Predicting Movie Genres Based on Plot Summaries

Article · January 2018

1 author:

Quan Hoang
Monash University (Australia)
2 PUBLICATIONS **15** CITATIONS

Predicting Movie Genres Based on Plot Summaries

Quan Hoang
University of Massachusetts-Amherst
qhoang@umass.edu

Abstract

This project explores several Machine Learning methods to predict movie genres based on plot summaries. Naive Bayes, Word2Vec+XGBoost and Recurrent Neural Networks are used for text classification, while K-binary transformation, rank method and probabilistic classification with learned probability threshold are employed for the multi-label problem involved in the genre tagging task. Experiments with more than 250,000 movies show that employing the Gated Recurrent Units (GRU) neural networks for the probabilistic classification with learned probability threshold approach achieves the best result on the test set. The model attains a Jaccard Index of 50.0%, a F-score of 0.56, and a hit rate of 80.5%.

1 Introduction

Supervised text classification is a mature tool that has achieved great success in a wide range of applications such as sentiment analysis and topic classification. When applying to movies, most of previous work has been focused on predicting movie reviews or revenue, and few research was done to predict movie genres. Movie genres are still tagged through a manual process in which users send their suggestions to email address of The Internet Movie Database (IMDB). As a plot summary conveys much information about a movie, I explore in this project different machine learning methods to classify movie genres using synopsis. I first perform experiment with Naive Bayes using bag-of-word features. Next, I make use of the pretrained word2vec embeddings [21, 3] to turn plot summaries into vectors, which are then used as inputs for an XGBoost classifier [9]. Finally, I train a Gated Recurrent Unit (GRU) neural network [10] for the genre tagging task.

The rest of the report is organized as follows. Section 2 discusses the related work. Section 3 describes the dataset used for this project. Section 4 outlines the models for experiments. Section 5 presents experiment results. Finally, Section 6 summarizes the paper and discusses directions for future work.

2 Related work

[20] proposes a Naive Bayes model to predict movie genres based on user ratings of the movie. The idea is that users are usually consistent with their preference and prefer some genres over the others. The evaluation metric is the percentage of movies that the model predicts correctly at least one of the true labels. The focus of my project is natural language processing, so I attempt to predict movie genres using only movie plot summary. Moreover, I adopt more rigorous metrics such as F-score or Jaccard index.

[7] attempts to classify movie scripts by building a logistic regression model using NLP-related features extracted from the scripts such as the ratio of descriptive words to nominals or the ratio of dialogues frames to non-dialogue frames. For each movie scripts, the model, based on extracted features, estimates the probability that the movie belong to each genre and takes the k best scores to be its predicted genres, where k is a hyper-parameter. The experiment is done on a small dataset with only 399 scripts and the best subset of features achieves an F1 score of 0.56.

[25] investigates different methods to classify movies' genres based on synopsis. The methods examined include One-Vs-All Support Vector Machines (SVM), Multi-label K-nearest neighbor (KNN), Parametric mixture model (PMM) and Neural network. All these methods use the term frequency inverse document frequency of the words as features. The dataset used for experiment is relatively small with only 16,000 movie titles for both the train and test sets. In addition, the experiment is limited to only predicting only 10 most popular genres, including action, adventure, comedy, crime, documentary, drama, family, romance, short films, and thrillers. Overall, SVM achieves the highest F1 score of 0.55.

As a movie can belong to several genres, this project is related to the multi-label classification problem, which was examined in some previous work. [22] introduces two extensions of the AdaBoost algorithm [12] for multi-class, multi-label text categorization. The first extension turns the multi-label problem into multiple, independent binary classification problems, while the second ranks the labels so that the correct labels receives the highest ranks. Both extensions are trained using weak hypotheses as one-level decisions trees that check the presence or absence of a term in a given document. For the second extension, the objective was to minimize the average fraction of pairs of labels that are misordered, a quantity called the empirical rank loss:

$$\frac{1}{m}\frac{1}{|Y_l||\mathcal{Y}\setminus Y_l|}|\left\{(i,j)\in Y_l\times(\mathcal{Y}-Y_l): r_i\left(\mathbf{x}_l\right)\leq r_j\left(\mathbf{x}_l\right)\right\}| \tag{1}$$

where $\mathcal{Y}$ is the set of all label, $\mathbf{x}_l$ and Y_l are respectively the feature vector and the set of true labels for the l-th example, $r_i\left(\mathbf{x}\right)$ is the rank value function for label i, and m is the total number of training examples. The predicted labels are those with the top k highest ranks, where k is a hyperparameter. Trained and tested on some multi-label subset of the Reuter-21450 dataset, the rank-based extension achieves slightly worse performance.

Inspired by the ranking method in [22], [11] propose a kernel method to learn the ranking function. Instead of fixing a number k as the label size for prediction, this method also learn a threshold $t\left(\mathbf{x}_l\right)$. More specifically, for an input vector $\mathbf{x}_l$ and the rank values $\left(r_1\left(\mathbf{x}_l\right), r_2\left(\mathbf{x}_l\right), ..., r_Q\left(\mathbf{x}_l\right)\right)$ where the set of labels is $\mathcal{Y}=\{1,2,...,Q\}$, the threshold value is defined as:

$$t\left(\mathbf{x}\right)=\arg\min_t|\left\{k\in Y_l\right\}\ s.t.\ r_k\left(\mathbf{x}_l\right)\leq t|+|\left\{k\in(\mathcal{Y}\setminus Y_l)\right\}\ s.t.\ r_k\left(\mathbf{x}_l\right)\geq t| \tag{2}$$

When the minimum is not an unique value but a segment, the threshold is chosen to be the middle of the segment. The threshold is modeled using linear least squares.

[26] applies this idea to neural networks by slightly modify the rank loss function as:

$$\frac{1}{m}\frac{1}{|Y_l||\mathcal{Y}\setminus Y_l|}\sum_{i\in Y_l,\ j\in(\mathcal{Y}\setminus Y_l)}\exp\left(-\left(r_i\left(\mathbf{x}_l\right)-r_j\left(\mathbf{x}_l\right)\right)\right) \tag{3}$$

The use of the exponential function is to severely punish the model for assigning higher rank value to the wrong labels.

This project employs several machine learning methods to predict movie genres, including Naive Bayes, XGBoost, and recurrent neural network (RNN). XGBoost [9] is an advanced and efficient implementation of the gradient boosting algorithm [8, 13, 14], an ensemble method that sequentially add new predictors that are trained on the residual errors made by previous predictors. Unlike the standard gradient boosting algorithm, XGBoost has a regularized objective and and it makes use of second-order approximation to quickly and greedily fit a new predictor at each iteration. XGBoost was the algorithm of choice winning solutions of many machine learning and data mining challenges.

To extract features to be used in XGBoost, I make use of the word2vec framework proposed in [21], which learns high-dimensional word embeddings. Word2vec learns embedding by training a neural network to predict neighboring words. The idea is that semantically similar words tend to occur near each other in text, so embeddings that are good at predicting context words are also good at representing similarity. There are two model architectures in word2vec to learn word embeddings: continuous bag-of-words (CBOW) and continuous skip-gram. In the CBOW architecture, the model predicts the current word from a window of surrounding context words. In the continuous skip-gram architecture, the model predicts the surrounding window of context words using the current word.

Vector embeddings learned by word2vec was shown to capture relations between words. For example, the result of the embedding vector('*king*') - vector('*man*') + vector('*woman*') is a vector close to vector('*queen*'). For this project, I use the pretrained 300-dimensional embeddings trained on part of the Google News corpus for 3 billion words [3]. The average of the embedding vectors of all words in a movie plot is then used as the vector representation of the plot.

Taking average of the embedding vectors of all words in a plot might be a lossy summary and fail to capture the sequential relationship. Therefore, I consider Recurrent Neural Networks (RNN), which are a powerful family of neural networks for processing sequential data. RNNs take one input from the input sequence at each time step. Hidden units in RRNs take inputs not only from the data or from previous layers as in traditional neural networks, but also from themselves in the previous time step. This allows RNN units to take information from the past sequence of inputs. Parameters in RNNs are shared across time steps and learned by gradient descent optimization methods with the Back Propagation Through Time (BPTT) algorithm [24], which computes the gradients of the loss function with respect to the parameters.

Gradients in RNNs, however, tend to vanish or explode as explored in depth in [16, 6], making it difficult to train RNNs. The LSTM networks [17] address this problem by introducing a complex unit called *memory cell*. The central feature of LSTM's memory cell is the constant error carrousel (CEC), which a self-connection for each cell state, thus allowing the gradient signal to stay constant as it flows backward across time steps. As memory cells interract with each other, they are equipped with *input gates* and *output gates* to protect themselves from perturbation. These input and output gates use multiplicative operations to control the network's sensitivity to each unit's inputs (or outputs).

Cell states in the original LSTM model, however, could become very large and saturate the output gates, thus leading to gradient vanishing. This problem was termed "*internal state drift*" [17]. To tackle this issue, [15] introduced *forget gates*, which reset cell states when the gates decide that the cell states' content is out-of-date. "Reset" does not mean to set the cell states to zero immediately, but to gradually reset by multiplying with a number between 0 and 1. This architecture allows the gradients to flow for long duration, and the LSTM has been found extremely successful in many applications such as speech recognition or machine translation.

Many LSTM variants have been proposed. Among them, Gated Recurrent Unit (GRU) [10] has become increasingly popular. GRU is similar to LSTM but combines the forget and input gates into a single update gate. As GRU has a simpler architecture than LSTM, I will use this architecture for the experiments with RNNs. In addition, I experiment with a variant of GRU that I term Self-normalizing GRU (SNGRU). In a GRU cell, the short-term memory states are squashed to the range between -1.0 and 1.0 by the hyperbolic tangent (*tanh*) function before going through the update gates. As a result, gradient signals become weaker when flowing through update gates during backward passes. The motivation of SNGRU is to avoid using the squashing function. To keep cell states bounded, SNGRU applies layer normalization [5], which helps the cell states converge to a normal distribution.

3 Data

To prepare the dataset for experiments, I extract the movie plot summaries and their corresponding set of genres from the IMDB datasets. The raw data files, including the plot list and the genre list files, are downloaded from [1]. The genre list file has 1,433,423 pairs of movie name and genre. It is noteworthy that there are k pairs with the same movie name in the file if the movie belongs to k genres. The plot list file has 345,920 pairs of movie and plot summary. There are 338,789 movies for which both genres and plot summaries are available. Selecting only movies belong to at least one of the 20 genres, which will be listed below, results in 283,355 movies. Finally, filtering out movies with more than 200 tokens in their plot summaries leaves 255,853 movies. I divide the data into train and test sets according to the 80/20 proportion. As a result, there are 204,682 movies in the train data, and 51,171 movies in the test data. There are 15,187,708 tokens and 266,084 unique word types in all of the train plot summaries. Of these, 60,325 word types occur 5 times or more.

The genre names and the percentages of movies in them are: *drama* (46.0%), *comedy* (27.9%), *thriller* (11.2%), *romance* (11.0%), *action* (9.7%), *family* (8.1%), *horror* (7.7%), *crime* (7.3%), *adventure* (7.0%), *animation* (6.2%), *fantasy* (5.9%), *sci-fi* (5.6%), *mystery* (5.4%), *biography* (5.2%), *music* (4.9%), *history* (4.7%), *war* (2.8%), *western* (2.4%), *sport* (2.4%) and *musical* (2.2%).

In the train dataset, there are 4,145 unique sets of genres. Each movie belongs on average to 1.84 genres. The distribution of the number of genres a movie belong to are: 1 (50.5%), 2 (25.2%), 3 (17.1%), 4 (5.1%), 5 (1.5%) and 6 or higher (0.6%).

Of all the movie plot summaries, 11% have 50 or fewer tokens, 56% have between 50 and 100 tokens, 25% have between 100 and 150 tokens, and 8% has between 150 and 200 tokens. The average tokens per plot summary is 89.5.

4 Methods

4.1 Naive Bayes

For this task, I experiment with three machine learning methods. The first method is to use Naive Bayes, a simple approach that was shown very effective for many text classification tasks. Naive Bayes makes two key assumptions. The first is the bag-of-words assumption, meaning that the order of words does not matter. The second is the conditional independence assumption. For document $\mathbf{d} = w_1 w_2 ... w_N$ and given the class c, the probabilities $P(w_i|c)$ are independent. Thus:

$$P(w_1, w_2, ..., w_n|c) = \prod_{i=1}^{N} P(w_i|c) \tag{4}$$

The equation for the class chosen by a Naive Bayes classifier is thus:

$$c_{NB} = \arg\max_{c \in C} \frac{p(c) \prod_{i=1}^{N} P(w_i|c)}{P(\mathbf{d})} \tag{5}$$

$$= \arg\max_{c \in C} p(c) \prod_{i=1}^{N} P(w_i|c) \tag{6}$$

I consider two approaches to apply Naive Bayes to the multi-label classification task involved in this project. The first approach is to build a binary Naive Bayes classifier (bNB) for each genre. The second approach is to build a multinomial Naive Bayes classifier (mNB) model. For each plot $\mathbf{d}$, MNB can estimate the posterior probability $p(c|\mathbf{d})$ in the right hand side of Eq. 5. The model predicts that a movie belong to a genre c if $p(c|\mathbf{d}) > p(c)$, where $p(c)$ is the prior probability of the genre c. The intuition is that the model assigns a movie to a genre if the model has a stronger than prior belief in the presence of the genre after reading the movie's plot summary.

4.2 Word2Vec+XGBoost

One problem with Naive Bayes is that the bag-of-word feature as well as the conditional independence assumption are not realistic and may not capture the meaning of sentences in text. Therefore, the second method is to employ rich, contextual representations for words learned by the word2vec framework [21]. Each plot summary is turned into a vector representation by taking average of the embedding vectors of all words in each the text. The assumption is that the average vector can be a good semantic summarization of the plot summary.

The plot vector representation can then be fed into XGBoost [9]. I apply XGBoost to the multi-label classification task in a similar spirit to the ranking method [11] discussed in Section 2. Instead of learning the rank values, however, I train an XGBoost classifier that output the probabilities $P(c|\mathbf{x})$ that the movie represented by $\mathbf{x}$ belongs to the genre c. To make predictions based on these probabilities, a XGBoost regressor is trained to predict the probability threshold for each categorical distribution.

To train the XGBoost classifier, the train data is modified by turning each pair of plot vector and its corresponding set of correct genres $(\mathbf{x}, C)$ into k pairs of $(\mathbf{x}, c)$, where $k = |C|$ and $c \in C$. To train the XGBoost regressor, the probabilities $P(c|\mathbf{x})$ estimated by the XGBoost classifier for each movie in the train data are used as features, and the probability threshold computed based on the corresponding set of correct genres is used as the target. More specifically, the probability threshold is defined as:

$$t(\mathbf{x}) = \arg\min_t |\{c \in C\} \ s.t. \ P(c|\mathbf{x}) \leq t| + |\{c \in (\mathcal{Y} \setminus C)\} \ s.t. \ P(c|\mathbf{x}) \geq t| \tag{7}$$

where $\mathcal{Y}$ is the set of all genres. At test time, the XGBoost classifier and regressor estimate the probabilities of each genre and the threshold, and output genres with higher probabilities than the threshold. I use the XGBoost Python Package [2] to train the XGBoost classifier and regressor. Instead of reinventing the wheel, I make use of the pretrained 300-dimensional embeddings trained on part of the Google News corpus for 3 billion words [3].

4.3 Recurent Neural Networks

Lastly, I train a RNN for this task. The RNN learns not only the embedding for each word, but also a nonlinear transformation of a sequence of embedding vectors. Thus, it might learn a more suitable representation of plot summaries for the genre tagging task. The key assumption in RNN is the same function can be used to transform current state and input into a new state. This idea of parameter sharing across time steps makes it possible to apply the model to examples of different length.

More specifically, I train a Self-normalizing GRU (see Section 2) network with 2 layers, each of which has 128 memory cells. The input to the network a sequence of 128-dimension vectors, each representing a word in a plot summary. The network learns vector representations for a vocabulary of 60,000 words, including the top 55,998 words that appear at least 5 times in the train dataset, "UNK", which represents other words, and "EOS", which stands for end of sequence. Given the last state $\mathbf{h}_N$, a linear transformation is applied to output a 20-dimension output vector y:

$$\mathbf{y} = W \times \mathbf{h}_N + \mathbf{b} \tag{8}$$

where W is a 128×20 matrix and $\mathbf{b}$ is a 20-dimension bias vector.

Three approaches are considered to make the final predictions using the output $\mathbf{y}$. In the first approach, each element y_c is fed to a logistic sigmoid function to estimate the probability of the genre c:

$$\sigma_c(y_c) = \frac{1}{1 + \exp(-y_c)} \tag{9}$$

The model makes prediction for each genre independently, and predict genre c if $\sigma_c(y_c)$ is larger than 0.50. This model is called Binary GRU.

In the second approach, $\mathbf{y}$ is interpreted as the rank values as discussed in Section 2; the loss function is defined in Eq. 3, and the threshold values on the train dataset are estimated following Eq. 2. The model following this approach is called Rank GRU.

In last approach, y is fed into a softmax function to estimate the probability of each genre:

$$P(c|\mathbf{x}) = \frac{\exp(y_c)}{\sum_{k=1}^{20} \exp(y_k)} \tag{10}$$

The target probability for each genre c is $\frac{1}{k}$ where k is the number of correct genres if c is in the set of correct genres, and is 0 otherwise. The network is trained to minimize the cross entropy between the predicted and the target categorical distribution. The threshold values on the train dataset are estimated following Eq. 7. The model following this approach is called Multinomial GRU.

For both the second and third approaches, an XGBoost regressor is trained to predict threshold values on the test dataset as described in 4.2. Implementation of the GRU network is in TensorFlow [4].

5 Experiment

5.1 Hyperparameter selection

There is no hyperparameter to tune for the Naive Bayes models. Due to limited time for experiment and priority, the default setting for XGBoost is used. For the GRU network, the hyperparameters include the number of layers, the number of units in each layer, the recurrent dropout keep rate

Hyperparameters	Values
Number of layers	[1, 2, 3]
Number of units	[128, 256]
Dropout keep rate	[0.5, 0.6, 0.7, 0.8, 0.9]
Learning rate	[0.0002, 0.0005, 0.001, 0.003, 0.005, 0.008, 0.01, 0.02, 0.05]

Table 1: Different values considered for hyperparameter selection.

[23] and the learning rate for Adam optimizer [18]. Different values for each hyperparameter are considered as shown in Table 1

To tune the hyperparameters, I divide the dataset into the train, validation and test sets according to the 70/10/20 proportion. The hyperparameters that yield the highest softmax loss or rank loss in the validation set are selected. The optimal settings are 2 for the number of layers, 128 for the number of units in each layers, 0.8 for the dropout keep rate and 0.01 for the learning rate.

5.2 Evaluation metrics

A weak metric is the "*hit rate*", proportion of examples where the model predicts at least one correct genre. However, tagging all movies with all genres can yield a hit rate of 100%. To punish for incorrect predictions, I adopt the Jaccard index, which is defined as the number of correctly predicted labels divided by the union of predicted and true labels, $\frac{|T \cap P|}{|T \cup P|}$. To further compare performance of different methods, I calculate the confusion matrix, accuracy, precision, recall, and F-score for each genre as well as for all of the test data. These metrics are defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$Fscore = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{14}$$

where TP, FP, TN and FN are true positive, false positive, true negative and false negative respectively.

5.3 Baseline

I use two approaches to estimate the baseline. The first approach is a simple heuristic that predict the top 1 (*drama*), top 2 (*drama* and *comedy*) or top 3 (*drama*, *comedy* and *action*) most popular genres for all movies, resulting Jaccard indices of 29.2%, 28.6% and 22.8% and hit rates of 45.3%, 66.4% and 71.3% respectively. So, tagging all movies to *drama* and *comedy* is a reasonable baseline. The second approach is to build 20 binary Naive Bayes model for each genre, using casefolding and the simple whitespace tokenizer. This method achieves an Jaccard index of 36.3%, which is a significant improvement over the heuristic baseline. The hit rate, however, is only 55.5%. Additionally, the F-score in this approach is 0.44.

5.4 Experiment results

Table 2 summarizes experiment results of different methods discussed in Section 4. Multinomial GRU achieves the highest Jaccard index of 50.0% and F-score of 0.56. Binary-based models such as Binary GRU and Binary Naive Bayes attain high precision (67.4% and 60.7% respectively), while rank-based models such as Multinomial Naive Bayes, rank GRU and Multinomial GRU obtain high recall (56.3%, 56.1%, and 51.3% respectively). This result is plausible as binary-based models make decision for each genre independently, and return positive prediction only when it is very confident. Meanwhile, rank-based models make decisions based on relative rank values (or conditional probabilities) among different genres, thus output positive prediction more often. This observation is confirmed by Figure 2,

Methods	Hit rate	Jaccard	F-score	Precision	Recall
Binary Naive Bayes	74.2%	42.6%	0.52	60.7%	45.0%
Multinomial Naive Bayes	**82.9%**	46.6%	0.53	50.7%	**56.3%**
XGBoost	74.6%	42.9%	0.49	55.3%	44.4%
Binary GRU	70.6%	46.5%	0.53	**67.4%**	44.0%
Rank GRU	82.7%	48.5%	0.56	56.6%	56.1%
Multinomial GRU	80.5%	**50.0%**	**0.56**	61.0%	51.3%

Table 2: Summary of experiment results for different methods.

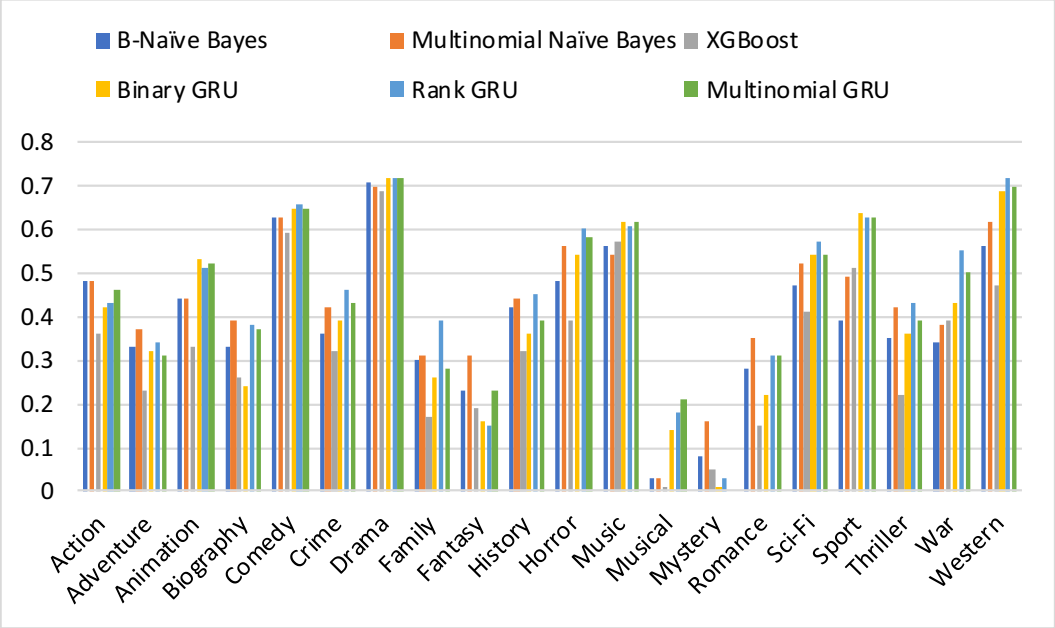

Figure 1: F-score by genre.

which shows that Multinomial Naive Bayes (in orange) and Rank GRU (in light blue) predict on average more genres per movie. As a result, these models get the highest hit rates, while Binary GRU and Binary Naive Bayes get the lowest. Overall, Multinomial GRU and Rank GRU achieve better balance among different metrics.

Figure 1 shows F-score by genre of each model. It can be seen that GRU models (the last 3 columns in gold, light blue, and green) consistently attain higher F-score than Naive Bayes models (the first 2 columns in blue and orange) and XGBoost (the middle column in gray). XGBoost consistently under-performs across genres. Tuning hyperparameters might improve this model a little bit, but this result strongly suggests that taking average of embeddings vector of all words in a plot summary might be too lossy and does not yield a good representation. The Naive Bayes models are competitive with GRU models on the most popular genres such as *Drama* (46% of train examples), *Comedy* (27.9%), *Thriller* (11.2%), *Romance* (11.0%) and *Action* (9.7%), but under-performs on less popular genres. So, the GRU models generalize to less popular genres better. One reason for this behavior is that Naive Bayes takes the bag-of-word and conditional independence assumption, and the data is not highly skewed. As a result, Naive Bayes models are biased towards the most popular genres.

Figure 2 shows the size distribution of the predicted genre sets by different methods. One problem of the binary-based models is that many movies are tagged with no genre. Binary Naive Bayes (in blue) and Binary GRU (in gold) respectively predict empty set for 15% and 9% of test examples. The size distributions of the genre sets predicted by the rank-based models are closer to the true distribution.

The following subsections discuss some models in further details.

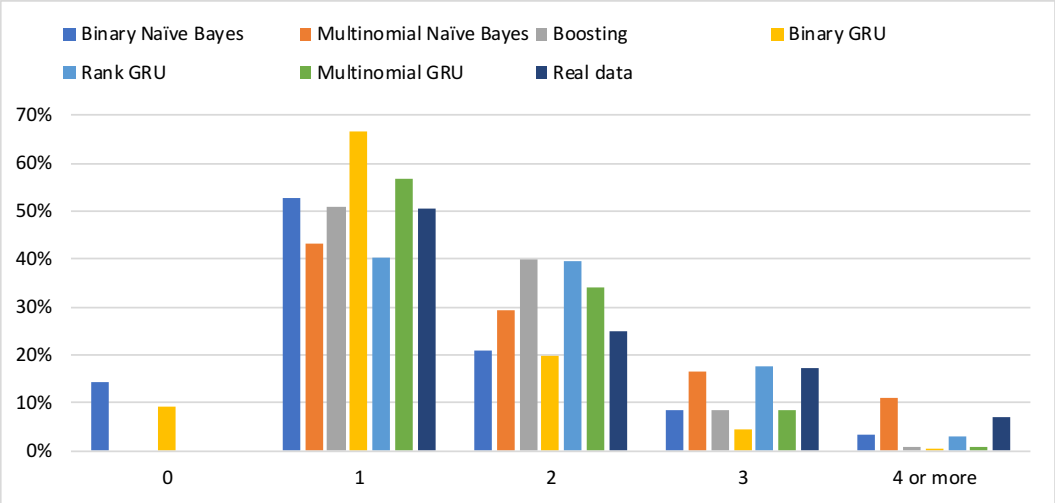

Figure 2: Size distribution of the predicted genre sets.

	Comedy	Drama	Sport	War	Western
Precision	54.1%	60.2%	80.0%	65.3%	91.1%
Recall	75.3%	85.1%	35.5%	27.1%	47.5%
F-score	0.63	0.70	0.49	0.38	0.62

Table 3: The Multinomial Naive Bayes model's precision, recall and F-score for some genres.

5.4.1 Naive Bayes Models

As discussed in Section 4, I consider two approaches to apply the Naive Bayes to the movie genre tagging task. The first approach is to build 20 binary Naive Bayes models for each genre. This model, when preprocessed with the whitespace tokenizer and casefolding, is used as the baseline. To better process the data, I apply casefolding, remove all stop words, and use the NLTK word tokenizer. As a result, this approach achieves a Jaccard index of 42.6%, which is a significant improvement over the whitespace-tokenizer version's 36.3% score. Similarly, F-score improves from 0.44 to 0.52.

Using the same word normalization procedure, Multinomial Naive Bayes attains a Jaccard index of 46.6% and F score of 0.53.

Figure 1 further compares F-score by genre on the test data. Multinomial Naive Bayes (second columns in orange) attains much higher F-score than Binary Naive Bayes (first columns in blue) for less popular genres. The common between the two models is that they tend to have high recall for very popular genres such as *drama* or *comedy*, and high precision but low recall for less popular genres such as *sport*, *war* and *western* as illustrated in Table 3. This result is plausible given the skewed dataset, which makes the Naive Bayes models to predict the popular genres more often, and to predict the less popular genres only when it has strong belief.

5.4.2 Multinomial GRU

This subsection discusses further the Multinomial GRU model. Tab. 4 shows Multinomial GRU's predictions on some random test examples. These few examples illustrate that predicting movie genres is a challenging task even for human. Plot summaries do not convey all the information about a movie, and are sometimes ambiguous. The first example is about a guy being misunderstood by his girlfriend, who becomes upset and runs away. The correct genre is *romance* but the model predicts *drama*, which is quite plausible. In the second example, the model correctly predicts *western*, but incorrectly predicts *action*, perhaps due to "*chases and catches*" or "*robber*". In the third example, the predicted genres are *drama* and *thriller* while the true genres are *drama* and *comedy*. However, the plot with details such as "*stalking*", "*murder*" and "*more people to kill*" suggests *thriller* more than *comedy*. It should be noted that the Jaccard index is only 33.3% in this case. In the fourth

example, there is no sign of *musical* unless one knows that Frank Sinatra is singer, while *biography* seems a reasonable choice.

In the fifth example, the model correctly predicts *comedy* and *family*, but fails to include *fantasy*, *mystery*, *romance*, and *sci-fi*. Of these missed genres, it is possible to detect *mystery* or even *fantasy* from the plot summary, but there is no clue to *romance* or *sci-fi*. As the only sign of comedy in the plot summary is "*many comical situations*", I changed it into "*many unexpected situations*" and the model predicts *family* and *horror* instead. The probabilities of *family*, *comedy* and *horror* are changed from 33%, 27% and 9% to 29%, 13% and 21%, respectively.

The sixth and seventh example shows the model's bias towards the most popular genres such as *drama* or *comedy*. In the sixth example, the model assigns a probability of 27% to *drama*, 24% to *horror*, 15% to *thriller* and 14% to *mystery* and selects only *drama* and *horror*, although there is no information in the plot summary suggesting *drama*. Similarly in the seventh example, the model assigns a probability of 46% to *romance*, 25% to *comedy* and 11% to *drama* and predict *romance* and *comedy*. The correct genre is *romance* only.

The model is also sensitive to some words in a plot. In the last example, the model assigns a probability of 22% to *sci-fi*, 15% to *animation*, 11% to *action*, 11% to *fantasy*, and 9% to *horror*, and it predicts *sci-fi* and *animation*. The model is obviously not biased towards *animation* because only 6.2% of train examples belong to the *animation* genre. It turned out that the culprit is "*miniaturized*". When removing this word from the plot, the model assigns 31% to *sci-fi*, 13% to *action*, 12% to *horror*, 7% to *fantasy* and 7% to *animation*, and it correctly predicts only *sci-fi*. The first example also illustrates this problem. When "*drives away with her*" is changed into "*falls in love with her*", the model changes its prediction from *drama* to *drama* and *romance*.

Finally, I qualitatively evaluate the model's learned word embeddings by looking at the nearest words to some random words. The nearest words to a certain word are those in the vocabulary with embedding vectors that have the smallest cosine distance with the word's embedding vector. Table 5 shows top 10 nearest words of 6 nouns, 6 verbs, and 6 adjectives. Nearest words to nouns are very relevant. The results, however, are mixed for verbs and adjectives. Words such as *kill*, *love*, *escape* and *cruel* have relevant nearest neighbors, but *manipulate*, *preserve*, *arrogant*, and *beautiful* have totally irrelevant neighbors. One possible reason for this behavior is that verbs and adjectives are often used in a wider range of contexts than nouns. *Hang*'s neighbors such as "*smothering*" or "*re-connect*" seem related to different meanings of the word. *Miniaturized* is an interesting example as it appears only 7 times in the train data, making the context specific. As a result, among its nearest words are *karaati*, *gnomes*, *alchemist*, *meadow* and *fairy-tale*, which may explain its influence on the model's prediction of *animation* in the previously discussed example.

6 Conclusion

This project explores several Machine Learning methods to predict movie genres based on plot summaries. This task is very challenging due to the ambiguity involved in the multi-label classification problem. For example, predicting *adventure* and *thriller* against true labels of *adventure* and *action* yields a Jaccard Index of only 33.3%.

Word2vec+XGBoost performs poorly as the average of embedding vectors of words in a document proves a weak representation. A future direction is to apply doc2vec [19] to learn richer representations of documents. Experiments with both Naive Bayes and GRU networks show that combining a probabilistic classifier with a probability threshold regressor works better than the k-binary transformation and the rank methods for the multi-label classification problem. Using a GRU network as the probabilistic classifier in this approach, the model achieves impressive performance with a Jaccard Index of 50.0%, F-score of 0.56 and hit rate of 80.5%.

There are several potential directions to improve the GRU network. As 46% of words in the vocabulary occur fewer than 20 times in the train data, most word embeddings get only a few weight updates. Using pretrained embeddings might be better for these words. In addition, about 75% of tokens are turned into "UNK". As a result, the GRU network learns the same embedding for these words. Finding a way to adapt the UNK's embedding based on context might make the network more powerful. Finally, the data is highly skewed, making the model biased towards popular genres such as *drama* or *comedy*. Dealing with this issue would further improve the performance.

Plot Summary	Predictions	Targets
The story is a typical misunderstanding in a relationship. Marco kisses a woman on the cheek and drives away with her, overseen by his jealous girlfriend Marie who writes him a letter and runs away to clear her mind. After hours of looking for her, Marco finally find his raging girlfriend and tells her who that mysterious woman really is.	Drama	Romance
Red Ryder (Bill Elliott as Wild Bill Elliott) chases and catches a bank robber, but the robber's boss, Denver Jack (Roy Barcroft) has him released by a crooked lawyer, Larry Randall (Robert Grady). Later, Randall decides he wants to reform and tells Denver he is quitting. Denver has him framed on a murder charge and Randall is to be hanged...unless Red Ryder and Little Beaver (Bobby Blake) can uncover evidence to prove his innocence.	Western, Action	Western
Pickups, the new film directed by Jamie Thraves, is about a man called Aidan (conveniently enough, played by Aidan Gillen) who is suffering from insomnia, back trouble and the breakdown of his marriage. Aidan finds solace in a number of strangers he picks up, although he's now concerned someone is stalking him. Work is getting on top of him too, he murdered a couple of people last week and he still has more people to kill.	Drama, Thriller	Comedy, Drama
In the 1950s and 1960s 'Frank Sinatra' (qv) was the head of the infamous "Rat Pack". He, 'Sammy Davis Jr. (I)' (qv), 'Dean Martin (I)' (qv), 'Peter Lawford' (qv) and 'Joey Bishop (I)' (qv) worked and played together. This film dramatizes their volatile relationships with each other and the Kennedys, 'Marilyn Monroe' (qv), mobster 'Sam Giancana' (qv), 'Judith Campbell Exner' (qv) and the FBI. Sinatra helps 'John F. Kennedy' (qv) get elected in 1960 with a little help from Giancana. Lawford, married to a Kennedy, is an unhappy go-between. Davis is fighting racism and insecurity. Campbell is sleeping with both Giancana and JFK who is also sleeping with Monroe.	Drama, History, Biography	Drama, Musical
The father of the Olsson family is responsible for many people's not being able to receive any TV broadcasts. Therefore, the family escape to an old castle in the country, to celebrate their Christmas there, away from all angry people and dangers. However, the three children soon discover that something's not right in the old castle. It seems haunted and it bears a centuries old secret. Many comical situations occur when the children try to investigate the haunting, but the parents don't understand what's going on.	Comedy, Family	Comedy, Family, Fantasy, Mystery, Romance, Sci-Fi
Kaden Mackenzy, a teenage girl with the ability to communicate with the dead, stumbles upon a secret that has been haunting the small town of Peabody, Massachusetts for ages. Kaden's gift suddenly becomes her curse as these interactions soon unfold a mystery of a historic and vicious nature.	Drama, Horror	Horror, Thriller
A barista discovers and experiences a romance in potency between aromas and sensations in a coffee shop. A coffee barista discovers and wraps himself in the magic of the sensations with all the atmosphere created by the combination of the aromas of the coffee; People who frequent the place unexpectedly experience feelings of romance, and she exposed in this environment, will soon experience feelings of ecstasy and unimaginable pleasure.	Comedy, Romance	Romance
The Investigator, a benevolent alien from a distant galaxy, selects an Earth boy and girl, John and Julie, to assist him in his mission to make their world a better place. The pair are miniaturized to assist The Investigator more easily, and assigned to prevent the theft by Stavros Karanti of a 14th century masterpiece from a church on Malta. John and Julie are presented with a car and a boat, scaled to accommodate their miniaturized size, and set out to thwart Karanti's plans...	Animation, Sci-Fi	Sci-Fi

Table 4: Multinomial GRU's predictions for some random test examples.

Words	Nearest words (from left to right)
Adventure	ropes, unsupported, roped, paddle, rainforest, superheroines, goblin, superheroine, shahenshah, huliau
Cigarette	unethical, porn, twenty-something, uneducated, seated, wonderfully, reevaluate, rekindles, pretense, charm
Batman	researched, gotham, organization, enemies, samurai, super, alliance, bhutan, responsibly, politicized
Cowboy	cowboy, marshal, broncho, saloon, rustling, cattlemen, stagecoach, ranch, outlaws, ranchers, settler
Earth	earth, spacecraft, orbs, portal, universe, preternatural, species, activating, quantum, dimension, cosmic
Mother	child, issue, five-year, family, care, familial, caring, mourning, wait, ferret
Escape	protector, humbled, tastes, danger, michaelis, exterminated, protect, barbarians, pitiless, legit
Kill	killing, attacked, lethal, fury, homicide, deadly, ruthless, captor, shortcut, kills
Love	attracted, kiss, crush, playboy, him.but, lovers, eun, i.e, woo, eun-ha
Manipulate	prosecutors, slattery, temps, malti, mousy, psychiatry, joking, 1989, loveless, bombard
Hang	conditions, re-connect, osman, smothering, heterogeneous, helps, krishna, posted, stanislav, feels
Preserve	breasts, jayne, operated, spinning, dismantling, guerrero, shell, gardens, geography, mavis
Arrogant	sheer, rhodes, zap, stealthily, adversaries, unbeknownst, stereotyped, elso, fliers, suzy
Beautiful	automaton, rong, wang, radha, floats, repent, chooses, troublemakers, dreams, stampeded
Cruel	observation, bothering, consulted, marat, ae, mennonite, repentance, seward, bribery, viewfinder
Happy	odette, instant, keynes, aficionados, miserably, drowning, shine, gossip, west, flirts
Sad	despise, analogy, stems, sang, gauri, obsessing, sympathizes, cabinets, language, favouring
Miniaturized	meadow, gladiatorial, dispenser, karaati, gnomes, pricing, can-do, alchemist, fairy-tale, dogpatch

Table 5: Examples of top 10 nearest words defined by cosine distance between the Multinomial GRU model's learned word embeddings.

References

[1] Imdb data. `ftp://ftp.fu-berlin.de/pub/misc/movies/database/`.

[2] Xgboost python package. `http://xgboost.readthedocs.io/en/latest/build.html`.

[3] Pretrained google news vectors. `https://code.google.com/archive/p/word2vec/`, 2013.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[7] Alex Blackstock and Matt Spitz. Classifying movie scripts by genre with a memm using nlp-based features, 2008.

[8] Leo Breiman. Arcing the edge. Technical report, Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[11] André Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687, 2002.

[12] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[13] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[15] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[16] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91, 1991.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[20] Eric Makita and Artem Lenskiy. A multinomial probabilistic model for movie genre predictions. *arXiv preprint arXiv:1603.07849*, 2016.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.

[23] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*, 2016.

[24] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[25] Ka wing Ho. Movies genres classification by synopsis, 2011.

[26] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.