Sentiment Analysis System

David Barbas Rebollo

Applications of Computational Linguistics
April, 2021

1 Introduction

The objective of this assignment is to create a sentiment analysis system.

The TASS2017_T1 dataset has been used to perform this task. These dataset is contains tweets in Spanish to be analyzed by the developed system. It contains 1008 training tweets, 506 validation tweets and 1899 test tweets. Each individual tweet has an identifier along with a sentiment tag referring to positive (P), negative (N), neutral (NEU) or none (NONE), and obviously the tweet itself.

Machine learning methods are then used to optimize the classification of the tweets.

2 Data processing

Firstly, the dataset is parsed from 3 XML files, respective to the subsets of data using the *ElementTree* library. Obtaining their id, content and sentiment, with the exception that the test set does not have assigned labels.

Secondly, all the tweets are normalized by removing stopwords, punctuation and transforming user names, hastags and url's into special tokens. They are "#user", "#hastag" and "#url" respectively. Lastly, the tweets are tokenized by *NLTK*'s, a NLP library, *TweetTokenizer*.

Finally, to be able to train machine learning models these texts must be vectorized. In this work, the *TfidfVectorizer* from the *Scikit-Learn* library has been used. This way it is possible to convert text into a TF-IDF matrix and check the relevance of each n-gram in the set of texts to obtain the most relevant features.

3 Experiments and results

The experiments performed involved the search of the best possible parameters for different machine learning classifiers. This can be easily done creating a *Pipeline* and performing a *GridSearchCV*. This way, *f1-score* could be maximized via a 5-fold stratified cross validation.

For the TF-IDF vectorization n-grams where used and the following parameters where considered:

- N-gram range: [(1,2),(1,3),(2,3),(3,4),(3,5),(3,6),(4,5)]
- Max document frecuency: [0.3,0.4,0.5,0.6,0.7,0.8,0.9]
- Min document frecuency: [1,2,3,5]

The classifiers used are all from the *Scikit-learn* library. The following classifying where considered along with their explored parameters:

- Logistic Regression
  - Inverse of regularization strength: [1,10,100,1000,10000]
- Support Vector Classifier
  - Regularization Parameter: [1,10,100,1000,10000]
  - Kernel: ['linear', 'rbf']
- Random Forest Classifier
  - Number of trees: [50,100,150,200,300]
- K Neighbors Classifier
  - Number of neighbors: [3,5,7,11,15,21,25]

The results obtained, with the best calculated parameters during training, are shown in the table below:

Table 1: Results of experimentation for the TASS2017_T1 task.

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.50	0.40	0.38	0.38
Support Vector Classifier	0.53	0.38	0.39	0.37
Random Forest Classifier	0.53	0.38	0.39	0.37
K Neighbors Classifier	0.48	0.41	0.39	0.40

Given the results above, the best model seems to be K Neighbors Classifier. However, considering *f1-score* it is possible to obtain a better classifier with Support Vector Classifier.

3.1 Fine tuning Support Vector Classifier

The following table shows the parameters modified with the evaluation metrics:

Table 2: Results of experimentation for the TASS2017_T1 task fine tuning Support Vector Classifier.

Model	N-gram range	Min DF	Regularization	Kernel	Accuracy	Precision	Recall	F1-Score
Orginal	(4,5)	2	10	Linear	0.50	0.40	0.38	0.38
Fine-tuning 1	(4,5)	2	10	RBF	0.55	0.48	0.38	0.36
Fine-tuning 2	(3,5)	1	100	Linear	0.53	0.46	0.42	0.43

The final experimentation shows, the linear kernel is able to yield better results than the rbf kernel. Also, expanding the n-gram range, decreasing the minimum document frequency and increasing the regularization enables to obtain the best classifier.