MOVIE GENRE PREDICTION USING NLP TECHNIQUES

Atharva Gokhale, Anchit Nagwekar, Parth Thakur

University of Waterloo

ABSTRACT

Natural Language Processing (NLP) is a field that enables computers to interpret textual data and pre-process it for machine learning algorithms. Determining movie genres accurately helps audiences to make informed choices about the movie they would like to watch. Movie title and its synopsis forms a good source of information for determining its genre. We have implemented classical, deep learning and state of art simple transformer algorithms to classify movies accurately based on their genre given their synopsis. We have designed a user interface which enables the user to input the synopsis of the movie, display the predicted genre, provide a feedback loop and recommendations of top three movies having the same genre and maximum similarity with the movie under consideration. A key feature of our system is the feedback loop which plays a vital role in modifying the genre of a particular movie in our training data set. Users can provide a different genre for the movie based on their perspective and provide feedback. Eventually, we can expect that the training data set will be modified based on the viewers perspective and the model will be retrained after a certain number of feedbacks. Thus, our model over time will predict the movie genre based on the perspective of the majority population and not based on the genre defined by the production house

Index Terms— *Movie Genre, Classification, Feedback, Recommendation*

1. INTRODUCTION

A movie genre is a category that the movie belongs to depending either on the way it is narrated or the response that it achieves in terms of emotions from its audience. In reality a movie can belong to a single or multiple genre, however we have built our dataset with just one genre for a movie. A movie genre is determined by the production house initially and by audience and critiques after its release depending upon personal perception. Audiences often decide the movie which they want to watch by considering a number of factors like the actors and the actresses in the movie cast, the story line, the characters played, directors, etc. However, a movie genre plays a very important role in making the final choice. We expect that the users would come back and provide their feedback related to the predicted genre based upon their personal choice.

We consider a multi-class classification problem which takes in the plot synopsis of the movie as input and spits out the genre it belongs to as output. A movie synopsis is a description of the movie which explains the plot and the story in brief. The synopses and the genres of the movies in the training dataset have been initially picked up from the original source (IMDb Datasets). However, the genres may be modified over time as users provide feedback about the genres according to their perception.

We built an end to end solution for predicting the movie genre and suggest similar movies. A web interface was created using the flask micro-web services [1] which lets the user enter the synopsis and title of the movie of which the genre needs to be predicted. We use these features to predict the genre using various classical and deep learning methods. We start with the basic Multi-Naive Bias Classifier (MNB) and eventually progress with more complex methods of deep learning and transformer algorithms in order to increase the prediction accuracy. The final genre classification is done using the BERT transformer classification algorithm as we achieved the highest results compared to existing classical and deep learning models which have been used for movie genre prediction. After predicting the genre of the movie under consideration we provide two additional functionalities which include the user feedback and recommendations. The feedback mechanism enables the user to report the genre of the movie as per their understanding. The recommendation engine provides top three movies of the similar genre which are closely related to the movie title entered by the user. The recommendation engine has been implemented with multiple methods such as cosine similarity and the Prediction IO Machine Learning server.

2. LITERATURE REVIEW

2.1. Classical Machine Learning Approaches

Classical machine learning approaches are one of the easiest to implement and provide compelling results. The most common approaches for text classification include, MNB (Multinomial Naïve Bias), SVM (Support Vector Machines) and Decision Trees. Previous work done on these algorithms for text classification has been discussed in this section.

2.1.1. Naïve Bias Classifier

Naïve Bayes classifier is one of the oldest yet a very effective classification algorithm. It assumes that the probability of occurrence of a feature is independent of one another [2]. However, even after the incorrect assumption in case of sequential data, the performance of Bayes theorem for text classification is considerable . There are 2 methods which have been generally used by the Bayesian model [2], the 1st one being the Multivariate Bernoulli Model. Here the words in a text or a "document" are treated as a binary string. Each document is treated as a Bernoulli experiment and the probability of a document belonging to a certain class is calculated. The second method is called a Multinomial model, [2], [3]. Here the "term frequency" is considered while using the Bayes Theorem to calculate probability of occurrence of a particular class. Here, instead of the entire document, each word is treated as multinomial distribution, which is responsible for the probabilistic class of the document. The results shown by [2] reveal that MNB works better in most scenarios.

2.1.2. Support Vector Machines

Support Vector Machines use the principle of setting a decision boundary while separating classes linearly. [4] and [5] provide a detailed description of how SVM along with an RBF kernel can be implemented for the words in a document into a term frequency vector, where each word is assigned a numeric value based on the number of times it occurred in a document. This can then be passed onto the SVM classification model which separates classes based on their geometric boundaries. [4] compared this model to K-Nearest Neighbors and found out that SVM performs consistently better for different datasets, whereas KNN performs better only for certain conditions when the hyper parameter are tuned. [4] claims SVM is a good algorithm for text categorization due to the nature of the data being sparse, all features being important for separation and the data is generally linearly separable. However, SVM's are generally more complex to Naïve Bias and KNN. [5] uses SVM to predict movie genres based on their synopsis, and lays emphasis on the importance of terms or the tokens in the selection of the hyper parameter to get the most optimum results.

2.1.3. Decision Trees

Word2Vec + gradient boosting is used by [6] pointing out the shortcomings of the Bayesian model where the probabilities of occurrence are considered independent which is not generally the case. Here Word2Vec creates embedding for certain words which are more commonly associated to a particular genre which is then fed to a gradient tree boosting model. A similar approach has been recommended by [7] where Doc2Vec has been used for embedding the data before the data has been passed to a Random Forest Classifier. Random Forests combines low depth decision trees and takes an aggregate of all the trees to come up with a prediction with reduced computation time.

2.2. Deep Neural Networks

We have enormous number of research articles who have proposed their deep neural network models for multi-class text classification problems. Many of these research work uses the audio-visual content such as the movie trailer [8], advertising banners [9] to extract the necessary features related to the movie to classify it into a particular target class.

2.2.1. Convolutional Neural Networks

Our first and baseline deep learning approach towards the multi-class text classification problem is using the Convolutional Neural Networks (CNN) in accordance with the model proposed by [7]. According to [10], CNN performs better in "extracting local and position-invariant features" from text data which makes it a potential model for classification problems. In order to further increase the model accuracy, our next approach is using recurrent neural networks which are used explicitly from sequential data [11].

2.2.2. Recurrent Neural Network

Bidirectional Long Short Term Memory (LSTM) achieves promising results in movie genre prediction according to experiments conducted by [12]. According to [12] using bidirectional LSTM outperforms the basic recurrent neural network model and classical machine learning models such as Support Vector Machine (SVM) and Linear Regression. Even though LSTM achieves better prediction accuracy on unseen data, the training time is significantly high for large datasets.

2.3. Transformers

The major advantage that transformers offer is their attention mechanism which can have an infinite reference window [13]. This is where they have an edge over methods like RNN, GRUs and LSTMs. Transformers are different based upon encoder-decoder sub-parts [14]. Encoders learn about the context and the language used in the text data whereas decoders learn the mapping between multiple languages. There are various existing models like Google's Bidirectional Encoder Representation from Transformer (BERT) which consists of stacked Encoders.

Encoder based transformers do not have recurrence like there is in RNNs. Thus they have to add positional information in the embedding. Sine and cosine functions are used to calculate positional encodings which are then added to the respective embedding vector. This ensures that the position of the word in a sentence is considered while forming the embeddings. Bidirectional Encoder Representation from Transformer (BERT) by Google is trained in 2 phases. The pre-training phase involved training over BooksCorpus (800 M words) and English Wikipedia (2500 M words). The fine tuning phase involves tuning the model for data particular to the task. These enable BERT to learn valuable insights regarding the language. [15][16]

3. IMPLEMENTATION

3.1. Dataset

We explored the IMDb website for obtaining movie synopses which is updated regularly. Various datasets exist on popular sources like kaggle, github, etc. [17] [18]. However, these sources contained movie synopses along with multiple tags and some unnecessary features as far as our problem is concerned. Our multi-class problem formulation called for a single genre label for each movie sample. Thus, we scraped the movie synopses and only the first genre provided by the IMDb website by using the python package beautiful soup. Our data consists of a unique IMDb movie id, movie title, movie synopsis and their tags. Some samples had missing information about the synopsis. We retrieved this information from the IMDb website using the unique IMDb movie ids in the dataset through web scraping. Our final dataset spans ten movie genre classes namely Comedy, Drama, Action, Crime, Adventure, Documentary, Horror, Biography, Animation and Fantasy. Each movie sample in the dataset definitely belongs to one of these classes. We sampled this dataset to include about 10,000 total samples with random distribution with respect to the genres they belong to. We randomly partitioned the dataset into training and testing sets using a ratio of 9:1. The following pie chart shows the distribution of the samples in our dataset according to their genres.

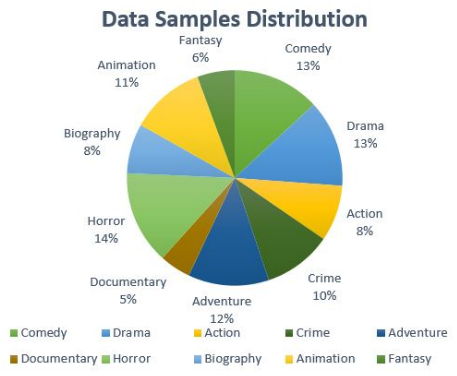

Fig. 1. Distribution of the movie samples according to genres [1]

3.2. Data cleaning and Pre-processing

The dataset formed above needed to be cleaned before making it ready to be given as input to a model. Some of the raw synopses which were incomplete and couldn't be scraped from the IMDB website needed to be eliminated. We queried the genre column and made sure that it only had one word belonging to the genres mentioned in the previous subsection. We also made sure that no unwanted symbols were part of the movie title column as well.

All the text was converted to lowercase to maintain consistency. We also removed all special characters from the synopsis text as these do not contribute to the overall inference which can be derived from written text and simply add to the size of the dataset. The next step we followed was removing stop words from all the synopses. Stop words are supportive words like 'the', 'a', 'an', 'is', 'for', 'of', etc. which help in sentence construction. Like punctuation marks, even stop words do not play a role in the genre. So it is safe to remove them. The next step in pre-processing the data was lemmatizing the data text. Lemmatizing the data refers to converting words into the smallest possible dictionary root word. It helps in reducing redundancy and limit the training vocabulary size. We use the wordnet lemmatizer from Natural language Tool Kit. In order to limit the number of words, we put a cap of 100 words for the synopses. We used zero padding for synopses with under 100 words and truncated any which had more than 100. Thus, the dataset is now ready for passing on as input to a machine learning model.

3.3. Application User-Interface

3.3.1. Webpage Design and Feedback Loop

It is our attempt to provide an end-to-end solution for the challenge of movie genre prediction based on the synopsis of the movie and additionally recommend similar movies to the user. We have used the flask web framework available in python [1]. Flask being a micro-framework it aims to maintain lightweight simplicity and provide flexibility. Flask proves to be a perfect solution for our problem considering its property of doing one task at a time and doing it well. We have developed a basic web-application using flask framework which enables the user to input the movie synopsis and movie title in the text-boxes provided 2. On submitting the request to predict the genre, a new web page appears with the predicted genre of the movie and suggested recommendations along with the provision to give a feedback as observed in figure 3.

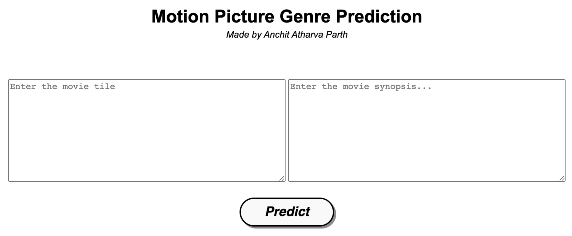
Motion Picture Genre Prediction
Made by Anchit Atharva Parth
Enter the movie tile
Enter the movie synopsis...
Predict

Fig. 2. Web-page Interface to enter the movie synopsis and title [1]

We have further added few more functionalities to the application which enables the users to provide a genuine feedback for the prediction and provide recommendations of the movies belonging to the same genre. The feedback mechanism is a simple way-out that provides the user an opportunity to correct the training dataset. We append the mode of the feedbacks given by the user to the target-label column of the training dataset and retrain the model after 5 feedbacks for a particular movie title. Here, we assume that the feedback given by the user is correct as per their interpretation of the movie. This modification added to the application provides a feedback loop to the process and improves the performance of the model eventually.

3.3.2. Recommendation Engine

Initially we tried to build a basic recommendation engine based on the features set such as the cosine similarity between the TF-IDF vectors of the words belonging to the movie title section having similar genre as the predicted one. However, the performance of this recommendation engine didn't seem to be at par in comparison to those built with the state of art recommendation engines such as PredictionIO.

Finally we tried to use Apache's Prediction IO Machine learning server to build a recommendation engine which will provide recommendations to the user based on the genre of the synopsis entered for prediction. Apache's PredictionIO is an open source Machine Learning platform which provides access to many template galleries which consist of a support system for providing recommendations. According to [19] we will use the Recommendation template with PredictionIO support system. We just need the data which includes authorized links to all the movies we will be recommending and most importantly the genre that our model has predicted for a given text input. We display the poster of the recommended movies on the web-page which will be eventually redirected to the hyperlink of the movie. Currently, we plan to recommend 3 movies with the same genre as predicted and which are quite similar to the movie of which synopsis is entered by the user.

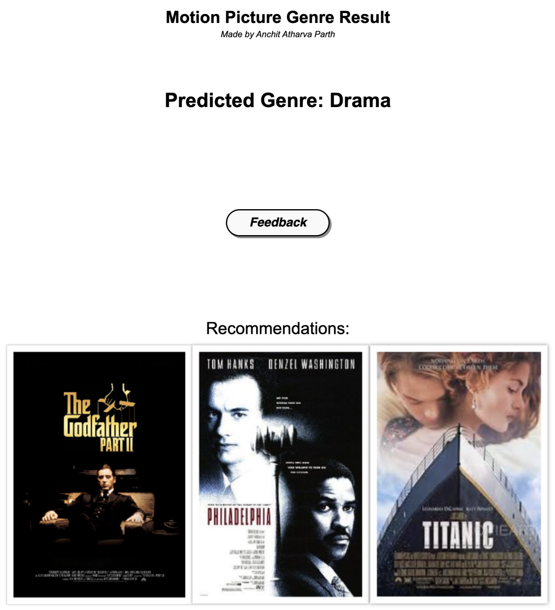

Fig. 3. Interface displaying predicted genre and Recommendation along with feedback loop [1]

3.3.3. Feature Engineering

Features are vectorized for each classical machine learning methods as: preprocessed data is converted into numeric format using count vectorizer initially. Later, we used TF-IDF vectorizer which uses term frequency in each document to convert the strings into a numeric value. Additionally, we deployed n-gram (n=3) in TF-IDF vectorizer which extracts the training features and

achieves better results.
Majority of the research articles related to the movie genre prediction using deep learning techniques have used the conventional features such as Word embeddings based on Word2Vec, Continuous Bag of Words (CBOW), ngrams and few dataset specific handcrafted features [20]. According to [12] the classification models using the word embeddings alone underperforms as the word embeddings can only handle the seen words in the test dataset. We extract the features from the training data using the character embeddings, word embeddings and sentence embedding in order to form an accurate word vector even if the word do not exist in the vocabulary of the training data-set as proposed by [21].

3.4. Classical Machine Learning Techniques

3.4.1. Multinomial Naive Bias (MNB) Classifier

It is one of the most successful classical machine learning algorithms which have been used for text classification. We have set this approach as a baseline approach. Taking into consideration the description of movies as the features initially, MNB was implemented and the accuracy on the testing set was observed. In the next step, the title was added to the synopsis as the training features, and MNB classifier was implemented. We observed that generally the accuracy of the latter method turned out to be better since there was an inclusion of more features. Thus, we decided to use both, the title and the description as the input as sometimes, titles have less features but provide a better overview of the genre, e.g. "Space Exploration" being the title is capable enough to link the word "Space" to the genre "Sci-Fi". MNB does not have any tunable parameters, and the model is implemented with the default hyper-parameters.

3.4.2. Support Vector Machines (SVM)

Another classical machine learning algorithm which was used is SVM. The data was converted from a string to a TF-IDF vector which is fed into the machine learning algorithm. The algorithm uses boundaries to separate various data points having different classes created by plotting the various features. This algorithm works geometrically and is better in our case compared to naïve bias as the inputs are transformed using the RBF kernel and then the classes are separated. A grid search was conducted for the hyper parameter $c \in (0.01, 0.05, 0.1, 1, 5)$ and the parameter giving the most accurate result was chosen (C=5). Similar to the implementation of the Naïve Bias model, both sets of input features were tried, one with just synopsis as the input and then a combination of the synopsis and the title as the input. We observed a similar trend to MNB where the second set of features was more accurate in predicting the genre. A consensus was made to use features which are mixture of both, the plots and the titles for the models here on, since it was proving to be more accurate consistently.

3.4.3. Hybrid Machine Learning Model

Having performed MNB and SVM we did not achieve satisfactory results. We were not planning on using decision trees in our approaches due to their unacceptable performance in the presence of a large number of features [22]. However, we decided we could use them in an ensemble model combining MNB, SVM, Random Forest Classifier and Gradient Tree boosting. The predictions for SVM and MNB were achieved as described above and the predictions for Gradient Tree Boosting and Random forests was obtained by using TF-IDF vector as input and setting a low tree depth (max depth = 10) and increasing the number of trees to (n estimators = 1000). This was done to reduce the time complexity and increase accuracy. The outputs from all 4 methods achieved were combined and the majority classification (mode) from all for methods for a testing example was considered as the final classification.

3.5. Deep Neural Networks

3.5.1. Preprocessing the training and test data

It is necessary to implement few preprocessing steps before feeding the training data to the neural networks. In our case, the movie title and its description (synopsis) are the input features. Vectorizing the input data is the first step as textual data cannot be fed to neural networks. Then, we pad the word vectors to limit the vector length to a particular value in order to maintain uniformity of all the input samples. We truncate the extra words in the synopsis section or zero-pad the ones with less number of words. We have limited the vector length to 128 considering the training time.
We have 10 target classes representing 10 genres of the movies we have considered in our classification problem. These target labels are converted to categorical state. On predicting the genre of the synopsis entered by the user, the predicted output will be probability values ranging between 0 and 1. The output layer activation function will be responsible for deducing the accurate target class from the output values. In this case, we use the categorical cross-entropy loss to monitor the training loss as well as the test loss.

3.5.2. Embedding Matrix for Neural Network Models

Formulating the word embeddings using the training data or using the pretrained embedding model is the necessary step in case of any language processing problems. This step forms vector corresponding to each word in the dataset. The word vectors capture the grammatical functions and the meaning of the words upon which we can perform mathematical operations to classify them into the appropriate target class. In our case, we can say that the training vocabulary is very limited which encourages the use of pretrained embedding models which are trained on a very large dataset which supposedly include majority of the words in a particular language.

As a baseline approach we trained a word2vec model using the our training data to form the embedding matrix. We observed that the test accuracy was in the range of 55%-60% which can be considered as a poor classification model. Thus, we opted for using the unsupervised pretrained models that used co-occurrence frequencies of the words to generate the word vectors. Later, we used the pretrained 100-dimensional Glove model [23] developed in Stanford University and trained on the twitter dataset. Using the embedding matrix formed using the glove embeddings, the model achieved significant increase in the test accuracy. Using the 300-d pretrained glove model can further increase the accuracy but will also increase the training time for the available hardware resources. We also tried using the fasttext word embedding model [24] developed by Facebook to construct the embedding matrix.

There are few research articles and implementations who have used the word2vec, glove and fasttext embeddings to train their neural networks in order to build an efficient classification model. However, using the word embeddings constructed using any of the above mentioned algorithms, we can further improve the model performance by using the character embeddings. Even though the pretrained word embedding models are trained on a very large vocabulary, there is huge probability that few words appearing in the train and test data are absent in the model vocabulary. These words are called as Out of Vocabulary (OOV) words and are assigned random vector values.

Thus, we need a different approach to handle these OOV words in our dataset. We have implemented the character level embeddings as a solution to the problem which uses a 1-dimensional convolutional neural network to determine the vector representation of the words by looking at the character level composition of each word in our dataset [25]. The 1-D CNN goes through multiple scans of the word character by character. The scanner can focus on multiple characters at a time and extract the necessary information. Finally, a word vector representation is built using the information extracted by the scanners from the characters of the word.

We can observe that the format of embedding matrix built using character embeddings is similar to that of embedding matrix built using word embeddings. This method helps us to extract the meaning not only from individual words but also from the fragments of words that can contribute towards classification. Eventually, we concatenate the embedding matrix built using character embedding to the matrix built using the word embeddings. We could observe that the model prediction accuracy increases considerably with the updated method.

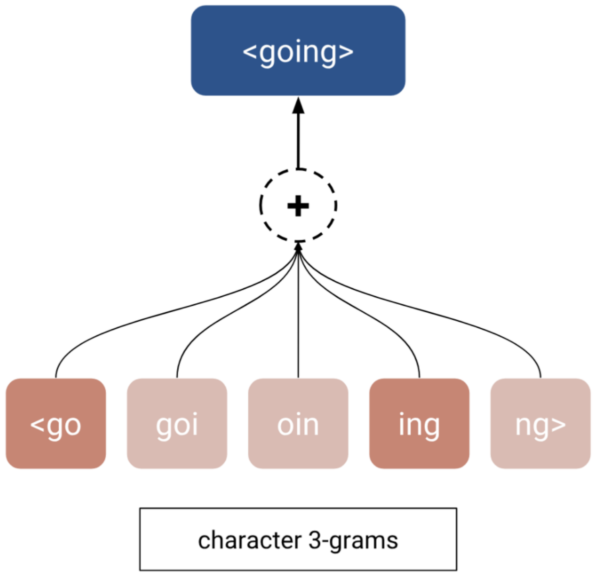

Fig. 4. Algorithmic Representation of Character Embeddings [26]

3.5.3. Convolutional Neural Network (CNN)

In our approach of developing a convolutional neural network for predicting the movie genre we have made few necessary modifications to the existing approaches in order to determine which embeddings best suit our dataset in order to achieve the best test accuracy. The training time of CNN is the lowest as compared to the Recurrent Neural Networks and Transformer architectures, thus, we have experimented with different word embeddings while training the CNN model. We have deployed an embedding layer where the weight matrix values are borrowed from the embedding matrix for the words that are present in the training data. Next, we have added multiple 1-D convolutional layers which extract the information from the training data using different filters along with a particular activation function.

At every stage, the filter value, either 0 or 1 is multiplied with the word embedding values resulting in 0 or 1 on passing through the activation layer. Thus, the convolution filters establish relation between two adjacent words, frequency of two words repeating after each other and relation between two words with a word in between. The stacked convolutional layers helps in extracting more complex features from the training data. We have eventually reduced the number of filter from 512 to 64 with each stacked convolutional layer. Also, we have added the maxpooling layer after each convolutional layer in order to generalize the feature values and reduce the feature size. Next to the stacked convolutional layers, we have used 3 fully connected dense layers (hidden neurons = 128) with dropout layer (dropout rate = 0.35) after each dense layer to avoid the model overfitting on the training data. Finally, we have used the softmax activation function as we want to classify the test samples into multiples classes. While compiling the model we have used the ADAM optimizer and categorical cross entropy loss which is the preferred combination of hyper-parameters for multi-class classification.

Now, considering the word embeddings, we initially started with the word2vec model trained on our training data as our baseline model. Further, to increase the accuracy we opted for 50-d Glove pretrained model and also deployed 50-d fasttext pretrained model. Increasing the dimensionality of the word embeddings increase the accuracy marginally while the training time increases significantly. Next, we tried to introduce some novelty to the existing implementations by using the character embeddings to construct the word embedding matrix. We observed that, the best test accuracy for CNN model was observed for the last approach.

3.5.4. Recurrent Neural Networks (RNN)

We know that the recurrent neural networks are more preferred for sequential data such as text. Recently, we can observe that large emphasis is given on using Long Short Term Memory (LSTM) networks for language related problems. LSTM's is a version of RNN with an extra memory cell to store timely information. Using the standard recurrent neural network model for text classification yields poor results as compared to the CNN model. Thus, we used the vanilla LSTM model for better results with the standard values of the hyper-parameters as mentioned in the LSTM layer defined by Keras. We can observe a slight increase in the test accuracy using the LSTM model as compared to the RNN model.

Further, we used the bi-directional LSTM model which can be considered as the state-of-art technique for majority of the text classification problems. The bidirectional LSTM model has two LSTM networks, the first network accesses the data in forward direction while the other accesses the data in reverse direction. Thus, we can say that the network can access the data in past and future at the same time while training the model. We have deployed three variants of the bidirectional LSTM model. The first one is a single layer bidirectional LSTM model, the second one is the stacked bidirectional LSTM model and the third one is the concatenated model. Obviously, the stacked model achieved better test accuracy as compared to the first one but the training time is significantly high.

We tried a different approach where we extracted the features from the movie title and movie synopsis sections separately while training the network [27]. Here, we used two separate bidirectional LSTM network to extract the information or the feature set from the movie title and movie synopsis respectively. Later, we concatenated the two models using the concatenation layer and then fed the training features which were extracted earlier to the fully connected dense layer which is responsible for classifying the training samples into respective target classes.

For all the recurrent neural network models, we have used the character embeddings to construct the embedding matrix. The LSTM layer in the basic LSTM model includes 120 hidden neurons. Similarly, in case of all bidirectional models we have used 120 hidden neurons and dropout rate of 0.35. In the output layer we have used the softmax activation function along with 10 neurons as we have 10 target classes. Again, the optimizer used is ADAM and the metric mentioned while training the network is categorical cross-entropy loss.

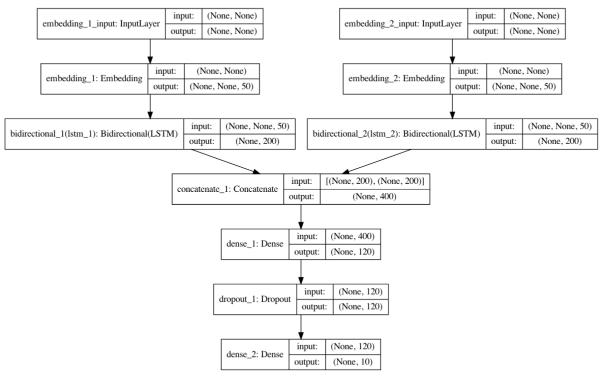

Fig. 5. Concatenated Model Architecture

We could observe that using separate bidirectional LSTM networks for movie title and movie description did not surpass the test accuracy achieved using the stacked bidirectional LSTM. Thus, we can say that using the movie title and its description together proves to be more beneficial in context of extracting necessary information from the training data. However, the concatenated model outperformed the model accuracy achieved using the CNN and basic LSTM models.

3.6. Transformer model: BERT

Bidirectional Encoder Representations from Transformers (BERT) involves joint conditioning on left and right context in the layers to pre-train representations from the text which are bidirectional. It uses the Masked Language Model (MLM) [28] which randomly masks few tokens from the input with the goal of predicting the masked word on the basis of its context. It also makes use of Next Sentence Prediction (NSP) [28] for pre-training. As mentioned in section 2.3, the self-attention mechanism lets BERT to model downstream tasks irrespective of the type of text - single or pairs. The steps for each task include plugging in the specific inputs and outputs for the task to BERT and then fine-tuning the parameters for making this pre-trained model learn according to our application and dataset.
We implemented BERT using the uncased model with 12 layers which has a total of 110M parameters [27]. It has 12 attention heads. We used the inbuilt classification model which uses the word embeddings created using the BERT architecture. We changed the learning rate from the default value of 0.00004 to 0.00002. We tried implementing the transformer model for various values of maximum sequence length. Maximum test accuracy was obtained for a maximum sequence length of 256. This transformer model was trained for 2, 5 and 7 epochs of which 5 proved to be the best. The highly pretrained nature of BERT justifies the highest accuracy we achieved compared to all the previous approaches.

4. RESULTS AND ANALYSIS

While implementing our models on the synopsis based features we realised that only using movie titles or their synopsis individually for genre classification saved a lot on the time, however, they weren't very accurate. This may be due to the fact that titles may sometimes be named metaphorically, sarcastically or simply out of context, e.g. a title "How to be a murderer" may be thought of as a crime or a thriller movie, however it is a comedy movie, which can be clearly classified by supplementing synopsis along with a title. Hence we use both the synopsis and the title of the movie as features for movie genre classification.
After implementing the classical machine learning techniques, we observe that MNB performs the worst having an accuracy of 54.78% on the test dataset and the SVM model provides an accuracy of 59.57% on our test dataset, which is more than MNB. We speculate MNB has the worst accuracy due to the fact that it treats the probability of occurrence of each feature as independent which is not the case with text-based data, since some of the words are interlinked e.g. getting "bear" after the word "teddy" is much more probable than getting the word "gelato" therefore, this assumption that the features occur independently is not correct. On the other hand, SVM using the RBF kernel to classify non-linear data makes it capable to form interlinks between words and hence, the accuracy score increases. While using the hybrid model we are gathering classification data from both of the algorithms discussed above along with shallow decision trees, whose hyper-parameters have been tuned. This works out to be the best amongst the classical models and has an accuracy score of 61.23% on the test dataset. This is due to the fact that it incorporates the best features from all classical models. However, It is important to point out the fact that the time complexity of MNB is much lesser as it simply computes the probability whereas SVM applies an rbf transform and then computes the classes, and the hybrid model, need to undertake hundred's of decision tree calculations before providing a classification.

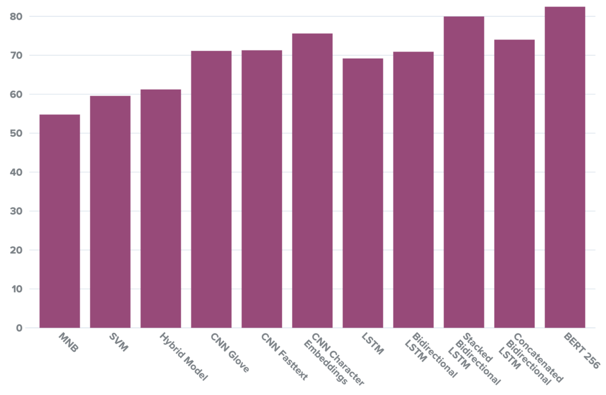

Fig. 6. Test Accuracy for all the implemented models

We can observe that using deep neural networks improves the test accuracy in comparison to the classical machine learning techniques as we are able to extract more complex information which helps in accurate classification. We could achieve the best test accuracy ranging near about 79% using the stacked bidirectional LSTM. However, in all deep learning models we could observe that the training and validation accuracy reached a minimum mark of 98% and 94% respectively. This scenario indicates that the models are overfitted on the training data as the test accuracy is comparatively low.
The primary reason for lower test accuracy is the limited training data which enforces us to train the model for more number of epochs to reach a certain value of training accuracy. We have tweaked are training data so that there are similar number of training samples for each target class. Thus, we could achieve similar test accuracy in predicting each target class individually. We had to limit the number of fully connected layers and the

input dimension considering the hardware resources at our disposal. Constructing a deeper neural network with more training samples and with wider training vectors will surely increase the test accuracy. Thus, we further relied on the simple transformer architecture for the best performance as it is pretrained on a very large dataset and includes state of art techniques such as attention layers which significantly improves the performance of the model.

Considering the performance of our prediction model is up to mark, we tried to evaluate our recommendation engine manually. We verified that the three recommended movies belong to the same genre and are quite similar to the predicted movie title. We plan to use some conventional metrics to evaluate the performance of our recommendation engine in near future.

Finally, with BERT, we achieved an accuracy of 82.47%. This was by far the best performance considering all the approaches we implemented. This was expected given the highly pretrained nature of BERT. The fact that BERT is trained on a huge vocabulary set lays the foundation for high performance. This can be associated with the deep understanding of English language and better context interpretation which is achieved in BERT.

Table 1 shows a performance comparison of the approaches we implemented for our problem against the hyper-parameter settings and preprocessing steps followed in [22] and [12] for classical machine learning techniques (Random Forest) and deep neural networks respectively. These existing approaches were tested on our problem under consideration and then compared with the performance achieved using the approach we followed.

Sr. No.	Models	Existing Approach	Our Approach
1	Classical Machine Learning Techniques	59.79 [22]	61.23
2	Deep Neural Networks	74.11 [12]	79.97
3	Transformer Models		82.47

Table 1. Comparison of test accuracy between existing approaches and our approach

5. CONCLUSION

In summary, we started with implementing the classical machine learning techniques to classify the movies into 10 defined target classes. We could achieve best test accuracy for a hybrid model which is custom designed for our problem statement. Later, we switched to implement deep neural networks with various architectures and fine-tuned hyper-parameters. The best test accuracy for deep neural networks is obtained for stacked bidirectional LSTM for certain set of hyper-parameters. For all the deep learning techniques, we used the character embeddings created using a shallow convolutional neural network. We could increase the model accuracy in comparison to the existing deep learning models using pre-trained word embeddings. Finally, we implemented BERT, the state of art transformer architecture which is pretrained on a large dataset. We deployed the pre-trained BERT model after pre-tuning it for our problem. We achieved the best test accuracy in comparison to all the previously implemented models.

Metrics need to be identified for evaluating the performance of the recommendation engine. We are currently in the process of determining the appropriate metric for this purpose. Secondly, in order to improve the detail to which the genres represent the movie, the problem formulation can be modified for a multi label classification problem wherein each movie sample has multiple associated genres. Approaches suitable for such problems can be then applied to achieve better results. Finally, this work can be extended by applying RoBERTa [29] developed by Facebook which has proved to be better in various other applications. It is trained on larger data with longer sequences and uses dynamic masking pattern as compared to static masking pattern on BERT which may lead to even better performance.

6. REFERENCES

[1] M. Grinberg, *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[2] A. Mccallum and K. Nigam, "A comparison of event models for naive bayes text classification," *Work Learn Text Categ*, vol. 752, 05 2001.

[3] E. Makita and A. Lenskiy, "A multinomial probabilistic model for movie genre predictions," *CoRR*, vol. abs/1603.07849, 2016.

[4] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, pp. 137–142, Springer, 1998.

[5] N. Fei and Y. Zhang, "Movie genre classification using tf-idf and svm," in *Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City*, ICIT 2019, (New York, NY, USA), p. 131–136, Association for Computing Machinery, 2019.

[6] Q. Hoang, "Predicting movie genres based on plot summaries," *arXiv preprint arXiv:1801.04813*, 2018.

[7] V. Battu, V. Batchu, R. R. R. Gangula, M. M. K. R. Dakannagari, and R. Mamidi, "Predicting the genre and rating of a movie based on its synopsis," (Hong Kong), Association for Computational Linguistics, 1–3 Dec. 2018.

[8] H. Zhou, T. Hermans, A. Karandikar, and J. Rehg, "Movie genre classification via scene categorization," pp. 747–750, 10 2010.

[9] M. Ivašić-Kos, M. Pobar, and I. Ipsic, "Automatic movie posters classification into genres," *Advances in Intelligent Systems and Computing*, vol. 311, pp. 319–328, 01 2015.

[10] M. d. Koning, "How to build a recurrent neural network to detect fake news," Feb 2020.

[11] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," 2017. cite arxiv:1702.01923Comment: 7 pages, 11 figures.

[12] A. M. Ertugrul and P. KARAGOZ, "Movie genre classification from plot summaries using bidirectional lstm," 02 2018.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[14] M. Allard, "What is a transformer?," Mar 2020.

[15] P. Suleiman Khan, "Bert, roberta, distilbert, xlnet-which one to use?," Oct 2019.

[16] R. Horev, "Bert explained: State of the art language model for nlp," Nov 2018.

[17] Cryptexcode, "Mpst: Movie plot synopses with tags," Apr 2019.

[18] Andybarthel, "Movie database," Dec 2015.

[19] V. Ojha, "How to build a recommendation engine using apache's prediction io machine learning server," Jul 2017.

[20] K. Gupta, "Predicting movie genres based on plot summaries," Jun 2019.

[21] E. Ma, "Besides word embedding, why you need to know character embedding?," Jan 2019.

[22] T. Pranckevicius and V. Marcinkevičius, "Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification," *Baltic Journal of Modern Computing*, vol. 5, 01 2017.

[23] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[24] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," 2016. cite arxiv:1612.03651Comment: Submitted to ICLR 2017.

[25] M. Antonio, "Word embedding, character embedding and contextual embedding in bidaf-an illustrated guide," Sep 2019.

[26] "Character embedding algorithm."

[27] A. Gokhale and R. Karwayun, "Fake news challenge: Stance prediction using deep neural networks," Aug 2020.

[28] V. Slovikovskaya, "Fake news detection empowered with bert and friends," Dec 2019.

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Ro{bert}a: A robustly optimized {bert} pretraining approach," 2020.