ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Assignment 3

Text Analytics

Submitted by:

- Panagiotis Antoniozas
- Spyros Mastrodimitris Gounaropoulos
- Panagiota Tavoularea

Introduction

For the 3rd assignment we chose as before the “movie reviews” text, provided by the sklearn library. We give some helpful metrics for interpreting the data set in the tables that follow.

Number of Total Reviews	***Number of Positive Reviews***	***Number of Negative Reviews***
2000	1000	1000

Avg. Number of Words per Review	***Avg. Number of Words per Positive Review***	***Avg. Number of Words per Negative Review***
629	665	593

Avg. Number of Characters per Review	***Avg. Number of Characters per Positive Review***	***Avg. Number of Characters per Negative Review***
3893	3662	4124

Text Preprocessing

Same steps were taken as they had been previously, in further detail, we insert our data to the variables X,Y. Variable x is a list and contains all the reviews.Y is an array containing the corresponding annotations for each review (Y=0 : negative review , Y=1 : positive review).

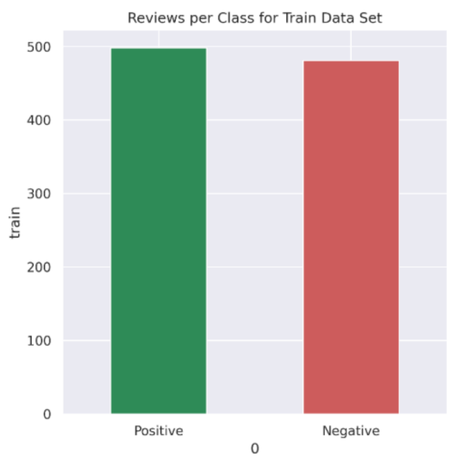
For the text preprocessing we removed all special and single characters. Also, removed all digits, converted the text to lowercase and performed lemmatization (process of reducing words to their root form).

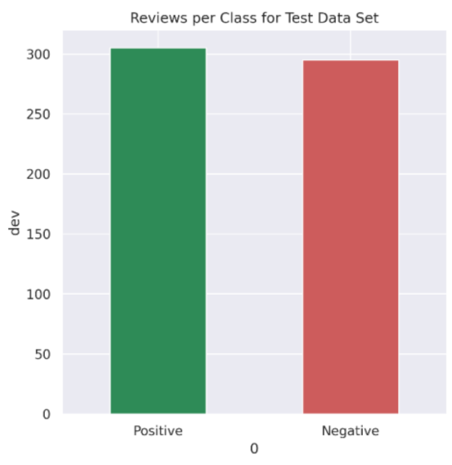
Below, we present an example of a sentence for a specific review, before and after the text preprocessing.

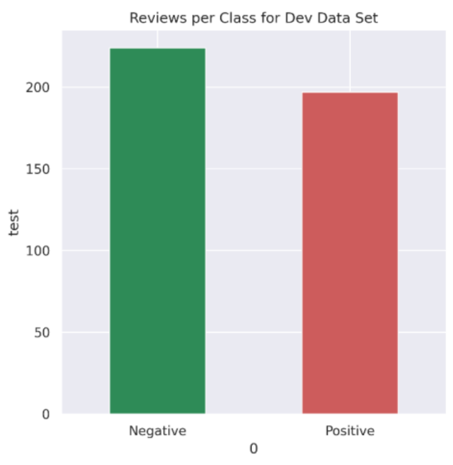
Before the Text Preprocessing	*arnold schwarzenegger has been an icon for action enthusiasts , since the late 80's , but lately his films have been very sloppy and the one-liners are getting worse .*
After the Text Preprocessing	*arnold schwarzenegger ha been an icon for action enthusiast since the late but lately his film have been very sloppy and the one liner are getting worse*

Splitting Data to Train, Test and Dev

Then, using the 70/30 rule, we divided our data set into training and testing. Then, we divide them into train and dev sets on the already-created train set using the 70/30 method. We display the review frequencies by class and by data set in the charts below.







TF-IDF Creation / Feature Selection

We applied the TfidfVectorizer() to all of the data sets in order to create the TF IDF features. Additionally, we excluded stopwords from our vocabulary and only preserved the top 5000 features, ranked by term frequency across the corpus.

Then, we used Singular Value Decomposition (SVD) for feature selection,to determine the class's most informative terms (positive or negative). The advantage of using SVD for feature selection is that it reduces the dimensionality of the dataset while preserving as much information as possible. Moreover, SVD is computationally efficient and numerically stable.

After using SVD, our final vocabulary is made out of the 1000 most informative elements that we maintained.

Baseline Model

We trained the algorithm on our train set in order to create the baseline model. Each data point is simply assigned to the class with the highest reviews by the algorithm. We utilised such a naive model, as a comparison measure to the others' models' performance.

Logistic Regression

We now again train the Logistic Regression algorithm, as he had the best results in the previous assignment. For the Logistic regression model we fit the train data set. All the necessary metrics that will be presented were measured, including confusion matrix, classification report and AUC scores per class and per data set.

MLP

In order to further reduce the features of our data we used SVD. Our observations showed that by reducing the number of features there was a decrease of overfitting risk. Hence, following multiple experiments, we decided to use 150 components. Further reduction resulted in worse results as the MLP model failed to interpret the complexity of the task.

Additionally, we fit the train data set for the MLP and perform hyperparameter tuning on the development set (rule : 70/30) for the following parameters:

Hidden Layers : The hidden layers are the layers of neurons between the input layer and the output layer. The optimal number of hidden layers for an MLP depends on the complexity of the problem being solved and the amount of available data. We trained our model for 1 to 3 layers with step 1. The tunner chose 3 hidden layers plus the input and output layers. The activation functions for the input was tanh and for the hidden layers relu twice and one sigmoid.

Units : The "units" in a hidden layer refer to the number of perceptrons or neurons in that layer.The number of units in a hidden layer is a hyperparameter that can be tuned to improve the performance of the MLP on a given task. Increasing the number of units in a hidden layer can allow the MLP to learn more complex patterns in the data, but may also lead to overfitting if the number of units is too large. Conversely, reducing the number of units can help prevent overfitting, but may also limit the capacity of the model to learn complex patterns. After, following several experiments we concluded with max_value = 512 and step = 32.

Epochs : The "epochs" refer to the number of times the entire training dataset is used to update the weights of the neural network. The MLP classifier was evaluated on different numbers of epochs, we set the maximum number at 100 and also we implemented early stopping optimization with patience set at 5.

Learning rate : The "learning rate" controls how much the weights of the network are adjusted during training. As a result of the hyperparameter tuning, the value of learning_rate: 0.0001 was selected.

Dropout : Dropout randomly sets a fraction of the input units to zero during each training iteration, which can help to prevent overfitting and improve the generalisation performance of the model. We tried out dropout rate 0.2 and 0.5 values.Finally, we used dropout for the first and third layer with probability of activation equal to 0.2 .

Batch size : The batch size is the number of input samples that are fed into the network at once. It can affect the speed and stability of the training process, which can indirectly impact the resulting model's performance. Batch size was evaluated on the train set for three different values 8,12 and 16 and according to the findings, it was determined that the best one was 12.

Evaluation of the Models

In the following tables we present precision, recall, F1, precision-recall AUC scores, for each class per classifier. We also present in the same tables, the same scores' macro averages per classifier. As we can see below, the MLP exhibits generally positive outcomes. For the train set, its macro average F1 is around 88%, while for the test and dev sets, it can reach 86%. The model is skilled at thoroughly memorising the training data and is able to successfully and accurately categorise unknown cases into both classes. Based on the metrics, as we can see below, the MLP compared to the best classifier of the previous work, where in our case is the logistic regression, are on average at the same levels.

Train Set

Classifier	***Precision***		***Recall***		***F1***		***Precision - Recall AUC***		***Macro-Averages***			
	Pos	***Neg***	***Pos***	***Neg***	***Pos***	***Neg***	***Pos***	***Neg***	***Precision***	***Recall***	***F1***	***Pr -AUC***
Baseline	0.51	0	1	0	0.67	0	0.75	0.74	0.25	0.50	0.34	0.75
Logistic Regression	0.97	0.99	0.99	0.97	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.99
MLP	0.91	0.85	0.85	0.91	0.88	0.88	0.95	0.95	0.88	0.88	0.88	0.95

Test Set

Classifier	***Precision***		***Recall***		***F1***		***Precision - Recall AUC***		***Macro-Averages***			
	Pos	***Neg***	***Pos***	***Neg***	***Pos***	***Neg***	***Pos***	***Neg***	***Precision***	***Recall***	***F1***	***Pr -AUC***
Baseline	0.51	0	1	0	0.67	0	0.75	0.74	0.25	0.50	**0.34**	**0.75**
Logistic Regression	0.83	0.87	0.88	0.81	0.85	0.84	0.93	0.92	0.85	0.84	**0.84**	**0.93**
MLP	0.87	0.84	0.84	0.87	0.86	0.86	0.93	0.91	0.86	0.86	**0.86**	**0.92**

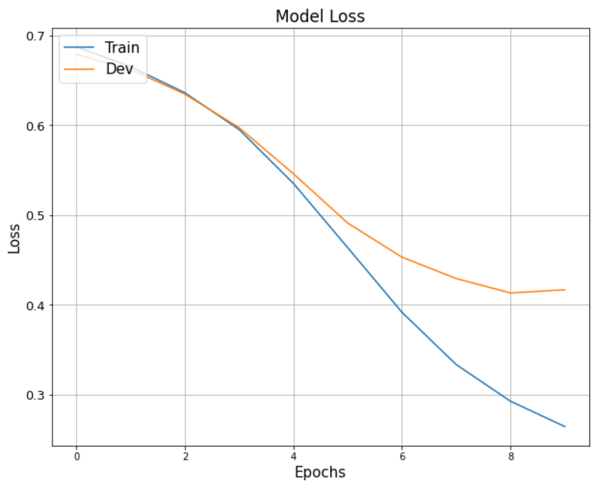
Development Set

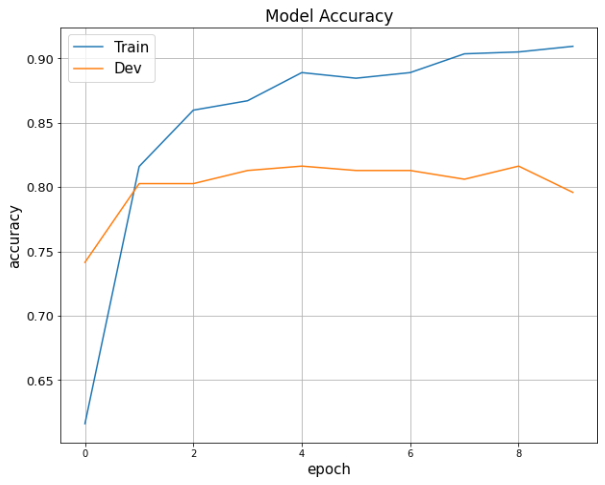
Classifier	***Precision***	***Recall***	***F1***	***Precision - Recall AUC***	***Macro-Averages***

	Pos	***Neg***	***Pos***	***Neg***	***Pos***	***Neg***	***Pos***	***Neg***	***Precision***	***Recall***	***F1***	***Pr -AUC***
Baseline	0.47	0	1	0	0.64	0	0.73	0.76	0.23	0.5	0.32	0.75
Logistic Regression	0.82	0.87	0.86	0.83	0.84	0.85	0.92	0.93	0.85	0.85	0.85	0.93
MLP	0.88	0.84	0.81	0.90	0.84	0.87	0.90	0.93	0.86	0.85	0.86	0.92

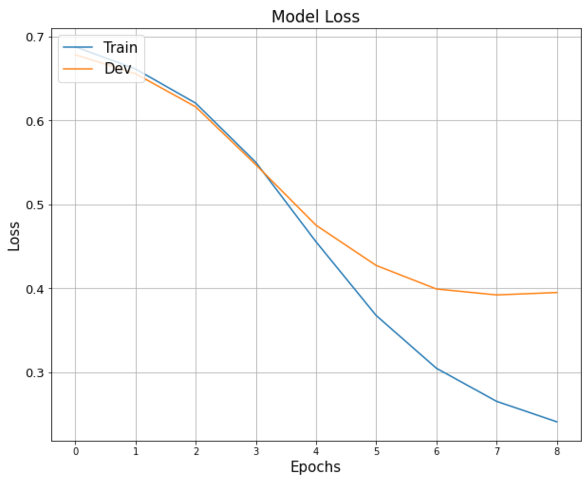
Evaluation of the Learning curves

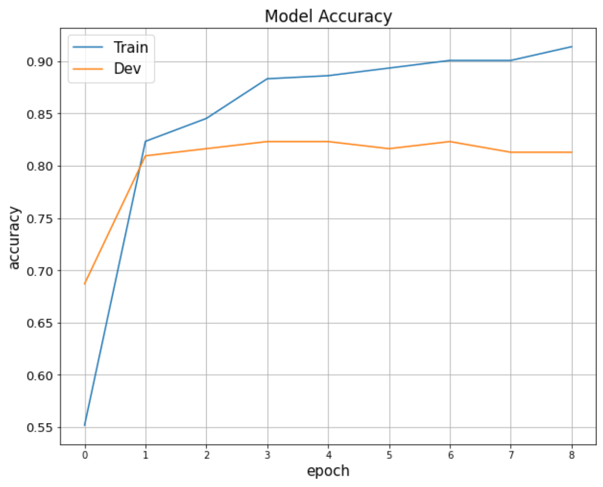
Batch Size = 12



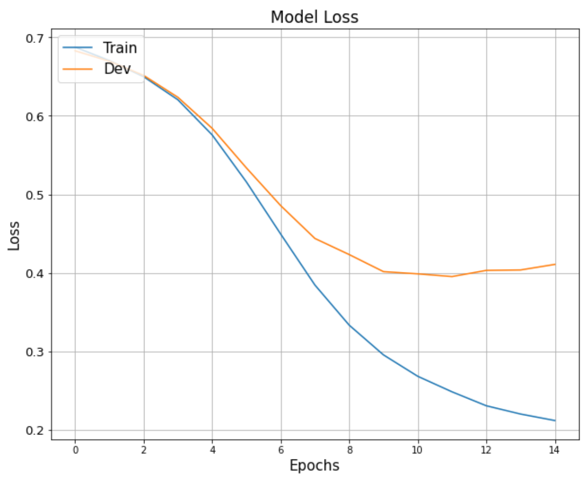


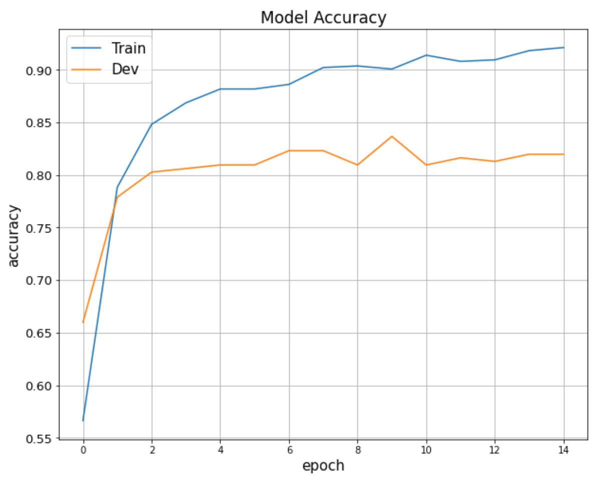
Batch Size = 8





Batch Size = 16





Model Accuracy

In a learning curve of our MLP, we want to see an improvement in model accuracy as the training progresses. This improvement in accuracy indicates that the model is learning and becoming better at making predictions. Typically, we would like to see the training accuracy and validation accuracy both increase with training, but at different rates. Initially, the training accuracy will increase rapidly as the model learns to fit the training data, but the validation accuracy may increase more slowly, this is because the model may be overfitting.

Observing the graphs up top, the model accuracy of the dev set for batch size 12, increases more swiftly than the other ones. On the side, max value of model accuracy is succeeded at the 5th epoch.

As a conclusion, taking into account the above graphs as well as the general performance of the model based on the parameters we set for it, the batch size 12 ended up being a slightly better choice by achieving a test accuracy of 86% to a test accuracy of 85% for batch sizes 8 and 16.

MLP vs Logistic Regression

MLP outperformed Logistic Regression slightly, as it achieved a macro-accuracy score of 0.86 on the test set compared to Logistic Regression's score of 0.84. While both models performed well, the MLP's higher accuracy score suggests that it was better able to classify the data. However, Logistic Regression was ultimately chosen over MLP because MLP was deemed to be very complex and required extensive tuning to achieve that extra 2% accuracy. In contrast, Logistic Regression was simpler and easier to interpret, while still providing good predictive performance. While accuracy is an important metric, other factors such as model complexity, interpretability, and computational efficiency also need to be considered when choosing a model. Therefore, despite MLP's better performance on the accuracy metric, Logistic Regression was chosen as the preferred model for our use case.