Basic twitter sentiment analysis with Pyspark and Kafka

Rongrong Su
STATISTICS AND ACTUARIAL SCIENCE
University of Waterloo
Waterloo ON CANADA
r2sue@uwaterloo.ca

Introduction

This project is about processing live data using Spark's streaming APIs and Python. In this project, I perform a basic sentiment analysis of real-time tweets. Specifically, I count the number of negative and positive words at each time step and visualize them with line graph. In this project, Apache Kafka, a distributed stream processing system, is also implemented. This project is based on uncompleted project of ADBI 591 at NCSU. For more details, the description of project of ADBI 591 at NCSU is given in Project-SparkStreaming document. This report contains three parts: methodology, conclusion and evaluation.

Methodology

- Apache Kafka

Apache Kafka is an open-source distributed streaming platform that enables data to be transferred at high throughput with low latency. To be more explicit, it is a messaging system using which one can forward the messages to other applications like Spark, HDFS etc. It has built-in client APIs that developers can use when developing applications that interact with Kafka. It is fast, scalable, durable and fault-tolerant. At its core, Kafka is a Pub/Sub system built on top of logs, with many desirable properties, such as horizontal scalability and fault tolerance. Kafka has since evolved from a messaging system to a full-fledged streaming platform (see Kafka Streams). Next are the steps for running Kafka and using Twitter streaming API:

1. Install and run zookeeper
2. Install and start Kafka server
3. After setting the Kafka, it is ready to accept messages on dynamically created Kafka topics. Then I create a topic named 'twitterstream' in Kafka.
4. In order to download the tweets from twitter streaming API and push them to Kafka queue, a python script twitter_to_kafka.py is provided. The required twitter authentication tokens is in twitter_to_kafka.py.

- Spark Streaming

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Kafka, Flume,

Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and live dashboards.

In this project, spark streaming part is in twitterStream.py document, which writes the running total count of the number of positive and negative words that have been tweeted. In addition, I plot the total positive and negative word counts for each time step. The word lists for positive words and negative words are given in the positive.txt and negative.txt files respectively. The twitterStream.py contains three main functions: load_wordlist(filename), stream(ssc, pwords, nwords, duration), make_plot(counts).

load_wordlist(filename)
This function is used to load the positive words from positive.txt and the negative words from negative.txt . This function returns the words as a list or set.

stream(ssc, pwords, nwords, duration)
This is the main streaming function consisting of several steps like starting the stream, getting the tweets from kafka, and stopping the stream. I take the streamed tweets and find the number of positive and negative words. The tweets variable is a DStream object on which I can perform similar transformations that I could to an RDD. Currently, tweets consist of rows of strings, with each row representing one tweet. We can view this structure by using the pprint function: tweets.pprint()

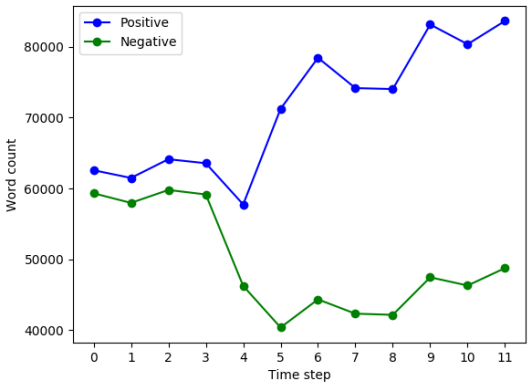
I want to combine the positive word counts together and to combine the negative word counts together. Therefore, rather than mapping each word to (word, 1), I can map each word to (“positive”, 1) or (“negative”, 1) depending on which class that word belongs. Then, the counts can be aggregated using the reduceByKey function to get the total counts for “positive” and the total counts for “negative”. The DStream would store the counts at each time step, not the running total. To get the running total, I must use the updateStateByKey function. For more details, read the section about this operation. It is this DStream (the one with the running totals) that I should output using the pprint function.

The other DStream (the one that doesn’t store the running total) can be used to make the plot that shows the total counts at each time step. However, I first must convert the DStream into something useable. To do this, use the foreachRDD function. With this function, I can loop over the sequence of RDDs of that DStream object and get the actual values.

make_plot(counts)
In this function, I use matplotlib to plot the total word counts at each time step. This information should be stored in the output of our stream function (counts).

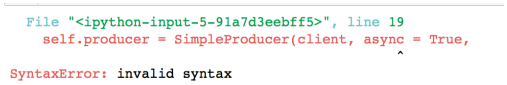
Conclusion

Finally, the plot of number of negative and positive words is given as follows.



Evaluation

This project is implemented with Python 3.6 instead of Python 3.7 because of the Python 3.7 compatibility issue.



So, it is my further job to figure out how to implement this project with figure out.

References

1. Kafka introduction, https://kafka.apache.org/intro.
2. Manisha Malhotra, Streaming Real-time Twitter feeds using Apache Kafka, https://www.linkedin.com/pulse/streaming-real-time-twitter-feeds-using-apache-kafka-manisha-malhotra.

3. satvikshetty04, twitter-sentiment-analysis, https://github.com/satvikshetty04/Twitter-Sentiment-Analysis.