Multi-class Classifier Training

Anindya Dutta and Aarushi Goel

October 23 2017

1 Training dataset

1.1 Sources

We required an emotion dataset that had multiple classes. After looking through, we came across two open-source datasets that have emotion classification for tweets. We will be using the resources by Crowdflower (a CSV file with **40000** examples) to train our model for the emotions. We use `pandas` to combine the data in our model.
Our model consists of pairs of sentences and the emotion they represent. The script `makedata.py` creates our dataset. The result is a dictionary.

For example, to load the CSV file, we write the following:

```
import pandas as pd
df = pd.read_csv('raw_data/text_emotion.csv', usecols=(1,3))
```

1.2 Background research

We tried our hands on a number of existing Python libraries developed for sentiment analysis. Specifically, a lot of our understanding of the ways we could implement came from `pysentiment 0.1.2.1`, `sentimental 2.2.3`, `vaderSentiment 2.5` and `textblob`. The main two methods of classifying are count-based and probability-based. However, all the above mentioned libraries only perform binary classification. Our work is a logical extension of the method used in `textblob`, which uses a probabilistic approach to sentiment classification.

1.3 Classes for classification

In binary classification, we classified our data as `positive` and `negative`. Using the CSV file, we have the following 13 sentiments:
`['love', 'empty', 'surprise', 'sadness', 'neutral', 'relief', 'enthusiasm', 'happiness', 'worry', 'hate', 'boredom', 'fun', 'anger']`
The data is pre-processed into a dictionary where each sentiment is a key, with all the lines for that particular sentiment as its values. Our goal is to classify the input text into one of these classes. We are experimenting with Naive Bayes' Classifier to calculate our results.

The library that we have taken reference from is TextBlob. TextBlob uses a Naive Bayes Analyzer to output a sentiment. However, the result is only a binary value `pos` or `neg`. We define a new module called `NaiveBayesAnalyzer` that can accommodate for multi-class classification.

2 Processing data for Naive Bayes Classifier

2.1 Member variables for class `NaiveBayesAnalyzer`

The dictionary created in (1) is passed as a parameter to the `NaiveBayesAnalyzer`. The dictionary has 13 keys as mentioned in (1.3).

2.2 Algorithmic component

The class `NaiveBayesAnalyzer` is responsible for training and analyzing strings. We use a 80-20 divide for the examples in our model (32000 for training, 8000 for testing). Feature engineering is an important aspect for Naive Bayes. For example, with the features being the default bag of words, the accuracy for 8000 models is **0.17**. Update it to word frequency improves the accuracy to **0.20**.

We have engineered a list of features for the model. They include word (unigram), bigram and trigram frequencies. However, for the tweets, because of language model differences, these features do not improve the model. The class `FeatureExtractor` compiles the set of above mentioned features into a dictionary for every training example for the `NaiveBayesAnalyzer` to classify.
Next Step: Look for a better training dataset.

2.3 Results of Feature Engineering

The current issue is with the dataset. We are looking for a dataset that can represent the natural English language. The language model can improve based on this training data set and will help in better classification of the real test data, which is the fiction text from Project Gutenberg. Once we have a better dataset, we will add in more features such as morphology stemming, exclusive vocabulary for sentiments, and average line lengths.

3 Next Steps

3.1 Training data

We need to search for well-classified training data that has sentences classified into sentiments. We have looked up on this and there are a few resources that have very limited number of lines (20-30).

3.2 From Sentences to Chapters

The program is working with correct outputs for sentences from the validation dataset in Twitter. Our next step is to use the Gutenberg API and link it to this classifier.

3.3 Merge Sentiments (Considering)

Classifying into 13 sentiments may result in lower confidence even on correct classification. We are considering the pros and cons of bringing down the number of classes to 5-6.