Machine Learning Engineer Nanodegree

Capstone Proposal

Mark Bastian
April 24th, 2019

Political Tweet Classification

Domain Background

Twitter is a common platform for people, including politicians, to express their views. Often tweets are highly polarized and opinionated, especially given the current political climate in the United States as well as the rise of fake "Trolling" tweets. Given this, it is valuable to have tools to answer such questions as:

- Is this a tweet from a real person?
- What are the political leanings of a tweet?
- Can I determine the party of the author of a tweet?

Besides being interesting, such a tool could be useful for doing things such as targeted advertising of candidates to twitter users in general. Users that express sentiments that correlate well to one party would be good targets for advertising for issues or candidates with similar sentiments and values.

A massive amount of related work exists, including:

- Actionable and Political Text Classification using Word Embeddings and LSTM
- Leveraging Deep Learning for Political Leaning Classification
- On Classifying the Political Sentiment of Tweets
- Text Classifiers for Political Ideologies
- Topic-centric Classification of Twitter User's Political Orientation

Problem Statement

The problem I wish to solve is classification of the author of a tweet into one of the two major US political parties (Republican or Democrat). Specifically, the model or models will take a tweet or tweet length text as input and produce a prediction of whether the tweet's author is a Democrat or Republican. The input is expected to be political in nature, so I will not detect non-political tweets and I will limit the output categories to

only the two major parties.

The dataset I will train on is sourced such that I know beforehand the party of the tweet. This can be broken up into testing and training sets to measure goodness of model fit.

Datasets and Inputs

I will use the Democrat Vs. Republican Tweets dataset found here.

This dataset provides 86,460 tweets divided roughly evenly (over 42,000 tweets per party) and is sufficiently large to produce sizeable testing and training sets. As the authors are all politicans it is implicitly categorized by the author's political affiliation.

Solution Statement

I would like to create two models and compare the results. One will use a standard text classification approach from sklearn (e.g. a Bayesian Classifier) and the second will use content from the second half of the course (a deep neural network, CNN, or RNN - not covered in the course).

Benchmark Model

As part of my solution, my benchmark model will be a Naive Bayes Classifier. My hope is that a neural network will produce better results. Similar solutions on Kaggle produce accuracies of about 74%.

Evaluation Metrics

The model will be evaluated using an 80/20 split of the data into training and testing data. Standard metrics such as precision, recall, f1 score, and a confusion matrix can be used to determine how good the resulting model is.

Project Design

From my research, I believe two possible network encodings that could produce good results would be LSTM/RNNs or CNNs. LSTMs were not covered in the class, but many resources exist to explain them as well as describe their use for text classification. Additional resources can be found here, here, or here. Although LSTMs seem like a natural choice for this problem, recent work has show that CNNs also work well for text problems. CNNs also can be trained in parallel, whereas LSTM and RNN training is a linear process so is much slower.

My planned workflow is to:

1. Load the data and create an 80/20 train/test sample set.
2. Create a simple Naive Bayes Classifier as a first model.
3. Create a CNN based classifier as a second model.
4. At this point, I expect quite a bit of tuning will be required on the CNN (e.g. number of layers, size of layers, etc.). I also expect it will take some work and learning to get input text encoded correctly. For example, I know I'll need to master variable-length input encoding to make this work but I haven't fully researched the topic yet.
5. Once I'm happy with the models, I'll report various metrics to compare them (e.g. accuracy, precision, recall, f1).