Natural Language Analytics

2nd assignment

Spyratos Angelos

Dsc17021@uop.gr

A Question

We prepared the dataset by firstly removing the special character “ from all files using regexes to avoid confusion while reading the csv with pandas (multiple tweets merged in one tweet). We also removed some timestamps from 8 tuples in the test set that produced an error on load.

Then we tried multiple methods and we ended up with Naïve Bayes being my final option as a classifier. We first did a preprocessing step to the tweets by removing unavailable tweets and changed many things in the tweets like changing emoticons to words like ‘__EMOT_SMILEY’, hashtags to ‘__HASH_ ‘, etc. (source: https://github.com/ayushoriginal/Sentiment-Analysis-Twitter). We also used the code from the same repository to create a feature vector*(values show below) for my 2nd method. As a first method we choose to run a pipeline to create vectors for ngramms 1,2,3(we removed extremely rare words that occurred in less than 0.2% of the tweets) , calculate the TfIdf measures and take into account the length of the tweet as an extra feature and passed it as input to a MultinomialNB classifier. For our second method we simply used the feature vector described below and passed it as input in a MultinomialNB classifier. In both cases the chosen value for the smoothing parameter was 1.

Results:

(Except from accuracy we also calculated the weighted-averaged precision, recall and f1-scores for all classifiers. The reason we used the weighted average was because of the imbalance of the classes in the dataset. Also you shall find 3 different results for each method as we found a bit of high differences when using: **a) only the train and test data**, **b) train+dev_train data and test+dev_test data** (merged together), **c) train data and test+dev_test data**) (we did not need to perform a grid search so the dev train-test sets could only be used as extra data.

1st method:

	Accuracy	Precision	Recall	F1 score
a)	60.35%	58.18%	60.35%	56.69%
b)	60.14%	58.62%	60.14%	56.64%
c)	59.54%	57.49%	59.54%	55.88%

2nd method:

	Accuracy	Precision	Recall	F1 score
a)	58.21%	61.45%	58.21%	51.03%
b)	40.91%	46.29%	40.91%	34.78%
c)	57.03%	61.01%	57.03%	50.90%

3rd method (Text Blob):**

	Accuracy	Precision	Recall	F1 score
a)	55.93%	55.20%	55.93%	52.41%
b)	54.99%	54.81%	54.99%	51.44%
c)	54.99%	54.81%	54.99%	51.44%

4th method (Pattern):**

	Accuracy	Precision	Recall	F1 score
a)	55.86%	55.03%	55.86%	52.31%
b)	54.92%	54.61%	54.92%	51.34%
c)	54.92%	54.61%	54.92%	51.34%

B Question

We can observe that both of our methods seem to outperform the pretrained algorithms in both terms of Accuracy and Precision-Recall (except when the dev-train set is also used to train our algorithms). We can also observe the the second method seems to have a lower accuracy scope but it has a higher precision score than the first method, a higher precision is desired in such cases as sentiment analysis so we could say that the second method seems as a bit better final choice than the first method. The reason we get such “low” scores in general is because of the nature of the dataset that contains tweets from completely irrelevant topics. This is the reason why both pretrained algorithms produce a lower end result as they where mostly targeted for sentiment analysis in domain specific tweets like “products” or “voting preferences”.

** Both tools produced a polarity measure between -1 and 1. We decided to do the following mapping of polarities to our labels : [-1,-0.3) : negative, [-0.3,0.3] : neutral, (0.3,1] : positive. We actually cheated no ourselves as the documentation for both tools indicated that a good measure could be to think of polarities greater than 0.1 as positive while less than that as negative.

Last but not least, we chose not to do a stopword removal or other tasks like stemming or lemmatizing as we thought that both stopwords like “no”, ”not”, etc. as well as words ending with n’t would provide good info in terms of sentiment’s as they indicate negation.

*Feature vector for second method consists of:

```
'hasAddict': False,
'hasAwesome': False,
'hasBad': False,
'hasBroken': False,
'hasBug': False,
'hasCant': False,
'hasCool': False,
'hasCrash': False,
'hasDifficult': False,
'hasDisaster': False,
'hasDont': False,
'hasDown': False,
'hasEasy': False,
'hasExcite': False,
'hasExclaim': False,
'hasExpense': False,
'hasFail': False,
'hasFast': False,
'hasFix': False,
'hasFree': False,
'hasFrowny': False,
'hasFuck': False,
'hasGood': False,
'hasHappy': False,
'hasHate': False,
'hasHeart': False,
'hasIncredible': False,
'hasInterest': False,
'hasIssue': False,
'hasLike': False,
'hasLol': False,
'hasLose': False,
'hasLove': False,
'hasNeat': False,
'hasNever': False,
'hasNice': False,
'hasPerfect': False,
'hasPlease': False,
'hasPoor': False,
'hasSerious': False,
'hasShit': False,
'hasSlow': False,
'hasSmiley': False,
'hasSuck': False,
'hasTerrible': False,
'hasThanks': False,
'hasTrouble': False,
'hasUnhappy': False,
'hasWin': False,
'hasWinky': False,
'hasWow': False
```