|  |
| --- |
| **Twitter Sentiment Analysis** |
| Outline:  Introduction  Data Description  Baseline Experiments  Enhancement Experiments  Overall Conclusion  Task Distribution |
|  |

***Introduction:***

We have a set of tweets annotated as **Positive**, **Negative** and **Neutral.** What we need to accomplish is to train a classifier using this data in order to be able to classify any new tweet introduced correctly.

We are allowed to use any type of supervised classifiers in addition to feature extraction, parameter grid tuning, Bagging technique, Boosting technique…etc. aiming to improve our model results in terms of accuracy and F-score metrics.

***Data Description:***

Our training data is introduced in three text filed broken down as beow:

|  |  |
| --- | --- |
| File Name | Number of entries |
| twitter-2013train.txt | 9685 |
| twitter-2015train.txt | 489 |
| twitter-2016train.txt | 6000 |

Each file is divided into three columns, first column represents a unique id for each tweet, second column is the annotation or the label of the tweet and third column is the tweet context.

Our testing data is introduced in one CSV file:

|  |  |
| --- | --- |
| File Name | Number of entries |
| test.csv | 3097 |

We are provided the unique id and the context of each tweet

Tweets are all in English language and all the punctuations like comma and quotations are represented by their ASCII representation.

Also Emojis and emoticons are represented by their known equivalent characters [For example: ☺ 🡪 :) ]

***Baseline experiments:***

Here we are required to train a standalone classifier and reach the best possible accuracy out of the following basic steps:

**Cleaning Data 🡪Preprocessing 🡪 Vectorization 🡪 Classification**

**Cleaning Data:** In this part in order to increase the accuracy of our classifier we cleaned the data from usernames, URLs, numbers, special chars and hashtags.

**Preprocessing:** We applied the below techniques in order

* Case-folding
* English Stop words Removal
* Stemming using PorterStemmer

**Vectorization:** Both CountVectorizer and TfidfVectorizer vectorizers were applied and results were compared

**Classification:** The following list of classifiers were experimented

[naive\_bayes – LogisticRegression – Decision Tree – RandomForest -SVM]

\*\*All experiments were performed using 10 fold Cross-Validation

*Results in terms of Accuracy & F-score:*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Classifier | Average Accuracy | Average F-score |
| Count Vectorizer | naive\_bayes | 0.624 | 0.548 |
| **LogisticRegression** | **0.662** | **0.614** |
| Decision Tree | 0.609 | 0.556 |
| RandomForest(est=20) | 0.648 | 0.581 |
| RandomForest(est=60) | 0.662 | 0.586 |
| SVM(c=1) | 0.628 | 0.596 |
| Tfidf Vectorizer | naive\_bayes | 0.606 | 0.438 |
| LogisticRegression | 0.653 | 0.563 |
| Decision Tree | 0.595 | 0.549 |
| RandomForest | 0.651 | 0.567 |
| SVM(c=1) | 0.667 | 0.609 |
| Word2Vec | LogisticRegression | 0.600 | 0.541 |

From the above table, we can deduct that using Count Vectorizer with Logistic Regression classifier yielded the best results in terms of both accuracy and F-score metrics

Therefore, we are going to consider this as our baseline results which we seeking to improve in the coming steps.

Also note that vectorization using word embedding, Word2Vec, was applied but the results were not very promising

***Enhancement experiments:***

1. Applying Grid search

**Goal**: Improve the best results yielded in the baseline experiment

To do so, parameter tuning is applied to both using Count Vectorizer and Logistic Regression classifier aiming to extract the best features from both and thereby improving our model

**Steps**:

* Pipeline is clreated to combine both Count Vectorizer and Logistic Regression classifier
* Parameter combinations are set as below
  + - * Vectorizer max\_df: 0.25, 0.5, 0.75
      * Vectorizer ngram\_range: (1, 1), (1, 2),(1,3)
      * Vectorizer min\_df : 1,2,3
      * Classifier C: np.logspace(-2, 0, 3)
* Best parameters are extracted as below and fed to the baseline vectorizer and classifier in order to improve their metrics
* Classifier C: 1.0
* Vectorizer max\_df : 0.5
* Vectorizer min\_df : 1
* Vectorizer ngram\_range : (1, 2)

**Results**: Average Accuracy: 0.675 / Average F-score : 0.616

**Conclusion**: Both accuracy and F-score increased.

Grid search is an effective and time saving technique in parameter tuning

1. Bagging

**Goal**: Improve the best results yielded in the Grid search experiment

**Steps**:

* BaggingClassifier is imported
* Best parameter form step 1 are fed to both classifier and vectorizer
* LogisticRegression repressor is fed to the input of BaggingClassifier as a base estimator and number of estimators used is 50&400

**Results**: 50 Estimators 🡪 Avg Accuracy: 0.672/ Avg F1: 0.622

400 Estimators 🡪 Avg Accuracy: 0.676 / Avg F1: 0.618

**Conclusion**: As we can see, there is a slight increase when using 400 estimators.

However, the time consumed and such slight increase did not encourage us to continue using Bagging technique

1. Boosting:

**Goal**: Improve the best results yielded in the Grid search experiment

**Steps**:

* AdaBoostClassifieris imported
* Best parameter form step 1 are fed to both classifier and vectorizer
* LogisticRegression repressor and Decision Tree Classifier are fed to the input of AdaBoostClassifier as base estimators and number of estimators used is 50

**Results**: LogisticRegression 🡪 Avg Accu: 0.578 / Avg F1: 0.620

Decision Tree 🡪 Avg Accu: 0.589 / Avg F1: 0.619

**Conclusion**: As we can see, there is a significant decrease in terms of accuracy.

Therefore, Boosting technique is not going to be used

1. Feature Extraction:

**Goal**: Improve the best results yielded in the Grid search experiment

**Steps**:

For each tweet:

* Count question marks
* Count exclamation marks
* Count upper case letters
* Count smile faces
* Count hash tags
* Count mentions

Feed the result vectors to the main vector of each tweet

**Results**: Average Accuracy: 0.679 / Average F-score : 0.620

**Conclusion**: Both accuracy and F-score increased.

Feature extraction is useful in our case

***Overall conclusion:***

* It is extremely important to learn about your data set and give yourself enough time to explore it before getting involved with analysis and classifications
* There is no single classifier that always perform well with a certain kind of data. It totally depends on the shape of your data set
* It is important to tune your parameters and extract the best out of them
* Data cleaning and preprocessing have a direct effect on your results