Artificial Intelligence Assignment 2 Documentation

Athanasios Lakes sdi1800090

November 2022

1 Introduction

Starting off, this assignment was completed after deliberate following of the professor's notes as much as laboratories' examples. However, for the better understanding of the concepts, I trusted third party sources covering similar notions and aspects that helped me get a deeper insight of Sentiment Analysis using Logistic Regression.

2 General Idea

The idea of the implementation is more or less straightforward compared to similar implementations on the internet:
.

- Data Import
- Text Processing
- Data Processing
- Vectorization Of The Text
- Training
- Predicting

3 Text Preprocessing

After reading a bunch of articles covering the concept of Sentiment Analysis on reviews I came to the conclusion that tidying the text is an essential part of the process. That is, removing **noise text**, **cleaning text from unwanted characters** such as *special characters*, **lemmatization**, and **removing of stop words**.

To be honest I tried excluding all this text processing from the model and found out it has fairly good prediction and training scores. However it is important to mention that this can be considerered random cause looking at the vocabulary generated by the vectorizer one can find words such as ***would have been*** or ***there was*** being included as features, providing no evidence whether or not the review is **positive** or **negative**. In the end I decided to keep these functions that **tidy** the text for novelty reasons and maybe some small efficiency improvements.

4 Data Preprocessing

Here I am estimating whether a review is a negative or positive based on the rating the user gave.

$$0 \leq rating \leq 4 \textbf{ negative review}$$

$7 \leq rating \leq 10$ **positive review**

If the review is negative I assign the value 0 to its respective rating value and 1 if it s positive

.

Meanwhile, I am splitting the data into two parts with 80 percent of the initial data being used to train the model and the rest 20 to validate it. I experimented with different splitting percentages like 0.3, 0.15 and 0.25 and found out they more or less produce the same results considering the fact that you keep do not change the rest of the hyperparameters immensely

.

5 Vectorization

For the Vectorization of the text I used TfidVectorizer.

- max features = 5000
- ngram range = (1,2)
- norm = l2

After trying different things with CountVectorizer (changing max features, ngram range etc.) I chose to go for the tfidfVectorizer as it produced noticeably better results. As a matter of fact, using the CountVectorizer I managed a prediction score of 0.86 at maximum where as with tfidf it went close 0.9 with no experimenting effort.

The greater the **max features** number the higher the train score which is obvious regarding the fact that it takes more and more words as features and consequently fits better to the current data. However, in countVectorizer the

value 5000 resulted in a highly overfit model where in tfidf that is not the case.

ngram value is set to (1,2) after experimentation. When I tried no value the score was closer 0.8 with similar results when the integral increased to (1,3). (1,2) however seems to increase the score making it close to 0.9.

norm value is set to 'l2' meaning *"The cosine similarity between two vectors is their dot product when l2 norm has been applied"(Scikit learn website)*. Using l1 produced lower train and test score.

6 Training

Training follows the same procedure with the code written in lab classes. I experimented with the number of mini batches that are trained in each loop but found that 10 iterations and a test size of ***(iterNumber * 0.1 + 0.01)*** works just fine.

*you can check the respective graphs in the folder