Report

Assignment 1: Sentiment classification

Name: Eva Richter

Fruit: 

Contributions of each member to the assignment (brief):

I worked on this assignment alone because I was in Germany during that week and it was easier for me to work on the assignment according to my own schedule due to my situation.

1. Pre-processing

Training data: 7456
Validation data: 2130

To design a sentiment classifier that, within the given method, achieves high accuracy, I added several preprocessing steps. First, I removed the stop words, since they are usually function words that contribute little meaning to the sentences and are therefore not needed for sentiment classification. The same applies to punctuation, which is why I removed any punctuation marks as well. However, I have excluded negative words such as negated verbs (e.g. haven't), as they are crucial to the meaning of the sentence and thus the sentiment classification.

In addition, I replaced the contractions, such as "don't, with the two-word version of the contraction (“do not”) to ensure that the model recognizes the words correctly. If we had been working with embeddings, this step would not be necessary, but since I am using a count vectorizer in which each word is represented as only one number, it makes sense to include this preprocessing step.

Finally, I have lemmatized the remaining words, i.e. reduced them to their base form, which carries the meaning of the word. Thus the vocabulary can be reduced.

2. Feature definition

For feature definition, I used CountVectorizer. The main motivation for using this tool, which can be imported from the Scikit-learn library, is that texts can be represented as a fixed dimension vector. In classical feature selection, features are selected manually, but for cases like sentiment analysis, it becomes increasingly difficult to improve the results for general cases, considering

rule-based features like the number of positive words, number of negative words etc. This is a tedious task which requires domain expertise and more human resources. Using a vectorizer is a simple statistical approach where the result could be involved by simply tracking the distribution of words in text all over the corpus.

The CountVectorizer converts a given set of words into a frequency representation. The CountVectorizer works by the following four steps:

Step 1: The vectorizer takes all the input texts (sentences) and tokenizes it to get tokens. If the unique number of tokens in the corpus is k, then k would be the dimension of the vector.

Step 2: The vectorizer creates a matrix with texts or documents as rows and all unique word as columns.

Step 3: Next, the vectorizer counts the number of times each token is present in each text and fills up the matrix.

Step 4: Finally, the rows are the vectors for each text while the columns are the vectors for each token:

text1 = “This food tastes delicious”
tag1 = "Positive”

text2 = “This candy tastes not so good”
tag2 = "Negative"

	This	food	candy	tastes	delicious	not	so	good
Text 1	1	1	0	1	1	0	0	0
Text 2	1	0	1	1	0	1	1	1

Table 1: Matrix created by CountVectorizer

Nevertheless, using this CountVectorizer is not the best method available, as nowadays there are high-level featurizers that would also transfer the semantic similarity between words and therefore give better results.

For this assignment I took all the training data and passed it to the CountVectorizer. Then I got the representation of the training, validation and test dataset. The total number of **unique tokens** was **14241** and the **matrix for each text** was therefore **1x14241**.

3. Classifier

For the given sentiment classification task, we used a logistic regression based classifier, since the dependent variable (target) is categorical, i.e. we want to determine whether a word contains positive or negative sentiment. Since I used a CountVectorizer, each word is represented as a feature in a vector with a coefficient that can be either positive or negative:

$Ax + By + c < 0$, negative sentiment
$Ax + By + c > 0$, positive sentiment, where

A and B are the corresponding coefficient,
x and y are the features,
c is the intercept.

This means that the higher above zero the coefficient for a word, the more positive the sentiment that word contributes to the meaning of the sentence, and conversely, the lower below zero the coefficient for word, the greater the negative weight that word contributes to the sentence. The sum of these coefficients for the individual words is used to calculate whether a sentence is classified as positive or negative overall.

I implemented this by instantiating the logistic regression model from the scikit-learn library. Then, the vectorized input (using the CountVectorizer) was passed as input to the model and the two binary classification labels: pos and neg were passed as truth values for the model.

4. Evaluation; model analysis; error analysis

Model Analysis

Overall the **accuracy** on the **training data set** was **97%**, while on the **validation dataset** it was **75%** (precision: 75.59, recall: 75.02).

Similarly, the **intercept value** for the model is: **-0.1954846**, which is a constant generalized from training data and does not make much difference while predicting class for newer dataset.

Words	Coefficients
beautiful	1.7005037306994295
engrossing	1.6966346261173135
powerful	1.6700237731066925
enjoyable	1.6276080614618091
affection	1.5221764233589319
triumph	1.5090209303059836
refreshing	1.5014328349948887
entertaining	1.496747463668491
brilliant	1.4911424320171367
warmth	1.4576343605815913

Table 2: Words with the top 10 highest coefficient values

Words	Coefficients
dull	-1.8710669615557152
boring	-1.8575730194621458
fails	-1.83674815925
neither	-1.6838157997822594
bore	-1.6369085036768933
generic	-1.5809096308005972
flat	-1.5703334977202397
badly	-1.5513886924352676
mediocre	-1.4829497067640962
incoherent	-1.4810134111657305

Table 3: Words with the top 10 highest coefficient values

As we can observe, all the words (features) with higher coefficients are words with positive meaning. This way, the model pushes the text with positive words to have higher overall values and classifies it as positive.

At the same time, the top lowest coefficient words are negative words. The model pushes the text with negative words to have lower overall value and classifies it as negative.

Consequently, text with a higher number of positive words will make the text positive and text with a higher number of negative words, negative. Hence, I implemented the logistic regression with a count vectorizer instead of listing positive words manually and extracting features in a rule-based manner. Table2 and table3 conclude that the model could learn about the positive and negative words by observing the training data.

Error Analysis

Test accuracy : 76.3%
Precision on final test set: 77.13%
Recall on val-set: 74.8%

False Negative

1. **“**you can practically hear George Orwell turning over .”

Words	Coefficients
practically	-0.28732939385514245
hear	0.48738833156059463
george	0.4208888304729238
orwell	0.04523603747290871
turning	-0.20632194082873734

For the given sentence, the word “Turning” has a low coefficient which should make the sentence negative but there are other words that carry higher positive coefficients and as a result the sentence is positive.

2. “a well-acted , but one-note film .”

Words	Coefficients
well-acted	not found
but	-0.1269759220462641
one-note	0.4208888304729238
film	0.33481996090855654
turning	-0.20632194082873734

The model misinterprets the word “film” as a highly positive word. Similarly, few other words that carry the higher weight for a sentence to be negative (in ideal situation) are not found in the vectorizer and hence ignored by the model. This depicts the lack of semantic relation between words and failure of the model to clearly distinguish the positive and negative words in some cases, resulting in false prediction.

False positive

1. “not a strike against yang's similarly themed yi yi , but i found what time ? to be more engaging on an emotional level , funnier , and on the whole less detached .”

Words	Coefficients
not	-0.4149196568490948
strike	0.0280963898967651
yang	0.15558649693646565
similarly	0.2160620426066212
themed	0.21992365541045952
yi	not found

yi	not found
but	-0.1269759220462641
found	-0.43859773983113276
time	-0.026752377696383084
engaging	0.6895443164113253
emotional	0.31333447318864266
level	0.019412020635233215
funnier	-0.7037650356200188
whole	-0.34488760893253234
le	-0.40534594954228675
detached	-0.6657296223243041

In the example above, negative words carry much more weight in the model. Words like “funnier” which could occur in both contexts are creating problems. Here the word funnier occurs in a positive context, but the model has learned that the word occurs only in the negative context. This proves that analyzing a model based on individual words is not the best idea. Instead, we should try to make the model see the context to obtain better results. This problem could be solved easily by a higher level of contextualized embeddings.

2. “Although I didn't hate this one , it's not very good either . it can be safely recommended as a video/dvd babysitter .”

Words	Coefficients
although	0.6695068355926632
not	-0.4149196568490948
hate	-0.2403674333551896
one	0.015910212411649676
not	-0.4149196568490948

good	0.6186080972502642
either	0.2874785163739617
safely	0.018606177971040732
recommended	0.27774593159593064
video/dvd	not found
babysitter	not found

This sentence is quite tricky to differentiate, even for a human. It sounds more condescending. In such a case, our simple logistic model does not predict it as a positive sentence. Here, the model got an equal proportion of positive and negative words.

Finally, to increase the accuracy score for the given test dataset, using a more semantic representation of words would be a great idea. Furthermore, if we have a high dimension representation of words, a higher level of machine learning model may perform well. Similarly, making models learn the context is important. Simply looking at positive or negative words may not be enough, as sentences can be complicated and the semantic meaning of a sentence cannot be generalized simply by looking at words but rather by the relation between these words.