Movie Genre Prediction Using Deep Learning

Submitted by

Anika Rahman	**160204012**
Ashikur Rahman	**160204025**
Mayisha Farzana	**160204028**

Submitted To

Mr. Mir Tafseer Nayeem
Assistant Professor

Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dhaka, Bangladesh

March 2021

ABSTRACT

As the amount of capital spent on film making becomes higher, the need to forecast a film's success early on has risen. Various measures have been taken to satisfy this need. The majority of the proposals have been based on movie ratings, trailer movie excerpts, and social media postings. Both of these, though, are only available after a film has been made and released. We suggest a deep-learning-based approach to predict a movie's success using only its plot summary text to allow an earlier prediction of its results. The findings of the suggested procedure's feasibility assessment are discussed in this article, which is accompanied by discussions and future works.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Movie plot summaries are expected to deliberate movies' genre as many viewers read the plot summaries before deciding to watch a movie. In this project, we perform movie genre classification from plot summaries of movies. Movie plot summaries reflect the movie genre, such as action, drama, horror, comedy, documentary, musical, etc., so that people can easily capture the genre information of the movies from their plot summaries. Significantly, several sentences in the plot summaries are highly representative of the genre of the movie. People generally read the plot summaries of movies before watching them get a concept about the movie. Therefore, plot summaries are written so that they reveal the genre information to the people. For example, if the plot declares humorous obstacles that must be overcome before lovers eventually come together, the movie will likely be a romantic-comedy. In this regard, there is a disguised representation of genre information in the movie plot summaries. We aim to learn this hidden representation. In other words, our objective is to classify the genres of the movies from their plot summaries, considering genre information represented by each individual sentence. By this process, representations of plot summaries can be used for movie recommendations. It can be inferred whether a plot summary reflects the genre of the movie it belongs to. Therefore, this method can be beneficial during the preparation of movie plots.

Websites like Netflix, HBO Go, Amazon Prime provide lists of movies based on genres; this makes it easier for viewers to select the movie that fascinates him/her based on the genre he/she is more inclined towards. Tagging movies is a complicated process and usually involves a manual process where the movies are allocated to one or more Genres based on the proposals sent by the users and consumers. If we can automate the process of tagging movies, it would be smoother, save time, and be more reliable than an untrained person. Categorizing movies makes it trouble-free for the audience to find which one they like and will want to see. Placing a film in a genre or division does not mean that it is of low quality

or lacks originality or creativity.

Moreover, when it comes to movies, most recent studies have concentrated on predicting movie ratings or sales, with relatively little research conducted on predicting movie genres. Movie genres are also tagged manually, with users sending their feedback to The Internet Movie Database's email address (IMDB). Since a plot description may relay much knowledge about a script, we tried to use different models to classify the movie genres.

Allocating genres to movies is a highly challenging task because the genre is a peripheral feature that is not present in a movie frame, so off-the-shelf detection models cannot be easily adapted to this context. As movies are not one-dimensional, one movie can spread several genres. That is why movie genre classification from the plot is a challenging task. For this, many movie plot summaries have been extracted and got down to work using this idea of multi-label classification.

Chapter 2

Related Work

In this section, we represent the related work for the movie genre prediction task. To the best of our knowledge, the best methods for genre classification/prediction were designed exclusively for single-label classification. Nevertheless, in real-world scenarios, a movie hardly belongs exclusively to a single genre.

[1]looks over different methods to categorize movies' genres based on the synopsis. The methods examined four different approaches, including One-Vs-All Support Vector Machines (SVM), Multi-label K-nearest neighbor (KNN), Parametric mixture model (PMM), and Neural network. All these procedures use the term frequency-inverse document frequency(TF-IDF) of the words as features. The dataset used for this experiment is relatively small, with only 16,000 movie titles for both the train and test sets. Moreover, the experiment is limited to only predicting only ten most popular genres: action, adventure, comedy, crime, documentary, drama, family, romance, short films, and thrillers. Overall, SVM attains the highest F1 score of 0.55. Here, the datasets are completely different.

[2] indicates a Naive Bayes version to predict movie genres fashioned on consumer rankings of the film. The idea is that customers are genuinely steady with their desire and like a few genres over the others. The evaluation metric is the share of movies that the version predicts efficiently as a minimum of genuine labels. Since the concern of my challenge is natural language processing(NLP), I attempt to predict movie genres using the use of the best movie plot summary.

[3]tries to categorize movie scripts by constructing a logistic regression model using the NLP-associated features extracted from the scripts, including the ratio of descriptive phrases to nominals or the ratio of dialogues non-dialogue frames. For every movie script, the model, primarily based totally on extracted capabilities, estimates the chance that the movie belongs to every genre and takes the k best rankings to be its expected genres, where k is a hyper-

parameter. The test is performed on a small dataset with the most effective 399 scripts, and the best subset of features achieves an F1 rating of 0.56.
In this project, we will try to capture a film and find out about the genres in which it falls using various modeling and vectorizers. We are going to use some models to try to improve genre prediction results. A review will be done on how the data set is processed and representative features selected, and the prediction accuracy improved.

Chapter 3

Project Objective

Our main objective is to implement an efficient model that can predict movie genres such as Thriller, Action, Comedy, Drama, Music, Western etc. by analyzing movie plot/summary. It helps to search specific types of movies which saves time. Besides, it helps consumers to know what to expect from the movie in advance. To perform this work, we've divided the main task into 5 subtasks. The subtasks are given below and also a flowchart is attached here. Using sentiment analysis methods, we attempted to accomplish the following goals in our research:

- Dataset Creation
- Dataset Preprocess
- Trained with Logistic Regression
- Trained with Deep Neural Network
- Analyze and Compare the results by presenting ablation experiment

3.1 Dataset Creation:

We have collected a dataset from Github. In the dataset, the movie plots have been classified into genre labels such as drama, action, romance, crime, war, music, etc. The raw dataset includes movie titles, movie plots, and movie genres. We extracted the movie plot summaries and their corresponding set of genres from the dataset to prepare our dataset for experiments. We used the training set to evaluate our approach, which contains crowd-sourced summaries from the actual users. The corpus contains 4456 movie plot summaries and their genres.

3.2 Data Preprocessing:

As seen from the below figure, the provided dataset has no missing values for any features. Text preprocessing is traditionally an important step for natural language processing (NLP) tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better. These various text preprocessing steps are widely used for dimensionality reduction. We use the following text preprocessing steps on the movie plot.

- Removing Stop Words in 'English'
- Converting all words into Lower Case
- Removing Punctuation and make it token-separator
- Tokenizing
- Removing accents

3.2.1 Removing Stop Words in 'English' :

"Stop words" are the most common words in a language like "the", "a", "on", "is", "all". These words do not carry significant meaning and are usually removed from texts. We use the Natural Language Toolkit (NLTK), a suite of libraries and programs for symbolic and statistical natural language processing, to remove stop words.

3.2.2 Converting all words into Lower Case :

CountVectorizer has a parameter lowercase that defaults to True. It converts all characters to lowercase before tokenizing. We convert every word into lowercase as features are case-sensitive.

3.2.3 Removing Punctuation and make it token-separator :

Regular expression denoting what constitutes a "token", only used if analyzer == 'word'. The default regular expression selects tokens of 2 or more alphanumeric characters (punctuation is completely ignored and always treated as a token separator).

3.2.4 Tokenizing:

Tokenization is a process that splits an input sequence into so-called tokens. Token can be word, sentence, paragraph etc. Override the string tokenization step while preserving the preprocessing and n-grams generation steps. Only applies if analyzer == 'word'.

3.2.5 Removing accents :

It removes accents and performs other character normalization during the preprocessing step. 'ascii' is a fast method that only works on characters that have a direct ASCII mapping. 'unicode' is a slightly slower method that works on any characters. None (default) does nothing. Here, we've kept it as default='None'.

3.3 Trained with Logistic Regression:

By using Logistic Regression, we've trained our dataset into 3 ways. The first method is to train Logistic Regression with TF-IDF on our dataset. The second way is to train with Logistic Regression along with Count Vectorizer. And finally, the last and third way is to train with Logistic Regression along with Binary Count-Vectorizer.

3.4 Trained with Deep Neural Network:

By using Deep Neural Network, we've trained our dataset into 3 ways. The first method is to train Deep Neural Network with TF-IDF on our dataset. The second way is to train with Deep Neural Network along with Count Vectorizer. And finally, the last and third way is to train with Deep Neural Network along with Binary Count-Vectorizer

3.5 Analyze and Compare the results by presenting ablation experiment:

At last, we've analyzed and compared our results of 2 different models using Evaluation Metric. The flowchart of our working procedure is given in the next chapter.

3.6 Sample Input and Output:

Input: "John McBut when a bomb goes off in the Bonwit Teller Department Store the police go insane trying to figure out what's going on. Soon, a man named Simon calls and asks for McClane. Simon tells Inspector Walter Cobb that McClane is going to play a game called S̈imon Says.̈ He says that McClane is going to do the tasks he assigns him. If not, he'll set off another bomb. With the help of a Harlem electrician, John McClane must race all over New York trying to figure out the frustrating puzzles that the crafty terrorist gives him. But when a bomb goes off in a subway station right by the Federal Reserve things start to get heated."

Output: Action [2]

Chapter 4

Methodologies

In this project, mainly we have trained models using Logistic Regression and Deep Learning Network and determined which model is better based on the ablation experiment. To perform this,we've to do several things such as Text Encoding, Preprocessing Dataset etc. These are described in the following two sections. An overview of our proposed model is shown below:

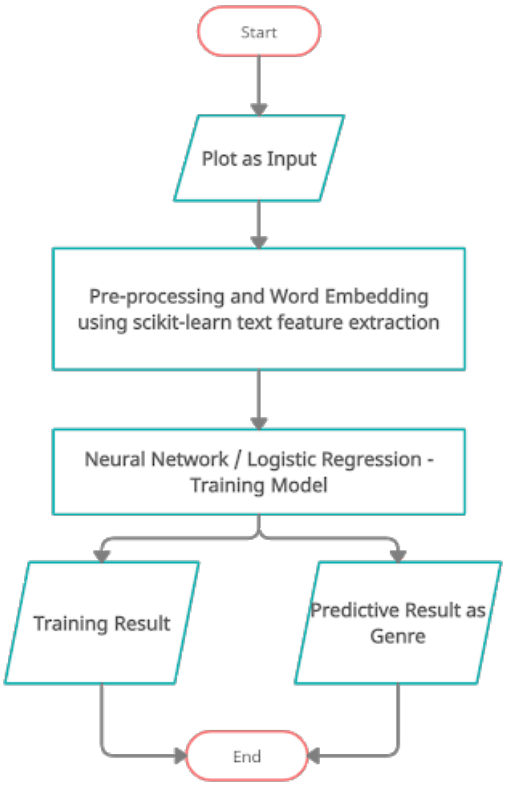

Figure 4.1: An overall procedure of movie genre prediction

4.1 Text Encoder:

When using machine learning algorithms, we are unable to deal explicitly with text. We would instead translate the text to numbers. Text encoding is converting text data into unique numeric vectors that machine learning algorithms can use. A few text encoders that were considered for this are listed below.

- **Word Count with Count Vectorizer (Bag of Words Model):** The Count Vectorizer is a primary method for tokenizing a collection of text documents, generating a vocabulary of recognized words, and encoding new documents using that vocabulary. The result is an encoded vector of the entire vocabulary's length and an integer count of how many times each word appears in the text.
- **Word Frequencies with TF-IDF Vectorizer (Bag of Words Model):** In the standard CountVectorizer model above, we used just the term frequency in a document of words in our vocabulary. In TF-IDF, we weight this term frequency by the inverse of its popularity in all documents. For example, if the word "movie" showed up in all the documents, it would not have much predictive value. It could be considered a stop word. TF-IDF is obtained by down weighing its counts by one divided by its overall frequency.

4.2 Preprocessing the Dataset:

In order to transfer the input data to the model, we need some changes. We will initialize the vocabulary based on the given corpus provided and create a dataset with given labels. We will initialize it for binary, count, and TF-IDF vectorizer and set binary vectorizer as default. As we have used the TF-IDF vectorizer, it will return us a TF-IDF value if we pass the fit transform value. After that, we will verify the word vector length and corpus label length. These two lengths should be equal for labeling. Otherwise, we cannot label it. We will convert the vector into the tensor type, add labels, and append it to the dataset. This is the preprocessing step.

4.3 Multilabel classification algorithm:

Classification is a predictive modeling problem that involves outputting a class label given some input. It is different from regression tasks that involve predicting a numeric value. Typically, a classification task involves predicting a single label. Alternately, it might entail

predicting the probability through two or more class labels. In both instances, the classes are mutually exclusive, ensuring the classification task assumes that the input belongs to one class only. Some classification tasks require predicting more than one class label. This suggests that class labels are not mutually exclusive. Multi-label classification is the term used to describe these functions. It is the technique to predict multiple predictions. We have to handle a few things differently in multi-label classification.

- **Activation Function :** In a neural network, an activation function specifies how the input's weighted number is converted into an output from a node or nodes in a layer. Activation functions usually are differentiable, which means that the first-order derivative can be determined for a given input value. Since neural networks are normally trained using the backpropagation of error algorithm, which uses the derivative of prediction error to update the weights of the model. Hidden layer activation functions are usually selected depending on the form of neural network architecture.To train our model, we have used the RELU activation function. This function provides better results and is simple to implement. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained.
- **Loss Function:** The difference between two probability distributions for a given random variable or sequence of events is measured by cross-entropy. Cross-entropy measures the number of bits needed to describe or transfer an average event from one distribution relative to another distribution, based on the concept of entropy from information theory. Since we are using a RELU activation function, we have to go with cross-entropy loss for our models.

4.4 Classification Models:

4.4.1 Neural networks for Multiple labels:

Neural network models can be configured to support multi-label classification and perform admirably, depending on the specifics classification task. Neural networks can explicitly assist multi-label labeling by merely defining the number of target labels as the number of nodes in the output layer. A task with three output labels (classes), for example, would necessitate a neural network output layer with three nodes. Relu is used as the activation function to introduce non-linearity into the prediction. The cross-entropy loss function must then be used to match the model. The details for configuring a multi-label neural network model are:

- The number of nodes in the output layer matches the number of labels.

- RELU activation for each node in the output layer.
- Cross-entropy loss function.

We will use some hyperparameters to train our model and Adam optimizer of stochastic gradient descent.

4.4.2 Logistic Regression for Multiple Labels:

Logistic Regression is designed for two-class classification problem. Here we will use a modified logistic regression model for our training dataset for multi-class classification. The Softmax function is used to solve multiple classification problems. We perform the model fitting test by applying some hyperparameters. Classification is done to predict a value with some small discrete values as outputs. Softmax function will compute the probability that the training sample will belong to which class was given the weight and net input. The details for configuring a multi-label logistic regression model are:

- In order to map predicted values to probabilities, it uses the Softmax function.
- Softmax function will compute the probability.
- Cross-entropy loss function.

We will take Adam as the advantage of sparse features and obtain a faster convergence rate than regular SGD with momentum.

Chapter 5

Experiments

5.1 Dataset and Feature Selection

This project aims to predict all of the potential movie genres depending on the storyline. Initially the provided [4] dataset consists of over 4k observations of movies. There are movies which are associated with movie titles, movie plots and movie genres.

Number of distinct Genres Collected	Number of Movie Titles Collected
19	**4456**

The list of Genres and their Distribution we found from our dataset is given below :

Serial No	Genre	Distribution	Serial No	Genre	Distribution
1	Animation	245	2	Action	765
3	Comedy	1339	4	Adventure	223
5	Biography	204	6	Drama	876
7	Crime	328	8	Fantasy	28
9	Mystery	22	10	Romance	17
11	Sci-Fi	13	12	Documentary	238
13	Family	11	14	Horror	127
15	Thriller	9	16	Short	7
17	Western	1	18	War	2
19	Musical	1			

The distribution of movies towards different genres is shown in the above figure. We can see that the number of ‘Comedy’ movies is more than all other genres and ‘Western’ and ’Musical’ movies are less than any other genres.

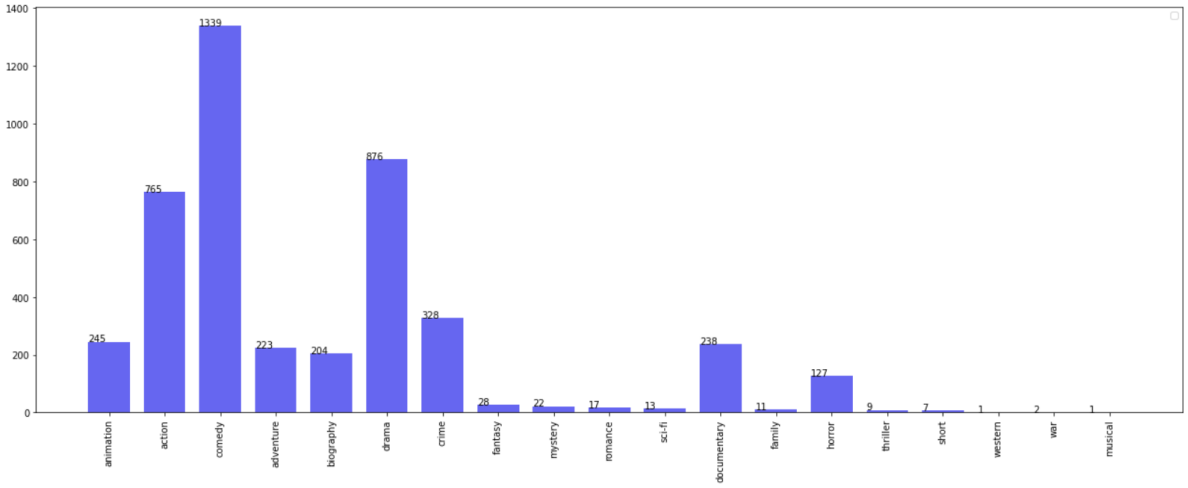

Figure 5.1: Number of movies per genre

We observe the following

- Lowest Genre movies is Western and Musical (only 1 movie in each genre)
- Highest Genre movies is Comedy (1339 movies), followed by Drama (876 movies)

5.1.1 Train/Test Split:

Now let us first split the provided data into train and test sets into 80:20. It was a multi-label dataset previously with highly imbalanced labels, then we drop a few columns and now we consider only single - label (Short and Musical having just 1 sample each and Comedy with >1300 samples). We split the provided data set into a ratio. We have taken 80% data as training and 20% for testing. After splitting, our dataset contains **Train : 3564 and Test : 892** . Using this, we will have a mixed dataset to classify the test samples correctly.

5.2 Evaluation Metric

The data consists of highly balanced data where each movie is classified into a single genre out of a maximum of 19. In order to measure the quality of the mathematical or machine learning process, evaluation metrics are used. For any project, evaluating machine learning models or algorithms is essential. There are several types of evaluation metrics available to evaluate a model. These include classification accuracy, logarithmic loss, confusion matrix etc. We compute accuracy, precision, recall, and F1 score to compare the performance of our movie genre prediction model.

- **Accuracy:** Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observations. It is conveniently suitable for binary as well as a multiclass classification problem.

  In here, we record our 19 genre predictions for each movie and calculate our accuracy as (Number of correct genre predictions for each movie)/(19*number of observations). Precision and Recall are metrics that measures the performance on the lowly occurring 'positive' class.

- **Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Precision(Genre=Action) = (Number of movies 'correctly' identified as Action Genre)/(Total number of movies that have been identified as Action Genre). Precision is the right choice of evaluation metric when we want to be very sure of our prediction.

- **Recall:** Recall is the ratio of correctly predicted positive observations to the all observations in actual class. Recall for a genre is given by Recall(Genre=Action) = (Number of movies 'correctly' identified as Action Genre)/(Total number of Action Genre movies in the data set)

- **F1 Score :** F1 Score is the Harmonic mean of Precision and Recall. F1 Score is the harmonic mean of precision and recall values for a classification problem.
  F1 Score = 2*(Recall * Precision) / (Recall + Precision)

  If our precision is low, the F1 is low, and if the recall is low again, our F1 score is low. It can be very helpful if the data is imbalanced.

- **Support :** Support is the actual number of occurrences of a certain class in the specified data set. Imbalanced support in the training data can indicate structural flaws in the reported classifier scores and indicate the need to stratify samples or rebalance. Support will not change between models, but rather a diagnostic evaluation process. In our model, we can see the overall distribution in each label through support.

5.3 Results

5.3.1 Classification Model

The algorithms that were used in our project includes-

- Logistic Regression
- Deep Neural Network

5.3.2 Algorithm Summary

Below is a list of all the models + encoders we have tried out -

Series	Encoder	Model
1	TF-IDF	Deep Neural Network
2	TF-IDF	Logistic Regression
3	Binary Count Vectorizer	Deep Neural Network
4	Binary Count Vectorizer	Logistic Regression
5	Count Vectorizer	Deep Neural Network
6	Count Vectorizer	Logistic Regression

Table 5.1: Showing all the models and encoders list

5.3.3 TF-IDF + Deep Neural Network

To make predictions, we used the TF-IDF vectorizer in combination with a deep neural network. The input layer consists of three hidden layers in this case. Each node has its own set of hidden nodes. Many of the nodes are part of a dense layer that is totally connected. To introduce non-linearity into the estimation, Relu is used as the activation function. Finally, we have a 19-neuron output layer (which corresponds to each genre). For optimization, we have used Adam Optimizer.
Hidden layers parameter are -
num hidden1 = 550, num hidden2 = 250, num hidden3 = 80. The optimal hyperparameters are given below:

Parameters	Values
Batch Size	180
Number of Iterations	9000
Learning Rate	0.001
Optimizer	Adam

Table 5.2: Hyperparameter Settings for TF-IDF + Deep Neural Network

	label	precision	recall	f1-score	support
Animation	0	0.27	0.20	0.23	49
Action	1	0.51	0.62	0.55	156
Comedy	2	0.58	0.61	0.60	269
Adventure	3	0.25	0.08	0.12	49
Biography	4	0.00	0.00	0.00	39
Drama	5	0.34	0.51	0.41	170
Crime	6	0.18	0.23	0.20	56
Fantasy	7	0.14	0.17	0.15	6
Mystery	8	0.00	0.00	0.00	2
Romance	9	0.00	0.00	0.00	3
Sci-Fi	10	0.00	0.00	0.00	2
Documentary	11	0.67	0.27	0.39	51
Family	12	0.00	0.00	0.00	3
Horror	13	0.00	0.00	0.00	32
Thriller	14	0.00	0.00	0.00	3
Short	15	0.00	0.00	0.00	1
Western	16	0.00	0.00	0.00	1
	accuracy		**0.44**		892

Table 5.3: Showing results using TF-IDF vectorizer and Deep Neural Network

5.3.4 TF-IDF+Logistic Regression:

To make predictions, we used the TF-IDF vectorizer in combination with logistic regression. It takes a weighted combination of the input features and passes it through a softmax function. Logistic regression is relatively robust to hyperparameter settings.
The optimal hyperparameters are given below:

Parameters	Values
Batch Size	180
Number of Iterations	9000
Learning Rate	0.001
Optimizer	Adam

Table 5.4: Hyperparameter Settings for TF-IDF + Logistic Regression

	label	precision	recall	f1-score	support
Animation	0	0.64	0.20	0.31	45
Action	1	0.56	0.68	0.62	146
Comedy	2	0.50	0.76	0.61	276
Adventure	3	0.27	0.16	0.20	38
Biography	4	0.00	0.00	0.00	43
Drama	5	0.37	0.44	0.40	173
Crime	6	0.35	0.15	0.21	60
Fantasy	7	0.00	0.00	0.00	5
Mystery	8	0.00	0.00	0.00	2
Romance	9	0.00	0.00	0.00	2
Sci-Fi	10	0.00	0.00	0.00	5
Documentary	11	0.86	0.32	0.46	60
Family	12	0.00	0.00	0.00	4
Horror	13	0.80	0.13	0.23	30
Thriller	14	0.00	0.00	0.00	1
Short	15	0.00	0.00	0.00	1
Musical	18	0.00	0.00	0.00	1
	accuracy		**0.49**		892

Table 5.5: Showing results using TF-IDF vectorizer and Logistic Regression

5.3.5 Binary Count Vectorizer + Deep Neural Network

We used the Binary Count Vectorizer in combination with a neural network to make predictions. In this case, the input layer is made up of three hidden layers. Each node has a collection of hidden nodes of its own. Many of the nodes are attached as part of a dense layer. Relu is used as the activation function to incorporate non-linearity into the calculation. Finally, a 19-neuron output layer is present (which corresponds to each genre). Adam has been used for optimization. Hidden layers parameter are -
num hidden1 = 550, num hidden2 = 250, num hidden3 = 80
The optimal hyperparameters are given below:

Parameters	Values
Batch Size	180
Number of Iterations	9000
Learning Rate	0.001
Optimizer	Adam

Table 5.6: Hyperparameter Settings for Binary CountVectorizer + Deep Neural Network

	label	precision	recall	f1-score	support
Animation	0	0.46	0.31	0.37	39
Action	1	0.56	0.59	0.58	150
Comedy	2	0.57	0.68	0.62	282
Adventure	3	0.32	0.18	0.23	45
Biography	4	0.24	0.15	0.18	40
Drama	5	0.32	0.41	0.36	177
Crime	6	0.34	0.28	0.31	65
Fantasy	7	0.00	0.00	0.00	5
Mystery	8	0.00	0.00	0.00	5
Documentary	11	0.68	0.48	0.56	44
Family	12	0.00	0.00	0.00	2
Horror	13	0.38	0.15	0.22	33
Thriller	14	0.00	0.00	0.00	3
Short	15	0.00	0.00	0.00	2
	accuracy		**0.47**		892

Table 5.7: Showing results using Binary Count Vectorizer and Deep Neural Network

5.3.6 Binary Count Vectorizer+Logistic Regression:

We used the binary count vectorizer following logistic regression to make predictions. Logistic regression is reasonably insensitive (or robust) to hyperparameter conditions. Given the weight and net data, the Softmax function will calculate the probability that the training sample will belong to the class. Finally, a 19-neuron output layer is present (which corresponds to each genre). Adamax was seen as an optimizer. The optimal hyperparameters are given below:

Parameters	Values
Batch Size	220
Number of Iterations	6000
Learning Rate	0.001
Optimizer	Adamax

Table 5.8: Hyperparameter Settings for Binary CountVectorizer + Logistic Regression

	label	precision	recall	f1-score	support
Animation	0	0.65	0.25	0.36	60
Action	1	0.53	0.61	0.57	153
Comedy	2	0.47	0.78	0.59	248
Adventure	3	0.11	0.03	0.04	40
Biography	4	0.43	0.08	0.13	38
Drama	5	0.39	0.46	0.42	180
Crime	6	0.34	0.12	0.18	80
Fantasy	7	0.00	0.00	0.00	10
Mystery	8	0.00	0.00	0.00	4
Romance	9	0.00	0.00	0.00	2
Sci-Fi	10	0.00	0.00	0.00	5
Documentary	11	0.90	0.47	0.62	38
Family	12	0.00	0.00	0.00	1
Horror	13	0.20	0.04	0.06	28
Thriller	14	0.00	0.00	0.00	4
Short	15	0.00	0.00	0.00	1
	accuracy		**0.47**		892

Table 5.9: Showing results using Binary Count Vectorizer and Logistic Regression

5.3.7 Count Vectorizer + Deep Neural Network

To make predictions, we used the Count Vectorizer following a neural network. The input layer, in this case, is made up of three hidden layers. Each node has its own set of hidden nodes. As part of a dense layer, all of the nodes are connected. To integrate non-linearity into the equation, Relu is used as the activation function. Finally, there is a 19-neuron output layer (which corresponds to each genre). We have used three hidden layers in here. Adamax Optimizer is used here.
Hidden layers parameter are -

num hidden1 = 550, num hidden2 = 256, num hidden3 = 64
The optimal hyperparameters are given below:

Parameters	Values
Batch Size	200
Number of Iterations	9000
Learning Rate	0.001
Optimizer	Adamax

Table 5.10: Hyperparameter Settings for CountVectorizer + Deep Neural Network

	label	precision	recall	f1-score	support
Animation	0	0.39	0.38	0.39	42
Action	1	0.60	0.58	0.59	171
Comedy	2	0.51	0.66	0.58	288
Adventure	3	0.22	0.15	0.18	34
Biography	4	0.29	0.14	0.19	36
Drama	5	0.39	0.41	0.40	181
Crime	6	0.31	0.24	0.27	62
Fantasy	7	0.00	0.00	0.00	3
Mystery	8	0.00	0.00	0.00	8
Romance	9	0.00	0.00	0.00	3
Sci-Fi	10	0.00	0.00	0.00	8
Documentary	11	0.62	0.28	0.38	36
Family	12	0.00	0.00	0.00	2
Horror	13	0.18	0.20	0.19	15
Short	15	0.00	0.00	0.00	2
Musical	18	0.00	0.00	0.00	1
	accuracy		**0.47**		892

Table 5.11: Showing results using Count Vectorizer and Deep Neural Network

5.3.8 Count Vectorizer+Logistic Regression:

To make predictions, we used the count vectorizer after performing logistic regression. Hyperparameter environments are relatively indifferent (or robust) to logistic regression. Softmax function will compute the probability that the training sample will belong to which

class given the weight and net input. Finally, there is a 19-neuron output layer (which corresponds to each genre). Adam was seen as an optimizer. The optimal hyperparameters are given below:

Parameters	Values
Batch Size	200
Number of Iterations	9000
Learning Rate	0.001
Optimizer	Adam

Table 5.12: Hyperparameter Settings for CountVectorizer + Logistic Regression

	label	precision	recall	f1-score	support
Animation	0	0.32	0.15	0.20	41
Action	1	0.57	0.63	0.60	150
Comedy	2	0.48	0.67	0.56	269
Adventure	3	0.19	0.07	0.11	40
Biography	4	0.43	0.06	0.11	48
Drama	5	0.34	0.45	0.39	181
Crime	6	0.24	0.09	0.13	78
Fantasy	7	1.00	0.25	0.40	4
Mystery	8	0.00	0.00	0.00	4
Romance	9	0.00	0.00	0.00	2
Sci-Fi	10	0.00	0.00	0.00	3
Documentary	11	0.66	0.48	0.55	48
Horror	13	0.50	0.14	0.21	22
Short	15	0.00	0.00	0.00	2
	accuracy		**0.45**		892

Table 5.13: Showing results using Count Vectorizer and Logistic Regression

In this report, we used multiple techniques to train a model to predict all of the genres (up to 19 potential genres) that a movie can be categorized into based on its plot. We looked at two text encoding algorithms (Count Vectorizer and TF-IDF Vectorizer) and different modeling algorithms (Logistic Regression, Neural Networks). Our ablations are compared in the table above. When we combine the TF-IDF vectorizer with logistic regression, we get an accuracy of (.49%), which is higher than all other models. We used multiple models and vectorizers here, and from the results, we can conclude that the Logistic Regression model estimates more reliably than any of them.

Chapter 6

Conclusion

Based on the experiments' results, we may conclude that using different model techniques to predict movie genres is very effective and challenging. Experimenting with a variety of features, as well as modeling, increased accuracy. Tfidf, vectorizer algorithms, and deep learning are just some of the methods we have used. The provided dataset was relatively small, and the plot's feature size was quite large. Deep learning did not do well due to a lack of data points. In future, we'll try to implement more modeling techniques such as SVM, KNN etc. to improve our accuracy than before.

References

[1] K.-W. Ho, "Movies' genres classification by synopsis," 2011.

[2] E. Makita and A. Lenskiy, "A multinomial probabilistic model for movie genre predictions," *arXiv preprint arXiv:1603.07849*, 2016.

[3] A. Blackstock and M. Spitz, "Classifying movie scripts by genre with a memm using nlp-based features," *Citeseer*, 2008.

[4] ishmeetkohli, "Dataset," *https://github.com/ishmeetkohli/imdbGenreClassification?fbclid=IwAR03-4ARb8qEB04kJqCa6tDe9pGINd8H-THYxyx8qQ2s-5En2BHBPqbkGHw*.