

Fall 2011

## Clustering of twitter technology tweets and the impact of stopwords on clusters

Surya Bhagvat  
*San Jose State University*

Follow this and additional works at: [https://scholarworks.sjsu.edu/etd\\_projects](https://scholarworks.sjsu.edu/etd_projects)



Part of the [Computer Sciences Commons](#)

---

### Recommended Citation

Bhagvat, Surya, "Clustering of twitter technology tweets and the impact of stopwords on clusters" (2011). *Master's Projects*. 323.

DOI: <https://doi.org/10.31979/etd.xyck-7nnx>  
[https://scholarworks.sjsu.edu/etd\\_projects/323](https://scholarworks.sjsu.edu/etd_projects/323)

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact [scholarworks@sjsu.edu](mailto:scholarworks@sjsu.edu).

# **Clustering of twitter technology tweets and the impact of stopwords on clusters**

A Writing Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By Surya

Bhagvat

2011

© 2011

Surya Bhagvat

ALL RIGHTS RESERVED

# Clustering of twitter technology tweets and the impact of stopwords on clusters

By  
Surya  
Bhagvat

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Teng Moh, Department of Computer Science

Dr. Mark Stamp, Department of Computer Science

Dr. Robert Chun, Department of Computer Science

## ABSTRACT

### Clustering of twitter technology tweets and the impact of stopwords on clusters

Year of 2010 could be termed as the year in which Twitter became completely mainstream. Twitter, which started as a means of communicating with friends, became much more than its beginning. Now Twitter is used by companies to promote their new products, used by movie industry to promote movies. A lot of advertising and branding is now tied to Twitter and most importantly any breaking news that happens, the first place one goes and tries to find is to search it on Twitter. Be it the Mumbai attacks that happened in 2008, or the minor earthquakes that happened in Bay Area in 2010 or the twitter revolution cause of the Iran elections, most of the tech and not so tech savvy viewers were following twitter rather than any main stream news channels. In fact most of the breaking news now comes on Twitter because of the huge number of user base rather than the traditional mainstream media.

The focus of this paper is clustering with the TF-IDF weighted mechanism of daily technology news tweets of prominent bloggers and news sites using Apache Mahout and to evaluate the effects of introducing and removing stop words on the quality of clustering. This project restricts itself to only tweets in the English language.

## **ACKNOWLEDGEMENTS**

My sincere thanks to Dr. Teng Moh for guiding me throughout the entire project and providing suggestions which helped me immensely for completing the project. Also my thanks to committee members Dr. Mark Stamp and Dr. Robert Chun.

## Table of Contents

1) Introduction	8
2) Related Work	9
3) Methodology for gathering and scrubbing the data	10
3a. Gathering the data	11
Identify Twitter Sources	11
Programmatically collect tweets	13
Twitter libraries/SDK	14
Collecting the Tweets	14
Challenges with using the Twitter API	15
Using Amazon Cloud for collecting the Tweets	16
3b. Scrubbing the data	18
3c. Preparing the data	19
4) Background on TF-IDF	20
5) Apache Mahout	21
6) Running the Clustering	
6a. Convert the input tweets to a sequence file	22
6b. Convert the sequence file to a vector	23
6c. Custom analyzer for stop words	25
6d. Running the clustering	26
7) Reading the cluster output	29
8) Evaluating the cluster quality	32
9) Conclusion	36
10) References	36

## List of Figures

Figure 1	Overall Process	10
Figure 2	Process involved till data is gathered	11
Figure 3	Different phases in gathering the data	11
Figure 4	Amazon EC2	15
Figure 5	Amazon cloud infrastructure	17
Figure 6	Process involved till data is scrubbed	18
Figure 7	Different phases in scrubbing the data	18
Figure 8	Preparing the data	20
Figure 9	Various steps in clustering	21
Figure 10	Sequence file format	22
Figure 11	Command to generate the sequence file	23
Figure 12	Directory structure after generating the vectors	24
Figure 13	Command to generate the vectors	25
Figures 14a-14d	Kmeans representation	27-28
Figure 15	Command to run the kmeans clustering	29
Figure 16	Command to run the cluster dumper	31
Figure 17	Database table structure	32



## 1) Introduction

Twitter initially started as a service where users who were friends or have common interests could share things together. The idea was very similar to group Short Message Service. However, within a year or so after twitter's launch it became very popular. The popularity of the twitter can be gauged by the fact that even the White House held a town hall meeting on July 6th, 2011 in which users had a chance to ask questions using the hashtag #AskObama[1]. Even our own SJSU has an official twitter account @SJSU [2]. Twitter uses hashtag mechanism as in the case of #AskObama so that users can simply search using the hashtag [6].

The impact of twitter on technology-related news and blogs is also tremendous. For example when it was found recently that CarrierIQ was installing root kit software on the mobile android and ios devices, the technology blogs that received most hits were those with Twitter accounts, specifically @Gizmodo[3], @Techcrunch[4], and @engadget[5]. These technology blogs would naturally post news on their respective blogs, but to entice readers and to reach a wider audience, the first place they posted news would be on their Twitter accounts.

Twitter, when it was first launched was given the moniker microblogging site and the reason being till that point all the blogs were one to two pages long but in the case of twitter that maximum you can post is 140 characters. The challenge would then be to convey the message in the most succinct manner as possible. To give the brief introduction, to tweet(post) something on twitter you create a twitter account on [www.twitter.com](http://www.twitter.com) and then to follow the other twitter users for example @BarackObama, @Gizmodo or @SJSU you simply use the follow button on the twitter website.

Being a technology and open source enthusiast and who uses twitter in a big way on a daily basis, for me the one thing that is of interest is to cluster the tweets to get related but different perspectives on the technology. For example if we take the case of HP touchpad fire sale we may also be interested in the tweets that have similar kind of fire sales going on in the electronics department, if we are following the CarrierIQ root kit fiasco we may be also interested in the tweets that have the security implications.

The related works done in this area and which serve as a reference point for this project include Tweets mining using WIKIPEDIA and impurity cluster measurement by Qing Chen et al.[8], Breaking News Detection and Tracking in Twitter by Swit Phuvipadawat et al.[7] and Using twitter to recommend real-time topical news by Owen Phelan et al.[9]. This project used the related work done by Qing Chen et al. [8] as a baseline and compared the results.

Clustering of the documents usually implies removing the stop words from the documents and then running the clustering algorithms on it. One cannot apply the same concept when it comes to the tweets, cause of the fact that each tweet is limited to 140 characters and applying the full set of English stopwords would lose the essence of the tweet completely resulting in meaningless clusters. Therefore, in this project there are two important things we need to consider, first to choose only the subset of the standard English stop words and second identifying the custom set of stop words applicable to our input set of tweets. The project used the open source Apache Mahout [12] as a technology platform for machine learning algorithms. To evaluate the quality of clustering, we compared the TF-IDF weights with our custom identified stop words.

The project also identified the API limitations with regards to Twitters API [13] and how this project used Amazon Cloud services [14] to overcome it. Also, Apache Mahout being relatively new, some of the features such as calculation of cluster quality and reading of clustered data were not user friendly. The paper discusses the extensions we wrote for reading the cluster data in a much more intuitive manner.

## **2) Related Work**

Mining of Twitter data is relatively complex since tweets are free-flowing text and are limited to 140 characters each [16]. One of the interesting approaches taken by Qing Chen et al. [8] in their paper was to use Wikipedia [15] as the search engine.

Qing Chen et al. proposed to use Wikipedia search by identifying the important words in a tweet, which the authors called, “the trend,” and then do a Wikipedia search based on the trend name [8]. When a search is done with Wikipedia, it could potentially return hundreds of records. The top five to six search records are then selected, and the short snippet in the search result is added to the tweet, which is then used as an input record. We deviated from this approach by using random thousands of tweets, running them against the clustering algorithms and evaluating the TF-IDF weights of the top terms. In addition, we used custom stop words. Swit Phuvipadawat

et al. used the hashtag #breakingnews and collected all the tweets with that hashtag [7]. Once the tweets were collected, they used Stanford Named Entity Recognizer to classify the keywords into proper nouns. Once this step was completed, similar tweets were grouped together with a number and a group label so that every message would belong to a particular group [7].

### 3) Methodology

There were three main phases with respect to data collection: gathering the data, scrubbing the data and preparing the data. Once the data was available then the next two phases would be to apply the clustering algorithms and then evaluate the results. The overview of the process is shown in Figure 1.

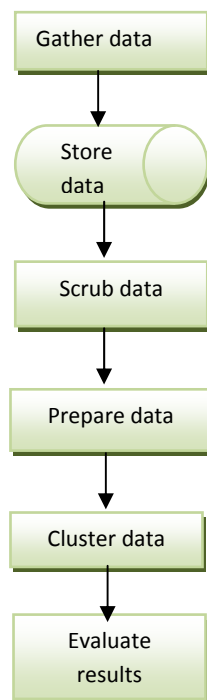


Fig 1.Overall Process

#### 3a) Gathering the data

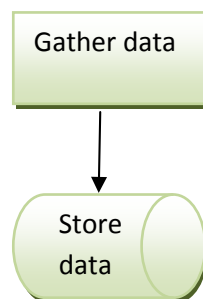


Fig 2. Process involved till data is gathered

The data gathering phase can be further divided into three steps: identifying the Twitter sources, programmatically collecting tweets and storing the data into the local disk. The overview of the process is shown in Figure 3.

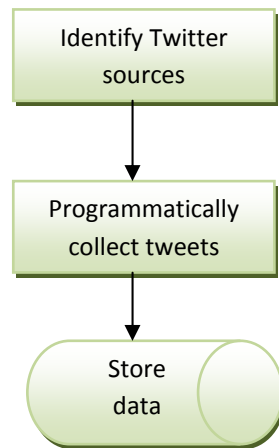


Fig 3. Different phases in gathering the data

#### Identify Twitter sources

Data gathering plays a major role in any project especially the one which involves machine learning and this project is no different. For this project, the dataset would be the technology tweets. The first step in this process is to establish the prominent news sources/bloggers from whom we would collect the tweets on a daily basis.

The sources that we chose include popular bloggers who have a large number of followers on twitter, well known news sites, and news blogs, Apple and Microsoft related sites cause of their popularity among consumers and blogs, security related blogs and deals sites. This ensures that we have as much diversification as possible in our tweets. The sources, the grouping and the additional information is provided in the t

Category	Twitter id	Details
Individual	KevinRose	KevinRose was the founder of Digg a popular crowd sourcing site, he has lot of inside information on the technology and has now started Milk a mobile application development company.
Individual	Jason	Jason Calacanis is an entrepreneur and blogger and has sold his company weblogs to AOL and is now the founder of Mahalo and the podcast This Week

		In Startups.
Individual	LeoLaporte	LeoLaporte was involved in technology related broadcasting shows right from early 90's. He currently runs TwiT.tv an online network which broadcasts shows on a daily basis.
News site	Wired	Covers technology news and how it impacts the culture, economy and politics.
News site	Cnet	Needs no introduction, if you want to know all about the technology then Cnet is the place to be.
News site	NyTimesTech	The technology stories of New York times
Gadget blog	Engadget	Covers the news about all the latest cool gadgets.
News blog	ArsTechnica	Covers the news and also publishes reviews, guides on the various technologies.
Controversial blog	Gizmodo	Controversial website which publishes all about digital culture and gadget news. They were the ones who caused a major furor when they leaked the lost iphone4 story.
Technology blog	GigaOM	Started by Om Malik, this covers the web 2.0 news as well as the related technology news.
Startup/Technology blog	Techcrunch	Needs no introduction, started by Michael Arrington this blog essentially covers all about technology startups. This is currently owned by AOL.
Social media news	Mashable	This site basically covers social media news.
Technology blog	ReadWriteWeb	Yet another technology blog.
Apple fan boys	MacWorld	The name itself suggests what it's all about. Covers anything and everything related to Apple and its products.
Everything Microsoft	EverythingMS	If Apple has MacWorld then Microsoft has EverythingMS.
Emerging tech	Techreview	Owned and published by MIT, covers emerging technologies and their impact.

## CS298 Report

Gadgets	Techland	Owned by TIME, mainly covers gadgets, about web in general.
Tech news from India	tech2eets	They publish tweets about all the technology news happening in India and around the world.
Nonprofit	TechSoup	Their goal is to help other non-profits and with the technology resources.
Security related	SecurityNews	Provides all the latest security news regarding hacking, malware, technology and so on.
Security related	Itsecnews	Very similar so security news.
Deals	Woot	Very popular one day one deal web site.
Deals	Fatwalletdeals	Twitter feed of the user contributed fat wallet deals.
Mobile(ios) app store	148Apps	Reviews all about ios apps, the name comes from the fact that when iphone was initially launched you could only have 148 apps on your iphone.
Mobile(ios) app store	Appcraver	One more ios app review site.

### Programmatically collect tweets

We used Twitter API as an asynchronous means of collecting tweets. One uses Twitter API by creating a developer account on their website, then submitting an application. Once approved, one can use Twitter API [13] to collect tweets. The challenge with using any external API is to understand its limitations with regards to the number of request per hour that an application can make. Twitter API has very strict restriction limits of 150 requests per hour for unauthenticated calls and 350 requests per hour for oauth authenticated calls [13].

As we can see the number is very low and once a developer id hits the limit, there is a chance that the application would be blacklisted and you would not be able to make the calls. Twitter provides REST API to make the calls or you can use open source Java libraries for using the Twitter API's.

### Twitter libraries/SDK

As mentioned in the Twitter developer documentation site [13], these are the available Twitter libraries for Java

- I. Scribe by Pablo Fernandez – an OAuth library
- II. Twitter4J by Yusuke Yamamoto – a Twitter API library (Java platform > v1.4.2, Android and GAE ready)

III. Twitter API ME by ernandesmj – a Twitter API library (xAuth only)

Choosing the Twitter library

For the project, the library we choose to use was Twitter4J. The reason that we choose this over the other two libraries is constant updates to the library and having much more robust api mechanism over the other two.

Collecting the Tweets

We need an automated mechanism where the program asynchronously collected Tweets from our chosen prominent bloggers and news sites on a daily basis. We use Twitter4j library and write a program called UserStream.java which collects the tweets on a daily basis. The algorithm for gathering the tweets is as follows

- I. **Algorithm 1.** Collecting Tweets Using the Twitter4J Library  
Instantiate the TwitterStream class
  - II. Implement the StatusListener class and the following methods  
**onStatus** – Whenever the change of status occurs this method will get triggered.  
**onDeletionNotice** – Called upon deletionNotice notices [17]  
**onTrackLimitationNotice** – This notice will be sent each time a limited stream becomes unlimited. If this number is high and or rapidly increasing, it is an indication that your predicate is too broad, and you should consider a predicate with higher selectivity.[17]  
**onScrubGeo** – Called upon location deletion messages [17].  
**onException** – When the Exception object is thrown.
  - III. Add the listener
  - IV. Call the filter method of the TwitterStream passing in the id's of the Twitter sources
- 

Challenges with using the Twitter API

As mentioned before Twitter is very strict with regards to the usage of its API and has very strict restriction limits of **150** requests per hour for **unauthenticated** calls and **350** requests per hour for oauth **authenticated** calls. Initial setup of collecting the tweets included running the above Java program UserStream.java from my home computer which asynchronously collected the data.

The program ran without any issues for couple of days and then ran into the Twitter API limit. Once Twitter recognizes your userid and the Oauth credentials along with the IP address as the cause of the rate limit problem, it blocks it immediately. So definitely using the twenty-five

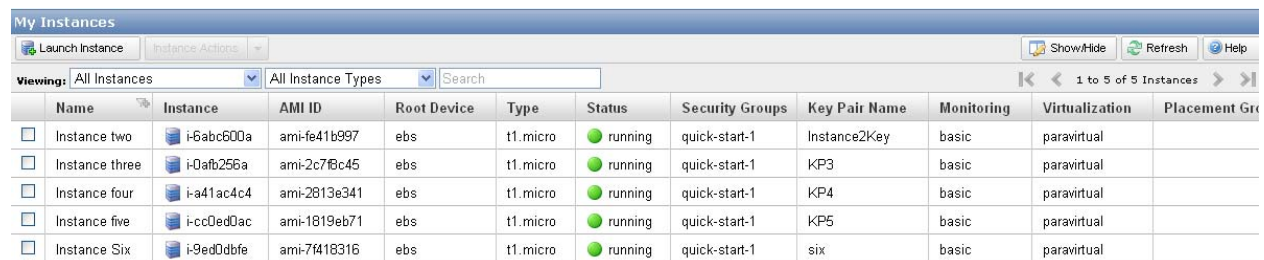
### CS298 Report

different resources and collecting the Twitter data over one home computer was not going to work. The only alternative then was to use the cloud based services like Rackspace or Amazon and use the virtual machines to collect the Tweets asynchronously on daily basis.

The choice of Amazon versus Rackspace basically boiled down to the cost of using the servers. Rackspace had a starting flat fee of \$10.95/month or 1.5 cents/hr [18] compared to Amazon of \$0.02 per hour for micro instances[19]. We chose Amazon EC2 for the project.

### Using Amazon Cloud for collecting the Tweets

We used Amazon EC2 Cloud's services [19] to overcome Twitter API's limitations by using five virtual machines and dividing the Twitter sources into five sources for each virtual machine.



The screenshot shows the 'My Instances' page in the Amazon Management Console. It displays a table with five instances, all in a 'running' state. The instances are named 'Instance two' through 'Instance six'. Each instance is a t1.micro type, using an ebs root device, and is part of the 'quick-start-1' security group. The key pairs are 'Instance2Key', 'KP3', 'KP4', 'KP5', and 'six' respectively. The monitoring is set to 'basic' and the virtualization is 'paravirtual'.

Name	Instance	AMI ID	Root Device	Type	Status	Security Groups	Key Pair Name	Monitoring	Virtualization	Placement Gr
Instance two	i-6abc600a	ami-fe41b997	ebs	t1.micro	running	quick-start-1	Instance2Key	basic	paravirtual	
Instance three	i-0afb256a	ami-2c78c45	ebs	t1.micro	running	quick-start-1	KP3	basic	paravirtual	
Instance four	i-a41ac4c4	ami-2813e341	ebs	t1.micro	running	quick-start-1	KP4	basic	paravirtual	
Instance five	i-cc0ed0ac	ami-1819eb71	ebs	t1.micro	running	quick-start-1	KP5	basic	paravirtual	
Instance six	i-9ed0dbfe	ami-7418316	ebs	t1.micro	running	quick-start-1	six	basic	paravirtual	

Fig 4. Amazon EC2 instances

Each of the virtual machine is using its unique twitter account, so for this project five separate Twitter accounts were used and each was running the same copy of UserStream. Twitter API also expects an application to be associated to a twitter account. The following is the account information for each box.

Box name	Associated twitter account	Application name
ec2-107-20-94-65.compute-1.amazonaws.com	surya1902	TestApplicationST2
ec2-50-17-54-102.compute-1.amazonaws.com	Suryab	Tweets Clustering
ec2-107-20-94-171.compute-1.amazonaws.com	Suryasignup	Tweets Clustering 1
ec2-184-73-120-69.compute-1.amazonaws.com	suryab_sjsu	Clustering of Tweets



The twenty-five twitter sources have been broken into five sources for each individual box and the data is asynchronously collected and it is pulled every week into the local disk as shown in the figure below

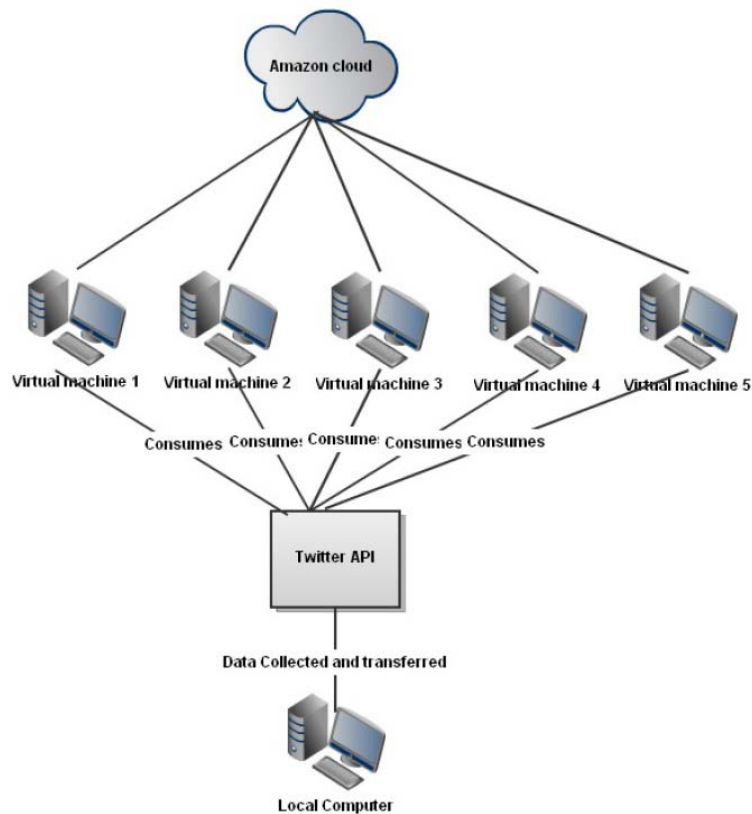


Fig 5. Amazon cloud infrastructure

After the above steps and copying the files over to the local computer on a weekly basis we would have the consolidated tweets for each box the file would look something like this

```

@chicpen75 - RT @mashable: The Last Words of Steve Jobs - http://t.co/OEgwodvB
@dmmapostolou - RT @mashable: The Last Words of Steve Jobs - http://t.co/OEgwodvB
@ShebaFamily - RT @mashable: 10 Creepy iPhone Photos for Halloween - http://t.co/mOnNIZfU
@sta2999 - @mashable always tweets right after me. i'm going to start reading all of it.
.....
  
```

### 3b) Scrubbing the data

Gather data

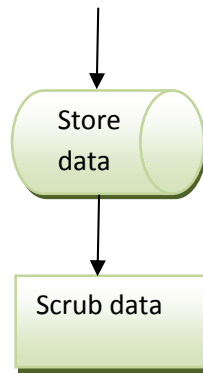


Fig 6. Process involved till data is scrubbed

The data scrubbing phase can be further divided into the following steps as shown in the figure below

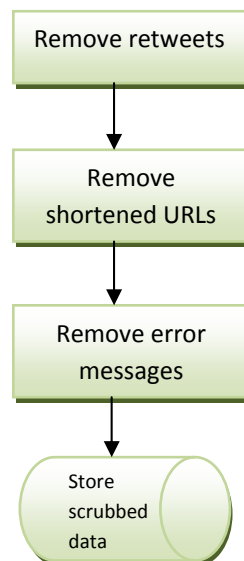


Fig 7. Different phases in scrubbing the data

Twitter provides the streaming API for near real-time access to the twitter data [13]. For our project we will be using the User streams which provides the information about the user like his direct messages, mentions, followers and so on. We are more interested in getting the direct messages of a particular user or source. The streaming API needs the oauth credentials of the twitter account and we must provide them.

Since we are using the Twitter4j and we are interested in the user resources we need to make use of the twitter4j.StatusListener and twitter4j.TwitterStream classes. When we make use of the status listener we not only obtain the tweets made by a particular source but we also end up

having the retweets as part of the stream. In this project we don't necessarily make use of the retweets, so we filter out the retweets. Also most of the tweets use URL shorteners at the end of

### CS298 Report

the tweets which actually point to the original link of the article, this would affect the clustering and as such we are going to strip away the shortened URL's like <http://bit.ly>. The algorithm for scrubbing the tweets is as follows

- i. Go through each Tweet and if the Tweet begins with @<source> filter it out.
- ii. Strip out the shortened URL's from the Tweets.
- iii. Write all the filtered tweets to a separate file.

As mentioned above even after breaking the sources into five different sources and having five Amazon based virtual machines in the cloud we would still run into issues with the API limit and the streamed source file would end up having all the information related to the connection as shown in below

```
Stream closed.Relevant discussions can be on the Internet at:
http://www.google.co.jp/search?q=70971d2e or
http://www.google.co.jp/search?q=000687bf
TwitterException{exceptionCode=[70971d2e-000687bf 70971d2e-0006876e], statusCode=-1,
retryAfter=-1, rateLimitStatus=null, featureSpecificRateLimitStatus=null, version=2.2.5-
SNAPSHOT(build: daa2ef4273f09c48cfedddf8317ce12d6f22997b)}
at
twitter4j.AbstractStreamImplementation.handleNextElement(AbstractStreamImplementation.java
:167)
at twitter4j.StatusStreamImpl.next(StatusStreamImpl.java:67)
at twitter4j.TwitterStreamImpl$TwitterStreamConsumer.run(TwitterStreamImpl.java:443)
Caused by: java.io.IOException: Premature EOF
    at
    sun.net.www.http.ChunkedInputStream.readAheadBlocking(ChunkedInputStream.java:53
8)
    at sun.net.www.http.ChunkedInputStream.readAhead(ChunkedInputStream.java:582)
    at sun.net.www.http.ChunkedInputStream.read(ChunkedInputStream.java:669)
    at java.io.FilterInputStream.read(FilterInputStream.java:116)
```

We need to clean up our tweets file by removing the irrelevant information and since the error message is not standardized we do this step manually.

### 3c) Preparing the data

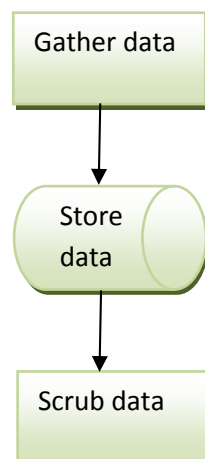




Fig 8. Process involved till data is prepared

Apache mahout expects each input of the text to be in its own separate file. When we used the Twitter streaming API and scrubbed the data, we ended with one file with all the tweets (retweets removed). We needed to make sure that each tweet inside the file was contained in a file of its own.

- i. **Algorithm 2.** Preparing the Input Data Take the filtered tweet file as the input.
  - ii. Read each line and effectively write it to a separate file, the file name begins with 'Twitter\_' count of the file.
  - iii. This would effectively serve as the input to the clustering algorithms.
- 

#### 4) Background on TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. Term Frequency represents the importance of a term in a specific document whereas Inverse Document Frequency represents the importance of a term relative to the entire corpus. Thus, under TF-IDF the stop words had smaller weight since they appeared frequently in the documents, and the non-stop words or the words that gave the gist of the topic had greater weight. One of the goals of the project was to see the effect of using the correct stop word.

#### 5) Apache Mahout

For this project we would be using Apache Mahout which has scalable machine learning libraries including the clustering algorithms [12]. Mahout is very new but has become quite popular as an open source library for mining of large data sets.

Some of the challenges with using a technology as new as Mahout is the lack of documentation. To understand the usage of the libraries going through the source code was the only option. Recently Manning on Oct 14<sup>th</sup>, 2011 released a book called Mahout in Action which was referenced in the project.

## 6) Running the clustering

Quite a lot of understanding and prep work needs to be done before running the actual clustering algorithms. The sequence is outlined and shown in the figure below

- i. Convert the input tweets to a sequence file.
- ii. Converting the sequence file to a vector.
- iii. Run the analyzer for the stop words.
- iv. Running the actual clustering algorithm.

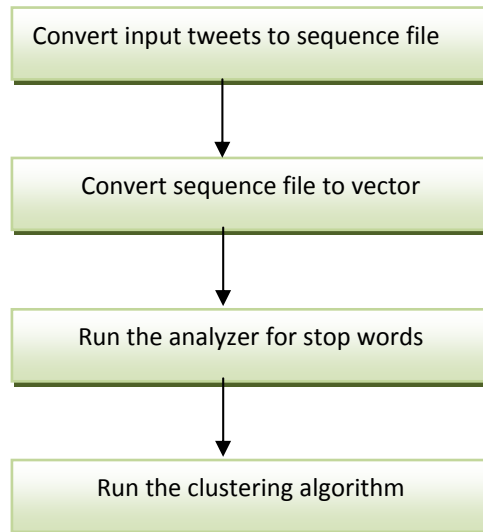


Fig 9. Various steps in clustering

6a & 6b) Convert the input tweets to a sequence file and then vector

Converting the input tweets into a vector would be the first step in clustering process. When we are trying to cluster text documents then we essentially represent document as a vector. We can think of a vector as something that contains the words as well as number of times each word occurs in the document. In Apache Mahout we need to first convert the input tweets into sequence file format and then from the sequence file format you create vectors.

There are two steps involved when using Apache Mahout in the generation of the vectors. The first step would be to generate the sequence file and the next would be to generate the vectors passing the sequence file as the input.

### Step 1: Generate the Sequence File

First run the following command to generate the sequence file in the chunked manner [20]

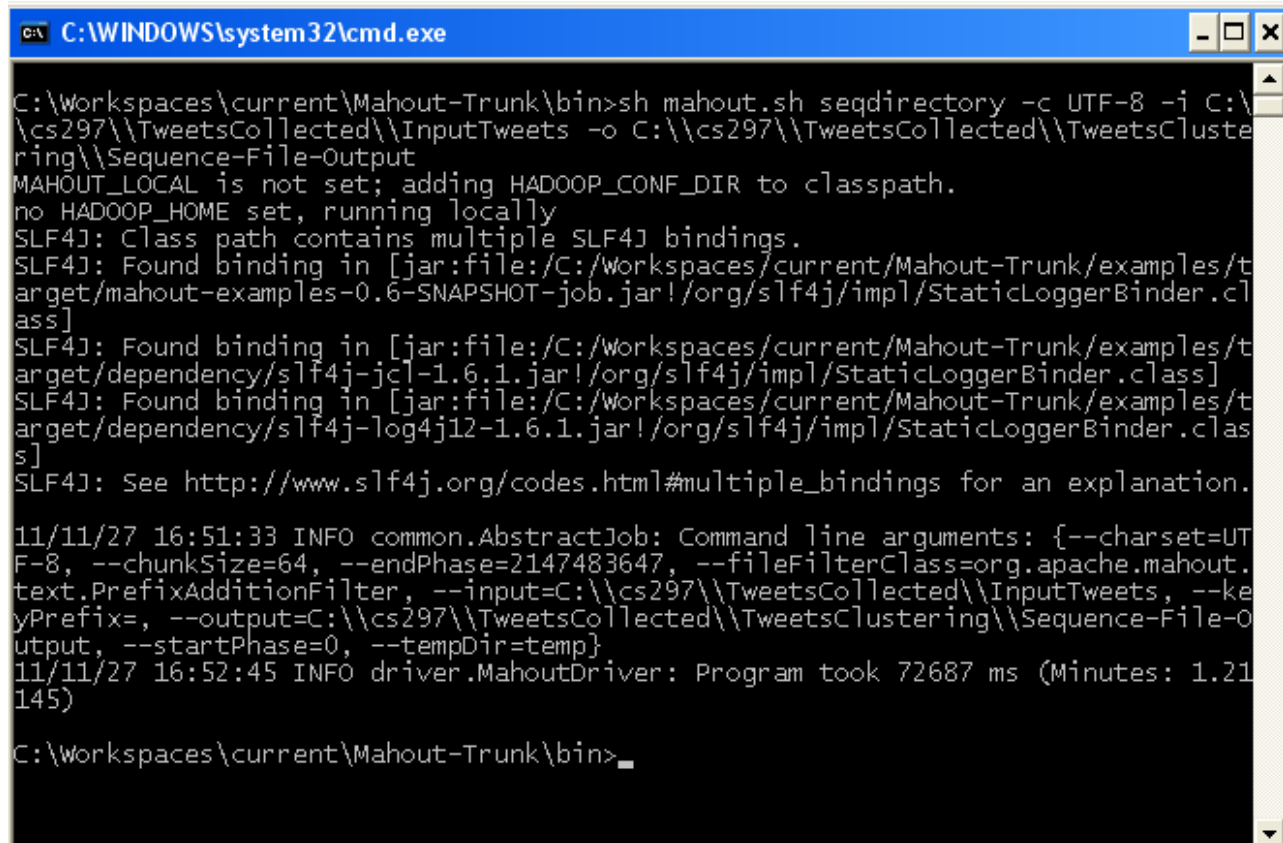
```
$MAHOUT_HOME/bin/mahout seqdirectory \--input <PARENT DIR WHERE DOCS ARE LOCATED> --output
<OUTPUT DIRECTORY> \-c <CHARSET NAME OF THE INPUT DOCUMENTS> {UTF-8|cp1252|ascii...}> \-
chunk <MAX SIZE OF EACH CHUNK in Megabytes> 64> \-prefix <PREFIX TO ADD TO THE DOCUMENT ID>>
```

The sequence file can be considered as sort of an intermediary file which essentially has the document id and the content of the text, in our case the tweet. It's not an ascii text file, so the content shown here is essentially a screen shot of how the sequence file looks like

```
SXU78Z9org.apache.hadoop.10.Text2org.apache.hadoop.10.Text#UUUUUUUUUUUUUUUUUUUU.gje00uo  
5&□"!..UUUUUUUU<UUUUUUUU  
F3/Tweet_1.txt&The Tide to Go Pen Will Get You Laid  
UUUUUUUUUVUUUUUUUSO  
/Tweet_10.txtGSperry shoes B1G1 50% off + 20% off + 10% cashback famousfootwear.com  
UUUUUUUUU1UUUUUUUSOSUO/Tweet_100.txtOCZ Vertex Plus OCZSSD2-1VTXP1L20G 2.5" 120GB SATA II MLC Internal Solid State Drive (S...  
UUUUUUUUUUUUUUUUUUUUDESUO/Tweet_1000.txt=RT @mashable: COMIC: Facebook Changes Getting out of Hand -
```

Fig 10. Sequence file format

The screen shot to generate the sequence file is shown below



```

C:\WINDOWS\system32\cmd.exe

C:\Workspaces\current\Mahout-Trunk\bin>sh mahout.sh seqdirectory -c UTF-8 -i C:\
\cs297\TweetsCollected\InputTweets -o C:\cs297\TweetsCollected\TweetsCluste
ring\Sequence-File-Output
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
no HADOOP_HOME set, running locally
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/t
arget/mahout-examples-0.6-SNAPSHOT-job.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/t
arget/dependency/slf4j-jcl-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/t
arget/dependency/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.clas
s]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.

11/11/27 16:51:33 INFO common.AbstractJob: Command line arguments: {--charset=UT
F-8, --chunkSize=64, --endPhase=2147483647, --fileFilterClass=org.apache.mahout.
text.PrefixAdditionFilter, --input=C:\\cs297\\TweetsCollected\\InputTweets, --ke
yPrefix=, --output=C:\\cs297\\TweetsCollected\\TweetsClustering\\Sequence-File-0
utput, --startPhase=0, --tempDir=temp}
11/11/27 16:52:45 INFO driver.MahoutDriver: Program took 72687 ms (Minutes: 1.21
145)

C:\Workspaces\current\Mahout-Trunk\bin>

```

Fig 11. Command to generate the sequence file

*Step 2: Generate the vectors from the Sequence File*

When one tries to cluster text documents, one represents each document as a vector. One can think of a vector as an entity that contains words as well as number of times each word occurs in the document. Mahout provides three different classes for vectors namely **DenseVector**, **RandomAccessSparseVector** and **SequentialAccessSparseVector**. From the Java API [21] the following are the users of the three classes

**DenseVector** – Implements the vector as an array of doubles

**RandomAccessSparseVector** – This vector only stores non-zero doubles

**SequentialAccessSparseVector** – Similar to **RandomAccessSparseVector** wherein it stores non-zero doubles and the elements would be accessed in a sequential fashion.

In our project we are going to use **SequentialAccessSparseVector**. The class to generate

**SequentialAccessSparseVector** from the sequential file is

```

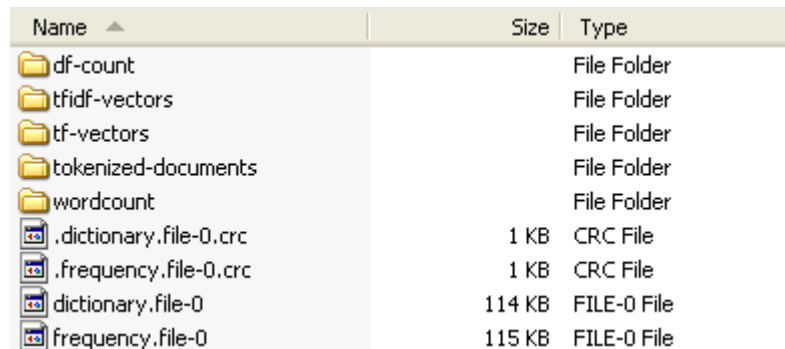
java org.apache.mahout.vectorizer.SparseVectorsFromSequenceFiles --input <SequenceFile> --output
<Output folder> --namedVector --minDF <int> --maxDFPercent <int> --weight TFIDF --analyzerName
<CustomAnalyzerClass>

```

The required and important parameters are described below

Parameter Name	Description
input	Location of the sequence file.
output	Location of the output folder where the sparse vectors will be written.
namedVector	Output vectors should be NamedVectors.
minDF	The minimum document frequency, the minimum number of documents the term should occur. If it is less than the provided value then it won't be the part of the dictionary.
maxDFPercent	The maximum document frequency, if the term occurs in more than the specified value then it is ignored.
Weight	In this case we use TF-IDF, we can also just use TF.
analyzerName	This should be a Java class that extends Lucene's Analyzer where we would define our own custom stop words.

Once the above command is run the following would be the output



Name	Size	Type
df-count		File Folder
tfidf-vectors		File Folder
tf-vectors		File Folder
tokenized-documents		File Folder
wordcount		File Folder
.dictionary.file-0.crc	1 KB	CRC File
.frequency.file-0.crc	1 KB	CRC File
dictionary.file-0	114 KB	FILE-0 File
frequency.file-0	115 KB	FILE-0 File

Fig 12. Directory structure after generating the vectors

The dictionary file contains the term to its ID, the tokenized-document folder is where the custom analyzer if specified is applied or the standard analyzer is used and the words are tokenized. The tfidf-vectors folder contains the actual vectors of the documents.

The screen shot to generate the vectors is shown below



```

C:\WINDOWS\system32\cmd.exe - java org.apache.mahout.vectorizer.SparseVectorsFromS...
C:\Workspaces\current\Mahout-Trunk\bin>java org.apache.mahout.vectorizer.SparseV
ectorsFromSequenceFiles --input C:\\cs297\\TweetsCollected\\TweetsClustering\\Se
quence-File-Output\\ --output C:\\cs297\\TweetsCollected\\TweetsClustering\\Spar
se-Vectors --namedVector --minDF 2 --maxDFPercent 50 --weight TFIDF --analyzerNa
me TweetsAnalyzer
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/t
arget/mahout-examples-0.6-SNAPSHOT-job.jar!/org/slf4j/impl/StaticLoggerBinder.cl
ass]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/t
arget/dependency/slf4j-jcl-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/t
arget/dependency/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.clas
s]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
11/11/27 16:56:12 INFO vectorizer.SparseVectorsFromSequenceFiles: Maximum n-gram
size is: 1
11/11/27 16:56:12 INFO vectorizer.SparseVectorsFromSequenceFiles: Minimum LLR va
lue: 1.0
11/11/27 16:56:12 INFO vectorizer.SparseVectorsFromSequenceFiles: Number of redu
ce tasks: 1
11/11/27 16:56:13 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName
=JobTracker, sessionId=
11/11/27 16:56:13 INFO input.FileInputFormat: Total input paths to process : 1
11/11/27 16:56:14 INFO mapred.JobClient: Running job: job_local_0001
11/11/27 16:56:14 INFO input.FileInputFormat: Total input paths to process : 1
11/11/27 16:56:15 INFO mapred.JobClient: map 0% reduce 0%

```

Fig 13. Command to generate the vectors

### 6c) Custom analyzer for the stop words

Mahout provides the class `org.apache.lucene.analysis.DefaultAnalyzer`. This class extends the `org.apache.lucene.analysis.Analyzer` class and provides a default constructor without any arguments. This is the class where we define the stop words and choosing the right set of stop words would play a crucial role in clustering.

By default Mahout's `DefaultAnalyzer` doesn't include any stop words. The goal was to compare the weightiness of the top terms in three runs; the first run with no stop words defined, the second with the chosen English set of stop words from the full set of MySQL stop words [22] and the final run with the custom stop words that we identified.

The following are the stopwords that are used for the second run i.e., regular English stopwords that we chose from the full set of MySQL stopwords [22]. Only a subset of the stopwords from the MySQL stop words was chosen. These stop words were compiled by running many rounds of the clustering algorithms and then comparing the results. If we include the full set of MySQL stopwords [22] and compared the results, the clusters that were obtained were essentially meaningless because of the 140 character twitter limit. We would like to compare the results of using the chosen regular English stopwords versus combining these with the custom

stop words that we have identified.

Regular English stopwords	"a", "able", "about", "across", "after", "all", "almost", "also", "am", "among", "an", "and", "any", "are", "as", "at", "be", "because", "been", "but", "by", "can", "can not", "could", "dear", "did", "do", "does", "either", "else", "ever", "every", "for", "from", "get", "got", "had", "has", "have", "he", "her", "hers", "him", "his", "how", "however", "i", "if", "in", "into", "is", "it", "its", "just", "least", "let", "like", "likely", "may", "me", "might", "most", "must", "my", "neither", "no", "nor", "not", "of", "off", "often", "on", "only", "or", "other", "our", "own", "rather", "said", "say", "says", "she", "should", "since", "so", "some", "than", "that", "the", "their", "them", "then", "there", "these", "they", "this", "tis", "to", "too", "twas", "us", "wants", "was", "we", "were", "what", "when", "where", "which", "while", "who", "whom", "why", "will", "with", "would", "yet"
---------------------------	---

The custom stop words in addition to the above words that are specific to the twitter world are the following. These words were obtained by collecting the tweets on a weekly basis, then running the clustering algorithms, verify whether the top terms that were generated added value to the cluster or not. If the word didn't really add value to that cluster it would be a stop word. This was a manual effort done on a weekly basis.

Custom Stop words	"you", "your", "takes", "chat", "live", "stage", "tells", "help", "become", "without", "doing", "come", "call", "needs", "find", "more", "check", "using", "before", "weekly", "found", "finds", "many", "first", "time", "others", "here", "much", "very", "during", "taking", "starts", "updates", "anything", "called", "works", "well", "haha", "comes", "five", "isnt", "keep", "really", "ways", "save", "thanks", "thank", "last", "take", "know", "want", "need", "based", "talk", "under", "getting", "gets", "looking", "back", "make", "down", "pass", "less", "next", "week", "grab", "gone", "soon", "lots", "trying", "goes", "http", "itself", "agree"
-------------------	---

The way the regular and custom stop words were identified was by collecting the tweets on a regular basis, running the clustering algorithms on the input, evaluate the impact the stop words have on the weight of the prominent terms and then include or exclude it from the set. This was a manual process which could be improved by means of an algorithmic approach.

#### 6d) Running the clustering

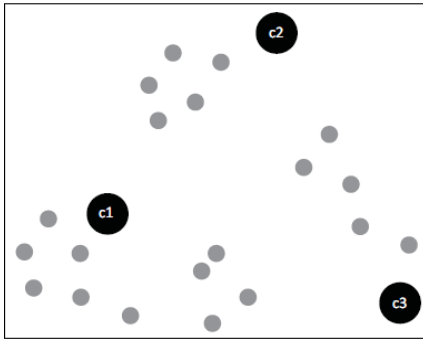
We would be using TF-IDF along with Kmeans clustering to cluster the tweets. The K means algorithm is as follows

- i. Assume there are n points and k groups in which you want to cluster these n points.
- ii. The first step would be to start with an initial set of k centroid points.
- iii. Define the max number of iterations.
- iv. The algorithm continues its processing and will keep refining the centroids until it comes to max number of iterations or the centroids converge.

### CS298 Report

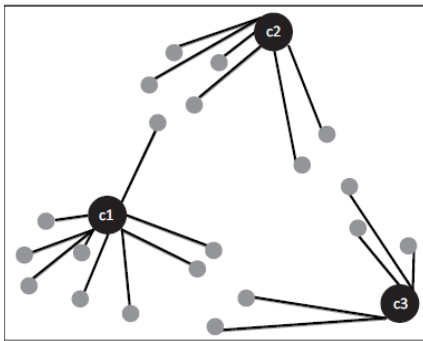
An example follows (taken from Mahout in Action book)

Assume initially we have three centroids  $c_1$ ,  $c_2$  and  $c_3$



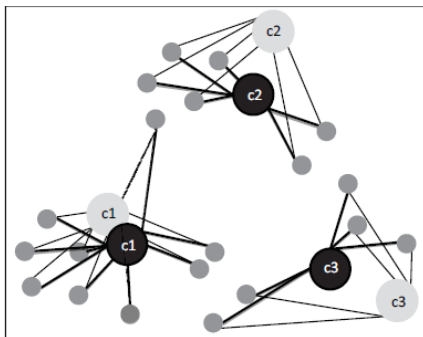
(Fig 14a)

The map stage assigns each point to the cluster nearest to it



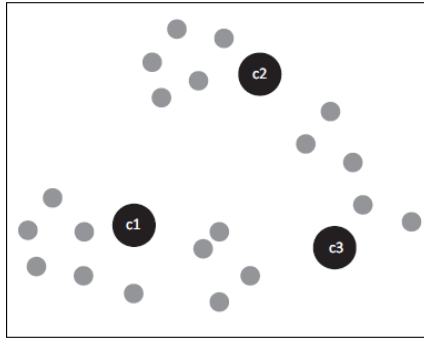
(Fig 14b)

In the reduce stage, the associated points are averaged out. This would result in the centroid location shifting as shown in the figure and produces new centroids.



(Fig 14c)

Once the iteration completes, this configuration from the above figure becomes the starting point for the loop and continues till the centroids converge to their final positions



(Fig 14d)

### Selecting the distance measure

From distance measures, such as CosineDistanceMeasure, EuclideanDistanceMeasure and TanimotoDistanceMeasure, we chose CosineDistanceMeasure after doing some trial runs.

### Command to run clustering

Mahout provides the class `org.apache.mahout.clustering.kmeans.KmeansDriver` to run the Kmeans clustering. The class takes the following parameters

```
java org.apache.mahout.clustering.kmeans.KMeansDriver -i <input location> -c <initial clusters location> -o <output location> -cd <distance> -x <iterations> -k <Number of clusters> -cl -dm <Distance measure>
```

The required and important parameters are described below

Parameter Name	Description
i	Path to the input directory.
c	Path to the initial clusters directory .
o	Output directory name.
cd	The convergence delta value.
x	The maximum number of iterations.
k	Number of clusters.
cl	Indicates that the clustering needs to be run after the iterations have taken place.
dm	The distance measure that needs to be used, in our case we use CosineDistanceMeasure.

The screen shot to run the kmeans clustering is shown below

```

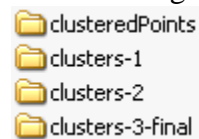
C:\WINDOWS\system32\cmd.exe
C:\Workspaces\current\Mahout-Trunk\bin>java org.apache.mahout.clustering.kmeans.KMeansDriver -i C:\\cs297\\TweetsCollected\\TweetsClustering\\Sparse-Vectors\\tfidf-vectors -c C:\\cs297\\TweetsCollected\\TweetsClustering\\initialclusters -o C:\\cs297\\TweetsCollected\\TweetsClustering\\clusters -cd 0.5 -x 25 -k 50 -cl -dm org.apache.mahout.common.distance.CosineDistanceMeasure
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/target/mahout-examples-0.6-SNAPSHOT-job.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/target/dependency/slf4j-jcl-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/target/dependency/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.

11/11/27 17:04:25 INFO common.AbstractJob: Command line arguments: {--clustering=null, --clusters=C:\\cs297\\TweetsCollected\\TweetsClustering\\initialclusters, --convergenceDelta=0.5, --distanceMeasure=org.apache.mahout.common.distance.CosineDistanceMeasure, --endPhase=2147483647, --input=C:\\cs297\\TweetsCollected\\TweetsClustering\\Sparse-Vectors\\tfidf-vectors, --maxIter=25, --method=mapreduce, --numClusters=50, --output=C:\\cs297\\TweetsCollected\\TweetsClustering\\clusters, --startPhase=0, --tempDir=temp}
11/11/27 17:04:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
11/11/27 17:04:26 INFO compress.CodecPool: Got brand-new compressor
11/11/27 17:04:26 INFO kmeans.RandomSeedGenerator: Wrote 50 vectors to C:/cs297/TweetsCollected/TweetsClustering/initialclusters/part-randomSeed

```

Fig 15. Command to run the Kmeans clustering

After running the Kmeans we get the following directory structure



This would vary depending on each run, in our case we choose the number of clusters to be 50 and the number of iterations before the clusters converge was set as 20, the clusters may converge before as is the case shown above.

## 7) Reading the cluster output

Mahout provides a utility called `org.apache.mahout.utils.clustering.ClusterDumper` which provides all the clusters information. The way to run this program is

```
java org.apache.mahout.utils.clustering.ClusterDumper -s <Sequence files directory for the clusters> -p <Directory containing point sequence files> -n <Number of top terms that we want to print> -d <Location of the dictionary file> -dt <The dictionary file type> -o <Output directory>
```

The required and important parameters are described below

Parameter Name	Description
s	This is the sequence file directory which is generated after running the Kmeans algorithm.
p	All the clustered points are stored in clusteredPoints directory, this would point the location of the directory.
n	The number of top terms (the terms with the highest weights that we want to print), in our case we are printing the top five terms.
d	This is the location of the dictionary file that was generated with the SequentialAccessSparseVector.
dt	The type of the dictionary file type whether it's sequential or text.
o	The output directory.

Once we run the above program we would get the following file in the output directory clusteroutput. If we look at the content of the clusteroutput file it's one huge file with all the clusters listed along with the weight for each term and the weights for the top five terms. A sample output is shown below (note this is just a sample, this file has lots of data)

```

CL-1(n=3576 c=[aakash:0.003, aardvark:0.004, aaron:0.005, ability:0.005, abuse:0.005, accepting:0.003, accepts:0.004, access:0.027, accessories:0.007, accident:0.003
Top Terms:
  iphone => 0.15075426587055726
  apple => 0.12412147780659481
  mobile => 0.11814728919291656
  jobs => 0.1115138864090512
  samsung => 0.1009606923032927
Weight: [props - optional]: Point:
1.0 [distance=0.9258744922524543] /Tweet_1018.txt = [deutsche:7.992, mango:5.227, monday:7.673, sets:7.673, telekom:8.125]
1.0 [distance=0.9527968879560713] /Tweet_1024.txt = [compaq:8.972, deal:5.823, ebay:6.852, notebook:7.673, refurbished:6.980]
1.0 [distance=0.9622165893364358] /Tweet_1030.txt = [creative:7.432, ecommerce:8.279, excellent:8.125, following:8.685, ideas:7.432, sites:6.893]
1.0 [distance=0.9398984734072342] /Tweet_104.txt = [brings:6.333, lovers:8.685, optinus:7.127, power:6.382, tegra:8.462]
1.0 [distance=0.8791553370303117] /Tweet_1044.txt = [camera:5.881, channel:6.704, iphone:4.768, night:6.739, system:6.200, viewing:8.685, vision:7.075]
1.0 [distance=0.8333065663730284] /Tweet_1045.txt = [games:6.221, iphone:4.768]
1.0 [distance=0.912478800989552] /Tweet_1047.txt = [breaking:8.125, cupertino:8.972, event:6.333, iphone:4.768, long:6.980, standing:8.279]

```

As we can see the cluster dumper output nicely prints out the top terms in each cluster and their weights along with all the files that form part of the cluster.

The screenshot to run the ClusterDumper utility is shown below

```

C:\Workspaces\current\Mahout-Trunk\bin>java org.apache.mahout.utils.clustering.ClusterDumper -s C:\\cs297\\TweetsCollected\\TweetsClustering\\clusters\\clusters-1 -p C:\\cs297\\TweetsCollected\\TweetsClustering\\clusters\\clusteredPoints -n 5 -d C:\\cs297\\TweetsCollected\\TweetsClustering\\Sparse-Vectors\\dictionary.file-0 -dt sequencefile -o C:\\cs297\\TweetsCollected\\TweetsClustering\\ClusterDumperOutput\\clusteroutput
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/target/mahout-examples-0.6-SNAPSHOT-job.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/target/dependency/slf4j-jcl-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/C:/Workspaces/current/Mahout-Trunk/examples/target/dependency/slf4j-log4j12-1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
11/11/27 17:18:53 INFO common.AbstractJob: Command line arguments: {--dictionary=C:\\cs297\\TweetsCollected\\TweetsClustering\\Sparse-Vectors\\dictionary.file-0, --dictionaryType=sequencefile, --distanceMeasure=org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure, --endPhase=2147483647, --numWords=5, --output=C:\\cs297\\TweetsCollected\\TweetsClustering\\ClusterDumperOutput\\clusteroutput, --outputFormat=TEXT, --pointsDir=C:\\cs297\\TweetsCollected\\TweetsClustering\\clusters\\clusteredPoints, --seqFileDir=C:\\cs297\\TweetsCollected\\TweetsClustering\\clusters\\clusters-1, --startPhase=0, --tempDir=temp}
11/11/27 17:19:04 INFO clustering.ClusterDumper: Wrote 50 clusters
C:\Workspaces\current\Mahout-Trunk\bin>

```

Fig 16. Command to run the cluster dumper

Problem with the cluster dumper output

Running the above program results in one file that contains all the cluster information and which is very difficult to read. Also, it is very difficult to associate the input tweet files in the cluster with the actual contents. To facilitate the viewing of the data in the clusters and to print the contents of the input tweets, we wrote a custom java program that employed the algorithm described below

---

**Algorithm 3.** Algorithm for Better Viewing of Clustering Data

---

- I. Opens the output file generated from ClusterDumper.
- II. Reads each line of the file and looks for the following strings CL- and VL-, this basically indicates the start of the cluster.
- III. From the position to the next CL- or VL- it reads each line, the following would be the lines Top Terms:
  - iphone => 0.15075426587055726
  - apple => 0.12412147780659481
  - mobile => 0.11814728919291656
  - jobs => 0.1115138864090512
  - samsung => 0.1009606923032927
- IV. For each cluster we create a new file and first write the top terms.

## CS298 Report

- V. The next line in the file generated from Mahout utility would be something along these lines as an example
- 1.0 : [distance=0.9258744922524543]: /Tweet\_1018.txt = [deutsche:7.992, mango:5.227, monday:7.673, sets:7.673, telekom:8.125]
- VI. The program parses this line, figures out what the file is in this case **Tweet\_1018.txt**, then opens this file and writes the contents of it.
- VII. The end result would be a file for each cluster with the top terms and the actual content of the Tweet included as shown below.

Top Terms:

changes => 4.999809711210189  
facebook => 2.0083499416228263  
reveal => 1.546768403822376  
consume => 1.0918245623188634  
mean => 0.7425660933217695

Tweet\_1000.txt RT @mashable: COMIC: Facebook Changes Getting Out of Hand -  
Tweet\_1039.txt Facebook changes getting to you? This #comic shows you how it could be worse  
Tweet\_105.txt Sears Ultra Plus Powdered Laundry Detergent 1/2 price \$11.99  
Tweet\_1082.txt Facebook Changes in a Nutshell [COMIC]

---




































## 8) Evaluating the cluster quality

To evaluate the quality of the cluster, we basically take the approach of calculating the TF-IDF weights for prominent terms with the following analyzers

- TweetsAnalyzer with no stop words.
- TweetsAnalyzer with common English stop words.
- TweetsAnalyzer with common English stop words + custom stop words.

Mahout doesn't really provide any mechanism to store the top terms in any database, we extend Mahout and provide a custom program that essentially writes the top terms to the database for each run.

The database that we use for this purpose is MySQL and the database table that we have created for this purpose is named TermsAndWeights and the structure is in the following fashion.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	RunType	varchar(200)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	Term	varchar(50)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	Weight	varchar(50)	latin1_swedish_ci		No	None		      
<input type="checkbox"/>	Field_1	varchar(50)	latin1_swedish_ci		Yes	NULL		      
<input type="checkbox"/>	Field_2	varchar(50)	latin1_swedish_ci		Yes	NULL		      

(Fig 16)

RunType – Basically describes the run type of the whole process, this could be one of NoStopWords, DefaultStopWords or CustomStopWords.



## CS298 Report

Term – The actual term.

Weight – The value of the assigned weight by that particular RunType.

Field\_1 – Additional field.

Field\_2 – Additional field.

**Algorithm 4.** Algorithm to track the top terms and insert into the databaseThe algorithm to track the top terms and insert into the database is given below

1. Opens the output file generated from ClusterDumper.
2. Reads each line of the file and looks for the following strings CL- and VL-, this basically indicates the start of the cluster.
3. Connect to the database.
4. From the position to the next CL- or VL- it reads each line, the following would be the lines

Top Terms:

```
iphone => 0.15075426587055726
apple  => 0.12412147780659481
mobile => 0.11814728919291656
jobs   => 0.1115138864090512
samsung => 0.1009606923032927
```

5. Read each top term and insert into the table along with the run type.
- 

We run the program with different analyzers

- TweetsAnalyzer with no stop words.
- TweetsAnalyzer with common English stop words.
- TweetsAnalyzer with common English stop words + custom stop words.

We then look at the top terms and their weights for each run.
















The results of the weights of the following five terms, siri, apple, iphone, augmented and with, for different runs, are shown in the tables below. These terms were chosen randomly

siri

←T→	RunType	Term	Weight	Field_1	Field_2
<input type="checkbox"/>  	DefaultStopWords	siri	1.5073364045884874	CLUSTER_22	NULL
<input type="checkbox"/>  	CustomStopWords	siri	3.5180675863362043	CLUSTER_16	NULL
<input type="checkbox"/>  	CustomStopWords	siri	0.608281407211766	CLUSTER_37	NULL

We notice for **siri** that when we didn't include any stop words it was not even part of the cluster but for default and custom stop words it carried the weights as shown above and for custom it appeared even twice.

apple

  	RunType	Term	Weight	Field_1	Field_2
  	NoStopWords	apple	0.5385988283673778	CLUSTER_38	NULL
  	DefaultStopWords	apple	0.6785155087709427	CLUSTER_20	NULL
  	DefaultStopWords	apple	1.981864897189317	CLUSTER_22	NULL
  	CustomStopWords	apple	4.620405804027211	CLUSTER_37	NULL

For the term **apple** the weight is more when we have used the CustomStopWords when compared to no stop words and default stop words




iphone

  	RunType	Term	Weight	Field_1	Field_2
  	NoStopWords	iphone	2.17898597097569	CLUSTER_38	NULL
  	DefaultStopWords	iphone	4.1664243917912245	CLUSTER_20	NULL
  	CustomStopWords	iphone	3.071035173501861	CLUSTER_12	NULL
  	CustomStopWords	iphone	0.45776439227646204	CLUSTER_16	NULL
  	CustomStopWords	iphone	0.8084317380731756	CLUSTER_37	NULL

iphone is relatively common term in the tech industry and as such for custom the value is little bit less than that of default though it appeared in more times than default.

augmented

We see the term like augmented appears only when we have the custom stop words and if we look at the cluster to which it is assigned its relevance is immediately apparent

  	CustomStopWords	augmented	1.5237760339464461	CLUSTER_36	NULL
---	-----------------	-----------	--------------------	------------	------

Tweet\_11533.txt Scientists testing HUD contact lenses on rabbits, hope to bring **augmented** reality to your eyeballs

Tweet\_1267.txt Lego's augmented reality at IDF, eyes-on (video)


































Tweet\_1269.txt 'What Was Here' project adds a pinch of history to augmented reality

Tweet\_12749.txt Playing with Augmented Reality

with

We see that the meaningless terms like with which doesn't add much value show up only for NoStopWords

### CS298 Report

  	RunType	Term	Weight	Field_1	Field_2
  	NoStopWords	with	0.8422637692204228	CLUSTER_3	NULL
  	NoStopWords	with	0.3438452297243579	CLUSTER_5	NULL
  	NoStopWords	with	0.3960316688456434	CLUSTER_20	NULL
  	NoStopWords	with	0.41346423462789844	CLUSTER_30	NULL
  	NoStopWords	with	0.39694931040734216	CLUSTER_33	NULL
  	NoStopWords	with	0.5274251368071629	CLUSTER_38	NULL
  	NoStopWords	with	0.49056213906726	CLUSTER_39	NULL
  	NoStopWords	with	2.053127784873136	CLUSTER_41	NULL
  	NoStopWords	with	0.6664629156996564	CLUSTER_42	NULL
  	NoStopWords	with	0.9887444247370181	CLUSTER_45	NULL

The consolidated table is shown below for the following five terms siri, apple, iphone, augmented and with

Term	No stop words	Default stop words	Custom stop words
Siri	Not part of cluster	1.5073364045884874	3.5180675863362043
apple	0.5385988283673778	0.6785155087709427	4.620405804027211
iphone	2.178985970975669	4.1664243917912245	3.071035173501861
augmented	Not part of cluster	Not part of cluster	1.5207760339464461
With	0.8422637692204228	Not part of cluster	Not part of cluster

Clearly, the important terms have more weightiness when we chose custom stop words compared to us using no and default stop words. Hence, choosing the correct set of stop words for microblogging sites such as Twitter is very important to get good clusters.

The second experiment that was conducted was to first collect sample of thousand random tweets and then use the approach taken by Qing Chen et al.[8] in their paper “Tweets mining using WIKIPEDIA and impurity cluster measurement”. As outlined in the paper we use wikipedia search by identifying the important words in the tweet which the authors call the trend and then do a wikipedia search based on the trend name. We combine the original text with the wikipedia text and use it as an input. Please note that since the tweets are collected on a daily basis there may not be any Wikipedia pages associated with the top terms. In this case we have used the original tweet as the input.

The results are shown for the top ten terms when using the above approach with default stop words versus the approach we took of using custom stop words and is shown in the table below. The second column shows the weights of the top terms under the approach that we took. The third column shows the weights of the terms under the approach that Qing Chen et al. took [8]. We noticed that terms had the highest weight when we chose the right set of stop words compared to combining the tweets with Wikipedia search results, showing that choosing the right set of stop words plays a major role in getting good clusters.

Term	Custom stop words	Wikipedia (Default stop words)
hadoop	5.9972124099731445	3.5919069877037635
internet	4.997677008310954	1.4604256780524003
wired	4.031572866439819	2.644033670425415
nokia	3.9821564934470435	4.583253523882697
free	3.9537076155344644	2.5828651428222655
storage	3.895374298095703	2.7575849056243897
interface	3.8416065216064452	1.6496093273162842
engadget	3.8365699276328087	2.8410439314665616
drop	3.7278315226236978	1.9779029173009537
million	3.5409763560575596	N/A

## 9) Conclusion

One of the main challenges in clustering Twitter data is that the length of each tweet is only 140 characters, and so, identifying the right set of custom stop words is very important.

This project was challenging because of the Twitter API limitations, evaluating and understanding cloud technologies such as Rackspace and Amazon and last but not least trying to understand Apache Mahout, which is a relatively new technology.

The work by no means is complete, one of the ideas that we can continue working on would be to use stop words and combine the twitter text with google news and then cluster the data. One other idea could be to see whether we can give a higher weightage to the tweet in the cluster based on how much the users are sharing it on social networking sites like Facebook and google plus.

## 10) References

[1] Schulman, Kori. "#AskObama at the First Ever Twitter @Townhall at the White House | The White House." The White House. N.p., n.d. Web. 21 Dec. 2011. <<http://www.whitehouse.gov/blog/2011/06/30/askobama-first-ever-twitter-townhall-white-house>>.

[2] "Twitter." Twitter. N.p., n.d. Web. 21 Dec. 2011. <<http://twitter.com/sjsu>>.

[3] "Twitter." Twitter. N.p., n.d. Web. 21 Dec. 2011. <<http://twitter.com/Gizmodo>>.

- [4] "Twitter." Twitter. N.p., n.d. Web. 21 Dec. 2011. <<http://twitter.com/Techcrunch>>.
- [5] "Twitter." Twitter. N.p., n.d. Web. 21 Dec. 2011. <<http://twitter.com/engadget>>.
- [6] "Twitter Help Center | What Are Hashtags ("#" Symbols)?." Twitter Help Center . N.p., n.d. Web. 21 Dec. 2011. <<http://support.twitter.com/articles/49309-what-are-hashtags-symbols>>.
- [7] Swit Phuvipadawat and Tsuyoshi Murata. 2010. Breaking News Detection and Tracking in Twitter. In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 03 (WI-IAT '10), Vol. 3. IEEE Computer Society, Washington, DC, USA, 120-123. DOI=10.1109/WI-IAT.2010.205 <http://dx.doi.org/10.1109/WI-IAT.2010.205>
- [8] Qing Chen; Shipper, Timothy; Khan, Latifur; , "Tweets mining using WIKIPEDIA and impurity cluster measurement," Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on , vol., no., pp.141-143, 23-26 May 2010  
doi: 10.1109/ISI.2010.5484758
- [9] Owen Phelan, Kevin McCarthy, and Barry Smyth. 2009. Using twitter to recommend real-time topical news. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 385-388. DOI=10.1145/1639714.1639794  
<http://doi.acm.org/10.1145/1639714.1639794>
- [10] Twitter API: Up and Running  
Learn How to Build Applications with the Twitter API  
By Kevin Makice  
Publisher:O'Reilly Media  
Released: March 2009
- [11] Mahout in Action  
By Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman  
Publisher:Manning Publications  
Released: October 14, 2011
- [12] "Apache Mahout: Scalable machine learning and data mining." Apache Mahout: Scalable machine learning and data mining. N.p., n.d. Web. 22 Dec. 2011. <<http://mahout.apache.org>>.
- [13] "Documentation | Twitter Developers." Documentation | Twitter Developers. N.p., n.d. Web. 21 Dec. 2011. <[dev.twitter.com](http://dev.twitter.com)>.
- [14] "Amazon Web Services ." Amazon Web Services . N.p., n.d. Web. 22 Dec. 2011.  
<<http://aws.amazon.com>>
- [15] "Wikipedia." Wikipedia. N.p., n.d. Web. 22 Dec. 2011. <<http://www.wikipedia.com>>.
- [16] "Twitter." Twitter. N.p., n.d. Web. 22 Dec. 2011. <<http://www.twitter.com>>

## CS298 Report

- [17] "StatusListener." Twitter4J - A Java library for the Twitter API. N.p., n.d. Web. 26 Dec. 2011. <<http://twitter4j.org/ja/javadoc/twitter4>
- [18] "Cloud Server and Virtual Server Hosting by Rackspace." Cloud Computing, Managed Hosting, Dedicated Server Hosting by Rackspace. N.p., n.d. Web. 26 Dec. 2011. <[http://www.rackspace.com/cloud/cloud\\_hosting\\_products/servers/](http://www.rackspace.com/cloud/cloud_hosting_products/servers/)>
- [19] ""Amazon EC2 Pricing." Amazon Web Services. N.p., n.d. Web. 26 Dec. 2011. <<http://aws.amazon.com/ec2/pricing/>>
- [20] "Creating Vectors from Text." Mahout Wiki. N.p., n.d. Web. 30 Dec. 2011. <<https://cwiki.apache.org/MAHOUT/creating-vectors-from-text.html#CreatingVectorsfromText-ConvertingdirectoryofdocumentstoSequenceFileformat>>
- [21] "AbstractVector (Mahout Math 0.6-SNAPSHOT API)." Search Lucene . N.p., n.d. Web. 30 Dec. 2011. <<http://http://search-lucene.com/jd/mahout/math/org/apache/mahout/math/AbstractVector.html>>
- [22] "MySQL :: MySQL 5.6 Reference Manual :: 11.9.4 Full-Text Stopwords." MySQL :: Developer Zone. N.p., n.d. Web. 30 Dec. 2011. <<http://dev.mysql.com/doc/refman/5.6/en/fulltext-stopwords.html>>.