

EECS 435: PROJECT FINAL REPORT

DATA MINING TWITTER

ARUNPRASATH SHANKAR

axs918@case.edu

INTRODUCTION

Twitter is a real time, highly social micro blogging service that allows us to post short messages of 140 characters or less; these messages are called tweets. Unlike social networks like Facebook and LinkedIn, where a connection is bidirectional, Twitter has an asymmetric network infrastructure of “friends” and “followers.” Assuming we have a Twitter account, our friends are the accounts that we are following and our followers are the accounts that are following us. While we can choose to follow all of the users who are following us, this generally doesn’t happen because we only want our Home Timeline to include tweets from accounts whose content we find interesting.

I believe Twitter is an important phenomenon from the standpoint of its incredibly high number of users, as well as its use as a marketing device and emerging use as a transport layer for third- party messaging services. It generates a tremendous amount of valuable social data and offers an extensive collection of flexible APIs which allows us to do just about anything we could reasonably expect to do with Twitter data before doing any heavy-duty development.

With overall throughput now far exceeding **50 million tweets per day** and occasional peak velocities in excess of **3,000 tweets per second**, there’s vast potential in mining tweet content. By applying some rudimentary analytic functions we can implement by taking advantage of the Twitter APIs to do a number of interesting things with its data. Due to the large update of Twitter, the amount of data available on the site and its easily searchable nature, it has become a great platform for data mining and information gathering. It’s also a powerful research tool. People increasingly use Twitter to share advice, opinions, news, moods, concerns, facts, rumors, and everything else imaginable. Much of that data is public and available for mining.

PROJECT IDEA:

In this project, I had tried to combine social web data (Twitter) with standard data mining analysis techniques along with visualization tools to answer the following kinds of questions:

- ❖ Who's making connections with social media?
- ❖ What they're talking about?
- ❖ How frequently are certain people communicating with one another?
- ❖ How symmetrical is the communication between people?
- ❖ What are people chatting about (and is it interesting)?

OBJECTIVES:

- ❖ To use adaptable scripts (Python) to harvest data from social network APIs such as Twitter.
- ❖ To slice and dice the collected data.
- ❖ To get a straightforward synopsis of the social web landscape.
- ❖ To analyze the social graph linkages that exists among Twitterers.
- ❖ To derive interesting insights about Twitterers by inspecting the entities that appears in their tweets.
- ❖ To visualize the connectivity of users and identifying clusters (groups) of users.
- ❖ To cluster similar tweets based on tf - idf and cosine similarity.
- ❖ To Apply Hierarchical Clustering to Tweets to mine for interesting clusters (groups).
- ❖ To build interactive visualizations with web technologies based upon HTML5 and Java Script toolkits.

DATA MINING TWITTER:

To start with, I downloaded live data from Twitter API, parsed it and used it to demonstrate some basic text processing. Also I queried the harvested data for controversial topics, in hopes of visualizing the two sides of the debate. Steps I followed to carry out my objectives:

- ❖ Harvest tweets for a given query
- ❖ Harvest current trends
- ❖ Find out the most frequently used words in tweets related to a given query
- ❖ Identify noise (stop words)
- ❖ Retrieve retweets
- ❖ Visualize the retweet graph
- ❖ Find similar tweets based on on tf -idf and cosine similarity metrics
- ❖ Find interesting word clusters by applying Hierarchical Clustering to tweets.

SURVEY OF RELATED WORK:

Due to the online visibility of social networks, the scientific research in this field counts several hot

topics. It is possible to identify at least three distinct, not necessarily disjoint, directions of research; they focus, respectively, on I) data collection techniques, II) characterization of online social networks and III) online social networks analysis.

In the first category with respect to data collection aspects, the work by Gjoka et al. [4] on OSNs (in particular on Facebook) is related to what I am aspiring to do in this project. In the second and third categories, I find works whose main goal is to discover properties of online social networks. Sometimes companies provide the complete OSN dataset, e.g. Ahn et al. [5] studied the graph of a South Korean OSN, named CyWorld, and its scalability properties, in order to correctly estimate degree distribution and clustering coefficient in smaller samples. Leskovec [6], in his PhD thesis, studied dynamics of large social networks analyzing data provided by Microsoft about the usage of their instant messaging platform, formerly called MSN. More frequently, social network services companies like Facebook are reluctant to share their data for research purposes. The only viable solution is to acquire this information crawling the front-end of the social networks, wrapping and managing public accessible data.

DATA SOURCE:

I used twitter real time data. The Twitter API only allows clients to make a limited number of calls in a given hour. This policy affects the APIs in different ways. The default rate limit for calls to the API varies depending on the authorization method being used and whether the method itself requires authentication.

- ❖ Unauthenticated calls are permitted **150 requests per hour**. Unauthenticated calls are measured against the public facing IP of the server or device making the request.
- ❖ OAuth calls are permitted **350 requests per hour** and are measured against the oauth_token used in the request.

I obtained my own consumer_key and consumer_secret by registering an application (dummy) with Twitter at <http://dev.twitter.com/apps/new>. These two items, along with the credentials returned through the “OAuth dance,” enables us to provide the application with access to our account data.

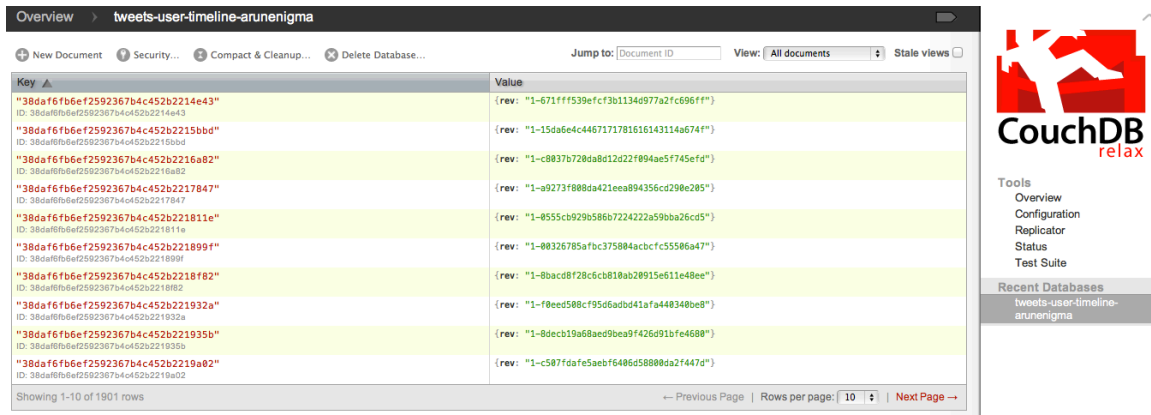
Data server Used:

Redis is an open source, advanced key-value store server. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets and sorted sets.

We can run atomic operations on these types, like appending to a string; incrementing the value in a hash; pushing to a list; computing set intersection, union and difference; or getting the member with highest ranking in a sorted set.

Database Used:

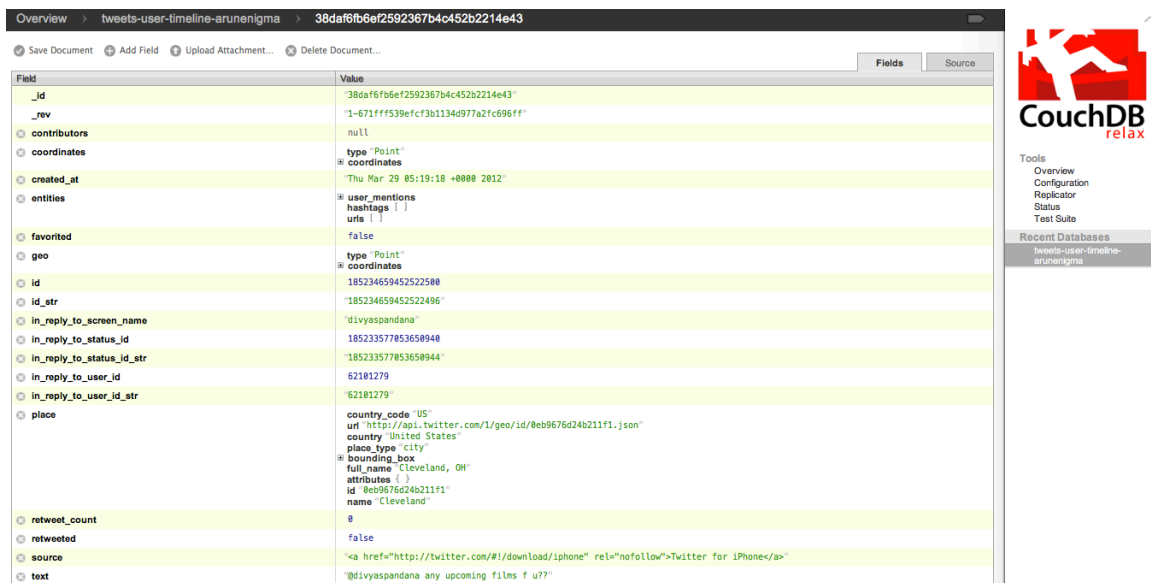
CouchDB: Web based database that can be easily queried and indexed using Python. We can store our data with JSON documents. Access our documents with our web browser, via HTTP. Query, combine, and transform our documents with JavaScript. CouchDB works well with modern web and mobile apps.



Key	Value
"38da6fb6ef2592367b4c452b2214e43" ID: 38da6fb6ef2592367b4c452b2214e43	{rev: "1-671fff539efcf3b1134d977a2fc696ff"}
"38da6fb6ef2592367b4c452b2215bbd" ID: 38da6fb6ef2592367b4c452b2215bbd	{rev: "1-15da6e4c4467171781616143114a674f"}
"38da6fb6ef2592367b4c452b2216a82" ID: 38da6fb6ef2592367b4c452b2216a82	{rev: "1-c8837b720da8d12d22f094ae5f745efd"}
"38da6fb6ef2592367b4c452b2217847" ID: 38da6fb6ef2592367b4c452b2217847	{rev: "1-a9273f888da421eeab94356cd298e285"}
"38da6fb6ef2592367b4c452b221811e" ID: 38da6fb6ef2592367b4c452b221811e	{rev: "1-8555cb929b586b722422a59bba26cd5"}
"38da6fb6ef2592367b4c452b221899f" ID: 38da6fb6ef2592367b4c452b221899f	{rev: "1-80326785afbc375804acbcfc55586a47"}
"38da6fb6ef2592367b4c452b2218f82" ID: 38da6fb6ef2592367b4c452b2218f82	{rev: "1-8bacdbf28c6cb818ab28915e611e48ee"}
"38da6fb6ef2592367b4c452b221932a" ID: 38da6fb6ef2592367b4c452b221932a	{rev: "1-f8eed588cf95d6add41af448348be8"}
"38da6fb6ef2592367b4c452b221935b" ID: 38da6fb6ef2592367b4c452b221935b	{rev: "1-8decb19a68aed9bea9f426d91bfe4688"}
"38da6fb6ef2592367b4c452b2219a02" ID: 38da6fb6ef2592367b4c452b2219a02	{rev: "1-c587fdafe5aebf6486d5888da2f447d"}

Showing 1-10 of 1901 rows

A snapshot of all my tweets stored in Couchdb



Field	Value
_id	"38da6fb6ef2592367b4c452b2214e43"
_rev	"1-671fff539efcf3b1134d977a2fc696ff"
contributors	null
coordinates	{type: "Point", coordinates: "Thu Mar 29 05:19:18 +0000 2012"}
created_at	"Thu Mar 29 05:19:18 +0000 2012"
entities	{user_mentions: [{screen_name: "divyaspandana", id: 62101279, id_str: "62101279"}], hashtags: [], urls: []}
favorited	false
geo	{type: "Point", coordinates: "185234659452522500"}
id	"185234659452522500"
id_str	"185234659452522496"
in_reply_to_screen_name	"divyaspandana"
in_reply_to_status_id	"185233577853658948"
in_reply_to_status_id_str	"185233577853658944"
in_reply_to_user_id	"62101279"
in_reply_to_user_id_str	"62101279"
place	{country_code: "US", url: "http://api.twitter.com/1/geo/id/8eb9676d24b211f1.json", country: "United States", place_type: "city", bounding_box: {full_name: "Cleveland, OH", attributes: {}, id: "8eb9676d24b211f1", name: "Cleveland"}}
retweet_count	0
retweeted	false
source	Twitter for iPhone
text	@divyaspandana any upcoming films f u??

A snapshot of couchdb page showing field- value of each attribute of a tweet

TOOLS AND METRICS USED:

Choice of Programming Language - Python

Python's intuitive syntax, amazing ecosystem of packages for data manipulation, and core data structures that are practically JSON make it an excellent tool that's powerful yet also very easy to get things up and running.

Also I have tried to show a broad array of useful visualizations in this project using a variety of visualization toolkits like Graphviz to HTML5 technologies such as Protovis.

- ❖ NLTK Module (Natural Language Processing with Python)
- ❖ Data Mining Metrics like Cosine Similarity, Frequency Analysis and Lexical Diversity, tf-idf, cosine similarity, pearson correlation coefficient
- ❖ Protovis is free and open-source, provided under the BSD License. It uses JavaScript for web-native visualizations; It accepts a search term as a command line parameter, fetches, parses, and pops up our web browser to visualize the data as an interactive HTML5-based graph.
- ❖ Graphviz: An approach for visualizing graphs of tweet data: exporting them to the DOT language, a simple text-based format that Graphviz consumes. Graphviz is open source graph visualization software.
- ❖ Clustering Algorithms like Hierarchical Clustering.

COLLECTING AND MANIPULATING TWITTER DATA

To tinker with Twitter's API, I used a minimal python wrapper around Twitter's web API that is available through a package called twitter.

Retrieving twitter trends and paging through twitter search results

Current Trends Sample (Code and Results in Project Folder)

```
{u'url': u'http://twitter.com/search/%23SongsIListenToEveryday', u'query': u'%23SongsIListenToEveryday',  
u'name': u'#SongsIListenToEveryday', u'promoted_content': None, u'events': None}
```

```
{u'url': u'http://twitter.com/search/%23TheWorstFeeling', u'query': u'%23TheWorstFeeling', u'name':  
u'#TheWorstFeeling', u'promoted_content': None, u'events': None}
```

```
{u'url': u'http://twitter.com/search/Chiquimarco', u'query': u'Chiquimarco', u'name': u'Chiquimarco',  
u'promoted_content': None, u'events': None}
```

```
{u'url': u'http://twitter.com/search/%22Bubba%20Watson%22', u'query': u'%22Bubba%20Watson%22',  
u'name': u'Bubba Watson', u'promoted_content': None, u'events': None} ...
```

Given that a topic is trending, I tried to grab some search results about it by using the search API to search for tweets containing a specific query and then print them out in a readable way as a JSON structure.

Twitter Search Results -- Public Timeline

```
http://t.co/74CLXtnm CWRU researcher Jonathan Karn seeks way to find HIV when it is hiding - Plain Dealer  
RT @tehgenerallee: Relay for Life at CWRU - just walked for 14 hours. Need a break before I continue onward until 7am.  
Relay for Life at CWRU - just walked for 14 hours. Need a break before I continue onward until 7am.  
RT @MaraSteines: CWRU Relay for Life!!! #whydoyourelay #fortheight #pinkribbon  
And I met Kinnell in almost the same way. He had some friends from CWRU working for Epic (he's at eClinicalWorks) who  
knew me.  
Academic - Staff Employment Opportunities @ CWRU - Case ...  
RT @MaraSteines: CWRU Relay for Life!!! #whydoyourelay #fortheight #pinkribbon  
CWRU researcher Jonathan Karn seeks way to find HIV when it is hiding - Plain Dealer http://t.co/74CLXtnm  
...
```

(Code and Results in Project Folder)

My code fetches and stores twelve consecutive batches (pages) of results for a query (q) with 100 results per page.

FREQUENCY ANALYSIS AND LEXICAL DIVERSITY:

LEXICAL DIVERSITY:

One of the most intuitive measurements that can be applied to unstructured text (tweets) is a metric called lexical diversity. It is an expression of the number of unique tokens in the text divided by the total number of tokens in the text, which are elementary yet important metrics in and of them.

Results for the query 'cwrU'

(Code and Results in Project Folder)

Total Words = 1472

Unique Words = 710

Lexical Diversity = 0.482336956522

Average Words = 14.72

One way to interpret a lexical diversity of around 0.48 would be to say that about one out of every two words in the aggregated tweets is unique. Given that the average number of words in each tweet is around 15 that translates to just over 7 to 8 unique words per tweet. Without introducing any additional information that could be interpreted as meaning that each tweet carries about 50 percent unique information.

The inference that we can derive from these results is how “noisy” the tweets are with uncommon abbreviations users may have employed to stay within the 140 characters, as well as what the most frequent and infrequent terms used in the tweets are. I have created a distribution of the words and their frequencies to visualize things below.

What are people talking about right now?

Among the most compelling reasons for mining Twitter data is to try to answer the question of what people are talking about right now. One of the simplest techniques we could apply to answer this question is basic frequency analysis. NLTK simplifies this task by providing an API for frequency analysis.

The Natural Language Toolkit (NLTK) is a popular module, which delivers a vast amount of tools for various kinds of text analytics, including the calculation of common metrics, information extraction, and natural language processing (NLP). It also offers a built-in frequency distribution and many other tools for text analysis.

Table 1: Basic Frequency Analysis

(Real Time Analysis for the query “cwru” as on 04/22/2012)

<i>Rank</i>	<i>Words</i>	<i>Frequency</i>	<i>Rank</i>	<i>Words</i>	<i>Frequency</i>
<i>Not Filtered</i>			<i>Filtered</i>		
1	#CWRU	32	1	#CWRU	32
2	to	30	2	CWRU	24
3	the	29	3	RT	19
4	CWRU	24	4	#cwru	13
5	a	23	5	CSU	11
6	for	21	6	Case	11
7	at	20	7	Western	8

8 RT 19	8 Bathurst 7
9 and 17	9 Emus 7
10 in 17	10 Reserve 7
11 of 14	11 University 7
12 with 14	12 great 7
13 #cwru 13	13 Great 6
14 on 13	14 Relay 6
15 The 12	15 find 6
16 CSU 11	16 lead 6
17 Case 11	17 @SGRho_ZetaCH 5
18 I 10	18 CWRU: 5
19 is 10	19 SGRhobics- 5
20 - 9	20 back 5
21 by 9	21 go 5
22 Western 8	22 #TEDxCLE 4
23 it 8	23 28-25, 4
24 Bathurst 7	24 @ 4
25 Emus 7	25 @pRHod1gy: 4
26 Reserve 7	26 @thinkintweets 4
27 University 7	27 BOARS 4
28 great 7	28 Brian 4
29 4 6	29 Bulldogs 4
30 Great 6	30 Dr. 4
31 Relay 6	31 Two 4
32 find 6	32 WIN!!!! 4
33 lead 6	33 battle 4
34 this 6	34 coming 4
35 @SGRho_ZetaCH 5	35 defeat 4
36 A 5	36 defending 4
37 CWRU: 5	37 forth 4
38 SGRhobics- 5	38 premiers 4
39 back 5	39 researcher 4
40 go 5	40 students 4
41 up 5	41 woRHOk 4
42 #TEDxCLE 4	42 #forthefight 3
43 28-25, 4	43 #pinkribbon 3
44 @ 4	44 #tedxcle 3
45 @pRHod1gy: 4	45 #whydoyourelay 3
46 @thinkintweets 4	46 (CWRU) 3
47 BOARS 4	47 2012; 3
48 Brian 4	48 23, 3
49 Bulldogs 4	49 Bomb 3
50 Dr. 4	50 Cleveland 3
...	...

50 Most Frequent Tokens (Includes Stopwords/Symbols)

#CWRU, to, the, CWRU, a, for, at, RT, and, in, of, with, #cwru, on, The, CSU, Case, I, is, -, by, Western, it, Bathurst, Emus, Reserve, University, great, 4, Great, Relay, find, lead, this, @SGRho_ZetaCH, A, CWRU:, SGRhobics-, back, go, up, #TEDxCLE, 28-25,, @, @pRHod1gy:, @thinkintweets, BOARS, Brian, Bulldogs, Dr.

50 Least Frequent Tokens (Includes Stopwords)

that, their, them, these, this..., though!, thru, tickets, tickets., tie, tmrw, today!, today's, today,, today., today?,

tons, top, track, trying, u, under, unveils, up!!!, ur, victory, visit, voted, w/nerf, w/orange,, waayyy, wasted!, way., website, week!, week:, well, wen, went, which, who, will, win, winners, work, working, worse., yes,, |

50 Most Frequent Tokens (After Filtering Out Stopwords)

#CWRU, CWRU, RT, #cwru, CSU, Case, Western, Bathurst, Emus, Reserve, University, great, Great, Relay, find, lead, @SGRho_ZetaCH, CWRU:, SGRhobics-, back, go, #TEDxCLE, 28-25,, @, @pRHodlgy:, @thinkintweets, BOARS, Brian, Bulldogs, Dr., Two, WIN!!!!, battle, coming, defeat, defending, forth, premiers, researcher, students, woRHOk, #forthefight, #pinkribbon, #tedxcle, #whydoyourelay, (CWRU), 2012:, 23,, Bomb, Cleveland

50 Least Frequent Tokens (After Filtering Out Stopwords)

success!, summer, summer., sure, swear, sweeping, taking, talked, team, this..., though!, thru, tickets, tickets., tie, tmrw, today!, today's, today,, today., today?, tons, top, track, trying, u, unveils, up!!!, ur, victory, visit, voted, w/nerf, w/orange,, waayyy, wasted!, way., website, week!, week:, well, wen, went, win, winners, work, working, worse., yes,, |

A very quick skim of the results above shows that a lot more useful information is carried in the frequent tokens than the infrequent tokens. I found that the frequent tokens refer to entities such as people, times, and activities, while the infrequent terms amount to mostly noise from which no meaningful conclusion could be drawn.

The first thing I noticed about the most frequent tokens is that “**CRWU**”, “**CWRU:**”, “**#cwru**”, “**CWRU.**” are top of the list. Given that it is the basis of the original search query, this isn’t surprising at all. Where it gets more interesting is when we skim the remaining tokens: there is apparently a lot of chatter about Relay, since the tweets were mined on 04/22/2012 (the day after the Relay for Life Event) as evidenced by the tokens *relay*, *#whydoyourelay*, *lead*, *go*. The point is that frequency analysis is a very simple, yet very powerful tool that shouldn’t be overlooked just because it’s so obvious. On the contrary, it should be tried out first for precisely the reason that it’s so obvious and simple.

As a final observation, the presence of “RT” is also a very important clue as to the nature of the conversations going on. The token RT is a special symbol that is often prepended to a message to indicate that we are retweeting it on behalf of someone else. Given the high frequency of this token, it’s reasonable to infer that there were a large amount of duplicate or near-duplicate tweets involving the subject matter at hand.

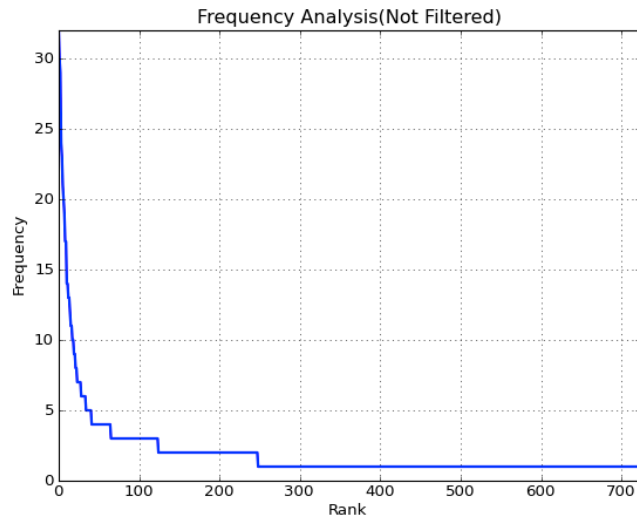


Figure 1: Frequency Distribution (Not Filtered) of words for the query “cwru”

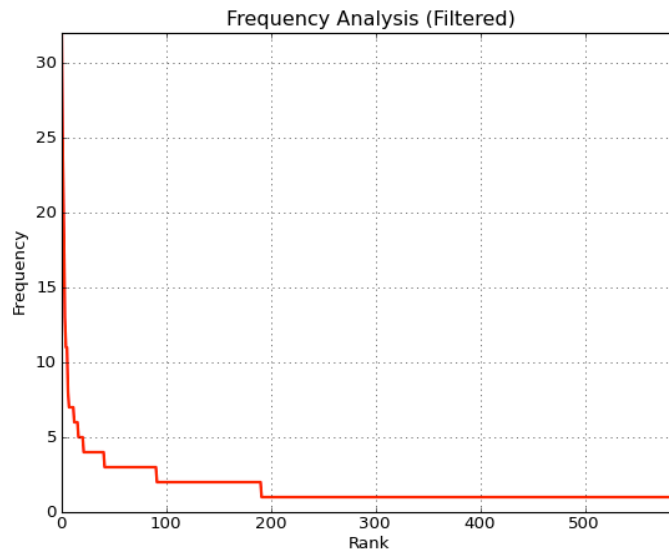


Figure 2: Frequency Distribution (Filtered) of words for the query “cwru”

(Code and Results in Project Folder)

By analyzing the list of words from frequency distribution, I found that common words like “the,” “to,” and “of” - stop words - are the most frequently occurring, but there’s a steep decline and the distribution has a very long tail. So what we can infer from the above figure is that a word’s frequency within tweets is inversely proportional to its rank in the frequency table. What this means is that if the most frequently occurring term in tweets accounts for $N\%$ of the total words, the second most frequently occurring term in the tweets should account for $(N/2)\%$ of the words, the third most

frequent term for $(N/3)\%$ of the words, etc. When graphed, such a distribution (even for a small sample of tweets) shows a curve that hugs each axis, as we can see in Figure above. An important observation here is that most of the area in the distribution lies in its tail.

By looking at Figure 1 and Figure 2, we can say that in Figure 1 there are more number of stop words and the initial most ranked words are essentially the stopwords. In figure 2, the distribution is plotted after removing the stop words/symbols because of which we can see a swift change in the ranking order as per the figure.

EXTRACTING RELATIONSHIPS FROM THE TWEETS:

Because the social web is first and foremost about the linkages between people in the real world, one highly convenient format for storing social web data is a graph.

I have used NetworkX to build out a graph connecting Twitterers who have retweeted information. Although the Twitter APIs do offer some capabilities for determining and analyzing statuses that have been retweeted, these APIs are not a great fit for our current use case because we'd have to make a lot of API calls back and forth to the server, which would be a waste of the API calls included in our quota. Since Twitter imposes a rate limit of 350 API calls per hour for authenticated requests; anonymous requests are limited to 150 per hour.

Besides, we can use the clues in the tweets themselves to reliably extract retweet information with a simple regular expression. By convention, Twitter usernames begin with an @ symbol and can only include letters, numbers, and underscores. Thus, given the conventions for retweeting, we only have to search for the following patterns:

- RT followed by a username
- Via followed by a username

I used the re (Regular Expressions) module to compile a pattern and extract the originator of a tweet in a lightweight fashion, without any special libraries.

Given that the tweet data structure as returned by the API provides the username of the person tweeting and the newly found ability to extract the originator of a retweet, it's easy to load this information into a NetworkX graph. I created a graph in which nodes represent usernames and a directed edge between two nodes signifies that there is a retweet relationship between the nodes. The basic steps involved are generalizing a routine for extracting usernames in retweets, flattening out the pages of tweets into a flat list for easier processing in a loop, and finally, iterating over the tweets and adding edges to a graph.

The built-in operations that NetworkX provides are useful to make sense of the data. Since we are looking at a very small slice of the overall conversation happening on Twitter - 500 tweets out of potentially tens of thousands (or more). For example consider the below distribution for the query 'obama', the number of nodes in the graph tells us that out of 500 tweets, there were 160 users involved in retweet relationships with one another, with 125 edges connecting those nodes. The ratio of $160/125$ (approximately 1.28) is an important clue that tells us that the average degree of a node is approximately one - meaning that although some nodes are connected to more than one other node, the average is approximately one connection per node.

The call to connected components shows us that the graph consists of 37 subgraphs and is not fully connected. The output of degree is a way to get the gist of how well connected the nodes in the graph are without having to render an actual graph. In this case, most of the values are 1, meaning all of those nodes have a degree of 1 and are connected to only one other node in the graph. A few values are between 2 and 9, indicating that those nodes are connected to anywhere between 2 and 9 other nodes. The gist of the graph is that it's mostly composed of disjoint nodes, but there is one very highly connected node.

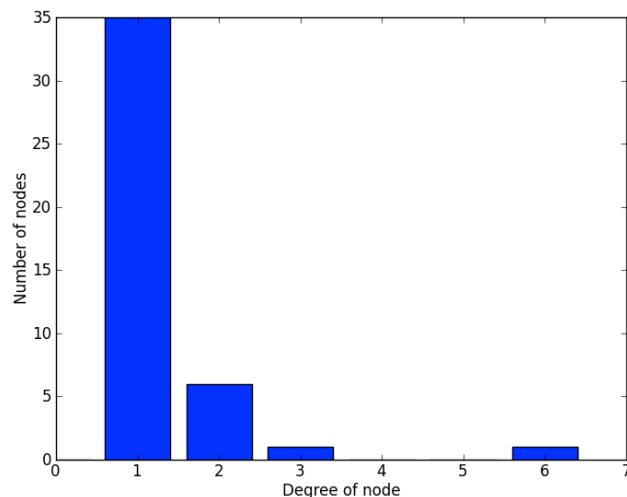
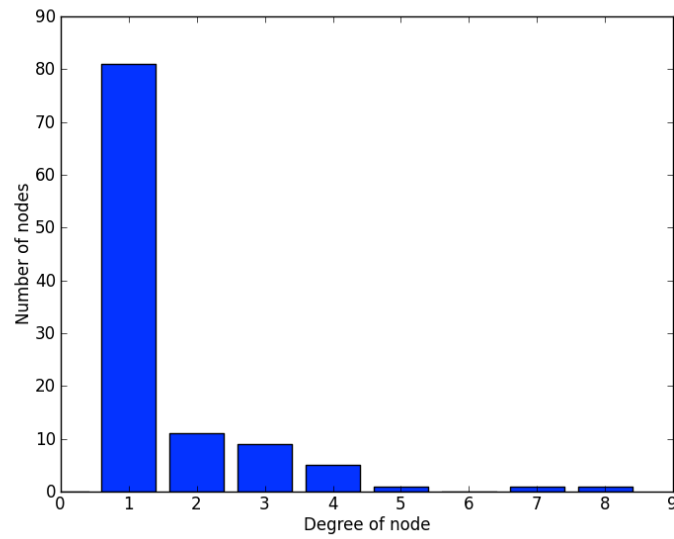


Figure 3: Distributions illustrating the degree of each node in the graph, which reveals insight into the graph's connectedness for the queries cwru (above) and obama (below)



VISUALIZING TWEET GRAPHS AND RETWEETS

An interactive Protovis graph with a force-directed layout that visualizes retweet relationships for the query “cwru” and “obama”

Different Layouts portraying the connectivity of users based on retweets

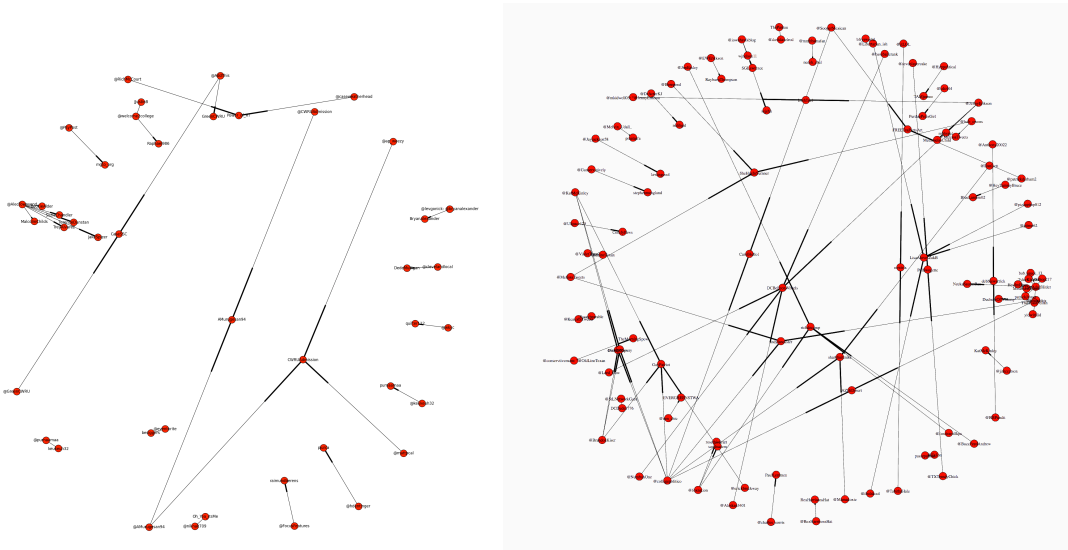


Figure 4: Fruchterman Reingold Layout for queries “cwru” (left) and “obama” (right)

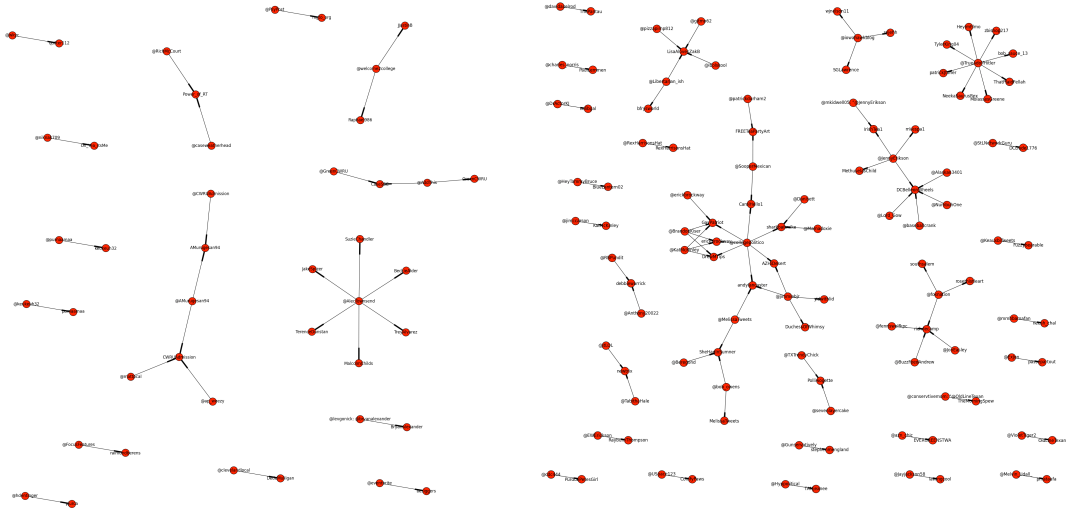


Figure 5: Graphviz Layout for queries “cwru” (left) and “obama” (right)

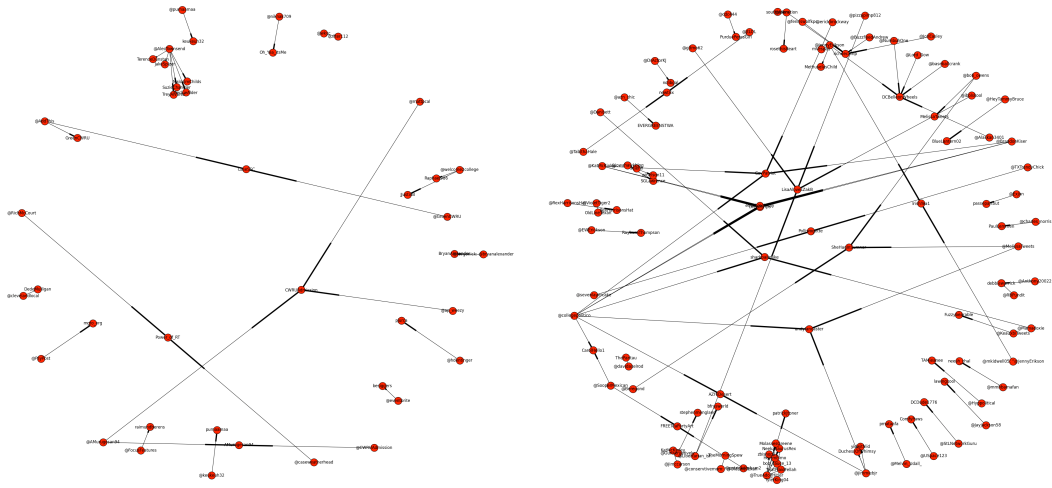


Figure 6: Spring Layout for queries “cwru” (left) and “obama” (right)

Building and analyzing a graph describing who retweeted whom

Query : 'cwru'

Number nodes: 82

Num edges: 60

Num connected components: 28

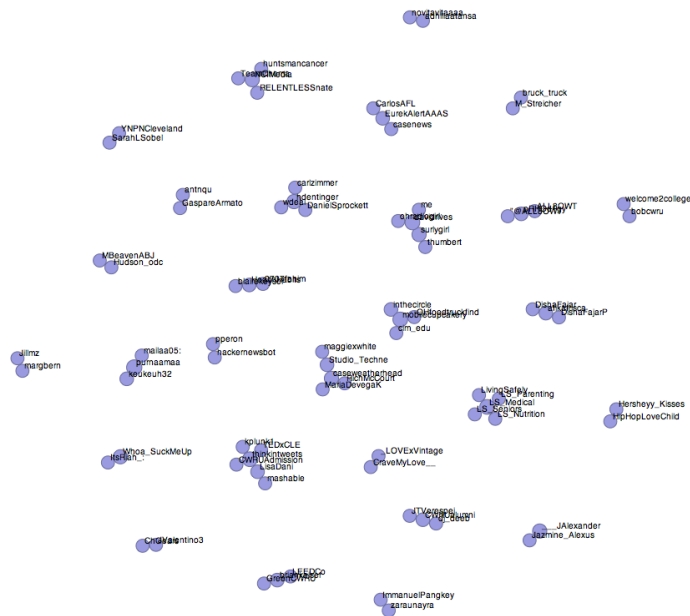


Figure 7: Retweet Graph showing retweet clusters for the query “cwru”



Query: 'Obama'

Number nodes: 299

Num edges: 215

Num connected components: 93

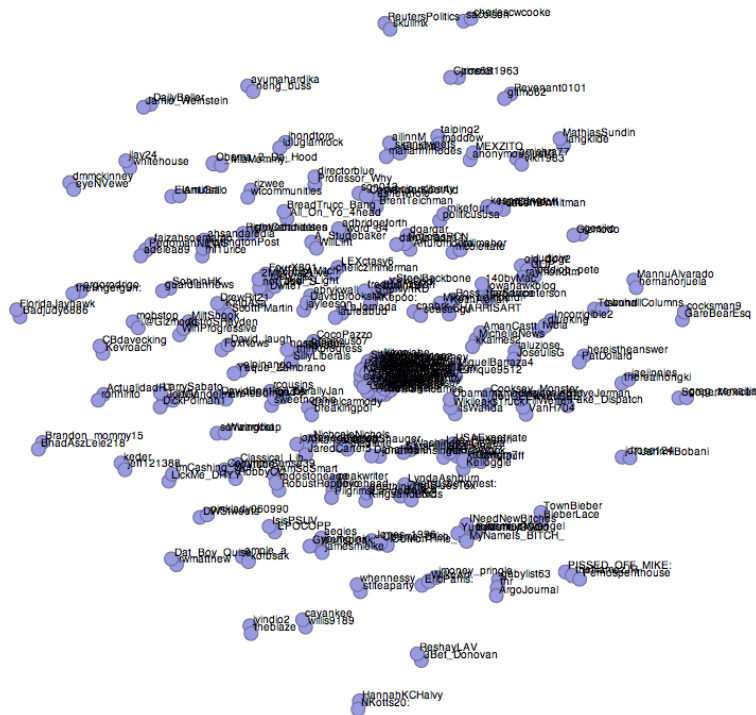
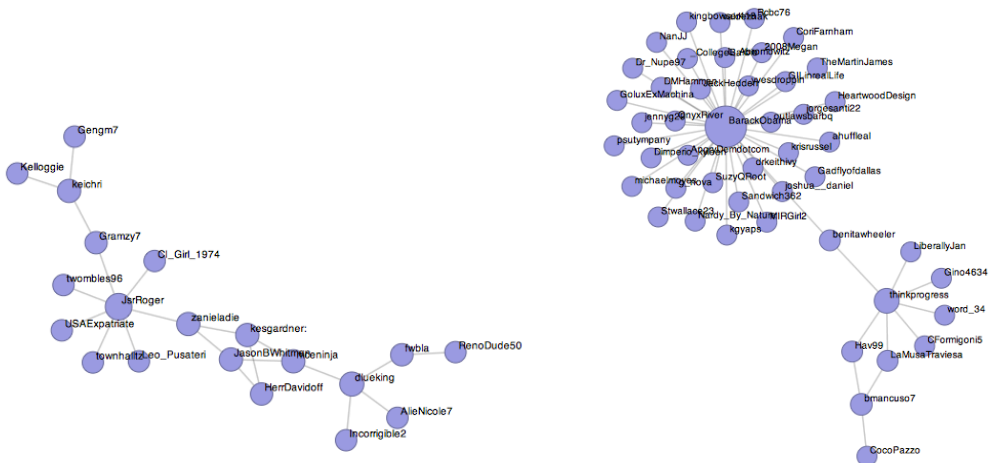


Figure 8: Retweet Graph showing retweet clusters for the query "obama"



Making sense of the textual information in tweets by using data mining fundamentals like TF-IDF, cosine similarity and collocation detection.

By using TF-IDF, we can retrieve relevant tokens (keywords) from tweets. TF-IDF stands for term frequency-inverse document frequency and can be used to query tweets by calculating normalized scores that express the relative importance of tokens in the tweets. Mathematically, TF-IDF is expressed as the product of the term frequency and the inverse document frequency, $tf_idf = tf * idf$, where the term tf represents the importance of a token in a specific tweet, and idf represents the importance of a token relative to the entire collection of tweets.

The inverse document frequency idf metric is a calculation that provides a generic normalization metric for the tweets, and accounts for the appearance of common terms across a set of tweets by taking into consideration the total number of tweets (three here) in which a query term ever appears. The intuition behind this metric is that it produces a higher value if a term is somewhat uncommon across the tweets than if it is very common, which helps to account for the problem with stopwords we just investigated.

Considering term frequency alone turns out to be a common problem when scoring on a tweet-by-tweet basis because it doesn't account for very frequent words that are common across many tweets. Meaning, all terms are weighted equally regardless of their actual importance. Using NLTK which provides list of stopwords that can be used to filter out terms such as "the", "a", "and", etc., but there may be terms that escapes even the best stopwords lists and yet still are quite common to specialized domains. `https` and symbols are such terms which need to be filtered. Consider the following sample of tweets.

Consider Three Sample Tweets mined for the query 'obama'

Tweet 1 → Mr. Obama pushes the urge to keep low student loan rates President Mr. Obama has the power to change

Tweet 2 → Obama touting affordable education, to keep low student loan

Tweet 3 → RT @BarackObama: Obama urges Congress to prevent student loan interest rates from doubling to save education

For example, the term "obama" (once normalized to lowercase) occurs twice in Tweet 1 and only once in Tweet 2, so Tweet 1 would produce a higher score if frequency were the only scoring criterion. However, if we normalize for document length, Tweet 2 would have a slightly higher term frequency score for "obama" ($1/9$) than Tweet 1 - ($2/19$), because Tweet 2 is shorter than Tweet 1 - even

though “obama” occurs more frequently in Tweet 1. Thus, a common technique for scoring a compound query such as “Mr. Obama” is to sum the term frequency scores for each of the query terms in each document, and return the documents ranked by the summed term frequency score.

Table 1: Sample term frequency scores for tokens → “mr.” and “obama” as explained above

<i>Tweets</i>	<i>tf(mr.)</i>	<i>tf(obama)</i>	<i>Sum</i>
<i>Tweet 1</i>	<i>2/19</i>	<i>2/19</i>	<i>4/19 (0.2105)</i>
<i>Tweet 2</i>	<i>0</i>	<i>1/9</i>	<i>1/9 (0.1111)</i>
<i>Tweet 3</i>	<i>0</i>	<i>1/16</i>	<i>1/16 (0.0625)</i>

Table 2: Sample term frequency scores for tokens → “the”, “obama” and “education”

<i>Tweets</i>	<i>tf(the)</i>	<i>tf(obama)</i>	<i>tf(education)</i>	<i>Sum</i>
<i>Tweet 1</i>	<i>2/19</i>	<i>2/19</i>	<i>0</i>	<i>4/19 (0.2105)</i>
<i>Tweet 2</i>	<i>0</i>	<i>1/9</i>	<i>1/9</i>	<i>1/9 (0.1111)</i>
<i>Tweet 3</i>	<i>0</i>	<i>1/16</i>	<i>1/16</i>	<i>1/16 (0.0625)</i>

Table 3,4,5 (below): Calculations involved in TF-IDF sample queries, as computed by Table 1 and 2

<i>Tweets</i>	<i>tf(mr.)</i>	<i>tf(obama)</i>	<i>tf(the)</i>	<i>tf(education)</i>
<i>Tweet 1</i>	<i>0.1053</i>	<i>0.1053</i>	<i>1.1053</i>	<i>0</i>
<i>Tweet 2</i>	<i>0</i>	<i>0.1111</i>	<i>0</i>	<i>0.1111</i>
<i>Tweet 3</i>	<i>0</i>	<i>0.0625</i>	<i>0</i>	<i>0.0625</i>

<i>Tweets</i>	<i>idf(mr.)</i>	<i>idf(obama)</i>	<i>idf(the)</i>	<i>idf(education)</i>
<i>Tweet 1</i>	<i>1.4771</i>	<i>1.0</i>	<i>1.4771</i>	<i>1.1760</i>
<i>Tweet 2</i>	<i>1.4771</i>	<i>1.0</i>	<i>1.4771</i>	<i>1.1760</i>
<i>Tweet 3</i>	<i>1.4771</i>	<i>1.0</i>	<i>1.4771</i>	<i>1.1760</i>

<i>Tweets</i>	<i>tf-idf (mr.)</i>	<i>tf-idf (obama)</i>	<i>tf-idf (the)</i>	<i>tf-idf (education)</i>
<i>Tweet 1</i>	0.1555	0.1053	0.1555	0
<i>Tweet 2</i>	0	0.1111	0	0.1306
<i>Tweet 3</i>	0	0.0625	0	0.0735

The same results for each query are shown in Table 6 (below), with the TF-IDF values summed on a per-tweet basis.

<i>Query</i>	<i>Tweet 1</i>	<i>Tweet 2</i>	<i>Tweet 3</i>
<i>obama</i>	0.1053	0.1111	0.0625
<i>mr. obama</i>	0.1555 + 0.1053 = 0.2607	0 + 0.1111 = 0.1111	0 + 0.0625 = 0.0625
<i>the obama education</i>	0.1555 + 0.1053 + 0 = 0.2607	0 + 0.1111 + 0.1306 = 0.2418	0 + 0.0625 + 0.0735 = 0.1360

From a qualitative standpoint, the query results make reasonable sense. Tweet 2 is the winner for the query “obama”, with Tweet 1 just a hair behind. In this case, the deciding factor was the length of Tweet 2 being much smaller than Tweet 1: the normalized TF score tipped in favor of Tweet 2 for its one occurrence of “obama”, even though “obama” appeared in Tweet 1 - two times. Similarly, Tweet 1 is also the winner for the compound search queries “mr. obama” and “the obama education”.

Querying Twitter Data with TF-IDF

Thus, by applying TF-IDF to the entire saved Twitter data in JSON file and by using NLTK module, we can pass in multiple query terms that are used to score the tweets (or tokens) by relevance.

(Code and Results in Project Folder)

Relevant Tweets filtered for the Query “cwru” and “research”

CWRU researchers find joint failures potentially linked to oral bacteria

Score: 0.0339214866024

CWRU researchers find joint failures potentially linked to oral bacteria

Score: 0.0339214866024

<http://t.co/RRq2mIGy> CWRU researcher Jonathan Karn seeks way to find HIV when it is hiding

Score: 0.0271371892819

<http://t.co/jgPTkCvs> Cleveland researcher receives Top 10 Clinical Research Achievement Award

Score: 0.0262617960792

<http://t.co/74CLXtnm> CWRU researcher Jonathan Karn seeks way to find HIV when it is hiding - Plain Dealer

Score: 0.0232604479559

CWRU researcher Jonathan Karn seeks way to find HIV when it is hiding - Plain Dealer <http://t.co/74CLXtnm>

Score: 0.0232604479559

#TEDxCLE great developments in #malaria research from Brian Grimberg, one of my former professors at #CWRU #Case

Score: 0.0218066699587

CWRU researchers find joint failures potentially linked to oral bacteria <http://t.co/8yvKjseP> #science #news #gumdisease

Score: 0.0203528919614

Full Results in Project Folder

FINDING SIMILAR TWEETS USING COSINE SIMILARITY

Using Vector Space Model And Cosine Similarity

If we consider the tweets to be a large multidimensional vector space that contains one vector for each tweet, and the distance between any two vectors indicates the similarity of the corresponding tweets. One of the most important things about using vector space model is that we can represent a query as a vector and find the most relevant tweets for the query by finding the tweet vectors with the shortest distance to the query vector.

For example, imagine a tweet that is defined by only two terms (“Case”, “Western”), with a corresponding vector of (0.45, 0.67), where the values in the vector are values such as TF-IDF scores for the terms. In a vector space, this tweet could be represented in two dimensions by a line segment extending from the origin at (0,0) to the point at (0.45, 0.67). In reference to an x/y plane, the x-axis would represent “Case”, the y-axis would represent “Western”, and the vector from (0,0) to (0.45, 0.67) would represent the tweet in question. However, interesting tweets generally contain hundreds of terms at a minimum, but the same fundamentals apply for modeling tweets in these higher-dimensional spaces; what could be applied to a 2-dimensional space can be equally as well applied to a 10-dimensional space or a 1000-dimensional space.

As it turns out, the cosine of the angle between any two vectors is a valid metric for comparing them and is known as the cosine similarity of the vectors. Intuitively, it might be helpful to consider that the closer two vectors are to one another, the smaller the angle between them will be, and thus the larger the cosine of the angle between them will be. Two identical vectors would have an angle of 0 degrees

and a similarity metric of 1.0, while two vectors that are orthogonal to one another would have an angle of 90 degrees and a similarity metric of 0.0.

We need to produce a term vector for each tweet and compute the dot product of the unit vectors for those tweets. We compute term vectors for a given pair of tweets by assigning TF-IDF scores to each component in the vectors. Because the exact vocabularies of the two tweets are probably not identical, however, placeholders with a value of 0.0 must be left in each vector for words that are missing from the tweet at hand but present in the other one. The net effect is that we end up with two vectors of identical length with components ordered identically that could be used to perform the vector operations.

Cosine Similarity

Here are two tweets to compare:

Tweet 1 → Mr. Obama pushes the urge to keep low student loan

Tweet 2 → Obama touting affordable education, to keep student loan low

<i>Tokens</i>	<i>Tweet 1</i>	<i>Tweet 2</i>
<i>mr.</i>	<i>1</i>	<i>0</i>
<i>obama</i>	<i>1</i>	<i>1</i>
<i>pushes</i>	<i>1</i>	<i>0</i>
<i>urge</i>	<i>1</i>	<i>0</i>
<i>keep</i>	<i>1</i>	<i>1</i>
<i>low</i>	<i>1</i>	<i>1</i>
<i>student</i>	<i>1</i>	<i>1</i>
<i>loan</i>	<i>1</i>	<i>1</i>
<i>touting</i>	<i>0</i>	<i>1</i>
<i>affordable</i>	<i>0</i>	<i>1</i>

We want to know how similar these tweets are, purely in terms of word count, ignoring word order and stop words. We begin by making a list of the words from both tweets:

We are not interested in the words themselves though. We are interested only in those two vertical vectors of counts. For instance, there are instances of 'student' and 'loan' in each tweet. We are going to decide how close these two tweets are to each other by calculating one function of those two vectors, namely the cosine of the angle between them.

The two vectors are:

$u: [1,1,1,1,1,1,1,0,0]$

$v: [0,1,0,0,1,1,1,1,1]$

The cosine of the angle between them is about **0.6682**.

These vectors are 10 - dimensional. A virtue of using cosine similarity is clearly that it converts a question that is beyond human ability to visualize to one that can be. In this case we can think of this as the angle of about 48 degrees, which is some 'distance' from zero or perfect agreement.

For example, suppose Tweet 1 contains the terms (A, B, C) and had the corresponding vector of TF-IDF weights (0.10, 0.15, 0.12), while Tweet 2 contained the terms (C, D, E) with the corresponding vector of TF-IDF weights (0.05, 0.10, 0.09).

The derived vector for Tweet 1 would be (0.10, 0.15, 0.12, 0.0, 0.0), and the derived vector for Tweet 2 would be (0.0, 0.0, 0.05, 0.10, 0.09). By reusing the TF-IDF calculations that were discussed previously, the exact scoring function could be any useful metric. TF-IDF (or some variation thereof), however, is quite common for many implementations and provides a great starting point.

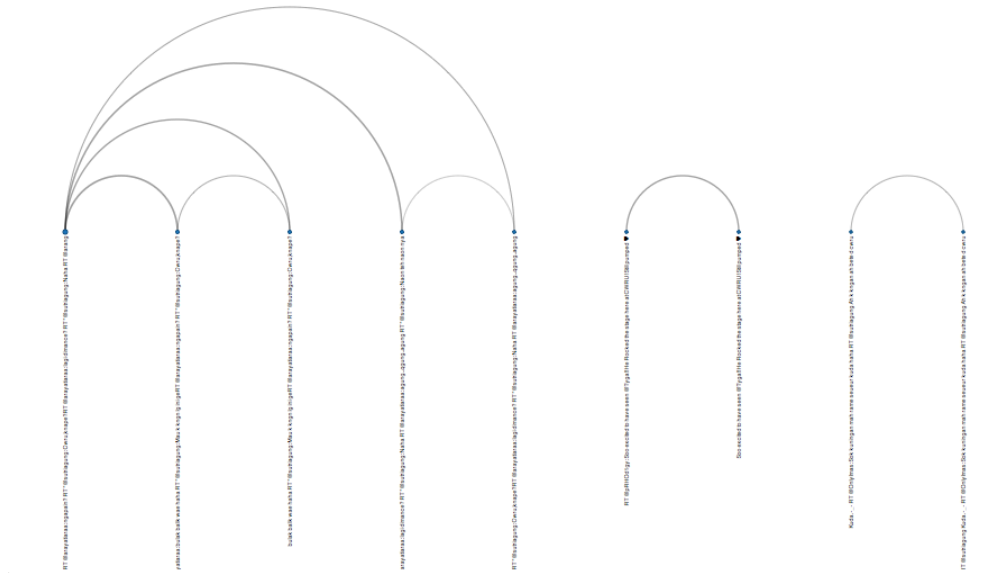
Instead of comparing just tweet vectors, we can compare our query vector and the tweet vectors. In terms of implementation, that means constructing a vector containing our query terms and comparing it to each tweet in the entire collection.

Visualizing Similarity with Graph Visualizations

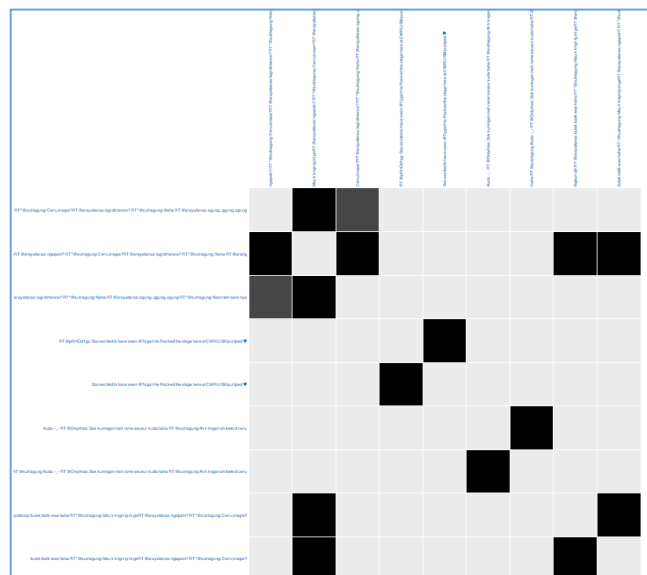
My code produces the arc diagram presented in Figure 6 and the matrix diagram in Figure 7.

The arc diagram produces arcs between nodes if there is a linkage between them, scales nodes according to their degree, and sorts the nodes such that clutter is minimized and it's easy to see which of them have the highest number of connections.

(Code and Results in Project Folder)



Matrix diagram (for the query “news”), with the color intensity of each cell varying as a function of the strength of the correlation of similarity score; a darker cell represents higher similarity. The diagonal in the matrix in Figure (below) has been omitted since the similarity scores for the diagonal would be perfect. Hovering over a cell displays the similarity score. An advantage of matrix diagrams is that there’s no potential for messy overlap between edges that represent linkages. A commonly cited disadvantage of arc diagrams is that path finding is difficult, but in this particular situation, path finding isn’t that important, and besides, it can still be accomplished if desired.



Matrix diagram (cwru)

PART 2:

HEIRARCHIAL CLUSTERING TWITTER DATA

Hierarchical clustering builds up a hierarchy of groups by continuously merging the two most similar groups. Each of the groups starts as a single item, in our case an individual tweet token. In each iteration, we calculate the distances between every pair of groups, and the closest ones are merged together to form a new group. This is repeated until there is only one group.

After hierarchical clustering is completed, we usually view the results in a type of graph called a dendrogram, which displays the nodes arranged into their hierarchy. The dendrogram I generated for the project data (Twitter) is shown in Figure below. This dendrogram not only uses connections to show which items ended up in each cluster, it also uses the distance to show how far apart the items were. Rendering the graph this way can help us determine how similar the items within a cluster are, which could be interpreted as the tightness of the cluster.

For the project, I have tried to generate a hierarchy of tweet tokens so to cluster the tweets thematically.

Distance Metric Used:

I have used Pearson correlation to determine how similar two tweet tokens are. While using twitter data, I found that some tweets contain more tokens or much longer tokens than others. By using, Pearson correlation we can correct for this, since it really tries to determine how well two sets of data fit onto a straight line.

Since Pearson correlation is 1.0 when two items match perfectly, and is close to 0.0 when there's no relationship at all. My code returns 1.0 minus the Pearson correlation to create a smaller distance between items that is more similar.

Each cluster in a hierarchical clustering algorithm is either a point in the tree with two branches, or an endpoint associated with an actual row from the dataset (in our case, a tweet). Each cluster also contains data about its location, which is either the row data for the endpoints or the merged data from its two branches for other node types. I have created a class called bicluster that has all of these properties, which I used to represent the hierarchical tree.

The algorithm for hierarchical clustering begins by creating a group of clusters that are just the original items. The main loop of the function searches for the two best matches by trying every possible pair and calculating their correlation. The best pair of clusters is merged into a single cluster. The data for this new cluster is the average of the data for the two old clusters. This process is repeated until only

one-cluster remains. Since it is very time consuming to do all these calculations, I stored the correlation results for each pair, since they will have to be calculated again and again until one of the items in the pair is merged into another cluster. Because each cluster references the two clusters that were merged to create it, the final cluster returned by this function can be searched recursively to recreate all the clusters and their end nodes.

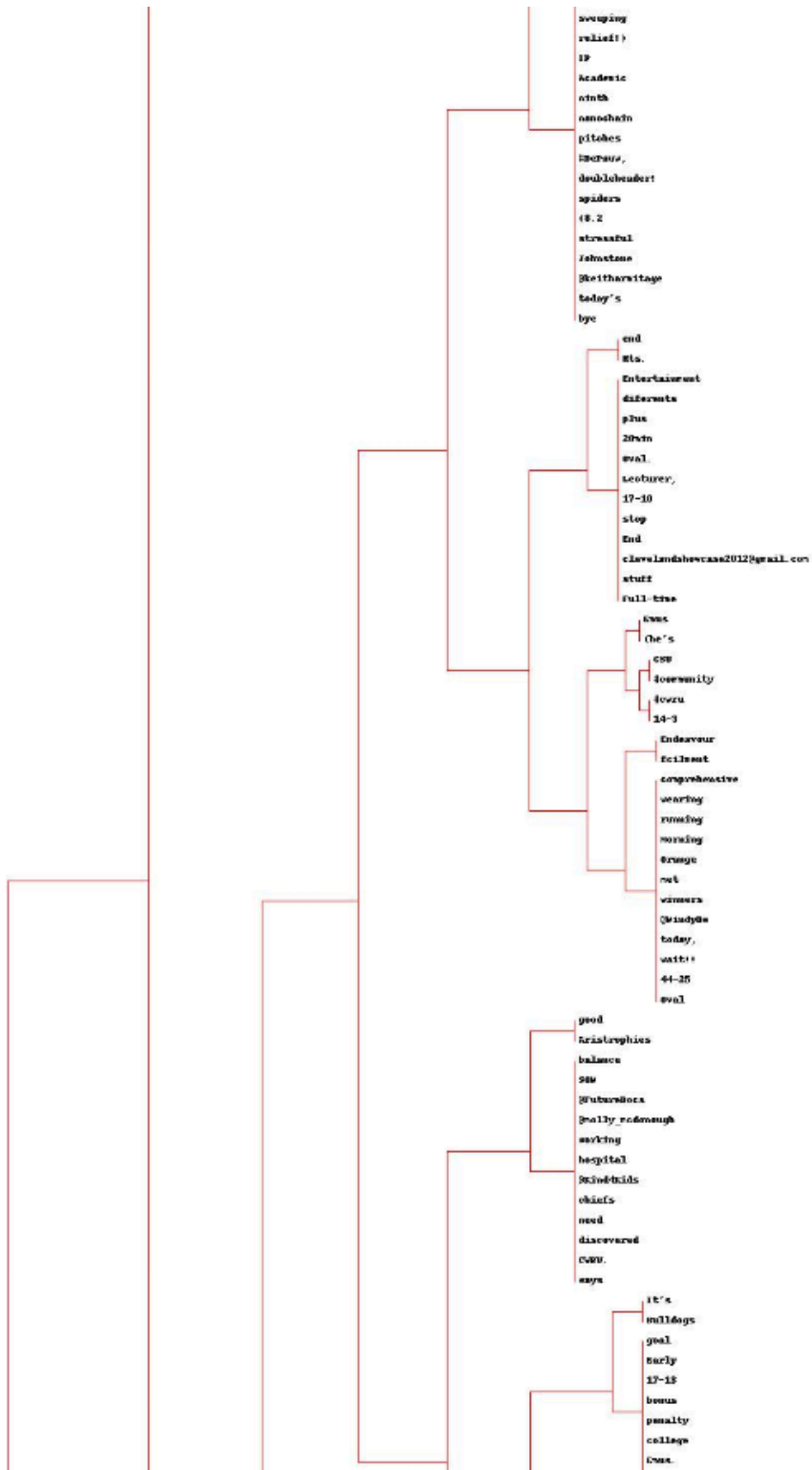
We can interpret the clusters more clearly by viewing them as a dendrogram. I used Python Imaging Library (PIL) for the purpose. The PIL makes it very easy to generate images with text and lines. It's often necessary to cluster on both the rows and the columns. For the project dataset, the columns represent words (Twitter Tokens), and it's potentially interesting to see which words are commonly used together.

The easiest way to do this using the functions we've written thus far is to rotate the entire dataset so that the columns (the words) become rows, each with a list of numbers indicating how many times that particular word appears in each of the tweets.

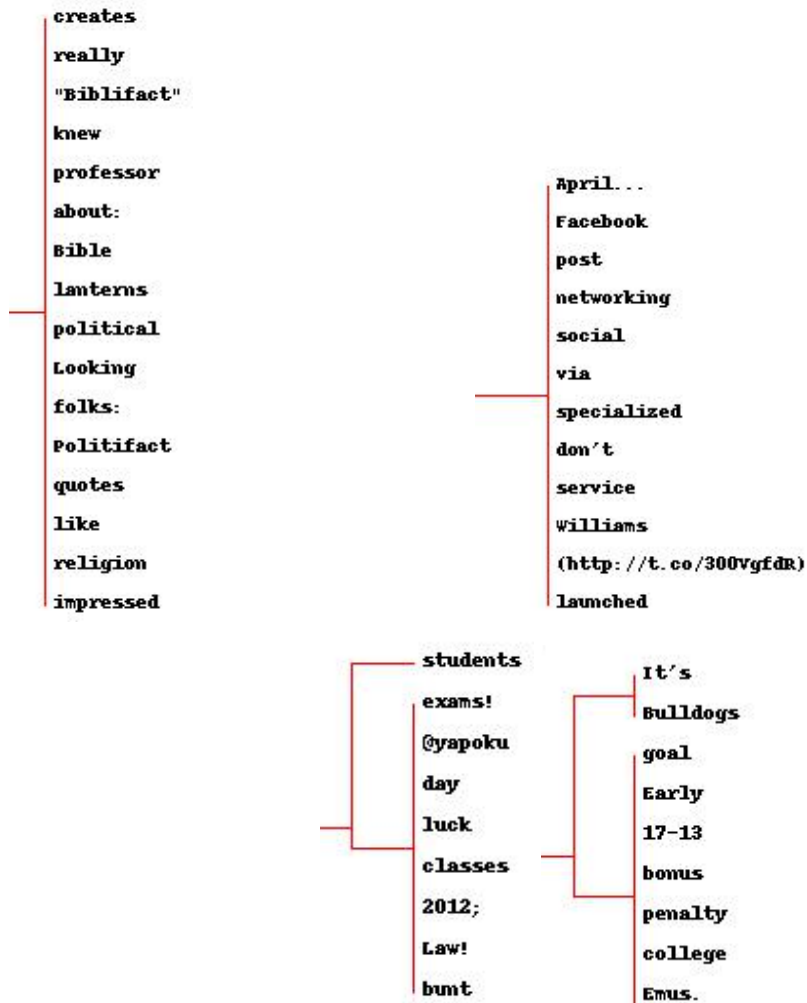
We can now rotate the matrix and run the same operations for clustering and drawing the dendrogram. As there are many more words than tweets, this will take longer than running the tweet clustering. Remember that since the matrix has been rotated, the words rather than the tweets are now the labels.

One important thing to realize about clustering is that if we have many more items than variables, the likelihood of nonsensical clusters increases. There are many more words than there are tweets, so we'll notice more reasonable patterns in the tweet clustering than in the word clustering. However, some interesting clusters definitely emerge, as shown in Figure below.

(Code and Results in Project Folder)



Dendrogram showing Hierarchical Clustering of tweets (cwru)



Interesting Clusters generated in the Dendrogram

Cluster 1 → is about Bible & Religion

Cluster 2 → is about CWRU's own social network launched in April (follow the link in the cluster), it's called CWRUandme.com

Cluster 3 → is about exams in Case Law school

Cluster 4 → is about a match between CASE and rivals

APPLICATIONS:

- ❖ Each tweet contains much more data than simply a username and the tweet itself. A tweet can contain data related to the author's biography, their URL, their location, the number of

users following this user, the amount of users they are following, along with country and other details. This treasure of information may be used for a variety of purposes. Companies can rank the relevance of a tweet by the importance of a person, they can track what people think about their store by comparing the location of the tweet to the location of their stores and so much more, the potential list is endless. I was quite surprised by what they have access to.

- ❖ Twitter is a favorite source of text data for analysis: it's popular (there is a huge volume of variety on all topics) and easily accessible using Twitter's free, open APIs which are easily consumable in JSON and ATOM formats.
- ❖ Social network analytics makes it possible to measure public sentiment with real-time data mined from Twitter, blogs and other social networks. Text analytics uses natural language processing to spot key words and to predict sentiment.
- ❖ By combining data from social networks with existing, structured data, including internal tweets; it's possible to obtain even better intelligence, leading to better decision-making.
- ❖ Social media has proved its Social networks also has a unique capacity to capture the mood of the moment and to spur powerful, impromptu actions that can have both a positive and negative impact.
- ❖ Data mining Twitter has great potential for the news business. Instead of painstakingly building a list of relevant people who sometimes prattle endlessly, we can capture in our web of interests only the relevant tweets produced by a desired group and the group it follows, all adding-up in real-time. This could be a great tool to follow developing stories and enhance live coverage.
- ❖ A permanent, precise and noise-free view of what's hot on Twitter is a key component of the 360° view of the web every media should now offer. As of today, many small businesses focused around tweet analytics are potentially making a fairly healthy profit by selling answers to certain classes of ad-hoc queries that customers demand.

REFERENCES:

[1] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In Proceedings of the 19th international conference on World wide web (WWW '10). ACM, New York, NY, USA, 591-600. DOI=10.1145/1772690.1772751 <http://doi.acm.org/10.1145/1772690.1772751>

[2] Raymond Kosala and Hendrik Blockeel. 2000. Web mining research: a survey. SIGKDD Explor. Newsl. 2, 1 (June 2000), 1-15. DOI=10.1145/360402.360406

<http://doi.acm.org/10.1145/360402.360406>

[3] Peter Mika. 2004. Social Networks and the Semantic Web. In Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI '04). IEEE Computer Society, Washington, DC, USA, 285- 291. DOI=10.1109/WI.2004.128

<http://dx.doi.org/10.1109/WI.2004.128>

[4] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. Walking in facebook: a case study of unbiased sampling of OSNs. In Proceedings of the 29th conference on Information communications, pages 2498 {2506. IEEE Press, 2010

[5] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. 2007. Analysis of topological characteristics of huge online social networking services. In Proceedings of the 16th international conference on World Wide Web(WWW '07). ACM, New York, NY, USA, 835-844. DOI=10.1145/1242572.1242685 <http://doi.acm.org/10.1145/1242572.1242685>

[6] J. Leskovec. Dynamics of large networks. PhD thesis, Carnegie Mellon University, 2008.

[7] Data Mining for Social Network Data Series: Annals of Information Systems, Vol. 12 2010, 2010, IX, 215 p. 68 illus.

[8] Jon M. Kleinberg. 2007. Challenges in mining social network data: processes, privacy, and paradoxes. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07). ACM, New York, NY, USA, 4-5.

DOI=10.1145/1281192.1281195 <http://doi.acm.org/10.1145/1281192.1281195>

[9] <http://journal.planetwork.net/article.php?lab=reed0704> for another perspective on the social web that focuses on digital identities.

[10] <http://www.stackoverflow.com>