**Luke Walsh**

**May 11, 2016**

## PySpark Streaming and NLTK Text Processing:

## Naïve Bayes Classification for Twitter Sentiment Analysis

**Problem/Purpose:** Build a PySpark Streaming application using Python’s NLTK library that analyzes the sentiment of Tweets about a given topic/search term in near-real time

**Big Data Set:**

Training data set:

-“Twitter Sentiment Analysis Dataset”

-~1.5 million Tweets already classified as positive or negative

-source: <http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/>

-format: csv

Classification data:

-Twitter live feed (streaming)

-https://stream.twitter.com/1.1/statuses/filter.json

-returns specified number of Tweets every interval specified

-format: returns JSON

**Hardware:**

* CentOS 6.7 VM running on Windows 10 VMWare Workstation 11.0

**Software:**

|  |  |
| --- | --- |
| **Technology/tools** | **Description** |
| Hadoop 2.6.0-cdh5.7.0 | Distributed file system with 1 data node (pseudo-distributed environment) |
| Spark version 1.6.0 | Cluster computing software; this app will use PySpark Streaming to analyze streaming data |
| Python version 3.3.3 | Language app is developed in |
| Python NLTK library | Natural Language Toolkit Library – a Python package that has useful tools for natural language processing such as the Naïve Bayes classifier this app uses |
| D3.js | JavaScript library used for visualizing results |

**Overview of steps:**

1. Train a model (Naïve Bayes) that will classify Tweets as positive or negative

2. Set up Twitter stream and format incoming tweets to run through the model

3. Run batches of Tweets through the classifier and capture results

4. Visualize

**Summary:**

* Using a Naïve Bayes model
  + Frequently used in text/document classification
  + Features are typically extracted based on most common words
  + Assumes independence between features
    - Not entirely valid assumption, especially in the context of sentences. But, still useful.
    - Using unigrams extracted from the tweets, so relationships between words is not know by the model
* Used SentimentAnalyzer (Python NLTK package) to:
  + Apply negation to all words (e.x. ‘not good’ -> ‘good’ becomes ‘good\_NEG’)
    - Even though we’re using unigrams, this allows the model to attempt to distinguish negated words
  + Break Tweets into unigrams (so ‘good\_NEG’ would be counted as a unigram)
  + Remove all stopwords so the features of the model aren’t words like ‘is’
  + Extract the top 200 most common unigrams to use in the classification model
* Train the classifier model -Python NaiveBayesClassifier (NLTK package)
* Test the model on some sentences

First create app on Twitter’s developer webpage and deal with API key’s/secret and user token/secret

Need authentication in order to access the Twitter stream

For more info: https://dev.twitter.com/oauth/overview

Use Python’s requests and requests\_oauthlib packages to setup connection

Requests\_oauthlib – handles authorization

Requests – allows the Spark streaming context make REST calls to the Twitter API

First create app on Twitter’s developer webpage and deal with API key’s/secret and user token/secret

Need authentication in order to access the Twitter stream

For more info: https://dev.twitter.com/oauth/overview

Use Python’s requests and requests\_oauthlib packages to setup connection

Requests\_oauthlib – handles authorization

Requests – allows the Spark streaming context make REST calls to the Twitter API

For each batch of Tweets that comes in, format the data to prepare to run through the classifier

Apply the Naïve Bayes model features to the tweets

Classify Tweets and capture output:

In this case we’re just capturing number of positive/negative tweets per batch

Use standard D3.js library to produce trend graphs (nothing fancy)

But! Potential for many other interesting visualizations

Ex: using Tweets’ location data to plot where the Tweet happened

Combined with sentiment analysis we could see how different regions feel about a subject on Twitter

* + Spark has very fast batch processing compared to something like Hadoop due to in memory computations
  + Spark is well suited for streaming applications and is scalable
  + Python NLTK library has many built-in tools to assist in NLP
  + Overall, PySpark Streaming and NLTK are very useful
* Cons:
  + No major cons from this demo project, but online research raised a few possibilities:
    - Python has useful tools but there are some things that are better handled or can only be done in Scala or Java, such as certain RDD operations
    - Scala and Java may potentially perform faster than Python. May depend on the application.

**YouTube URLs here:**