TWITTER NETWORK ANALYSIS

Project Summary

Nowadays, social network plays an important role to show people's attitudes and ideas. This project aims at exploring the social network and topic changing in the group of Women In Technology (WIT) over time. The users' behavior in Twitter has been used in this network analysis. There are four phases in the process of this social network analysis, which are, Network Definition, Language Analysis, Network Analysis and Topic Dissemination/Monitoring. The primary goal is to do a language analysis of tweets to find out the trend of topics and the secondary goal is to find the pool of Twitter users (say a network) who possibly share the same interests, which is derived from the language analysis performed in the first step. The analytical techniques including MapReduce, Natural Language Processing will be used to complete the process to identify the network and keywords. The project also involves building a near real-time Twitter streaming analytical pipeline with Amazon Web Services (AWS) for the phases of Network Analysis and Topic Monitoring. Amazon Kinesis Firehose, AWS Lambda (Python function), Amazon S3, Amazon Elasticsearch Service with Kibana will be integrated to complete the process.

Background

In the past decade, the digital landscape has changed dramatically. Fake news and social media vigilante reporting have become increasingly common. Information now spreads rapidly through the digital media landscape to a general public not yet skilled at detecting fraudulent news. While this new landscape presents a threat, it also presents an opportunity. Organizations now have the power to reach audiences and encourage change for good like never before.

Historically the technology sector has been dominated by men. But over the past decade a movement has grown to support and encourage both gender and ethnic diversity in the tech. With most movements there is an inevitable backlash. In September of 2017, the New York Times published an article covering the backlash against women standing up to harassment and bias in Silicon Valley. The vitriolic nonsense espoused by the sector participants in this article brought to light the reality that while gains have been made there is very obviously still work to be done to at a minimum ensure the work is not undone.

The concept behind this project was to explore and develop a framework and infrastructure that would allow an organization to identify other organizations and individuals in the digital landscape who would be interested and helpful in spreading information on a topic. The framework developed for this project would be useful for any individual, organization or business interested in better controlling information dissemination in the modern digital landscape. This project is important because the infrastructure developed could be used by organizations that lack the technological expertise or financial resources to implement this project on their own. For the purposes of this pilot our group decided to explore a topic that hit close to home and focus on women in technology(WIT).

The social and digital media landscape is comprised of a wide variety of platforms and venues such as Facebook, Twitter, Instagram, Reddit, and news websites. Twitter maintains a public API which allows users to access thousands of randomly selected tweets on a daily basis. Generally speaking, any tweet on twitter can be viewed by anyone with internet access. This functionality has lead to Twitter often being referred to as the “town square” of social media. Twitters emphasis on encouraging the dissemination of information and free speech lead to the choice to use Twitter as the social media platform for the project.

Methods

The framework and associated infrastructure requires the completion of four phases for an organization new to the process. The phases include: network definition, language analysis, network analysis and topic dissemination/monitoring. Each of these phases and the associated methods are described in the sections below.

<u>Network Definition</u>

In order to reach a desired audience, an organization must first figure out how to identify appropriate audience members. As a starting point for this process, a brainstorming session can be used to identify what we will refer to as "super users". Super users are individuals known to be influential in the arena of interest. The common followers among these super users can then be used to define a "base network" of users with a high likelihood of interest in the topic.

<u>Language Analysis</u>

The base network will serve as the basis for creating an exploratory data set of tweets that can be used to better understand how users think and communicate about the topic of interest. The exploratory analysis will focus on analyzing language commonalities and relationships. Thus, the hot topic or keyword will be extracted from the language analysis on tweets content. The outcome of the language analysis will be used in live stream tweet pipeline which is used to identify users for the network analysis.

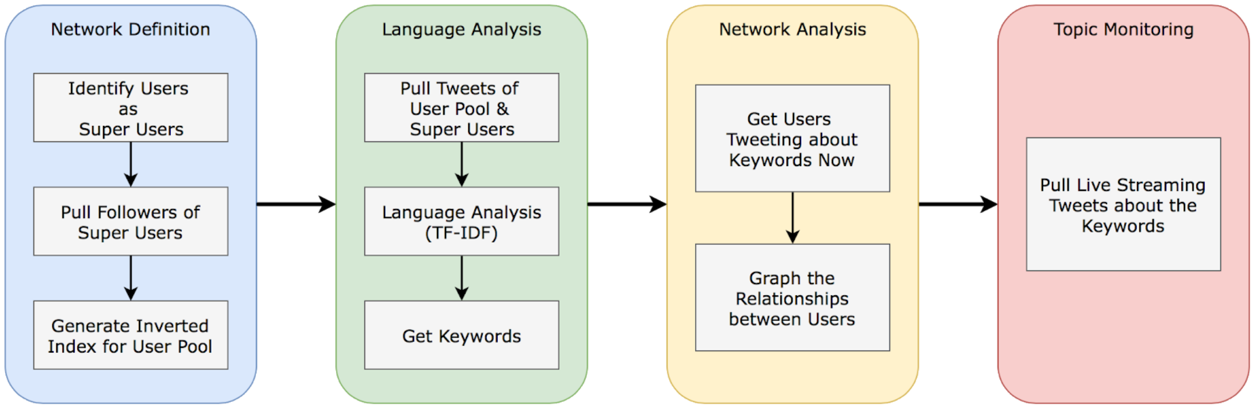
<u>Network Analysis</u>

The relationships between active users in the hot topic will be another interesting thing to study. Users with the most connections in the network can be seen the "leader" in the discussion about a specific topic or a keyword. This phase will take the users list from live stream tweets from the pipeline using as the network analysis.

<u>Topic Dissemination/Monitoring</u>

The last phase of the process is to use information gathered in the exploratory analysis to inform a monitoring system. The monitoring system will be comprised of a dashboard that presents the user with live streaming data from twitter as it relates to the influences and keywords identified in the previous phrase. This presents the user with a streamlined approach for monitoring twitter on an issue of importance.

The graphic below provides a big picture overview of the process.


Network Definition
Identify Users as Super Users
Pull Followers of Super Users
Generate Inverted Index for User Pool
Language Analysis
Pull Tweets of User Pool & Super Users
Language Analysis (TF-IDF)
Get Keywords
Network Analysis
Get Users Tweeting about Keywords Now
Graph the Relationships between Users
Topic Monitoring
Pull Live Streaming Tweets about the Keywords

The network analysis as well as the topic dissemination and monitoring were not completed in preparation for this paper. As such, it is not discussed in the implementation and results section. However, these components are discussed in the future work section at the end of the paper.

Implementation

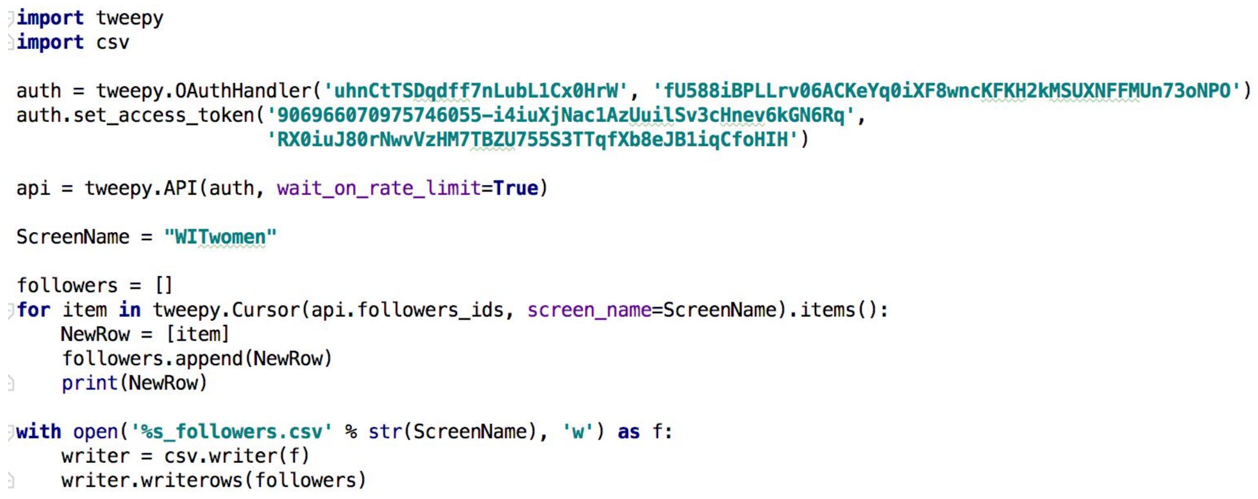
Infrastructure

The Amazon Web Service suite was used to complete the majority of the work on this project. Data was stored in the S3 cloud and mapreduce code was implemented using EMR. Python was the primary programming language and interfacing with the twitter API occurred using Python. AWS offers data streaming capabilities which will be a critical to implementing the monitoring component of this project in the future,

Network Definition

As discussed in the introduction, the topic of interest for this project is women in technology. The research team completed a brainstorming session and follow up research for the purposes of identifying a list of super users known to influence the public discussion of WIT. The super users twitter users include: @womenintech, @WITwomen, @windows, @microsoftwomen, @sherylsandberg, @ericajoy, @codefirstgirls. The users following at least 3 super users above can be considered as one node in the network in this study. Two steps should be taken in this phase: 1. Pulling down the user lists of super users with Twitter API; 2. Finding out the users following more than 3 super users with applying Map-Reduce on AWS.

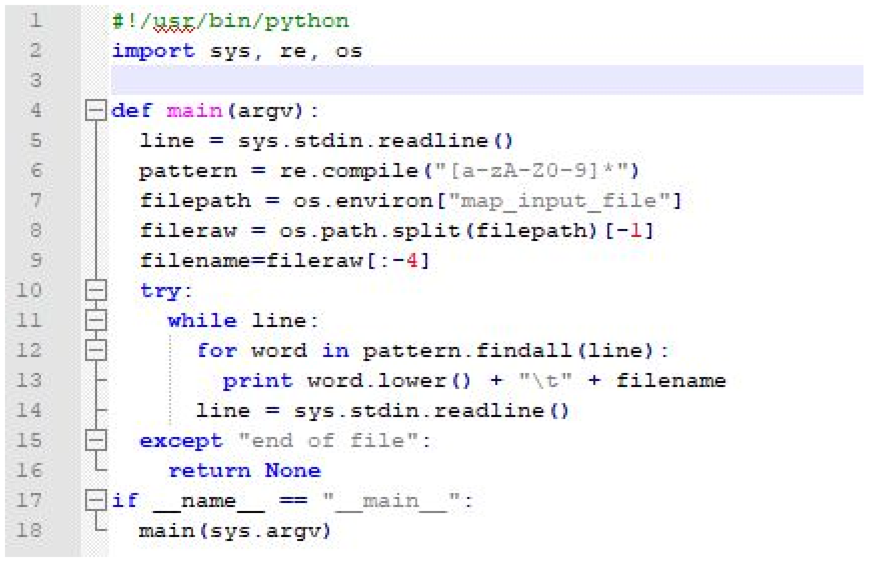
Doing a general analysis of twitter users' trends, we created a list of twitter users, mainly, account holders who have are an ideal example of "Women in Technology", are active twitter user and have a decent amount of followers. The twitter API was queried and userids for each super users followers were collected as using Python code below. This was implemented using python libraries tweepy and an API pipeline. A csv file stored the list of followers for each "superuser".



```
import tweepy
import csv

auth = tweepy.OAuthHandler('uhnCtTSDqdff7nLubL1Cx0HrW', 'fU588iBPLLrv06ACKeYq0iXF8wncKFKH2kMSUXNFFMUn73oNPO')
auth.set_access_token('906966070975746055-i4iuXjNac1AzUuilSv3cHnev6kGN6Rq',
                      'RX0iuJ80rNwvVzHM7TBZU755S3TTqfXb8eJB1iqCfoHIH')

api = tweepy.API(auth, wait_on_rate_limit=True)

ScreenName = "WITwomen"

followers = []
for item in tweepy.Cursor(api.followers_ids, screen_name=ScreenName).items():
    NewRow = [item]
    followers.append(NewRow)
    print(NewRow)

with open('%s_followers.csv' % str(ScreenName), 'w') as f:
    writer = csv.writer(f)
    writer.writerows(followers)
```

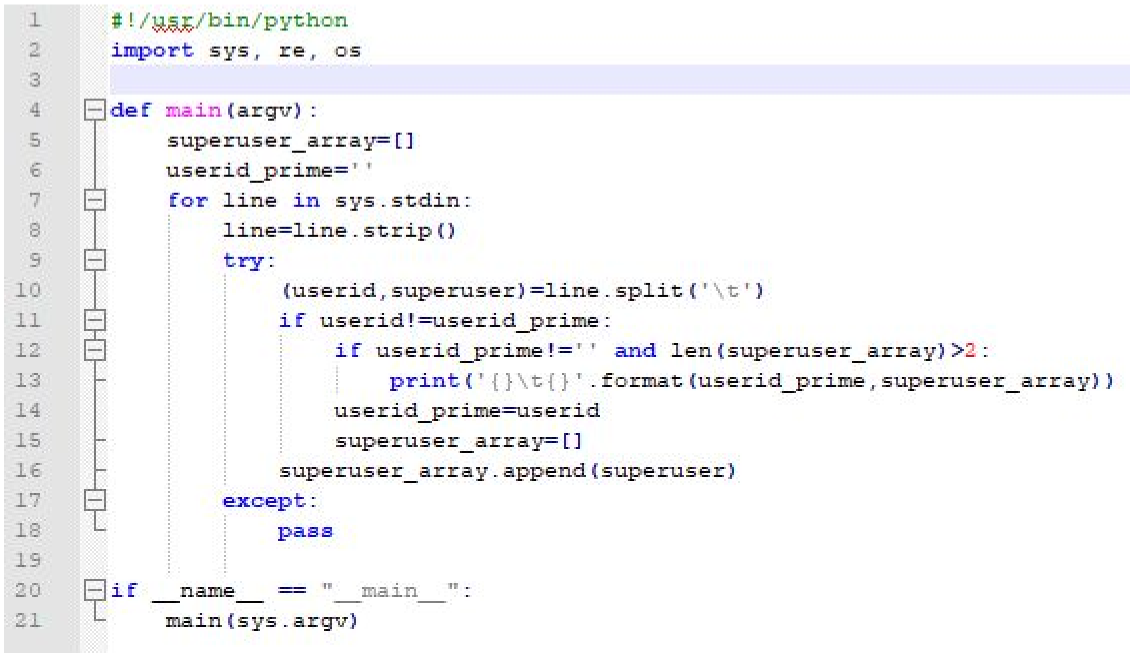
Given the large magnitude of followers for each super user, the aggregation of users was implemented using a map-reduce algorithm in amazon EMR. The mapper and the reducing as written in python are provided below.

Mapper



```
#!/usr/bin/python
import sys, re, os

def main(argv):
  line = sys.stdin.readline()
  pattern = re.compile("[a-zA-Z0-9]*")
  filepath = os.environ["map_input_file"]
  fileraw = os.path.split(filepath)[-1]
  filename=fileraw[:-4]
  try:
    while line:
      for word in pattern.findall(line):
        print word.lower() + "\t" + filename
      line = sys.stdin.readline()
  except "end of file":
    return None
if __name__ == "__main__":
  main(sys.argv)
```

Reducer



```
#!/usr/bin/python
import sys, re, os

def main(argv):
    superuser_array=[]
    userid_prime=''
    for line in sys.stdin:
        line=line.strip()
        try:
            (userid,superuser)=line.split('\t')
            if userid!=userid_prime:
                if userid_prime!='' and len(superuser_array)>2:
                    print('{}\t{}'.format(userid_prime,superuser_array))
                userid_prime=userid
                superuser_array=[]
            superuser_array.append(superuser)
        except:
            pass

if __name__ == "__main__":
    main(sys.argv)
```

One of the assumptions built into this analysis is that users who follow the super users will be interested in WIT. As can be seen in the mapper code, only users following at least 3 super users were considered in this analysis. This threshold was established boost the likelihood that the previously mentioned assumption would hold true.

Language Analysis.

TF-IDF is a commonly used natural language processing method and information retrieval method designed to analyze the language contained within all documents in a corpus in vector space. For our purposes, a document is a tweet and the corpus is all the tweets collected as part of phase I. The tf-idf weighting scheme assign to term t a weight in document d given by the following formula:

$$\text{tf-idf}_{t,d} = \log(1+\text{tf}_{t,d}) * \log(N/\text{df}_t)$$

where:

$\text{tf}_{t,d}$= the frequency of term t in document d
N= the number of documents
Df_t= the number of documents containing the term

The benefit to using this vectorization approach is that terms with a high collection frequency are given a lower importance score while terms with a high document frequency but a low collection frequency are weighted highly. This approach allows for the balance measuring importance as a function of the frequency with which a word is used versus the rarity of a word.

The assumption being that high frequency rare words and treated with similar importance to high frequency highly used terms.

Let D be the collection of documents in the corpus. Let T be the collection of terms (unique words) in our collection D.

The term frequency (tf) for a given time ti within a particular document dj is defined as the number of occurrences of that term in the dj th document, which is equal to ni,j: the number of occurrences of the term ti in the document dj .

Tf i,j =ni,j

The term frequency is often normalized to prevent a bias towards larger documents, as shown below:

tfi,j= ni,j ∑k nk,j

where ni,j is the number of occurrences of the term ti in the document dj. Note that we are using the total number of terms for normalization. Instead we can use the maximum as well.

The inverse document frequency (idf) is obtained by dividing the total number of documents by the number of documents containing the term ti , and then taking the logarithm of that quotient:

idfi =log |D| |{d:ti ∈d}|

with

|D|: total number of documents in the collection
|{d : ti ∈ d}|: number of documents where the term ti appears. To avoid divide-by-zero, we can use 1 + |{d : ti ∈ d }|.

For a given corpus D, then the tf-idf is then defined as:
(tf-idf)i,j = tf i,j ×idfi

A high weight in tf-idf is obtained by a high term frequency and a low document frequency of the term in the collection. For common terms, the ratio in idf approaches 1, bringing the logarithm closer to 0.

We need to compute:

1. Given a corpus of text, calculate tf-idf for every document and every term

2. Need to calculate, over the corpus, the following:
    a. Number of terms
    b. Number of unique terms
    c. Number of documents
    d. Number of occurrences of every time in every document and
    e. Number of documents containing each term

IMPLEMENTATION

To implement tf-idf using MapReduce we will divide it into 2 steps. In the first step we are going to count the documents and in the next we will count the terms in each document and then calculate the tf-idf as with both we have enough information to do so. So the general JobConfig consists of two jobs countTotalDocuments and calculateTfIdf.

From the input the tf-idf for terms in title and text will be calculated and written into a text file.

STEP 1: To count the documents for our calculation we used Hadoop Counters and read the result from our job runner. We only need a map phase and deactivate reducers.

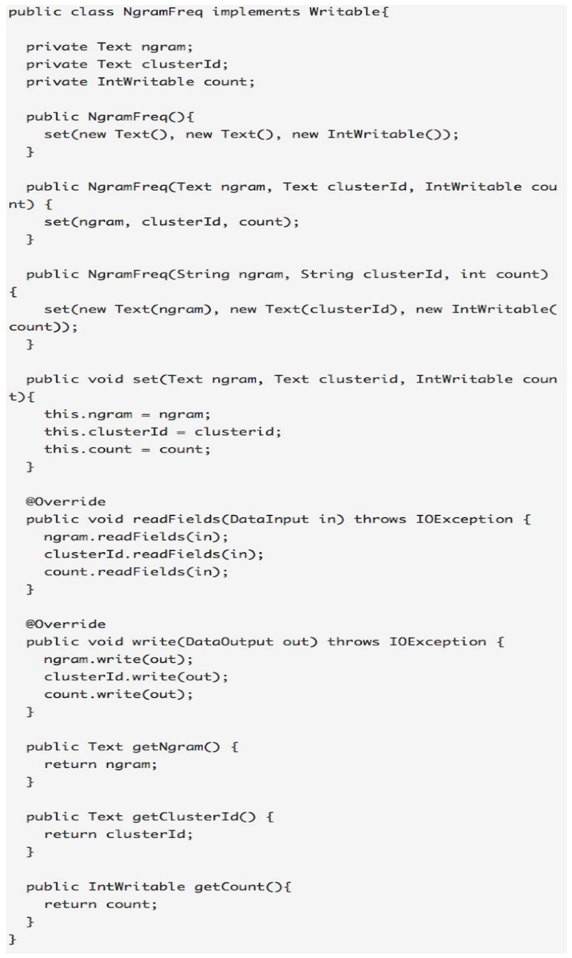
```
    public enum Count {
      TOTAL_DOCUMENTS
    }
    context.getCounter(Count.TOTAL_DOCUMENTS).increment(1);
  }
}
```

STEP 2: TF-IDF with Hadoop secondary sorting
 The next step is to calculate the tf-idf score for each term in a document. The overall idea of using secondary sorting for this is to have the values in the reduce phase grouped and sorted by the document id and the term itself. Like this:

doc1:a
doc1:a
doc1:b
doc1:c
doc2:a

doc2:a
doc2:b

We have to make sure that each word gets partitioned to the same reducer and at the reducer we have to assure that the terms are sorted and grouped by the document id. We had to define our own key class NgramFreqKey.class also created our own NgramFreq model. The key and model are later being used by the partitioner and group comparator at the reducer.



```
public class NgramFreq implements Writable{

  private Text ngram;
  private Text clusterId;
  private IntWritable count;

  public NgramFreq(){
    set(new Text(), new Text(), new IntWritable());
  }

  public NgramFreq(Text ngram, Text clusterId, IntWritable cou
nt) {
    set(ngram, clusterId, count);
  }

  public NgramFreq(String ngram, String clusterId, int count)
{
    set(new Text(ngram), new Text(clusterId), new IntWritable(
count));
  }

  public void set(Text ngram, Text clusterid, IntWritable coun
t){
    this.ngram = ngram;
    this.clusterId = clusterid;
    this.count = count;
  }

  @Override
  public void readFields(DataInput in) throws IOException {
    ngram.readFields(in);
    clusterId.readFields(in);
    count.readFields(in);
  }

  @Override
  public void write(DataOutput out) throws IOException {
    ngram.write(out);
    clusterId.write(out);
    count.write(out);
  }

  public Text getNgram() {
    return ngram;
  }

  public Text getClusterId() {
    return clusterId;
  }

  public IntWritable getCount(){
    return count;
  }
}
```

In cases where we need to control the way Hadoop partitions and sorts at ‘reduce’ level it offers the possibility of using a custom partitioning and group comparator. We are going to use this in our last step to calculate tf-idf.

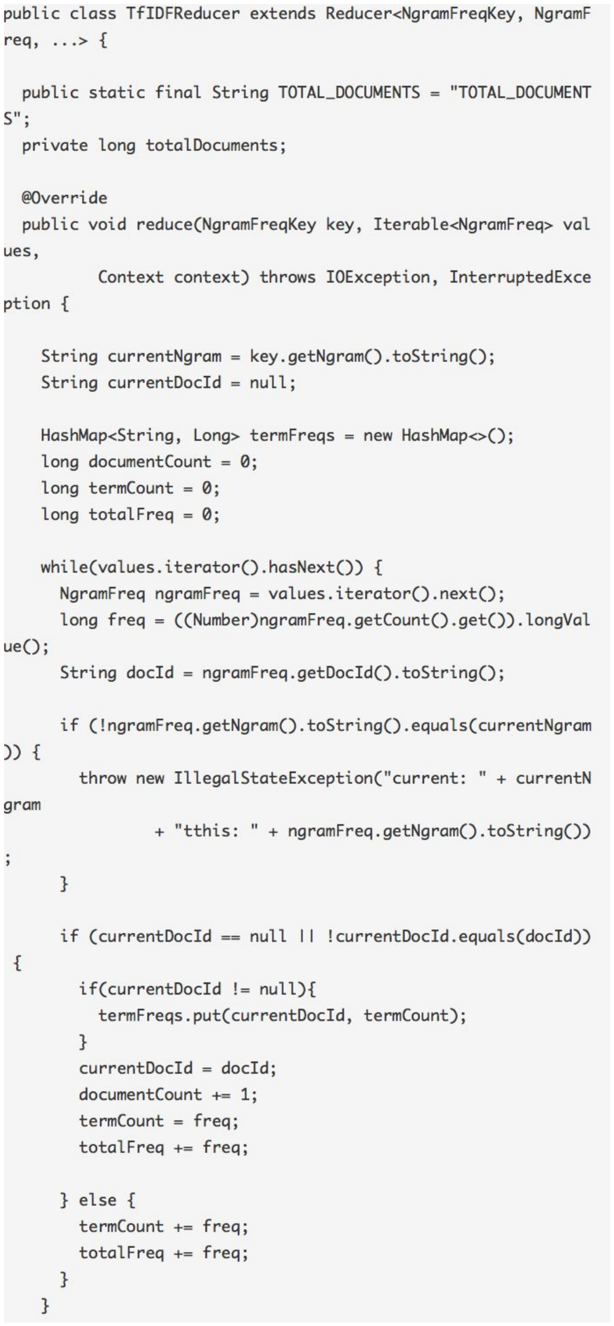
Our partitioner will make sure that we partition by the term itself only and not by the document ID contained in the key. By this we achieve a fairly good distribution and the possibility to count the occurrence of a term at the reducer.

```
public class NgramFreqKeyPartitioner extends Partitioner<Ngram
FreqKey, NgramFreq>{

  @Override
  public int getPartition(NgramFreqKey key, NgramFreq value, i
nt numPartitions) {
    return (key.getNgram().hashCode() & Integer.MAX_VALUE) % n
umPartitions;
  }

}
```

To achieve the sorting at the reducer as stated above Hadoop allows us to override the grouping and sorting. We write our own group comparator like this:

```
public class NgramFreqKeyGroupComparator extends WritableCompa
rator implements Serializable {

  NgramFreqKeyGroupComparator() {
    super(NgramFreqKey.class, true);
  }

  @Override
  public int compare(WritableComparable a, WritableComparable
b) {
    NgramFreqKey gka = (NgramFreqKey) a;
    NgramFreqKey gkb = (NgramFreqKey) b;

    return gka.getNgram().compareTo(gkb.getNgram());
  }
}
```

This guarantees that the reducer receives sorts the values in expected order, first by the document id and then by the term. Here is what our reducer looked like:



```
public class TfIDFReducer extends Reducer<NgramFreqKey, NgramF
req, ...> {

  public static final String TOTAL_DOCUMENTS = "TOTAL_DOCUMENT
S";
  private long totalDocuments;

  @Override
  public void reduce(NgramFreqKey key, Iterable<NgramFreq> val
ues,
          Context context) throws IOException, InterruptedExce
ption {

    String currentNgram = key.getNgram().toString();
    String currentDocId = null;

    HashMap<String, Long> termFreqs = new HashMap<>();
    long documentCount = 0;
    long termCount = 0;
    long totalFreq = 0;

    while(values.iterator().hasNext()) {
      NgramFreq ngramFreq = values.iterator().next();
      long freq = ((Number)ngramFreq.getCount().get()).longVal
ue();
      String docId = ngramFreq.getDocId().toString();

      if (!ngramFreq.getNgram().toString().equals(currentNgram
)) {
        throw new IllegalStateException("current: " + currentN
gram
                + "tthis: " + ngramFreq.getNgram().toString())
;
      }

      if (currentDocId == null || !currentDocId.equals(docId))
 {
        if(currentDocId != null){
          termFreqs.put(currentDocId, termCount);
        }
        currentDocId = docId;
        documentCount += 1;
        termCount = freq;
        totalFreq += freq;

      } else {
        termCount += freq;
        totalFreq += freq;
      }
    }
```

Results

The first two stages of this process were completed as described above in the implementation section. The purpose of the results section is to provide a summary of the outputs from each stage and discuss how they fit into the overall framework of this analysis.

Network Definition

The first step in the network definition process was to identifying all followers for the super users. The table below provides a summary of the total number of followers by super user.

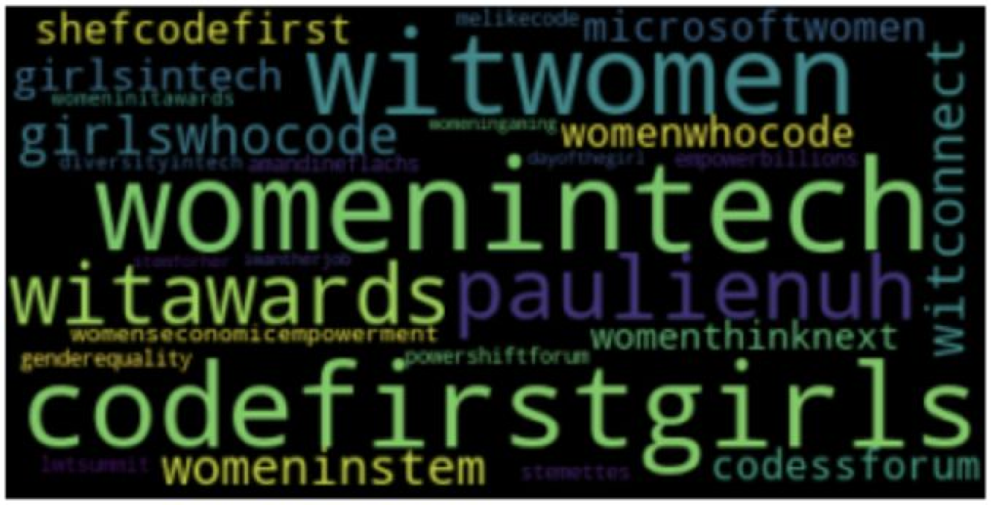
SuperUser	Number of Followers
@CodeFirstGirls	11,734
@EricaJoy	61,129
@MicrosoftWomen	38,867
@sherylsandberg	257,680
@window	16,514
@WITWomen	109,000
@WomenInTech	132,753

As mentioned in the implementation section, only users following at least 3 of the super users were included in the next phase of the project. For simplicity purposes, the table below provides the top most common set of super users followed by a follower.

SuperUser Set	Number of Followers
['WITwomen', 'ericajoy', 'microsoftwomen']	124
['WITwomen', 'microsoftwomen', 'ericajoy']	104
['ericajoy', 'WITwomen', 'microsoftwomen']	89
['microsoftwomen', 'WITwomen', 'ericajoy']	87
['microsoftwomen', 'ericajoy', 'WITwomen']	65

The twitter API was queried in order to collect all tweets from users meeting the above criteria. Through this process, the team discussed that Twitter has changed it's privacy settings such that all tweets are no longer public. Some of the users in our sample had adjusted their privacy settings such that there tweets could not be accessed via the API. A total number of 22,805 tweets were collected from the 9 users. Tweets could not be collected from 86 users setting their tweets as private.

After running TF-IDF on the text documents using Hadoop we mined important keywords which would be used in extracting more tweets. A word cloud sample of keywords is below:



Future Work

The future work proposed by this team includes completing the network analysis and developing infrastructure that allows a non-technical organization to implement the above analysis.

To monitor tweets and active users in the hot topic defined, a live streaming tweets pipeline would be created using AWS Firehose. Through the live pipeline, all real time tweets will be collected and be saved in the AWS S3 bucket directly. The main purpose is getting all tweets to see what people are discussing about in the hot topic in a certain period. The steps would help to implement this process:

- A producer device (in this case, the Twitter feed) puts data into Amazon Kinesis Firehose.
- Firehose automatically buffers the data (in this case, 5MB size or 5 minutes intervals, whichever condition is satisfied first) and delivers the data to Amazon S3.
- A Python Lambda function will be triggered when a new file is created on S3 and indexes the S3 file content to AWS Elasticsearch.
- The Kibana application runs on top of the Elasticsearch index to provide a visual display of the data.
- Use python script with boto3 package to start the pipeline engine. Then, the real time streaming tweets and the users' information will be feed in to S3 automatically.

Additionally, the outcome of the first two phases of the analysis would be a dashboard with live stream twitter data that could be used to monitor the influential users and topics. This would allow organizations to stay in the know with the individuals who are critical to the organization's efforts. Additional functionalities for the dashboard to considered could include:

Flag users who commonly use the keywords of interest but who had not previously been included in the organization's network

Create an interface directly with twitter so that users can tweet at influential individuals in from the dashboard

Given the current socio-political atmosphere in the United States it’s increasingly important that all organizations are on an equal playing field. Organizations with enough technical resources have the capabilities to drown out the voices of other important organizations. The development of this infrastructure will serve as a catalyst for trying to level the playing field.

Network analysis:

A basic network is defined by nodes and edges. In the case of this analysis, the nodes are the users and the edges are the connections between them. A number of different graphs will be developed as part of this analysis. We expect the review of graphs will be an iterative process. Areas of graph exploration are described in greater detail below.

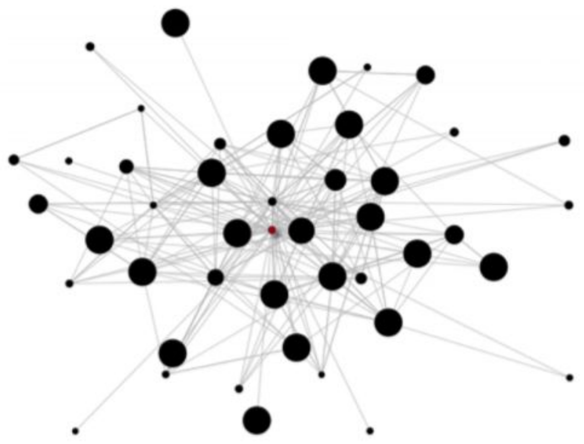
As a starting point, our team will visually analyze an undirected map of the network. Under this scenario, an edge would exist between users regardless of the nature of the connection. For example, an edge connecting Melinda Gates and another user could either be a user who follows Melinda Gates or a user who Melinda Gates follows.

While an undirected graph may be a good starting place, the twitter network is ultimately a directed graph. The reason being that relationships between nodes(users) is directional. For example, a user may follow Hillary Clinton on twitter but that does not mean that the relationship is reciprocal. Our group will explore directed graphs based on various types of twitter relationships(followers, re-tweets, etc.) Our group will also explore incorporating the relationship importance(or weight) of edges between users.

Ultimately, the goal of this analysis will be to identify key users for whom relationship development efforts could result in a boost to media awareness for our organization. These key users would include influencers and information disseminators.

Influencers are individuals who have a large number of followers within the network of interest. Given their large number of followers, these user can be considered as topic experts to community. The image below provides a graphic of a network that has been analyzed to identify influencers. The size of the node corresponds to the number of followers a given user has within the topical network. For simplification purposes, the author removed edges between nodes with a small number of connections.

Figure 1: Influencer Identification



Information disseminators are individuals who act as links between communities. This can be measured by calculating the betweenness centrality of a node. The betweenness of a node measures the fraction of shortest paths that pass through a given node. This will identify nodes that are represent single point failure. Therefore, without them information would not be passed between subgroups. Our group intends to conduct additional research regarding how to implement a page rank approach to calculate node centrality.

The method that we plan to use to implement the network analysis lies on the same lines of a SNAP(Stanford Large Network Dataset Collection) network's twitter data analysis [2].
The steps are:

1. Collect users using 'Snowball Sampling' technique: Snowball sampling uses a small pool of initial informants to nominate, through their social networks, other participants who meet the eligibility criteria and could potentially contribute to a specific study. The term "snowball sampling" reflects an analogy to a snowball increasing in size as it rolls downhill
2. Process the collected Twitter data to generate an output file of relationships between twitter accounts, generate an edge list. That is a list of relationships between twitter

accounts. A weight value is included, this value is the total number of followers for the first twitter account. The weight value can be used later to prune the network graph.

3. Visualize network data in a network graph using the NetworkX library:
- Create a directed graph (net.DiGraph) containing all the edge data including metadata.
- Remove nodes based on how connected they are to other nodes in the network (i.e. remove poorly connected nodes)

REFERENCES

1. https://en.wikipedia.org/wiki/Tf–idf
2. https://snap.stanford.edu/data/index.html#twitter
3. https://aws.amazon.com/neptune/
4. https://blogs.sequoiainc.com/twitter-fire-hoses-and-congress-oh-my/
5. https://aws.amazon.com/blogs/compute/parallel-processing-in-python-with-aws-lambda/
6. http://cs.boisestate.edu/~amit/teaching/430/handouts/ir-handout.pdf
7. https://en.wikipedia.org/wiki/Social_network_analysis\
8. https://aws.amazon.com/blogs/big-data/building-a-near-real-time-discovery-platform-with-aws/