

<Title of project placed here>

Akshata Bhat 2361611

Chinghan Cheng 2413917

Hu Liang 2387172

Yin Zhang 2412842

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

<Date of submission placed here>

**Abstract**

<Abstract goes here…>

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

<**Please note that you are under no obligation to sign this declaration, but doing so would help future students.>**

Name: Akshata Bhat Signature:

Name: Chinghan Cheng Signature:

Name: Hu Liang Signature:

Name: Yin Zhang Signature:**Contents**

[1 Introduction 1](#_Toc532821885)

[2 Survey 2](#_Toc532821886)

[3 Data collection 3](#_Toc532821887)

[3.1 News collection 3](#_Toc532821888)

[3.1.1 Implementation of Beautiful Soup 3](#_Toc532821889)

[3.1.2 Implementation of Requests 3](#_Toc532821890)

[3.2 Tweets collection 3](#_Toc532821891)

[3.2.1 Implementation of Streaming API 3](#_Toc532821892)

[3.2.2 Implementation of REST API 4](#_Toc532821893)

[4 Data processing 5](#_Toc532821894)

[4.1 Data storage and noise removing 5](#_Toc532821895)

[4.1.1 Implementation of MongoDB 5](#_Toc532821896)

[4.1.2 Remove duplicates, retweets and quotes 5](#_Toc532821897)

[4.2 Entity recognition and data filtration 6](#_Toc532821898)

[4.2.1 Implementation of Stanford NER 6](#_Toc532821899)

[5 Data Analysis 8](#_Toc532821900)

[5.1 Tweets grouping 10](#_Toc532821901)

[5.1.1 Prework 10](#_Toc532821902)

[5.1.2 Optimization 10](#_Toc532821903)

[5.1.3 Implementation of Dice Coefficient method 11](#_Toc532821904)

[5.1.4 Implementation of MinHash LSH function 11](#_Toc532821905)

[5.1.5 Comparison of two methods 12](#_Toc532821906)

[5.2 Data visualization 12](#_Toc532821907)

[5.2.1 Tweets distribution 12](#_Toc532821908)

[5.2.2 News key words 12](#_Toc532821909)

[6 Evaluation 13](#_Toc532821910)

[7 Conclusion 14](#_Toc532821911)

[References 15](#_Toc532821912)

**Chapter 1**

# Introduction

Social media has made a rapid advancement in the past few years, accelerated by the striking development of mobile devices, such as smart phones and portable laptop, as well as Internet. Twitter, as a representative social network site which has a micro-blogging format, becomes an information gathering platform with huge amount of data. It provides users an opportunity to share and discuss what is happening locally or globally in real time. According to recent statistics (OMNICORE, 2018), there are 326 million monthly active users in twitter and 500 million tweets are sent per day. So mining data in twitter and subsequently establish a topic model may have a great value with regard to various fields. Additionally, thanks to the geo-enabled function of it, it makes developers or researchers convenient and less time-consumed to detect hot topics in users’ posts or discussions based on geography. In consideration of instantaneity of tweets, trying to explore the relationship between tweets and news reports which often have a lag seems to be of great significance in predicting hot news and analysing crowds’ reactions.

During this research, a mass of tweets are collected. Simultaneously, news is also gathered from three news websites, the BBC Scotland, the Scotsman and the Herald. In our project, we collect news during five different periods of a day from Monday to Sunday via importing packages, Beautiful Soup and Reqeusts, in python. As to twitter data, we adopt Streaming API and REST API to get as much as data. Then we select MongoDB which is a cross-platform document-oriented database as our database to store location based data. In order to remove noise, StandfordNER package is imported through NLTK interface to do entity recognition and the output entities are the precondition of subsequent grouping. The purpose of our project is to firstly analyse the variation stream of tweets in different time periods of a day, and then try to group these tweets on the basis of different entities including person, organization and location. As a consequence of a series of filtering procedures, the hot topics in Glasgow may emerge. This may be useful to dig out the underlying relationship between tweets and official news released in the same corresponding periods.

This paper is organized as follows. Section 2 introduces previous others’ research related to our project. Section 3 mainly describes data collection and Section 4 is about processing data including news and tweets. Section 5 provides experimental results to make analysis and further visualize them. Conclusion is given in Section 6. Finally, Section 7 evaluates this project.

**Chapter 2**

# Survey

Topic detection in social medial is also known as event detection. Topic Detection and Tracking (TDT) has been a classic approach to address event detection from conventional media (Atefeh and Khreich, 2015). However, because twitter messages do not have to be well-written and structured but are restricted in text length, TDT is unsuitable to be applied in tweets analysis. Therefore, many researchers have made great effort on this field. Aiming to analyse the composite events from different twitter users but not only a single source, Zhou and Chen (2014) proposed a location-time constrained topic (LTT) graphical model to make a social message represented as a probability distribution and measured the similarity between messages through the distance between distributions. McMinn and Jose (2015) examined the role of named entities in event detection and proposed an entity-based approach to detect real-time events in Twitter. After testing with a huge amount of tweets, their approach is verified to have higher precision and lower computational complexity compared to state-of-the-art approaches. Doulamis, Kokkinos and Varvarigos (2016) developed an event detection algorithm to model twitter dynamics and finally proved that multiassignment clustering performs better than traditional graph partitioning methods. Liu et al. (2017) ever used twitter data with geo-tag to put forward a probabilistic graphical model to detect global and local topics in an offline manner.

When it comes to news, different news agencies publishing the same news in different versions is common. So the need of detecting same incident across different versions broadcasted online to avoid effort-consuming becomes essential. Goh, Soon and Haw (2013) pointed out that the named entities (NEs) existing in various versions of news can be used to observe the same incident. Tariq, Karim and Foroosh (2017) presented a sparse structure model, NELasso, to recognize and characterize the relevance among NEs mentioned in news articles.

Inspired by previous work above, we apply Named Entity Recognizer on both tweets and news in our project.

**Chapter 3**

# Data collection

## 3.1 News collection

By means of Beautiful Soup and Requests python libraries, online news contains title, time, URL and article is obtained from three news websites, the BBC Scotland, the Scotsman and the Herald. Five periods news of each day was captured during a week and finally it reached 5687 pieces of news totally.

### 3.1.1 Implementation of Beautiful Soup

Beautiful Soup is a Python library which can extract data from HTML and XML files (Beautiful Soup documentation, 2015). It performs well on navigating, searching and modifying the parse tree by working with different parsers. In our research, we approached the BeautifulSoup4.

### 3.1.2 Implementation of Requests

Requests library is an Apache2 Licensed HTTP library which allows programmers to send HTTP/1.1 requests by using Python language. Without adding query strings to URLs, it is automatic and convenient (Reitz, 2018).

## 3.2 Tweets collection

For gathering both real-time tweets and past context, Twitter APIs including Streaming API and REST API are implemented. We got nearly 2.2milloins tweets in total.

### 3.2.1 Implementation of Streaming API

Twitter Streaming API provides a convenient way to access to 1% of all real-time tweets (Matthew, 2013). Meanwhile, tweepy package is imported, which is the adaptation of Twitter Streaming API for Python platform. It exposes the Streaming API in an easy-to-use manner and with tweepy, it's possible to get any object and use any method that the official Twitter API offers (Github, 2018). In this research, it is provided for both collecting data randomly with ‘sample’ function and filtering data with geo-tagged from the UK with ‘filter’ function. The location parameter is provided at the latter function by bounding box theory. Bounding box theory means that geo-tagged tweets have 4 Longitude-Latitude coordinates which can indicate a specific area (TWITTER DEVELOPER, 2018). Therefore, the place where tweets are collected can be restricted.



### 3.2.2 Implementation of REST API

As a supplement of Streaming API, REST API which allows developers to retrieve the historical Twitter data is also used. Its filters like location, language and so on are various (Matthew, 2013). This report approach Search API and Timeline for collecting data based on keywords and user names. The keywords are applied by the burst entities, the user names are the accounts relative to the three news channels.



**Chapter 4**

# Data processing

## 4.1 Data storage and noise removing

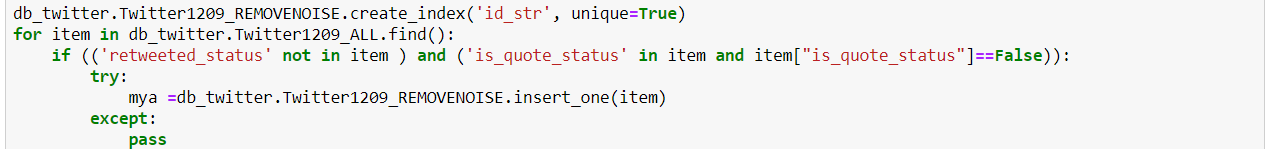
After completing data acquisition, we stored news and tweets into different databases in MongoDB. Then we removed duplicated news and those tweets which are retweets or quotes in order to avoid unnecessary errors when grouping them in Chapter 5.

### 4.1.1 Implementation of MongoDB

MongoDB is an open-source and free document-oriented database with querying and indexing that developers need. What makes it popular in developing modern apps and analysing data is its flexibility and scalability in storing data (MONGODB, 2018). It can be used to store any of the social web data in JSON-like files which allows data structure can be changed over time (Matthew, 2013).

### 4.1.2 Remove duplicates, retweets and quotes

The method of createIndex is used for removing duplicates through the features of the operation only scans the collection once, and if at least one index is to be built in the foreground, the operation will build all the specified indexes in the foreground (MONGODB, 2018). Therefore, this period depends on the parameter of ‘id\_str’ which is defined as the unique identifier for the Tweet and as if the tweet is unique it can be stored. In the next step, the retweets and quotes are removed by identify their ‘retweeted\_status’ which contains a representation of the original Tweet that was retweeted and ‘is\_quoted\_status’ which indicates whether this is a Quoted Tweet (TWITTER DEVELOPER, 2018).



## 4.2 Entity recognition and data filtration

(?) named entities were recognized in tweets and (?) in news by applying Stanford NER package. We stored these entities with their unique ids of raw data into new collections in MongoDB. It seemed that Named Entity Recognizer acted as a filter during this course.

### 4.2.1 Implementation of Stanford NER

Stanford Named Entity Recognizer is an implementation of a Named Entity Recognizer (NER) programmed in java. NER can label sequences of words in an inputted text which are names of entities, such as person, location and organization even gene and protein names (STANFORD, 2018).

Here in this project, in order to implement Stanford NER in analysis process with python language, NLTK package is imported to provide an interface to make it work. After data is extracted from MongoDB, we do recognize entities in both news and twitter data.

Take the process of news as an example:

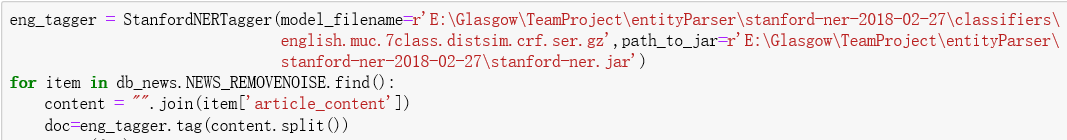


Figure 4.1: Example of entity recognition

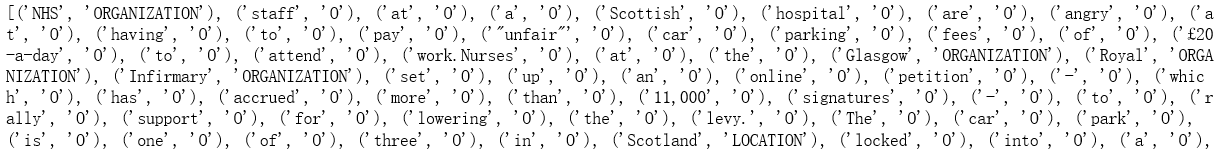


Figure 4.2: Example of recognition result (partly)

Then, all the entities of every piece of data are stored into a new collection with its original id and during this process data without entities was removed:

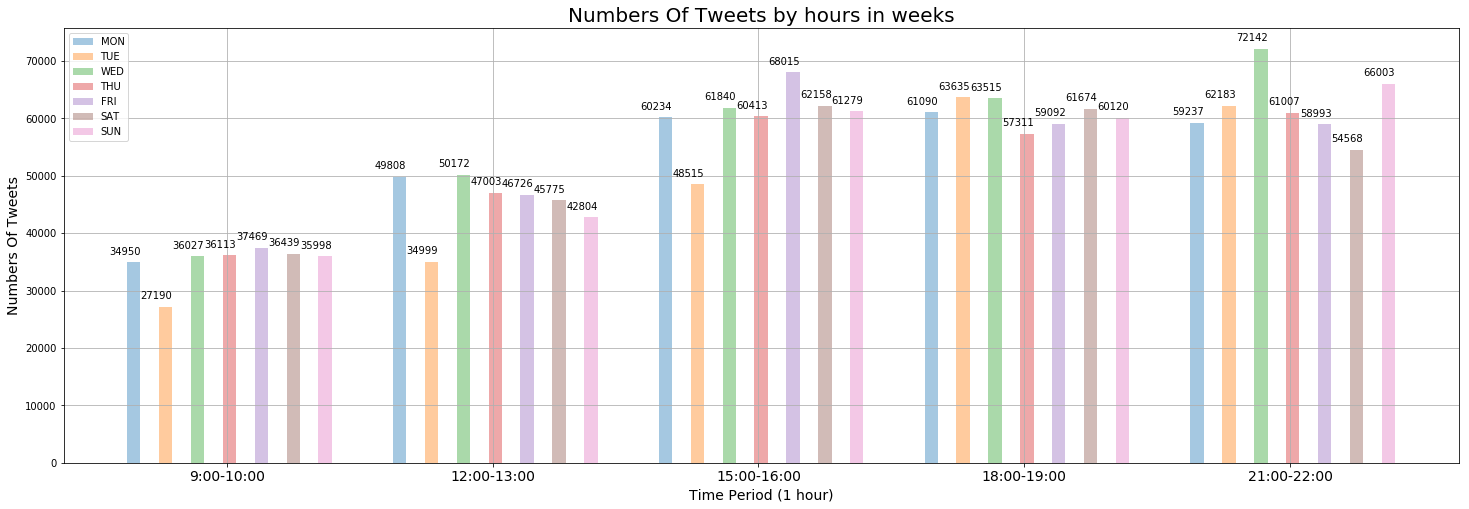


Figure 4.3: Storage of entities with id

**Chapter 5**

# Data Analysis

**Tweets:**



**Observation:** This plot shows the numbers of tweets are collected by hours in weeks. There are more tweets shown after 15:00 each day and the maximum spots on Wednesday night from 21:00 to 22:00. The following is on Friday afternoon from 15:00 to 16:00 then the Sunday night from 21:00 to 22:00.



**News:**

**(graph)**

**Observation:**

**(description)**

## 5.1 Tweets grouping

Some twitter users may not mark the place when they post tweets. In order to be aware of the place where tweets get released, we decide to group tweets and allocate the geo-information to those tweets without geo-tagged through this.

This research applies two methods in this part, one is MinHash LSH function, and the other bases on Dice Coefficient method.

### 5.1.1 Prework

After previous steps, only the tweets with entities are stored. The same entities are then grouped by aggregate function for showing the frequency of keywords. Specifically, count the number of tweets of every entity has been mentioned, then calculate the mean(μ) and the standard deviation(σ). We first plan to keep those with frequency larger than μ-2σ, however, the standard deviation is larger than we expected causing μ-2σ turn to negative. At last, the tweets with entities appear above-average tweets are kept and can be identified as burst entities.

### 5.1.2 Optimization

Firstly, when the data was collected regularly, some duplicated twitters were collected more than once. Some the duplication must be removed. The removing duplication is not working as expected, which we tried using nested loop to compare, find and remove the duplications. The preformation was unacceptable, which consumed more than 12 hours for each collection. To resolve this problem, unique index as the internal feature of MongoDB is chosen to do this task. It can finish the task in 5 minutes. It just needs us to generate the collection and create a unique index of id column on it, and then insert the data one by one. Hence, we just need to catch the unique error and let the process continue.

Another big performance issue we met is that when we started running the NER part code with StanfordNER, which is developed by java, on server, it was unexpected that took a so long time to divide sentences into words one dataset by one. The reason is the NLTK package, which is encapsulation of StanfordNER, calls java program consumed around 2 to 3 seconds for each item of dataset.

According to this reason, we considered to find a method to optimize this process. Firstly, multithreading method was tried for resolving it. However, lock must be used to grantee the independence of each thread and protect the private data would not be edited by another thread. It slowed the performance which just improve only 2 to 3 times as 30 threads.

### Thread function

class myThread (threading.Thread):

def \_\_init\_\_(self, threadID, name, q, rlock):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.name = name

self.q = q

self.rl = rlock

def run(self):

print ("Starting Thread: " + self.name)

process\_data(self.name, self.q, self.rl)

print ("Exiting Thread: " + self.name)

### Data Processing, queueLock is used to grantee the queue can be touched only one thread at once. RLOCK is used to protect the private data would not be changed by another thread.

def process\_data(threadName, q, rlock):

while not exitFlag:

queueLock.acquire()

if not workQueue.empty():

item = q.get()

queueLock.release()

collection\_twitter = db\_twitter.Twitter1204\_REMOVENOISE\_NER\_MultiThread

rlock.acquire()

doc=eng\_tagger.tag(item['text'].split())

rlock.release()

……

else:

queueLock.release()

### init thread, locks, queue

threadList = ["Thread-"+str(i) for i in range(1,31)]

rlock = threading.RLock()

queueLock = threading.Lock()

workQueue = queue.Queue()

threads = []

threadID = 1

### Generate threads

for tName in threadList:

thread = myThread(threadID, tName, workQueue, rlock)

thread.start()

threads.append(thread)

threadID += 1

starttime = datetime.datetime.now()

### Fill the workload queue

queueLock.acquire()

for word in db\_twitter.Twitter1204\_REMOVENOISE.find():

workQueue.put(word)

queueLock.release()

while not workQueue.empty():

pass

exitFlag = 1

for t in threads:

t.join()

print ("Starting Main Thread!!!")

endtime = datetime.datetime.now()

print("Time used:", endtime - starttime)

After that, we adopted multiprocessing method to make it less time-consuming through creating a Pool which is an embedded function of multiprocessing package. It can reduce to one of ten time to finish the task with 50 number of subprocesses.

if \_\_name\_\_ == "\_\_main\_\_":

client = pymongo.MongoClient(host='localhost', port=27017)

db\_twitter=client.TWITTER\_DB

starttime = datetime.datetime.now()

### Filling queue

print("Starting Filling Queue!\n")

manager = multiprocessing.Manager()

queue = manager.Queue()

for word in db\_twitter.Twitter1204\_REMOVENOISE.find():

queue.put(word)

client.close()

print("Finished Filling Queue!\n")

### Generate processes pool and apply tasks

pool = multiprocessing.Pool(processes = 30)

while not queue.empty():

item = queue.get()

pool.apply\_async(process\_data, (item, ))

print("Mark~ Mark~ Mark~~~~~~~~~~~~~~~~~~~~~~")

pool.close()

pool.join()

print("Sub-process(es) done.")

endtime = datetime.datetime.now()

print("Time used:",endtime - starttime)

The multiprocessing method is better than multithreading, however it still consumed too long time. Consequently, java code is developed, because the most time consumption is on calling java which spent 99% time to start a JVM and load jar packages into it. Java program can avoid this problem very well. At the end of this trying, less than 2 minutes are used to finish the whole task.

$ java -mx8G -cp /scratch2/project/stanford-ner-2018-10-16/stanford-ner.jar:bson-3.9.1.jar:mongodb-driver-core-3.9.1.jar:mongodb-driver-legacy-3.9.1.jar:mongodb-driver-sync-3.9.1.jar: MongoDBJDBC

Loading classifier from /scratch2/project/stanford-ner-2018-10-16/classifiers/english.all.3class.distsim.crf.ser.gz ... done [1.0 sec].

Connect to database successfully

Collection test choosing successfully!

All documents insert sucessfully!

Dec 18, 2018 3:03:57 AM com.mongodb.diagnostics.logging.JULLogger log

INFO: Closed connection [connectionId{localValue:2, serverValue:849790}] to localhost:27018 because the pool has been closed.

Processing Time： 106465ms

### 5.1.3 Implementation of Dice Coefficient method

The dice coefficient can be used for calculating the overlaps between two observations (Hersh et al., 2006).

In the research, the dice coefficient is applied for calculating the similarities between entities with overlapping tweets ID. For instance, If there are two entities A, B, each of them combines with several tweets ID, is the numbers of tweets ID overlapping in both A, B entity, , are the numbers of tweets ID in each A, B entity:



While the Dice (A, B) > 0.4, entity A and B would be considered as high coefficient then would be put into the same group.

As the result, there are … tweets in groups…

### 5.1.4 Implementation of MinHash LSH function

MinHash LSH is a combination of MinHash algorithm and Locality-sensitive hashing (LSH) method to estimate the Jaccard similarity coefficient with lower query cost than solely using MinHash (Github,2018).

(then detailed process and result)

### 5.1.5 Comparison of two methods

(from the result, effect or running time)

## 5.2 Data visualization

For better explanation, this section is designed to visualize after-grouping data including both tweets and news. Folium, a Python library which can create Leaflet maps (DOMINO DATALAB, 2015), is implemented on tweets. And Python Imaging Library (PIL) and Worldcloud library are imported to represent news visually (Python Graph Gallery, 2017).

### 5.2.1 Tweets distribution

(picture and its description)

### 5.2.2 News key words

(picture and its description)

**Chapter 6**

# Evaluation

**Chapter 7**

# Conclusion

# References

ATEFEH, F. & KHREICH, W. 2015. A Survey of Techniques for Event Detection in Twitter. *Computational Intelligence,* 31**,** 132-164.

*Beautiful Soup Documentation* [Online]. Crummy. Available: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> [Accessed 8 December 2018].

Doulamis, N.D., Doulamis, A.D., Kokkinos, P. & Varvarigos, E.M. 2016, "Event Detection in Twitter Microblogging", *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2810-2824.

Github. 2018. *Tweepy: Twitter for Python!* [Online]. Github. Available: <https://github.com/tweepy/tweepy> [Accessed 8 December 2018].

Github. 2018. *datasketch: Big Data Looks Small* [Online]. Github. Available: <https://ekzhu.github.io/datasketch/index.html#datasketch-big-data-looks-small> [Accessed 13 December 2018].

Goh, H., Soon, L. & Haw, S. 2015, "Automatic discovery of person-related named-entity in news articles based on verb analysis*"*, *Multimedia Tools and Applications*, vol. 74, no. 8, pp. 2587-2610.

Hersh, W.R., Bhupatiraju, R.T., Ross, L., Roberts, P., Cohen, A.M. & Kraemer, D.F. 2006, "Enhancing access to the Bibliome: the TREC 2004 Genomics Track", *Journal of biomedical discovery and collaboration*, vol. 1, no. 1, pp. 3-3.

McMinn, A. J. & Jose, J. M. Rea-Time Entity-Based Event Detection for Twitter. *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, 2015 France. Springer, pp.65-77.

MONGODB. 2018. Available: <https://www.mongodb.com/> [Accessed 9 December 2018]

MONGODB. 2018. *Documentation: createIndexes* [Online]. Available: <https://docs.mongodb.com/manual/reference/command/createIndexes/#dbcmd.createIndexes> [Accessed 7 December 2018].

Liu, H., Ge, Y., Zheng, Q., Lin, R. & Li, H. 2018, "Detecting global and local topics via mining twitter data", *Neurocomputing,*vol. 273, pp. 120-132.

OMNICORE. 2018. *Twitter by the Numbers: Stats, Demographics & Fun Facts* [Online]. OMNICORE. Available: <https://www.omnicoreagency.com/twitter-statistics/> [Accessed 5 December 2018].

Python Graph Gallery. 2017. *Wordcloud* [Online]. Python Graph Gallery. Available: <https://python-graph-gallery.com/wordcloud/> [Accessed 6 December 2018].

REITZ, K. *Requests: HTTP for Humans* [Online]. Available: <http://docs.python-requests.org/en/master/> [Accessed 8 December 2018].

Roos*.*2015. *Creating interactive crime maps with Folium* [Online].Domino datalab. Available: <https://blog.dominodatalab.com/creating-interactive-crime-maps-with-folium/> [Accessed 5 December 2018].

Tariq, A., Karim, A. & Foroosh, H. 2017, "NELasso: Group-Sparse Modeling for Characterizing Relations Among Named Entities in News Articles", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 10, pp. 2000-2014.

The Stanford Natural Processing Group, T. S. N. *Stanford Named Entity Recognizer (NER)* [Online]. Stanford University Available: <https://nlp.stanford.edu/software/CRF-NER.shtml> [Accessed 9 December 2018].

Twitter Developer. 2018. *Tweets Objects* [Online]. Available: <https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html> [Accessed 20 November 2018].

Twitter Developer. 2018. *Tweets Objects* [Online]. Available: <https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/geo-objects.html> [Accessed 20 November 2018].

Zhou, X. M. & Chen, L. 2014. Event detection over twitter social media streams*. THE* *VLDB Journal,* 23**,** 381-400.