1 COL772 Assignment 2 - Overview

Rachit Jain
2018ME10032

This assignment involved the task of text classification, more specifically Sentiment Mining, using classical Machine Learning Algorithms. As expected in all assignments, each student was asked to develop a solution from scratch. The problem for this assignment dealt with sentiment categorization system for tweets.

1.1 Input

A labelled set of 1.6 million tweets were provided with their labels for sentiment corresponding to each tweet and the tweet itself, in string format. Both the categories of labels, POSITIVE and NEGATIVE, were equally balanced, with 50% of the data for each category. The input was a dumped set of tweets which included the user IDs, URLs, Emoticons, etc. and needed significant amount of pre-processing before passing it through the pipeline. This has been discussed in detail later.

1.2 Output

The output expected a sentiment label corresponding to each of the tweet, with 0 representing NEGATIVE sentiment, while 4 for POSITIVE sentiment. This prompted that the problem is a binary class classification problem and the use of cross-entropy or binary-loss measures were the things that clicked immediately!

1.3 Evaluation Scheme

Before going ahead with creating a solution to the problem, one should sit back and think about a scheme that would allow him/her to review the impact of each addition in the code much smoothly later. Since this assignment was supposed to be evaluated on the F1 Score, having knowledge about the accuracy and F1 score for every experiment that we run in future would enable us to choose the parameters right.

1.4 Pre-Processing

This block was the heart of the entire pipeline. The tweets were exceptionally noisy and required significant amount of pre-processing for them to be good for the model to learn as features from. Some of the input tweets (with labels) and their corresponding category under which I segregated them are shown below:

Sentiment	Original Tweet	Processed Tweet	Category
"4"	"@YouLuvMe sure..... bighead "	"sure. bighead"	UserID & Repetition
"0"	"nobody put me on fololow friday this week.. im tight! lol"	"nobody put me on follow friday this week. I am tight laugh"	Repetition & Inbuilt Clictics
"0"	"Work laptop is officially dead .. Not happy at all.. "	"work laptop is officially dead not_happy at all"	Negation Included
"0"	"happy for Coach Stringer (HOF c/o 2009!)...now if I can only finish my term paper on her "	"happy for coach stringer hof co 2009 now if I can only finish my term paper on her"	Punctuations & Sarcasm

1.4.1 Techniques

1. Emphasize Capitals: If the entire word is capitalized, there is more emphasis that should be given. Ex: "I am extremely HAPPY v/s happy would indicate a much stronger sentiment. Thus, highlight these words on purpose.
2. Clean the Clutter: Remove UserIDs, mentions, URLs, websites, hashtags, quotes, etc. since they are not useful for the task for sentiment classification.
3. Translating Emoticons into words: To increase the effect of sentiment in the tweet, the emoticons like ':)' v/s 'xD' v/s ':)))' should be translated to different tokens to highlight their significance.
4. Remove Punctuation: To clear out unnecessary items from the tweet string
5. Handle Clitics: Expand the clitics into their general form, like "I'm" like "I am", "I'hv" to 'I have', "should've" to "should have", etc.
6. Handle ShortForms: Expand the shortforms to indicate the true nature of their sentiment, like 'ily' to 'i love you', 'gm' to 'good morning', etc.
7. Negation: Add negation to 2 words following negative word like 'not' and that converts the sentiments to opposite nature.
8. and many more small adjustments...

1.4.2 Tokenize

It is better to start with removing the extra spaces and any other pre-processing that is needed. Following this, we can convert each sentence into tokens which shall be directly going into the pipeline.

1.5 Pipeline

The ML pipeline follows next. Since this is a assignment that allows the use of classical ML algorithms only, and thus, a simple Naive Bayes and Logistic Regression implementation can do the work.

1.5.1 Experiments

Following are the details about the model that was used:

1. Logistic Regression vs Naive Bayes: The first came out to be better for the sentiment analysis problem. SVM and NB were poor in performance.
2. Count Vectorizer vs TFIDF Vectorizer: The former one gave better performance than the TF-IDF one upon experimentation. Maybe a few different hyper-parameter choice would have given a different result.
3. Unigram, Bigram or more?: Bigram add more emphasis and allowed a combination of words to be used. Trigrams reduced the performance.
4. Solver: The 'saga' solver gave better results.
5. Stemming vs Lemmatizer: None of the two gave any positive performance improvement. Stemming made the words different and the sentiment was hard to find out in that case.

1.5.2 Hyper-Parameter Tuning

One can try out different parameters for the above experiments and after a lot of hit and trail combinations, the final parameters were chosen accordingly.

1.6 Further Improvement

Additional gains in performance were made by adding a feature of lexicons taken from the data itself and then copying these special words like 'happy', 'love', 'laugh', 'bad', 'poor', 'sad', etc. multiple times in the sentence since the CountVectorizer adds weightage to the sentiment tokens.

Rachit Jain
2018ME10032