of 2020, in which the movie market was strongly affected by the unprecedented pandemic. Finally, the size of the dataset is decently large. It contains 23,608 movies that have been aired in the US, and only 1248 movies have no plot descriptions. All in all, we have a large amount of textual data in English for analysis.

Nevertheless, some caveats need to be kept in mind for this dataset when analyzing and concluding the findings. First, the dataset does not cover the whole year of 2019 since the timing of scraping was in the middle of 2019. Second, the dataset has filtered out the number of votes less than 10, so that the ratings are not too biased by minor reviews. Thirdly, although we manually removed a few outliers that run extraordinarily longer, the distribution of runtime is skewed to the left with a long tail to the right. Additionally, the numVotes feature is also not normally distributed, with a slight skewness to the right. These unbalanced features may influence the performances of models that are sensitive to outliers.

**Feature Engineering**

For model fitting, we employed different techniques to engineer the features. First, we find that the genre feature is a list of strings that indicate different genre types for a given movie. We convert the feature into 24 different columns with each column representing an individual genre. If a movie belongs to a genre, it will be marked as 1 in the genre’s column, otherwise, it will be set to 0. Therefore, a movie can be categorized into more than one genre, with all relevant genres coded as 1 and others coded as 0. Another categorical variable that needs to be encoded is the start year. We also change them into independent columns in a similar fashion.

We then transform the target variable average\_rating into three classes. The original ratings range continuously from 1 to 10. Keeping all 10 classes creates a huge burden for the model because our dataset is relatively small compared to the number of labels. With this consideration in mind, we decide to split the continuous variable average\_rating into three classes- for ratings below 5, we encode them as 0 (bad); for ratings between 5-7, we encode them as 1 (mediocre); for ratings larger than 7, they will be encoded as 2 (good).

To fit textual data into natural language models, we wrangled the plot information through several stages, including lowering all word cases, removing punctuations, removing stopwords, combining into sequences, and creating embedding. After data processing, we exported the cleaned data for modeling and analysis.

**Methods and Results**

Baseline

First, we conducted two classification baseline models -- one linear and one deep learning model -- both of which only include runtime, number of votes, start year, and genre as features to predict rating. The linear model has little prediction power with an accuracy of 0.541713, indicating that there may be nonlinear relationships between the independent variables and the ratings. The improvement from the neural network model confirms this suspicion, while the accuracy is still not satisfactory (accuracy: 0.587117). These two experimentations motivate us to incorporate NLP to extract more information from movie plots to boost the model’s predictive power.

Sentiment Analysis

Our first attempt is sentiment analysis using SentimentIntensityAnalyzer from NLTK’s VADER model. With the intuition that different tones of description may disclose the information of movie plots, we extract the sentiments from the textual data to improve the model. We create subjectivity, polarity, positive, negative, neutral, and compound scores from the movie plot. Additionally, we generate the length of the text and word counts of text in the dataset. For a better understanding of the top words used in the plots, we visualize the sentiment with word clouds and histogram. We incorporate some of these sentiment scores in the prediction model. However, as shown in the pair plot, sentiment information did not improve the prediction accuracy due to the absence of correlation between sentiments and the dependent variable.

Task-Specific Word Embedding

Discovering the limitation of sentiments, we hope to capture more semantic similarities between words in the plots using word embedding. We first use Keras Tokenizer to give each unique word in the plot feature an index. In total, there are 57007 unique vocabularies. We then use texts\_to\_sequences to translate each sentence into a sequence number based on tokenized indexes. We apply padding to each sequence of numbers to ensure that all text bodies are the same length. After transforming plots into sequences of numbers, we split the dataset into training (80%) and testing (20%).

Our first word embedding model learns embedding from the dataset only. We use the Embedding layer from Keras to perform the task. We first build a linear neural network (Embedding + Flatten + Dense) to perform predictions. The test accuracy rate for this method is only 0.46444544. Then we create a deep learning model with one GlobalAveragePooling1D layer and more dense layers, and we are able to increase the accuracy to over 0.5 (0.5116279). Nonetheless, this prediction power is still not strong enough.

Pre-Trained Word Embedding

Our second word embedding model uses a pre-trained 100 dimension version of Global Vectors for Word Representations (GloVe) trained on [Wikipedia 2014](http://dumps.wikimedia.org/enwiki/20140102/) and [Gigaword 5](https://catalog.ldc.upenn.edu/LDC2011T07). We read the pre-trained vector into a dictionary with a 100 number long vector mapping to each word. In total, the pre-trained GloVe has 400,000 words. We set the embedding\_dim to be 100 and prepare an embedding matrix shape num\_token by embedding\_dim (vocabulary size + padding (1)). We insert pre-trained vector representations into the matrix for words in the plot that are also in pre-trained GloVe. We record the indexes for words that are missing. In total, there are 12878 out of 57007 words that are not in pre-trained GloVe. For these missing words, a vector of 0s will be recorded. As expected, the prediction accuracy for using only pre-trained GloVe doesn’t offer much improvement (0.44722718 in linear and 0.5178891 in DNN).

Fine-tuned Word Embedding

The third word embedding method we use is a fine-tuned GloVe. To compensate for the missing words from the pre-trained GloVe, We employed the [mittens](https://www.aclweb.org/anthology/N18-2034.pdf) library to create new embeddings for words that are not included in GloVe. In word embedding, each of those missing words will have a vector with all elements being 0. Therefore, we can use the indexes of the missing words to insert the new embeddings into the embedding vectors so that the 0s vectors are replaced by the new vectors. This method yields prediction accuracy of 0.46109122 in linear and 0.52549195 in deep learning models respectively.

LSTM

After experiencing bottlenecks in word embeddings with simple neural nets, we decided to complicate the model architecture by including Long Short-Term Memory (LSTM) structures. LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. LSTM is particularly useful in situations where understanding historic context is important. Bidirectional LSTM has access to both past and future input features for a given time which is especially important in pattern recognition.

We use two bidirectional LSTM layers in the model and expect the model to learn from the neighbor textual information and enhance the prediction power. However, the model only returns 0.48300537 accuracy on the test dataset.

**Discussion**

We find consistent improvement in prediction switching from linear models to neural networks for baseline and various word embedding models. This demonstrates the superiority of neural nets in capturing non-linear and contextual information. Also, this improvement may come from the large amount of data we have.

We find that there isn’t much improvement in model performances between the three types of embeddings. Theoretically, fine-tuned word embedding should perform the best in capturing the semantic similarity between words in the document. However, this is not the case for the movie plots and below are a few possible reasons.

Firstly, GloVe is trained on a large number of Wikipedia passages and English Gigaword Fifth Edition, a comprehensive archive of news over years. These two data sources likely use formal, journalistic, and objective writing styles, whereas plot descriptions were written in more dramatic and casual fashions (e.g. “On New Year's Eve in London, four strangers find themselves on the roof of a building known for suicides.”). These stylistic and vocabulary differences are reflected in the fact that one-fifth of the words in the plots are missed in the GloVe vocabularies. The mittens model is effective in creating new, domain-specific embeddings for sentiment analysis tasks (eg. classifying IMDB movie reviews). However, it also suffers from the same problem with GloVe: they treat words that appear in similar sentences as having similar meanings. Words like ‘wide’ and ‘narrow’, or ‘Red Sox’ and ‘Yankees’ confuse those systems. They also have a hard time identifying negations in the sentences so that some important differences in sentence meanings are not captured. These shortcomings might have influenced the models’ ability to understand the content of the plots.

Secondly, GloVe may have included the most commonly used words in English, and hence, those words not being captured by GloVe seem less important for sentence and meaning representation. And this results in a similar performance between the three word embedding models.

Thirdly, the plot dataset, although only has 12 million tokens from 20,000 observations, has 57,007 unique vocabularies. In comparison, GloVe (6B) is trained on 6 billion tokens and produces 400,000 unique words. We believe that movie plots exhibit huge information density, conveying a large number of messages and content in a short sentence. The different natures between densely packed plot lines and news articles/Wikipedia pages weaken the embeddings’ abilities to represent the plots. The fact that each plot is so uniquely different from each other in terms of the language used shows that the diversity of plots is difficult to be represented in a predefined vector space. The embedding models simply don’t see enough similar sentences to identify and place similar words into the vector. The failure of the fine-tined pre-trained model exemplifies a challenge that some of the state-of-the-art language representations couldn’t effectively represent words in a dense information space.

Fourthly, our experimentation with different neural network architectures and word embeddings did not go through rigorous tuning processes. There are a few parameters such as embedding dimensions, the maximum number of words (max\_feature), maximum sentence length (max\_len), number of iterations for learning new embeddings that could be tuned. Yet, we only use minimal or baseline values as starting points. We suspect that the models would perform better after more thorough parameter searching.

Fifthly, it is possible that plots alone do not offer good information to predict public rating. In this case, public tastes for movies are perhaps so diverse that there’s no single winning plot formula that can summarize what a “good” movie is. This explanation is consistent with the concept of cultural pluralism in which multiple sub-cultures contribute to the diversity of society and are valued equally for their uniqueness.

Last but not the least, the aforementioned NLP techniques are more commonly used in language extraction and understanding, rather than in a prediction task. Therefore, even though we are able to extract the meaningful and important information from the movie plots, it does not guarantee that we can acquire better prediction power.

To conclude, we employed a wide range of techniques in the project such as data wrangling, feature engineering, sentiment analysis, building multiple neural network structures, and fine-tuning word embeddings. Although our models experience bottlenecks in performances, we identified a few reasons and ways to improve as stated above.