Jasen Jones, Miki Seltzer, Eric Whyne
W205 Fall 2015

W205 Final Project Progress Report

Project Summary

As described in our project proposal, our plan is observe electronics related tweets before, during, and after Black Friday and Cyber Monday.

Progress Summary

We've started collecting data, looked at the contents of what's coming into our systems to refine our collection filters, and experimented with several different streaming and processing architectures. Early observations of our collected data indicated that some of our filter terms were overly broad

Data Collection

As we experiment with different processing options we've been collecting data to file. Our scripts can be viewed here: https://github.com/ericwhyne/MIDS-SARD-Project The script is fairly straight forward, tracking terms using Twitter's Streaming API and dumping JSON records to a file who's name includes the current ISO-8601 formatted date. In order to ensure continued execution, Supervisord (http://supervisord.org/) is used. Supervisord recovers the program in case of a timeout due to connectivity, reboot due to power loss, or program error.

Our initial set of keywords resulted in about 25GB of data being collected per day. After removing overly broad keywords, we are currently seeing about 11GB of data on week days and 9GB of data on weekends.

Sentiment Analysis

We have implemented some basic sentiment analysis and are currently working on ways to integrate the results into our existing data store. Our overall concept is to rate tweets as positive, negative, or neutral depending on the text contained in the tweet. To do so we modified an algorithm from http://fjavieralba.com/basic-sentiment-analysis-with-python.html to tokenize each tweet and count the number of positive and negative words contained in the tweet based on sentiment dictionaries from academic literature. This method is limited in scope as it cannot account for things like sarcasm or historical trends, but should give us a basic understanding of aggregate sentiment values toward a product.

To score each tweet we tokenize the text and assign a descriptive tag for each token. "Good" would be tagged with "positive" while "terrible" would be tagged with "negative" for example. We have also accounted for preceding words that increase, decrease, or invert the

meaning of the original sentiment. Increasers like “completely”, “so”, and “totally” double the score of the following word, while decreasers like “kinda”, “barely”, and “sorta” halve the following word’s score. An inversion word like “not”, “won’t”, and “didn’t” plays a different role, reversing the value of a given token. This means that while “good” is normally positive, “not good” should be negative. We are also considering using existing sentiment dictionaries that include a gradient positive/negative scale rather than binary good/bad designators, but at this point we simply wanted to show proof of concept. We also intend to incorporate the tagging algorithm at the point of data collection so it can more easily be used during our future queries.

Processing Architectures

Our initial plan was to process the data using Hadoop batch. However, since we have a nice streaming data set, we decided to also invest time in setting up stream processing. Rather than point our streaming pipeline directly at the twitter API feed, we’ve decided to use Apache Kafka as a queue. Kafka gives us the option of using Balanced Consumers to stream processing directly, or using other streaming software such as Apache Storm or Spark Streaming which can read directly from Kafka Topics. Code for producing twitter to a Kafka topic and reading as a Balanced Consumer is included in our github repo (https://github.com/ericwhyne/MIDS-SARD-Project). After experimenting with Python code for consuming Kafka topics, we discovered that only one library supported the concept of Balanced Consumer and we had to increase the number of topic partitions to support this, since consumers simply coordinate their efforts by allocating partitions to each other. This means that in theory we should be able to scale to an arbitrary volume and velocity of data. In practice, we suspect that the coordination done through ZooKeeper between balanced consumers is going to cause problems. We have yet to see how well Storm or Spark Streaming adapt to Kafka Topics. Our Sentiment or other analytics, written as Python functions, should be easy to plug into whatever processing architecture we decide to use.