Machine Learning Engineer Nanodegree

Capstone Proposal

Swaraj Dash

October 5th, 2019

Proposal

Domain Background

In the past few years, there has been a huge growth in the use of microblogging platforms, such as Twitter. Spurred by this growth, companies are seeking ways to mine these platforms for information about what people think and feel about their products and services. Sentiment analysis, an application of Natural Language Processing, is the contextual mining of text which identifies and extracts subjective information in the source material, thus helping a business understand the underlying sentiment of their users - whether it is positive, negative or neutral.

Having seen my college senior and mentor work on documents classification on the Reuters dataset, it has always been a personal motivation to work on an NLP application. Understanding how computers process and analyze human language fascinates me. Hence, this capstone project will be about performing sentiment analysis on tweets for each of the six major U.S. airlines.

Research:
This Is How Twitter Sees The World : Sentiment Analysis Part One
Twitter Sentiment Analysis: The Good the Bad and the OMG!
Sentiment Analysis: Concept, Analysis and Applications
Sentiment Analysis of Twitter Data: A Paper from Columbia University

Problem Statement

The problem is to perform sentiment analysis using Machine Learning techniques. We will attempt to classify the sentiment behind a tweet as one of the three classes: positive, neutral or negative, given the tweet (text). This is a multi-class classification challenge.

This is a Supervised Learning exercise where we will apply Machine Learning algorithms to predict the sentiment of a tweet. We will demonstrate the performance of our models by statistical metrics like test accuracy scores.

Datasets and Inputs

For this project, I'll be using the Twitter US Airline Dataset from Kaggle (https://www.kaggle.com/crowdflower/twitter-airline-sentiment/).

Input Dataset: Tweets.csv (14,640 labeled tweets)

- Dataset size for training: 80% of data (11712)
- Dataset size for testing: 20% of data (2928)

The fields of the dataset are as follows:

- **tweet_id**: unique identifier for each tweet in the dataset
- **airline_sentiment**: contains one of ['positive', 'neutral', 'negative'].
- **airline_sentiment_confidence**: a floating point number in the range from 0.0 to 1.0, that specified the confidence of the *airline_sentiment* attribute.
- **negativereason**: specifies the reason for *negative* sentiment. It has 11 unique values, eg: 'Bad Flight', 'Late Flight', 'Lost Luggage', 'Cancelled Flight', etc.
- **negativereason_confidence**: a floating point number in the range from 0.0 to 1.0, that specified the confidence of the *negativereason* attribute.
- **airline**: contains the name of the airline service; one of ['Virgin America', 'United', 'Southwest', 'Delta', 'US Airways', 'American']
- **name**: contains username of the Twitter user
- **retweet_count**: an integer value containing the number of retweets for that particular tweet
- **text**: contains the original tweet itself
- **tweet_created**: contains the timestamp of when the tweet was made

Solution Statement

The idea is to apply Machine Learning algorithms to build a model that will predict the class of the sentiment of a tweet as positive, neutral, or negative. We will first try to get some understanding of the data with Pandas, Matplotib, Numpy, etc. Then we will apply some popular algorithms like SVM, Decision Trees, Random Forests, which fit the purpose of classification problems. The models will be trained on the training data that will consist of 80% of the dataset, and will be tested on the remaining 20%. We will then verify our models' accuracy scores. The data definitely requires some preprocessing, as it seems to have a lot of unnecessary attributes, that have no correlation with the test results. We may try both binary (positive or negative) or multi-class (positive, neutral or negative) classifications and see how the results vary.

Benchmark Model

During my model optimization exercise, we will be using a Logistic Regression classifier model as the benchmark model. We will try to beat this benchmark model, with proper hyperparameter tuning, given they are fitted to the same training and testing set. We will use the models' test accuracy scores as the basis of comparison.

Evaluation Metrics

The evaluation metric to be used for this problem will be the Accuracy Score. We will try to beat the benchmark model with our model. It is nothing but the ratio of total number of correct predicted outcomes to the total number of outcomes.

Let us try to understand:
Given are two lists, y_pred and y_true, that contain the predicted and actual outcomes respectively. For every position index i, compare the i-th element of y_pred with the i-th element of y_true and perform the following calculation:

1. Count the number of matches
2. Divide it by the number of samples

Consider this example:

`y_pred = [0, 2, 1, 3, 0], y_true = [0, 1, 2, 3, 0]`

We see matches on indices 0, 3 and 4. Thus:

`number of matches = 3 number of samples = 5`

Finally, the accuracy calculation:

`accuracy = matches/samples accuracy = 3/5 accuracy = 0.6`

Project Design

- **Programming Language**: Python 3.6
- **Libraries**: Pandas, Matplotlib, Natural Language Toolkit, scikit-learn

I plan to use the following workflow:

- Data Exploration
    - Loading libraries and data
    - Get an understanding of the dataset
    - Dimensions of the data
    - Statistical summary

- Data Preprocessing
    - Data cleaning
    - Removing unnecessary attributes
    - Identifying feature and target columns
    - Splitting of data into training and testing data

- Text Preprocessing
    - Deploy NLTK
    - Remove stopwords
    - Generate clean tweets

- Develop Models
    - Build ML models
    - Select the best models
    - Make predictions and evaluate test scores
- Model Improvement
    - Hyperparameter tuning to beat benchmark model
- Draw Conclusions