Project Report:

Team members:

- Jonas Brandenburg (if20b243), if20b243@technikum-wien.at
- Violeta Garcia Espin (if20b091), if20b091@technikum-wien.at
- Rawan Mousa (if20b129), if20b129@technikum-wien.at

Date: 09/11/2022.

Repository: https://github.com/FreshJonas/NLP-on-Movie-Plots

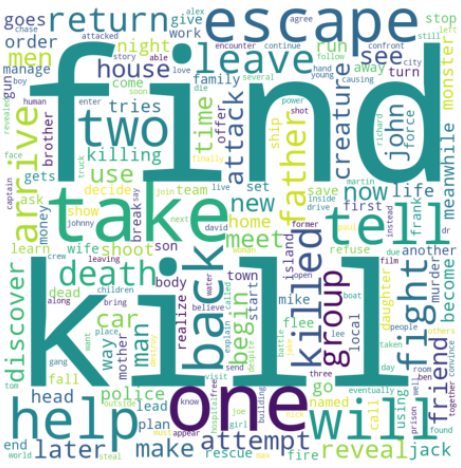
Summary:

This project aims to determine whether movie plots are from an american movie or an indian movie. This gets accomplished by using different Classification Methods like Logistic Regression, Naive Bayes Classifier as well as a Neural Network.

Task and/or data description:

The main goal of the project is to classify data that will get fetched from the Wikipedia API. The data consist of Indian Movie Titles and their plots as well as American Movie Titles and their plots. We received the data from the API by iterating over the Wikipedia Categories Indian_Movies_by_genre and American_Movies_by_genre. Only pages that had a plot section consisting of 200 to 6000 characters were included. We selected 2000 pages each of Indian and American movies.

The plots often included named entities like locations or names which we anticipated would make the classification very easy. This is why we chose to exclude these named entities in some of the classification models to verify whether this prediction was accurate.



Wordcloud for american films

Wordcloud for indian films

Methods:

Jupyter Notebooks - Code
Github Repository - Version Control

We used three Machine learning algorithms to classify the data.

Logistic Regression Classification:

After we read the dataset, we have started cleaning it by removing special characters and stop words, lowercasing the texts, and stemming the words thereafter we created count vectorize using Scikit-learn's CountVectorizer, which is used to convert a collection of text documents to a vector of token counts. Additionally, we split the data into a training and testing set which will allow us to evaluate the accuracy and see if the model is generalized well. This means whether the model is able to perform well on data it has not seen before. Moreover, as hyperparameters, we used a solver with a liblinear optimizer and calculated the accuracy with and without named entity removal.

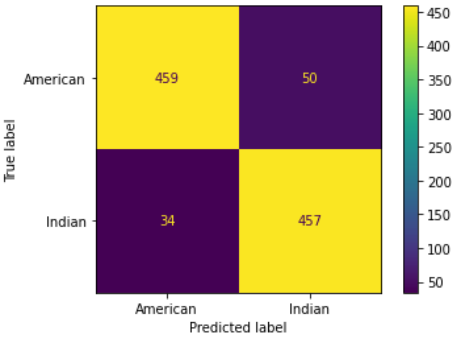
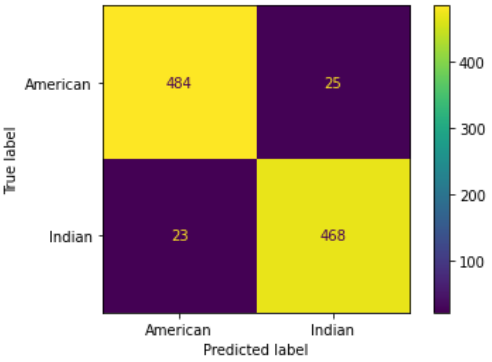
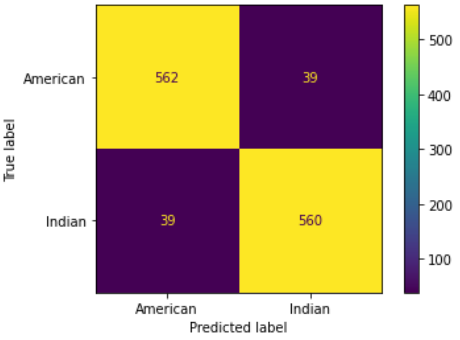
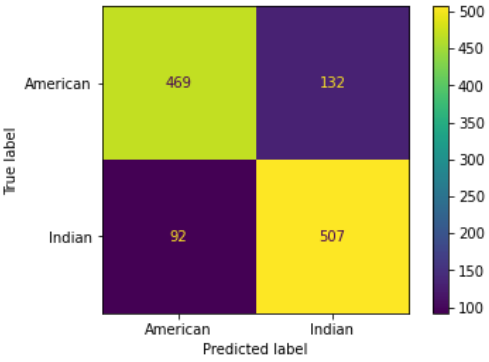
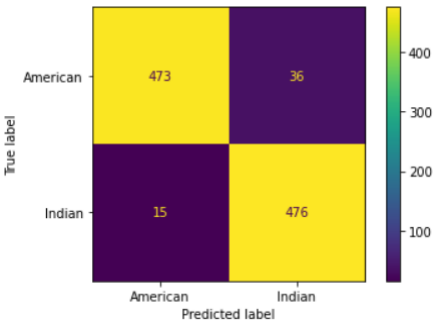
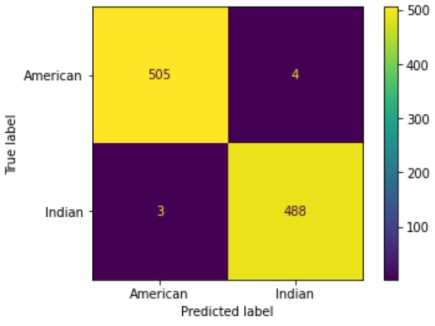
Naives Bayes Classification:
For the Naives Bayes Classification we have used the class MultinomialNB from the Scikit-learn library. The data preprocessing is identical to that described in the Logistic Regression section. After the data splitting the model is trained and predictions are made for the testing dataset. The performance of the model is evaluated with the method accuracy_score() from Scikit-learn library as well as through the representation of a confusion matrix. As in logistic regression, the model was evaluated with and without entity removal.

PyTorch Neural Network Classification:
The first step here was to create the data processing pipelines. These pipelines extract the plot (feature) as well as the target from the data tuples. They also perform all the necessary pre-processing on the plots. Here we used two different approaches. In one approach the plots were just lower cased and tokenized without further pre-processing. In the other approach we used the same cleaning function that was previously used in the logistic regression model and also removed stopwords named entities and special characters. This approach was significantly slower though (~ 10 times as slow) when running the model itself on the data. We then defined the classification model itself and defined a simple linear layer. We then ran the model with a batch size of 64 for 10 epochs.

Outcome and conclusions:

Implementing the Neural Network with the PyTorch library proved to be the biggest challenge in this project. Even though we used many online examples as inspiration it took quite long to get it all running and understand all involved components at least on a basic level. Once that along the implementation of the other two models was done we achieved results that were much better than previously anticipated.

Here are the results for the individual methods:

	Named Entities Removed	Named Entities Untouched
LogReg		
	Accuracy: 0.916	Accuracy: 0.952
NeuralNet		
	Accuracy: 0.935	Accuracy: 0.813
NaiveBayes		
	Accuracy: 0.958	Accuracy: 0.994

As it is visible in the graphics above the Naive Bayes Classifier performed better out of our chosen machine learning algorithms. This is not surprising since it is known to be very successful in classifying text documents. We think that the Neural Network could be

optimized to achieve an even better performance but we currently lack the required skills and knowledge to get to such results.

We did not expect that a classification between american and indian plots could be accomplished with such a high accuracy since we anticipated the used words to be very similar. This wasn't entirely the case though as can be seen in the wordclouds displayed in the data description section of this report. Here we can see some similarities like the word find which is commonly used in both plots as well as significant differences regarding the words kill and love (the former being overrepresented in the american plots while the ladder is overrepresented in the indian plots).

This also leads us to the question whether the movie selection could have been done more carefully. One approach for future improvements could be to only select movies from certain movie genres and then compare American and Indian Movies in this set.