Machine Learning Engineer Nanodegree Capstone Proposal

Emmanuel Dickson

July 10th, 2020

Tweet Sentiment Extraction

Domain Background

Sentiment analysis can be defined as the process of computationally interpreting and categorizing the feelings, emotions and even the intentions behind a text. This is customarily achieved through the use of Natural Language Processing (NLP) [1] and text analysis tools. With the increasing importance a good and positive online persona has on a person or company's brand, it has become more imperative for people to be tactful with their choice of words while conversing and airing their views on an online platform like Twitter. This project was a featured code competition on Kaggle.com[2].

Problem Statement

For this project, we are required to go beyond analyzing the sentiment of the text data obtained from tweets and provide accurate insight on which portion of the tweets influenced its sentiment classification. The problem as described on the Kaggle competition page is as follows:

"Your objective is to construct a model that can look at the labeled sentiment for a given tweet and figure out what word or phrase best supports it."

Datasets and Inputs

The dataset for this project was obtained from the Kaggle competition page. It comprises of three .csv files namely;

train.csv - for training our model. The training set comprises of 27,481 unique values with 40% labeled as neutral, 31% labeled as positive and the remaining 28% labeled negative.

test.csv - for testing our model. The testing set comprises of 3,534 unique values with 40% labeled as neutral, 31% labeled as positive and the remaining 28% labeled negative.

sample_submission.csv - a sample submission file to be used for prediction. This set comprises of 3,534 unique textIDs and for each ID, we are to predict the string that best supports the sentiment for the tweet in question.

There are four columns in the dataset:

- *textID* – unique ID for each piec34e of text
- *text* – the text of the tweet
- *sentiment* – the general sentiment of the tweet
- *selected_text* – the text that supports the tweet's sentiment. (This column is only present in the train.csv file).

The train dataset will be splitted into training (80%) and evaluation (20%) sets for training and tuning the model. Predictions will be made by passing the sample_submission.csv file to the model. The IDs contained in this csv file are for the items in the test dataset provided. Therefore the test data is being used to make predictions only and hence there is no need for labelling.

Solution Statement

The stated problem can be solved by applying sklearn's CountVectorizer[3] to the train dataset which will convert the collection of tweet texts into a matrix of words with their token counts. We can then assign weights to the top words for the various sentiment categories. With these weights assigned, predictions can be made on tweets which will return the subset of words with the largest weight sum. This solution will be clearly quantifiable and measureable.

Benchmark Model

A simple Vanilla Artificial Neural Network (ANN) will be used as the benchmark for this project. This is because the model is fast, easy to implement and can be used to make predictions which would give better results than random guessing for the problem at hand.

Due to the fact that this problem is from a kaggle competition, the score of the winning entry which was 0.73615 will be used as a secondary benchmark for this project's solution.

Evaluation Metrics

On the Kaggle competition website, the Jaccard index[4] was used as the metric to quantify the performance of submitted entries. This metric will equally be adopted for the evaluation of this project.

Project Design

The dataset will be preprocessed by dropping any NaN value present. Our dataset will be cleaned by removing punctuations, symbols, numbers, Stop words etc. Also, the emoticons will be replaced with their corresponding sentiment. URLs, hashtags and usernames will also be removed.

Exploratory Data Analysis (EDA) will be carried out on the dataset to obtain insight into its various features and how they would affect our proposed model.

Using sklearn's train_test_split function, the train dataset will be splitted into training and evaluation sets for training and tuning the model with the train set receiving an allocation of 80% of the dataset.

Using sklearn's CountVectorizer function, we will be able to convert the collection of tweet texts into a matrix of words with their token counts. We can then assign weights to the top words for the various sentiment categories.

The proposed Algorithm for weight calculation by Nick Koprowicz[5] will be used as shown below:

. For each class $j \in \{positive, neutral, negative\}$

a. Find all the words i in the tweets belonging to class j.

b. Calculate $n_{i,j}$ = the number of tweets in class j containing word i.

c. Let d_j be the number of tweets in class j. Calculate $p_{i,j} = \frac{n_{i,j}}{d_j}$, the proportion of tweets in class j that conain word i.

d. Let $w_{i,j} = p_{i,j} - \sum_{k \neq j} p_{i,k}$ be the weights assigned to each word within each class.

With these weights assigned, predictions can be made on tweets which will return the subset of words with the largest weight sum.

His proposed algorithm for finding the selected text is given below:

. For every tweet:

a. Let j be the sentiment of the tweet.

b. If $j ==$ neutral return entire text.

c. Otherwise, for each subset of words in the tweet, calculate $\sum_i w_{i,j}$, where i is the set of words in the tweet

d. Return the subset of words with the largest sum, given that it exceeds some tolerance.

With our model set up, the submission file will then be passed to it for prediction.

An alternative approach that could also work for this project would be to model the problem as a Question-answering system where given a question (is sentiment positive or negative?) and a context (tweet text), we train a transformer model to find the answer (selected_text). This approach was given by Jonathan Besomi[6].

References

1. https://en.wikipedia.org/wiki/Natural_language_processing
2. https://www.kaggle.com/c/tweet-sentiment-extraction/notebooks
3. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
4. https://en.wikipedia.org/wiki/Jaccard_index
5. https://www.kaggle.com/nkoprowicz/a-simple-solution-using-only-word-counts
6. https://www.kaggle.com/jonathanbesomi/question-answering-starter-pack