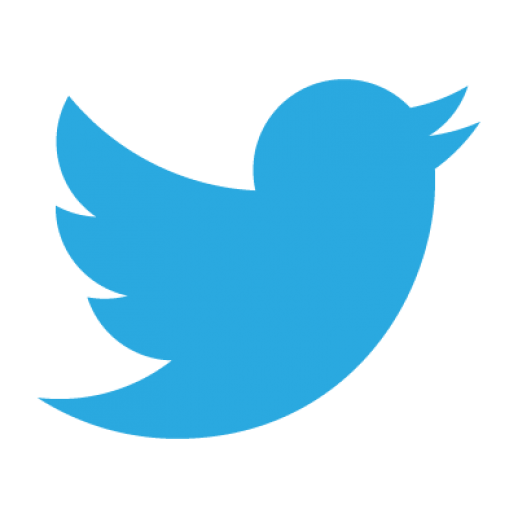
We Consult Everything (WCE)

Lokanadh Naveen Deevi | Sandra Sara Jacob | Arulmozhi Varman Kulasekaran | Betty Kurian

**Detecting trending Events from Twitter**



# **Introduction**

Since the surge in social media data, We Consult Everything Ltd (WCE) has acquired several clients to facilitate them in leveraging their massive amounts of data to make better informed business and operational decisions. As consultants for Dr. Who, we have formulated a high-level pipeline for detecting interesting events or trends from Twitter using big data techniques. As Dr. Who previously stated, Twitter tweets, if monitored and analyzed, can provide valuable information allowing users and organizations to obtain meaningful insights. The actionable plan consists of real-time detection on Twitter, the ability to convey interesting and trending events as well to receive pushed event information on a multitude of devices (phones, tablets, computers, etc.)

# **Actionable Plan**

To start the process, we will obtain a training dataset from Twitter Streaming API which consists of 6 months’ worth of tweets from last year stored in HDFS as key value pairs via Kafka zookeeper. After initial data preprocessing in Spark, to create a bag of words and then forming topic clusters with an LDA tuning package, human annotators will manually classify them as events and nonevents. Each of the event clusters are then labeled per their social features (i.e. user interactions/retweets/mentions/etc.) and Twitter centric features (i.e. non-real world events/mannequin challenge/giving Tuesday/etc.). Using these features, we train a classification model based on Naïve Bayes, for it’s speed, performance and scale, with cross validation to distinguish between events and nonevents. This facilitates reducing the amount of noisy data for real time tweets. Because clusters constantly evolve over time, the features will be intermittently updated for the old clusters by computing for newly formed ones.

The initial input of the real-time data would consist of the actual tweets from users from Twitter’s global Streaming API, being updated every hour. The real-time Twitter data would be pulled with low latency for processing. This streamed data will consist of each tweet’s username, tweet message, location and time. With Python coding, Kafka (using the Hosebird client library) will collect the high volumes of real time log data in JSON format from Twitter Streaming API and send it to Spark Streaming as discretized streams (DStreams) for fast pull analysis. Kafka is a service that is highly scalable, message durable and fault tolerant with quick recovery to move enormous amounts of geographically distributing streams of data into Spark for storage and analysis as they are becoming more integrated.

There are only a handful of systems that perform scalable, rapid efficient iterative batch processing well, thus Spark was selected for this plan. The iterative algorithms and interactive data mining along with enhanced programmability is better supported on Spark. It also makes it easy to use a single framework to satisfy all the processing needs while sharing and reusing code and business logic between streaming, batch and interactive pipelines. Once the data is in Spark, it needs to start getting processed. Spark Streaming will ingest the twitter data from Kafka, dividing the DStreams into batches via Scala, Python or Java scripts. These batches of input are resilient distributed datasets (RDD). The RDD are then partitioned and distributed for processing while being stored in memory for faster access with partitions being recomputed on failure. This allows Spark Streaming to then seamlessly integrate with the next Spark Engine component, MLlib (machine learning library).

Spark’s MLlib helps make machine learning easy and scalable by providing machine learning tools, featurization, pipelines, persistence/tuning and utilities in Python, Scala or Java. The incoming data will need to go through three phases: data preprocessing, data representation and data organization. The RDD batch inputs from Spark Streaming will initially need to be transformed via tokenization so it is broken down into a bag of words setting the delimiter to white space. After that has completed, stop words will be removed since they don’t carry much meaning by using the natural language toolkit package (NLTK) which is a corpus of stop words. The words will go through stemming to reduce them down to their root word form. From this we will then calculate the number of retweets, @usernames, links, hashtags, proper names (using Stanford Named Entity Recognizer (SEM) and frequency of multiword with special capitalization which make up the social and twitter centric features for classification purposes. For the data representation phase, this data will then be categorized as an event or nonevent using the classification model created with the training dataset, which is periodically being updated. In order to discover the topics of these events, topic modeling is conducted using Latent Dirichlet Allocation (LDA) with an LDA tuning package to select the number of topics. LDA is based on optimization which has been shown to produce good parameter estimates and dramatically faster on large datasets. It can analyze enormous streams of collections of documents in a short amount of time. After topic modeling, there may be similar topics that form from this analysis thus the next phase, data organization or threshold based clustering, will be performed to merge them with a cosine similarity distance threshold of .8 and above else considered an independent cluster. Since Twitter data is dynamically evolving and new events arise over time, clustering, which is unsupervised, does not need labeled data for training or any previous information on number of clusters, which provides an efficient approach. Clusters will be evaluated within a 48-hour span, so they will be dropped if there are no additional tweets within that timeframe. These final topic clusters and their associated messages will be ranked based on the volume of tweets and frequency of similar tweets within the last hour. Of this, the top 20 will then be selected as the trending events.

This information of the most frequent word of each cluster relating to the topic and the number of users tweeted on it will be sent back to Kafka. DOMO, a business cloud, will receive this data and with the help of an analyst, dashboards will be created which automatically get distributed via the Alert Center to all forms of electronic devices of Dr. Who. With Domo, Dr. Who has an easy to understand user interface to analyze data, interact with reports and slice and dice if need be. He is also able to only see specific data he is most interested in and hide the rest. It is an easy to use, safe and secure environment regardless of how tech savvy you are.

# **Challenges**

With all great plans, there can be some challenges that are faced. It’s best to be proactive and know ahead of time what the potential issues could be so you can plan accordingly. At the forefront, there is so much meaningless messages within Twitter posts such as personal info, random thoughts, opinions, rumors, etc. We may only obtain very little information from the 140-character Twitter post and in low quality form such as grammar issues, spelling mistakes, slang and typos. There can also be an immense amount of content related to nonevents such as sleep or food which have common characteristics of event content. There is are some underlying assumption that occurrence of messages related to the same event should be close in time and increase in frequency closer to an event as well as having no real specific definition of an event or nonevent. There is also an increasing number of spammers and content polluters. Additionally, in regards to the classification modeling, the training dataset of manually labeling is very labor intensive and time consuming. Since classification is done first the model may possibly remove the real-world events before even reaching the clustering phase. It may also be restricted in scope due to the sample size of the training set used. All this in addition with the data sparseness, lack of context and diverse vocabulary can negatively affect event detection performance and text analysis. We must consider highly scalable, efficient and untraditional approaches to accurately obtain these trending events.

# **References**

1. <http://web6.cs.columbia.edu/~gravano/Papers/2011/icwsm11-a.pdf>
2. <https://www.datanami.com/2015/11/30/spark-streaming-what-is-it-and-whos-using-it/>
3. <http://spark.apache.org/docs/latest/mllib-statistics.html>
4. <https://pdfs.semanticscholar.org/b0a8/d512169453d92fbe44bc77a96bcc9cb9c85f.pdf>
5. <http://web2.cs.columbia.edu/~gravano/Papers/2010/wsdm10.pdf>
6. <http://ingest.tips/2015/06/24/real-time-analytics-with-kafka-and-spark-streaming/>
7. <https://www.linkedin.com/pulse/flume-kafka-real-time-event-processing-lan-jiang>
8. <http://www.michael-noll.com/blog/2014/10/01/kafka-spark-streaming-integration-example-tutorial/>
9. <https://www.youtube.com/watch?v=RQEPONxBzBY>