DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Targeted Data Enrichment for an Existing Large Sparse Dataset

Zhouyang Xue

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Targeted Data Enrichment for an Existing Large Sparse Dataset

Zielgerichtete Datenanreicherung in einem großen dürftig besetzten Datensatz

Author:	Zhouyang Xue
Supervisor:	PD Dr. Georg Groh
Advisor:	Dietrich Trautmann M.Sc
Submission Date:	15.12.2016

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.12.2016 Zhouyang Xue

Abstract

The current work collected a dataset on the interaction between people to be used in future research on sentiment analysis. Based on messages sent from an individual to others, a crawler is build to able to identify individual with high likelihood of response. Based on a random forest model that analyzes features in message and frequent term count analysising the text body, the crawler was able to detect replyied individuals with 75% of acurracy. This allowed us to build a dense and strong connected social network and thus can works for more detailed analysis social researches.

Contents

1. Introduction

1.1. Introduction

In the past few years, sentiment analysis has been a popular and interesting topic in natural language processing research field. Most of the applications focus on extracting subjective information such as feedback, reviews or opinions on a particular subject or product.

Naturally, such studies require a large amount of data about emotional expressions. This leads the researchers to gather dataset from social network platforms, where people express and share their opinions and daily activities see [Yua16; Tha]. Moreover, it is easy to collect datasets from these platforms when the target is a specific topic or subject, not just because of the APIs provided by the platforms, but because of the one-directional nature of the communication: users post messages about that topic.

Differently from previous work, our research focuses on sentiment analysis among people, particularly the messages transmitted between two individuals. We are focusing on bidirectional communication, where users message each other. This can involve more complex dialog, in structure and length, and this requires gathering the complete interaction in order to be able to detect the exact emotions expressed.

Each dialog can be influenced by the relation between participants, and the expressions depend on habits or style of each person. In particular case, the dialogs between two friends can be negative but still occur. To detect these influences, it requires the complete interaction between two individuals. This is the reason why data gathering is no longer intuitive.

This thesis is focused on data collection on communication for future sentiments analysis work. We start from an existing dataset collected from previous work [Kos13] as base and refrence, then build an efficient crawler to expand the amount of data, with

the purpose of searching communications between users.

1.2. Related Work

The source platform of the existing dataset is Twitter, which provides a friendly API and has unique characteristics as a microblogging platform. The purpose of this particular dataset is not for sentiment analysis, so filtering and preprocessing is needed. In any case, it provides an excellent base source for this thesis.

Most of the works which gather sentiment dataset from Twitter, see [PP10; Nak+16; KWM11] their goal is analysing positive or negative expressions to a certain popular topic. And as most of other works, the collection was simply done by randomly getting post related with the chosen topic from random users. Such a method ignores the evolution of the opinion of a single person and discussions on the topic.

1.3. Motivation and Objective

To guarantee the quality of the dataset, the collection of this thesis is aimed at the users that communicate frequently with each other, and where the communication is bidirectional. The primary objective for this is collecting dataset and preparing them for future work on sentiment analysis between pairs of users.

There are two part of this thesis, starting from the existing dataset:

- Preparing dataset for sentiments analysis between people:
  1. Generate features indicating the sender and receiver of each message.
  2. Group messages of the same context, i.e., conversation grouping.
- Build a crawler aiming to identify messages between users that frequently communicate with each other.

1.4. Thesis structure

In chapter 2, sources and tools, methods and data mining techniques used in this thesis is presented.
In chapter 3, describe analysing and evaluation the existing dataset.
In chapter 4, the development of a general crawler.
In chapter 5, several method and process to identify possible frequent communicating users.
In chapter 6, presents discussing on further work and presenting the conclusions.

2. Background and Method

2.1. Twitter

Twitter is a social networking microblogging platform where its users can send and read text-based messages called "tweets". And Twitter is currently one of largest social networking platform with 300 million active users with 500 Million new tweets being public per day[2].

As a microblogging platform, every microblog tweets are limited to 140 characters or less. This size motivates user to focus and clever use of language in the content text. Unlike other social networking platforms (e.g. Facebook [3]), the default setting of tweets is publicly visible, but senders can restrict the message delivery to a limited group of users.

Another aspect is Twitter allows users to build one-directional relation with other interested users by using "following" option, this option allows users to have access and receive notification of tweets of interested users. And this supposes less restriction to form communications and dense social network.

2.1.1. Tweet's Terms

Essential terms about Twitter involve in this thesis:

Tweet 140 character limit message, can contain website link, emoji, image and audio.

User In the user interface, every user has a name to display and a unique username to identified themselves, and users have a profile with the fields: location, self-description link and a picture.

[2]according to social media statistic website `https://www.socialbakers.com/statistics/twitter/`
[3]`https://www.facebook.com`

Follower A user is able to choose the option "following" to another user and become a follower, in order to build a direct relation. The follower has notification and access to following user's tweets.

Protected user "Protected" is a security and privacy option, once a user actives this option, only the follower of the user can have access to the user's tweets. And also to become a follower of the protected user requires the user's approve.

Hashtag Users can add the hashtag symbol(#) before a relevant keyword in tweet's message body to index keywords or topics of the tweet.

Mention Tweet a mention tweet is a tweet that contains another user's username in the body of the Tweet text with the @ symbol before with purpose to address specific users. And a mention tweet works as a direct message from the author to the mentioned user.

Reply Tweet a tweet that replies another tweet, beginning with the "@username" in the text body to indicate the replied user.

Retweet Tweet a tweet aim to re-share content of another tweet, however, Twitter also allows users to add more context in the text to express any opinion or reference to another user of the re-share tweet.

2.1.2. Twitter API

Twitter has a friendly REST API, providing different functions for get access to Twitter data. And each of then function contains a limitation of the number of request per 15 minutes, called "rate limit". This restriction is used Twitter for the protection of abuse requesting, and it also restricts the work of this thesis in order to crawl tweets. This limitation has characteristics of overlapping between methods.

We are interested in mention tweets, and there are several functions can be used:

Function	Rate limit	Description
GET statuses/*user_timeline*	300	This method can return up to 3,200 of a user's most recent Tweets
GET search/*tweets*	450	Returns a collection of relevant Tweets created in last seven days matching a specified query
GET statuses/*lookup*	200	Returns Tweet JSON objects for up to 100 Tweets per request, as specified by comma-separated values passed to the id parameter. Tweets)

Table 2.1.: Twitter REST API functions

Another option is using stream API, it is similar of GET search/tweets and consist in establishing a connection to a streaming endpoint. It does not have any rate limit. The disadvantage is, as mentioned in the name of API, it only returns tweets created in last moment.

Results of the function are JSON objects that represent all relevant information of each tweet, see example in the following listing:

```
"created_at": "Mon Sep 03 13:24:14 +0000 2012",
"id_str": "242613977966850048",
"entities": {
  "hashtags": [],
  "user_mentions": [
    {"name": "Jason Costa",
      "id_str": "14927800",
      "id": 14927800,
      "screen_name": "jasoncosta"
    },
    {"name": "Matt Harris",
      "id_str": "777925",
      "id": 777925,
      "screen_name": "themattharris"
    },]},
"in_reply_to_user_id_str": "14927800",
"text": "@jasoncosta @themattharris Hey! Going to be in Frisco in
October. Was hoping to have a meeting to talk about @thinkwall if
you're around?",
"retweet_count": 0,
"in_reply_to_status_id_str": null,
"id": 242613977966850048,
"in_reply_to_user_id": 14927800,
"user": {
  "name": "Andrew Spode Miller",
  "created_at": "Mon Sep 22 13:12:01 +0000 2008",
  "location": "London via Gravesend",
  "id_str": "16402947",
  "utc_offset": -18000,
  "id": 16402947,
  "lang": "en",
  "geo_enabled": true,
  "time_zone": "London",
  },
"in_reply_to_screen_name": "jasoncosta",
"in_reply_to_status_id": 2342343
```

Figure 2.1.: Tweet JSON object example

2.2. Decision tree

Decision tree is a data mining technique used for both classification and regression. The idea is partitioning the feature space into regions and assigns a the correspond class to the region. Each partition is based maximizing the difference of information gain before and after of the split. With purpose to divide data into increasingly pure partitions.

The resulting model can be represented in a tree structure, for example, see Figure 2.2, where the leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

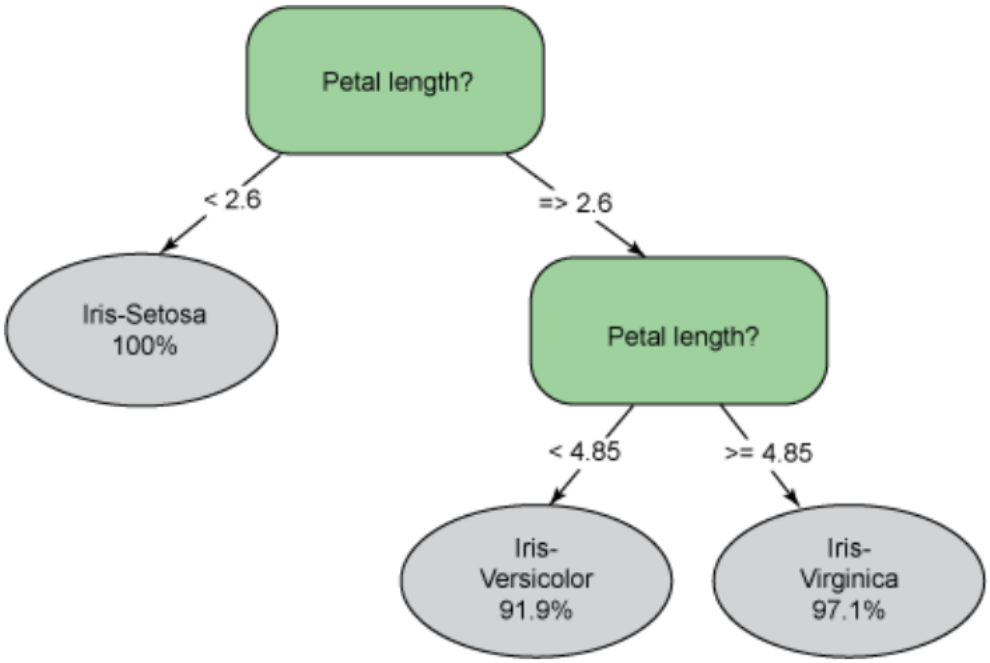

Figure 2.2.: Decision tree example from iris dataset

2.3. Random Forest Classification

Random Forest is an ensemble machine learning[1] algorithm, by combining multiple decision trees. It consists in constructing multitude decision trees on different subsample of the training dataset. Therefore to predict a new input, resulting output class is the mode of the result of all the trees. As a result, this algorithm improve the predictive accuracy and control over-fitting

2.4. Frequency Term Count

Frequency Term(TF) Count is a numerical statistic that measures the importance of a term in a document by counting the frequency of each term. Commonly used in text mining as a weighted factor.

Normally this feature is combined with inverse document frequency (IDF), where IDF are the terms that are often used in a language, but does not contain any information in the context, as stopwords in English, e.g. "the". The purpose of IDF is to avoid giving importance to those terms appear everywhere and do not characterize the topic or context of the document.

[1] Ensemble methods consist in combine multiple learning algorithms to obtain better predictive performance

3. Analysis and Evaluation of the Existing Dataset

3.1. Existing Dataset

As mentioned before, we starts working with an existing dataset from previous work [Kos13] in 2012. The aim of the crawling was to build a densely connected graph of a social network, by filtering network hubs (people with lots of friends and followers) and only crawling English speakers.

The crawling of the dataset was based on an Breadth-first, by intent to crawl the complete social network for each user. It starts with a obvious English speaker user in a queue and runs a loop to gather tweets from each user and all their friends and followers.

The dataset is stored in TUM server's MYSQL database with fields:

Field	Type	Field	Type
ID	bigint(20)	IN_REPLY_TO_SCREENNAME	varchar(255)
CRAWLED_AT	datetime	IN_REPLY_TO_STATUS_ID	bigint(20)
FIRST_CRAWLED_AT	datetime	IN_REPLY_TO_USER_ID	bigint(20)
CREATED_AT	datetime	RETWEET_OF_SCREEN_NAME	varchar(255)
IS_RETWEET	bit(1)	RETWEET_OF_STATUS_ID	bigint(20)
JSON	blob	RETWEET_OF_USER_ID	bigint(20)
LATITUDE	double	STATUS_TEXT	varchar(255)
LONGITUDE	double	USER_ID	bigint(20)
RETWEET_COUNT	bigint(20)	CURRENT_SCREEN_NAME	varchar(255)

Table 3.1.: Tweets fields stored in TUM's service

3.2. Data Processing Functions

This section presents functions to process tweets for the first goal of this thesis, to prepare the dataset for future work in sentiment analysis between users. And according to Section 1.3, the preparation consists:

- Extraction of the mentioned user IDs for each mention tweet, to identify receiver of each message.
- Grouping tweets into conversations, separating tweets into corresponding contexts.

As we are working in a large and sparse dataset, with JSON object compressed, the total size of the dataset archives 240 GB. So the efficiency is the primary requirement for these functions.

3.2.1. Mention List Generation

This process generates lists of users mentioned in each mention tweet. And the required information is extracted from Tweet JSON object, in the field *mentioned_users*.

The main issue in dealing with this process is how to store the mention lists. Because a mention tweet can mention more than one user, the relations between a tweet to its mentioned users has cardinality one-to-many. And the storage of this information needs to be considered carefully by the size of the table mentioned before

There are several considerations:

1. Transform user mentioned list concatenating *user_ID* into a string, separate by a space.
   - Advantage: saving memory as it only requires one extra new field in the tweet's table
   - Disadvantage: it is complicated to use native operation of MySQL for processing concatenate field or accessing any of user_id in a concatenate chain.

2. Store user mention list in different table from tweet table, and the table has the fields described in Table 3.2 :

   - Advantage: easy to process and implicates to reusability.
   - Disadvantage: the consumer of space and slow down the query access, as the information are separated in different, and building indexes in this size of the table is costly.

Field	Type	ContainsNull
ID	bigint(20)	NO
MENTIONED_USER_ID	bigint(20)	NO
USER_ID	bigint(20)	NO

Table 3.2.: mention list table description

3. Use graph database: the graph database works for one to many relation fields, but it is not in the scope of this thesis.

All tree options accomplish with the part of the goal, to be able to access the receiver of each tweet. Our decision is working with the second case as it gives conformity to later works in dataset evaluation and also for considering reusability for future sentimental analysis work, this option is most intuitive to understand.

3.2.2. Conversation Grouping

There is no directed way to group tweets into conversations, the only indication of conversation of a tweet is the field IN_REPLY_TO_STATUS_ID which is the identifier of preview tweet (replied tweet) of each reply tweets.

The intuitive implementation is for each tweet, access one by one of each preview tweet until reaching the tweet that initiates the conversation (tweet with field IN_REPLY_TO_STATUS_ID null), and then assign a unique conversation identifier to all the tweets of the conversation. This implementation can not develop with MySQL

native query, it requires a SQL API to provide a higher programming language environment such as Python.

This design does not guarantee the efficiency, when a function scans the entire table and try to match, make changes for each read row iteratively, this produce a N+1 problem[1]. Especially when the input table is large and sparse, single random position access is computationally costly if it is done. The solution is use native SQL query (UPDATE and JOIN statement) to do read, match and writing in single operation and target to entire database.

In order to do conversation grouping only with native MySQL query, two queries need to be used. The first query assigns the root tweets of conversation a unique conversation id, the root tweet is found by using JOIN statement to match each root tweet (with field IN_REPLY_TO_STATUS_ID null) with replying tweet.
The second query succeeds all the conversation ids to replying tweets. The second query needs to be run fro multiple times, until there is not more successor tweets.

3.3. Evaluation Criteria

Before to start the evaluation and summary the existing dataset, a standard of evaluation need to be established. As the goal of this thesis is to find users that have frequent interaction with each other we define follow standard:

A frequent interaction between two users A and B is defined as:

let T_{AB} be the number of total tweets between A and B, $T_{A\to B}$ be the number of tweets of A mentioning B and $T_{B\to A}$ the number of tweets of B mentioning A.
and we define 2 parameters:

$$MinimalTotalTweets \in \mathbb{N}_{>0} \tag{3.1}$$

$$MinimalTweetsRate \in [0,1] \tag{3.2}$$

[1]In database optimization term, N+1 problems is fetching table during a select in a suboptimal way

1. The total number of tweets between users A and B T_{AB} is the number of tweets of A mentioning B $T_{A\to B}$ add the number of tweets of B mentioning A $T_{B\to A}$ and is bigger equal than the MinimalTotalTweets.

$$T_{AB} = T_{A\to B} + T_{B\to A} >= MinimalTotalTweets \tag{3.3}$$

2. The minimum number of tweets between the number of tweets of A mentioning B $T_{A\to B}$ and the number of tweets of B mentioning A $T_{B\to A}$ is bigger equal than 10% of the total number of tweet between users A and B T_{AB}.

$$min(T_{A\to B}, T_{B\to A}) >= MinimalTweetsRate \cdot T_{AB} \tag{3.4}$$

The first criteria guarantee the interaction is frequently and the second criteria ensure bidirectional communication. In this thesis, to not be too restrictive and we agree on this values for the parameters is already enough for future work of sentiment analysis :

$$MinimalTotalTweets = 30 \tag{3.5}$$

$$MinimalTweetsRate = 0.1 \tag{3.6}$$

3.4. Processing Dataset

In this part, details about import the existing dataset from TUM's service and apply the processing method is presented.

First, we import the existing dataset from department server, due to the size of the dataset 240G, and the working hardware of the thesis can not store entire dataset, we only import tweets that accomplish the goal the thesis, which are:

- @ Tweets: as the mention list is not built yet, we import tweets that contain symbol "@" in the status text. Further cleaning needs to be done, because "@" symbols can appear as part of an email address or emoji and not a mention.
- Tweets without Mention but belongs to a conversation for conversation grouping.

Also, we are only considering fields in interested for the objectives of this thesis:

- STATUS_TEXT: message text.
- IN_REPLY_TO_STATUS_ID / IN_REPLY_TO_USER_ID : for conversation grouping.
- RETWEET_OF_STATUS_ID / RETWEET_OF_USER_ID : effects the interpretation of status_text.
- USER_ID: the sender of the tweet.
- CREATED_AT: for ordering the tweets.
- ID: to identifier each tweet.

After importing, we found out that particular mention tweets did not contain JSON objects. We decide to ignore these tweets as the mention list can not be build and recover the JSON objects of tweets is very slow by the limit rate of Twitter API (1200 Tweets per minutes), yet the implementation of the function to recover them is done.

Then we process the dataset by generating mention list and grouping in conversations as described in section 2. Once, the mention list is generated, we clean @ tweets that do not mention any user.

The final table to be considered for later works has fields:

Field	Type	Null
ID	bigint(20)	NO
CREATED_AT	datetime	NO
IN_REPLY_TO_STATUS_ID	bigint(20)	YES
IN_REPLY_TO_USER_ID	bigint(20)	YES
RETWEET_OF_STATUS_ID	bigint(20)	YES
RETWEET_OF_USER_ID	bigint(20)	YES
STATUS_TEXT	varchar(255)	YES
USER_ID	bigint(20)	NO
CONVERSATION_ID	int(11)	YES

Table 3.3.: Imported tweet table's fields

3.5. Dataset Summary and Evalutation

From TUM's server, 106,040,965 mention tweets are imported, and between them, 40% are reply tweets. With these size of dataset that corresponds single interactions, a high number of frequent interactions between users are expected. Yet the average of the number of tweets per user is low, and it might inference the expectation.

Basic summary about existing dataset and imported dataset:

Entire existing dataset	
Tweets with JSON object	211,681,918
Users	3,737,917
Average of number of tweets per user	59
Tweets without JSON object	11,331,926
Imported dataset	
Mention tweets	106,040,965
Reply tweets	42,758,430
Mention retweets	28,280,803

Table 3.4.: Overview about the existing dataset

Statistics obtained after conversation grouping and generating pair user list with frequent interaction: tabular

Conversations	2.906.513
Tweets involved in a conversation	8.381.500
Multiple-user conversations[1]	2.871.388
Average number of participating-user of multiple-user conversations	2.28
Pair of frequent interaction Users	85.812
Tweets between frequent interactions	5.635.012
Conversations between frequent interactions	519.268
Tweets in conversations between frequent interactions	1.455.457

Table 3.5.: Existing data processed summary

To evaluate the effectiveness of crawl we set up the crawl rate value by rate of tweets that satisfied the frequent interaction criteria:

$$CrawlerRateValue = \frac{Tweetsoffrequentinteractions}{Totaltweets} = \frac{5.635.012}{211.681.918} = 0.0266 \quad (3.7)$$

Only 2.66% of tweets crawled accomplish the frequent interactions criteria. And we can observe from **??**, this 2.66% of tweets contains 15% of conversations, which satisfies the goal of the thesis in communication term. Also, this approves the value of parameters are well defined. The further works of this thesis is to build a crawler with higher effectiveness.

Another interesting aspect is that most of the conversations have more one user participating, due to the completeness of crawled network. As this can guarantee a conversation is a single bidirectional interaction, this particular aspect can be an indication for later works, to focusing on tweets conversation in order to gather more interactions.

4. Crawler Development

This chapter describes the design of the crawler, used to gather tweets from Twitter. For this purpose, the main task of the crawler is to interact with the Twitter API and the local database.

Interacting with the Twitter API

The ultimate goal of this thesis is to collect frequent interactions between users, and this requires gathering tweets from users that mention each other in their tweets frequently. As described in section 2.1.2, the Twitter API does not supply functions to get mention tweets between two users. However, it offers various functions to gather tweets from any single user. Hence, the task to identify the users interacting frequently is accomplished as part of this work. This is described in chapter **??**.

The object of the crawler is to gather as many as possible tweets from each user to increase the likelihood of finding frequent interactions between users. This is the reason the crawler uses the function "GET statuses/user_timeline" of the Twitter REST API mentioned in section 2.1.2, which has the highest rate limit and provides oldest tweets compare to the other relevant functions.

The main challenge in interaction with the Twitter API is to manage the rate limit, the service "GET statuses/user_timeline" only provides 200 tweets per minute. According to section 3.5, applying the crawler developed in Koster's work [Kos13], in average only five tweets out of 200 were tweets of frequent interactions. To increase this percentage and therefore make the crawling process more efficient, a better crawling strategy is developed, as presented in next chapter. Another approach is using multiple clients accessing the Twitter functions simultaneously.Twitter allows this, assuming each person can have several accounts. In order to exploit this, the crawler needs to be implemented in multiprocessing.

Storing Datasets

Tweets downloaded from the Twitter function are of the JSON object format (see **??**), and from each JSON object is extracted the tweet features describe in Table 3.3 and mentioned user list (see Table 3.3). In addition, information about the author of each tweet also is extracted and stored, which will be used in future works. And to guarantee the completeness of the datasets, JSON object is also being stored.

The main issue of storing is always the storage space, each JSON object has at least 3 KB of size, for storing one million tweets, more than 3 GB of space are required. As a consequence, those datasets are stored in several locations. In order to adapt the storage of the datasets according to hardware situation, the software pattern single responsibility principle[1] is applied in the process of storing datasets, using different functions to manage each dataset.

4.1. Design of the Crawler

A detailed presentation of the crawler is presented in this section, the process of crawling is split into tasks defined previously and each task is implemented in a Python function.

The complete workflow of the crawler as follows:

1. The input of crawler is a list of user IDs, which correspond to the same input of function "GET statuses/user_timeline". Hence, the first task is splitting the input list into chunks and assigning them equally to available clients. Each client is an independent thread with a distinct Twitter account allowed to access the Twitter REST API and is in charge for rest of the tasks.
   Also the splitting based on steps, the resulting chunks are sequences of items with index $x = i + n * k$ such that $0 <= n < (j - i)/k$. Where n is the length of the entire list and k are the number of chunks. In other words, each $chunk_i$ contains user ID of the position are $i, i + k, i + 2 * k, i + 3 * k$and so on

[1] The single responsibility principle states that every module or class should have responsibility over a single part of the functionality provided by the software. `https://en.wikipedia.org/wiki/Single_responsibility_principle`

The reason for applying this splitting is to keep original order of input list, and this benefits in case the input list consists in a ranking, where users in the earlier position should be crawled first.

2. The second task is connected to Twitter Rest API, and call service "GET statuses/user_timeline" for each user. And parse each resulting JSON object.

3. The third task is compose by 3 part:

   – Store tweet with mentioned user list: Using Peewee API to connect database and send the dataset.

   – Store user information: same process as tweets.

   – Store JSON object: this part is particularly different from other two, is the more common situation the storage location is a file from an external disk. Since the running environment is multitasking, file lock is provided for all threads when editing the storage file.

5. Data Crawling

This chapter presents approaches to find out users with a higher likelihood of having a frequent interaction, to generate the input list for the crawler described in the preview chapter. The approaches are evolving iteratively by four milestones. Moreover, for each milestone, we crawled a certain number of users to evaluate and approve the used methods.

The first milestone is based on the pair of users which already interact frequently in the existing dataset. The rest of milestones are focused on exploiting new dataset from milestones, particularly estimate the likelihood of the mentioned users(not crawled) by the new crawled users mention back.

5.1. 1th milestone: Basic Crawl

As described in the Section 3.5, the existing datasets contain 85.812 pairs of users with frequent interaction. Assuming if two users frequently communicate in the past, they might interact to each another also at present. We decide to crawl to those pairs of user with expectation that those pairs of users keep their frequent interactions relations until now.

Initially, our plan was to crawl the entire 85812 pair of users, yet due to the schedule, we were only able to crawl 8.997 of users, which are approximate 4497 (8997/2) pair of users. Yet, the input list (8997 users) for crawler function is a list with all pair of user's id, so in each position of the list has two user's id and the list is order by the first user_id. As a consequence, the first users of each position are repeated several times in the list.

If we build a graph using the input list where the node is user and edges are frequent interactions. We can found out easily stars[1] in the graph, see Figure 5.1.

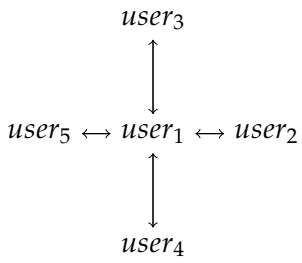

Figure 5.1.: Star graph: the node represents user and edges represents frequent interactions

We can observe, for each N nodes (users) there are N-1 edges (frequent interactions). In another word, by crawling N users we can get up to N-1 frequent interaction relations, instead of N/2 if the input list was randomly pairs of users.

This observation is particularly important for this milestone, as it indicates the expected number of frequent interaction in new crawled dataset is not longer the half of the number of the crawled user, as they are not independent pairs of users. Instead, from 8.997 crawled user, there are already 5273 frequent interactions between them in existing dataset, together with 863.525 mention tweets.

5.1.1. Result of Crawling and Evaluation

By the 8.997 crawled user, crawler gathers total of 26,437,355 tweets. Between tweets, 66% are mention tweets, 16% more compare than existing dataset. So these crawled users are more communicative than the average.

The following table represents statistics generated after Conversation Grouping, and we can observe only 0.3 (1.2%) million of 27 million tweets is involve in a conversation,

[1] In graph theory, a star S_k is the complete bipartite graph $K_{1,k}$: a tree with one internal node and k leaves, `https://en.wikipedia.org/wiki/Star_(graph_theory)`.

Tweets	26,437,355
Users	8,997
Average of number of Tweets per user	2,938
Mention tweets	17,709,152
Reply tweets	7,821,989
Retweet tweets	6,442,164
Repeated Tweets crawled in the existing dataset	1,043,681

Table 5.1.: First milestone common summary

and also only half of conversation have more than one participating, this due to incompleteness of the crawled social network, where not all participants of the conversations are crawled.

Conversations	115,257
Tweets involved in a conversation	331,000
Multiple-user conversations[i]	56,200

[i]At least two users publish tweets in the conversation

Table 5.2.: Conversation summary of the first milestone

New statistics about frequent interaction between crawled user of this milestone, these numbers are important to evaluate if the crawling achieves the expected frequent interaction:

Frequent interactions	3,215
Number of tweets of frequent interactions	741,236
Frequent interactions also in existing dataset	1,934
Tweet of Frequent interactions also in existing dataset	632,315
New Frequent interactions	1,281
Tweets of new Frequent interactions	108,921

Table 5.3.: Data Processed summary of the first milestone

Evaluation of the Crawling

Comparing the result of crawling with expected value, 1,934 of 5,273 interactions still satisfied the criteria of frequent interaction, but the number of tweets per interaction has a significant increased, with 632,315 tweets in 1,934 frequent interactions compare to 863,525 tweets in 5,273 frequent interactions from the existing dataset.
And together considering 1,281 new frequent interactions formalized with the same group of user in the new dataset, a development of the social network is observed. Therefore, this approves the theory of this milestone.

And to evaluate the efficiency of this crawling, we compute the crawler rate value with the number of tweets of frequent interaction and total crawled tweets of the new dataset. The result is 2.8%, which is slightly better than crawling of existing dataset.

$$Crawler\ Rate\ Value = \frac{Tweets\ in\ frequent\ interactions}{Total\ tweets} = \frac{741,236}{26,437,355} = 0.028 \quad (5.1)$$

But considering that this milestone, unlike the crawling of existing dataset, it is not crawling the entire network of each user. In fact, for most of the crawled users, this milestone is considering one or two interactions. To be more accurate, between 3,215 frequent interactions with 3,957 users participating, 3,385 users has less than two frequent interactions to another user.

To be more precise, more detail statistics about mention tweets are presented in the following table:

Mention tweets	17,709,152
Mention tweets send by crawled users to crawled users[i]	3,077,252
Tweets send by crawled users to crawled users in Frequent interaction	741,236
Tweets send by crawled users to crawled users not in Frequent interaction	2,336,016
Mention tweets send by crawled users to uncrawled users[i]	16,822,114

[i] these two types of mention tweets is overlapping as a mention tweets can mention several users.

Table 5.4.: Summary about mention tweets crawled in this milestone respect crawled users of this milestone

From above table, the number of the mention tweets considered in find frequent interaction between users crawled in this milestone are 3,077,252, which is only 17.6% of the total mention tweets. The rest of the mention tweets are sent to users that are not crawled yet, and in order to exploit this crawled datasets, and also use them as support, the next milestones focus on estimating the likelihood of these uncrawled users mentions back. As long as, any of these uncrawled mentions back, more new frequent interaction is found.

5.2. 2nd Milestone: Model Based on Tweet's Feature

As described in the section 5.1.1, this milestone focus on the expansion of the first milestone, particularly in uncrawled user mentioned by crawled user of the dataset of the first milestone. The idea is ranking them by the likelihood of they mentioning back to the user that is already crawled, in this way to form new frequent interactions.

With the purpose to obtain the likelihood of each mentioned uncrawled user, this milestone focus on observing features relevant of each tweet. In order to find out patterns that differentiate mention tweets in frequent interaction and not in.

So this milestone is about building a classification model to label mention tweets if they belong to frequent interaction. As from preview milestone, we can extract mentions tweets within and not in the frequent interaction. As so, the model is built in a supervised learning.

Therefore, these are features of a mention tweet with its hypothesis that we considered might have influence in the classification:

- If the tweet is a reply tweet: reply tweets are a basic indication of interactions.
- If the tweet is a retweet: users tend re-shared interesting content in so it might initiate a discussion. See [BGL10]
- The created local hour of the tweet: users are more active during not working times.
- Number of favorites of the tweet: this feature indicates the level of the influence of tweets. See [Cha+10]
- Number of retweets of the tweet: this feature indicates the level of the influence of tweets.
- Time passed from last mentioned tweet to the same user if a user mention another user several times in a very short time, it can be an indication of interactions.[1]

Due to the limitation of hardware and computational cost, we are not able to consider all the feature defined above to build the model. Especially, to compute time of last

[1]Statistics provided from, indicates most replies happends one hour after twitter post https://sysomos.com/inside-twitter/twitter-retweet-stats

mentioned tweet to the same user requires access all the mention tweets of the user. Therefore, we only used features can be extracted from the fields in the imported table see Table 3.3:

- If it is a reply tweet: extract from field IN_REPLY_TO_STATUS_ID
- If it is a retweet: extract from field RETWEET_OF_STATUS_ID
- The created local time hour: extract from CRAWLED_AT(in utc_time) and combine with the field utc_offset of the author to generate local hour.

As a consequence, considering less feature suppose less accuracy of the mode. To compensate this lack is to use a larger input to predict. And then build a larger ranking list, so event there is less accuracy, but the result ranking is large enough, that allows us only to consider the very first positions of rankings. For example use 1,000,000 mentioned users to predict and build ranking list, then only use 1,000 users of top rankings.

Considering the input features for the model are two boolean fields (if is reply and if is retweet) and one ordinal field (local hour of creation time), and although the exact probability is not interested in this case, we decide to use the decision trees classification model. And applying into random forest to avoid over-fitting.

5.2.1. Processing of Modeling and Ranking Users to Crawl

Processing of Modeling

First, the training dataset is build with mention tweet between user crawled in the first milestone, see table. The between these tweets, we label class 1 to the tweets in frequent interaction and class 0 to tweets not in any frequent interaction. Because of the field utc_offset is not available from all the users, and it is used for generating the feature created local hour, part of the tweets are filtered. Follow table summarizes the resulting training dataset:

And then fit the training dataset into the random forest classifier model.The fitting process is done using Sklearn API randomforestclassifier[1], by using a randomforest-

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

Mention tweets	2,650,952
Mention tweets in frequent interactions (Class 1)	663,913
Mention tweets not in frequent interactions (Class 0)	1987039

Table 5.5.: Summary of the training dataset for the classification of the second milestone

classifier with 100 decision trees.

Observing from Table 5.5, there is unbalance between the numbers of tweets in class 1 and 0. Also, in this thesis, the target tweets are tweets in class 1, so they are much prior than the class 0. With all these reasons, the weight of the classes need be adjusted giving more weight to class 1.

To find the reasonable difference of class weight, we decide to try several sets of class weights and evaluate the resulting model using Cross-validation[1] with ten chunks. In order to find a balance between sensitivity[2]. and specificity[3]

Class weight	Sensitivity	Specificity
0: 0.10, 1: 0.90	0.25	0.62
0: 0.15, 1: 0.85	0.34	0.82
0: 0.20, 1: 0.80	0.35	0.82
0: 0.25, 1: 0.75	0.41	0.80
0: 0.30, 1: 0.70	0.41	0.78
0: 0.35, 1: 0.65	0.44	0.75

Table 5.6.: Sensitivity and specificity respect different class weight for the random forest model

From table 5.6, we decide to use the class weight (Class 0:0.2 , Class 1:0.8) as the model has the best balance. And from the importance of features of the resulting model, see table[s], we can observe the features ifReply and ifRetweet stands the hypothesis.

[1]Cross-validation is a model validation technique by splitting training dataset into chunks, and test the model to each chunk with the model trained by another chunks.
[2]Sensitivity measures the proportion of positives that are correctly identified as such.
[3]Specificity measures the proportion of negatives that are correctly identified as such.

Resulting feature importance of the model

Feature	Importance
Created local hour	0.01979704
If is a reply	0.52091674
if is a retweet	0.45928622

Table 5.7.: Importance of each feature in random forest

5.2.2. Process of Ranking Users to Crawl

To generate the uncrawled user ranking list to crawl, we considered a group of 1,000 users crawled in the first milestone of the source users of this milestone. From this 1,000 source users, there are total 2,748,630 mention tweets send to the uncrawled users. Features vector are generated for each tweet and use the defined and trained random forest model to predicts the correspond class of each tweet.

Then for each pair of mentioned user and source user, count the number of mention tweets in class 1 between them. The reason of separate the count between the pairs of mentioned user and source user is to be accurate for later crawling evaluation, as it allows us to set the expectation of each mentioned user to crawl is at least one frequent interaction between the correspond source users.

Using the generated ranking list, we crawl for first 1,100 mentioned users with the highest count of the number of the mention tweets in class 1. And, the between those 1,100 users to crawl, 13 users are repeating, so the final number of users to crawl is 1,083.

The initial expectation of the result of this crawling is frequent interactions between the selected mentioned users and the source users, which can be a total of 1,100. However, repeating users do not influence the initial expectation by these repeating users correspond now more that one source users.

Due to account privacy level, some of the users are not crawled, the final number of crawled users is described in next section.

5.2.3. Result of Crawling and Evaluation

For this milestone, 971 users are crawled, with a total of 2,778,287 tweets. These users are also very active communicators with an average of 68%[1] of their tweets are mention tweets, comparing with 50% in average by crawled users of the existing dataset, see Table 3.4.

Tweets	2,778,287
Users	971
Average of number of Tweets per user	2,861
Mention tweets	1,894,071
Reply tweets	897,684
Retweet tweets	693,993
Conversations	36,389
Tweets involved in a conversation	78,487

Table 5.8.: Second milestone crawled dataset summary

The study of the effectiveness of the crawling of this milestone is different from the evaluation of the crawling of the first milestone and the existing dataset. As the expectation for those previous crawlings is frequent interactions between crawled users by each crawling.

Due to this crawling is an expansion of the dataset crawled from the first milestone, which corresponds a dense network. So the result of crawling a user that possible have a strong connection (frequent interaction) to a single source user of that dense network, there is high likelihood to find more interaction between the crawling user and closed social network of the single source user. This feature of the social network is not be controlled by the random forest model of this milestone.

To study of the effectiveness only about the random forest model of this milestone, without the influence of dense social network. The evaluation is focus only on the interaction between crawled user and corresponding source user.

[1]Persentage calculated by the number of mention tweets respect the total tweets from Table 5.8

Pairs of source user and successful crawled user	981
Frequent interaction between the pairs	526
Tweets between the Frequent interaction between the pairs	234,226
Tweets send by the source users to the crawled users	224,033
Frequent interaction tweets send by the source users to the crawled users	126,523
Tweets send by the crawled users to the source users	112,855
Frequent interaction Tweets send by the crawled users to the source users	106,798

Table 5.9.: Summary between source and crawled users of the second milestone

From above table, we can observe 526 (53.6%) of 981 predicted pair stands the frequent interactions. Together with 106,798 new Tweets crawled belongs to frequent interactions with the source users from the first milestone, which is 3.84% of total tweets. And this is the crawler rate value of this crawling with only the influence of the random forest model of crawling.

In addition, these 106,798 new Tweets combine with 224,033 tweets from the first milestone and turn this 224,033 tweets also into a frequent interaction tweets, which is 8.15% of tweets to be considered in the ranking process, see 5.2.2. So another point of view to evaluate the crawler rate value can be affected new frequent tweets with the sum of considered ranking tweets and new crawled tweets:

$$\begin{aligned} \textit{Crawler Rate Value} &= \frac{\textit{Tweets in the frequent interaction}}{\textit{Total tweets considered}} \\ &= \frac{224,033 \ + \ 106,798}{2,778,287 \ + \ 2,748,630} = 0.0598 \end{aligned} \tag{5.2}$$

The result is 5.98%, the effectiveness is of this crawling is 250% better than 2.26 of crawling of the existing dataset. An improvement is observed.

5.3. 3rd milestone: Model Tweet's Text

The approach of this milestone works as same as the second milestone; the goal is to crawl uncrawled user mentioned by crawled users of the dataset of the first milestone with the higher likelihood to mentioning back. In this milestone, we focus on tweets text body.

In every language, there are terms very often used in conversations, especially terms that work as confirmation or answer, for example, in English terms, "agree", "thank you", "okay"... Having these terms in the text body of a tweet can be a direct indication of the current tweet is in the conversation.

As mentioned in section 3.5, in average, a conversation is single bidirectional interaction. By having enough quantity of matching frequent terms fo conversations in the tweets text body from a crawled user mentioned to an uncrawled user, can turn this uncrawled user a good candidate to crawl.

So the basic idea of this milestone is to find a list of frequent terms in conversations, to able to ranks uncrawled users base on the count of frequent conversations terms appears in all mentioned tweets from a crawled user to them.

5.3.1. Frequent terms in conversations

To identify terms are frequent in conversations, we extract text body of 353.899 tweets that belongs to conversations from new dataset crawled in the first milestone. Using Twitter tokenize API[1] from NTLK, we extract all terms from those tweets.

And as mentioned in section 2.4, inverse document frequency terms need be considerate before counting the frequency. In a standard frequency terms analysis, inverse document frequency terms are stopwords without any meaning to the topic of the text like pronouns or common verbs in phrasal verbs, e.g. stopwords considered in the Koster's work [Kos13]. Yet in the case of this milestone, some stopwords are interested to be considering and not ignore, for example, "Okay", "indeed", "please" or

[1] Twitter parser, able to identify words, hashtags, mentions, and emojis in the tweets text body http://www.nltk.org/api/nltk.tokenize.html

any affirmation or denial.

Because of there is not source providing stopwords for conversations terms, we build the stopwords list based on 1000 frequent terms of 350.000 tweets (post) not involved in any conversations and neither contains any mentions.

And then we count the frequency of terms in the conversations tweets filtering by the stopwords list extract from non_conversations tweets and considering the 1,000 highest frequent, the result is 1000[1] frequent term used in the conversations.

Some word from the generated list accomplishes the expectation of affirmative word, e.g., "okay", "eh", "ya", and also words that initiate a question, "how's", "whatever" and "whose". Emojis like ";-)", ":P", and ":-(", and also special hashtag "ff"[2]

5.3.2. Process of Ranking Users to Crawl

For rank new user list to crawl, we considered a group of 1,000 users crawled in the first milestone as the source users, and not considered in the second milestone; the reason is to crawl more distinct dataset. And from this 1,000 users, there are total 2,532,109 mention tweets send to users that are not crawled.

Next step is to scan text body of each tweet to count the number of frequent conversation terms. And sum of total terms found in all tweets for each mentioned user. With the same purpose from the second milestone, in order to evaluate in the method of this milestone without the influence of dense network, the sum is done by separating for each pair of source user and mentioned user.

Them, 1,100 pair of users with the highest sum of terms in frequent conversation is select to crawl. Within 65 users are repeating, so 1,035 mentioned user is crawled. Due to account privacy level, some of the users are not crawled, the final number of crawled users is described in next section.

[1] See appendix A for full list of words

[2] In Twitter, FF is a short for FollowFriday, used commonly in Friday where people recommend and promote each other in each social network.

The condition for the initial expectation of the result of this crawling is same as the crawling of the second milestone, as many pairs of source user and crawling user are considered, 1,100 frequent interaction are expected.

5.3.3. Result of Crawling and Evaluation

In this milestone, 971 users are crawled, with a total of 2,778,287 tweets. All the crawling result summary is very similar from the second milestone, see tab:Second milestone crawled dataset Summary, there are active communicators with 65% of their tweets are mention tweets.

And we can also observe from following table, there is an increase in the number of conversation tweets and conversation found, compare with the second milestone see tab:Second milestone crawled dataset Summary. This approves correctness the list of frequent conversations term.

Tweets	2,787,262
Users	966
Average of number of Tweets per user	2,879
Mention tweets	1,816,084
Reply tweets	780,523
Retweet tweets	657,108
Conversations	44,508
Tweets involved in a conversation	99,591

Table 5.10.: Third milestone crawled dataset summary

Evaluation of the Crawl

The evaluation of this crawling has the same criteria as the second milestone, it focuses only on the interaction between crawled user and corresponding source user.

From above table, we can observe 566 (56.9%) of 994 predicted pair stands the frequent interactions with 109,924 new Tweets crawled. which is 3.94% of total tweets crawled in this milestone. Slightly improvement compare with the previous milestone.

Pairs of source user and successful crawled user	994
Frequent interaction between the pairs	566
Tweets between the Frequent interaction between the pairs	240,008
Tweets send by the source users to the crawled users	202,000
Frequent interaction tweets send by the source users to the crawled users	128,757
Tweets send by the crawled users to the source users	114,067
Frequent interaction Tweets send by the crawled users to the source users	109,924

Table 5.11.: Summary about interaction between source users and crawled users of the third milestone

And reusing the concept from equation 5.2, by summing the interaction tweets from both source and crawled users, and summing the total tweets considered for both cases.

$$\begin{aligned} Crawler\ Rate\ Value &= \frac{Tweets\ in\ the\ frequent\ interaction}{Total\ tweets\ considered} \\ &= \frac{128,757\ + 09,924}{2,532,109\ +\ 2,787,262} = 0.0448 \end{aligned} \tag{5.3}$$

The result is 4.48%, this effectiveness is lower than the second milestone, due to the less mentions tweets from source users.Yet this milestone has the advantage in discover tweets in conversation.

5.4. 4th milestone: Union of Second and Third Milestone

This milestone is about combining the features generated from the second and the third milestones. Also a simple evaluate the difference between of the ranking from both milestones.

As the evaluation of the methods of the previous milestone is approached, this milestone is concentrated in the crawling part. The primary goal is to crawl the users with the highest likelihood to mentions back any of the users in the dataset crawled until now (not just source users).

As a consequence, the evaluation of this milestone of crawling is based on entire new crawled dataset (from the previous tree milestones).

5.4.1. Process of Ranking Users to Crawl

The process is similar from previous milestones, 1,000 crawled users from the first milestone are considered as source users. And then compute the both ranking lists from the second and third milestones for all pairs of crawled user and mentioned user.

In order to compare both methods from two previous milestones, we decide to use simple criteria. By counting the number of coincident users by considering different numbers of the top of both rankings:

Number of top position taken	Number of the pairs of source user and mentioned user coincide in both ranking
100	47
500	224
1000	469
5000	3121
10000	6480

Table 5.12.: Overview of comparing rankings list generated from methods of the second and third milestone

From above, the similarity between user crawling rankings system from the second and third milestone are closer when extends the top number. Also from the evaluations of both milestone, indicate a similarity in crawling result, instead of using any model technique to find the balance, we decide to simply give more prior for ranking of the third milestone. As for the sentiment analysis work, is more interest in having more conversations (the thired milestone focus on conversation tweets).

To optimize the crawling, the sum of the features from both previous milestones is not longer separated by pairs of source user and mentioned user, for instead, the sum is done respect each mentioned users. In this way, users mentioned by several source users have higher positions of ranking, and a major number of frequent interaction is expected.

The respects this total sum ranking, we decide to select top 1,100 users by giving prior the feature from the third milestone than the second.

The estimation for the expectation of this crawling result is more complex, as describe in section 3.5, this milestone is expanding for a dense social network, especially users to crawl are select respect from mention tweets from all the source users.

5.4.2. Result of Crawling and Evaluation

In this milestone, 3,048,018 tweets are collect from 1,029 users. The standard statistics are similar from the crawling result of the second and third milestones, see tab:Second milestone crawled dataset Summary and tab:Third milestone crawled dataset Summary. As this based on the same rankings systems.

Tweets	3,048,018
Users	1,029
Average of number of Tweets per user	2,962
Mention tweets	1,995,666
Reply tweets	799, 657
Retweet tweets	825,724
Conversations	45,355
Tweets involved in a conversation	89,595

Table 5.13.: Fourth milestone crawled dataset summary

And the following table indicates real value of the crawling, as in this milestone, the summary is respecting on entire new crawled dataset, and not only source users:

Crawled users in frequent interaction	769
Between crawled users and the rest users of the dataset	
Mention tweets send by crawled users to the rest users of the datasets	673,499
Mention tweets send by the rest users of the datasets to the crawled users	1,527,545
Frequent interaction between the crawled user and the rest users of the datasets	1,489
Tweets in Frequent interaction between the crawled user of this milestone and rest of the datasets	320,843
Mention tweets send by crawled users to the rest users of the dataset in frequent interaction	150,010
Mention tweets send by rest user of dataset to crawled users in frequent interaction	153,913
Between crawled users of this milestone	
Frequent interaction between the crawled user of this milestone	458
Mention tweets between crawled users	426,615
Tweets in Frequent interaction between the crawled users	79,176

Table 5.14.: Summary about interaction between crawled users and entire new crawled datatset of the Fourth milestone

From above table, a high effective crawling is observed. Where 769(75%) crawled users of 1,029, have in average two frequent interaction with the rest of the user of the datasets. Whereas a total of 320,843 tweets(including from the rest of the datasets) are turned as frequent interaction. So within 3,048,018 crawled in this milestone 150,010 and 79,176 tweets are frequent interaction, which gains up an 7.5% of crawler rate value, almost double than the previous milestones.

Furthermore, 458 frequent interaction found between only crawled users in this milestone are an indication of a strong, dense social network formalized by the crawled dataset of preview milestones, as the crawled users are also a dense social network.

6. Conclusion

6.1. Further Considerations

In this thesis, we only worked in few aspects about tweets amount all the rest of relevant features, especially information about the author of each tweet are ignored. Furthermore, we only minimal did exploitation in the relations between users, and study the social network.

A few of the possible additions and extensions of the crawler could be:

1. Reconsidering the features ignored during the modeling of the second milestone.
2. Apply methods to identified sub-dense social network, in order generate an optimised source users for the ranking system.
3. Considering social relations between users, e.g., followers and list of groups.
4. Considering the social influence level of user by number of followers.

6.2. Conclusion

In this thesis, with the purpose to gather interaction between people in the social platform Twitter for future sentiment analysis between people.

We success to build two crawler systems able to identify if an individual replies during an interaction by analyzing all message send to him. Both systems achieve more than 50% of accuracy in identifying replies individual. The first system uses random forest classification model to predict the likelihood of being replied of each sent message by analysis the features of the message. The second system uses a list of frequent terms in conversation generated in this thesis, to identify possible conversations between two individual with only messages from one of them. Moreover, combine the both systems

and exploiting social network structure, we are able to recognize replying individuals with 75% of accuracy.

By using each system interactively in different milestones, we finally 35,050,922 tweets, which contains 8,437 users in frequent interaction with 2,032,203(5.8%) tweets. Together with the crawler designed, this thesis brings an strong connected social network dataset wiht guaranteed the completeness in interaction between nodes, and possible to extend. This datasets can provide a solid base for future social analysis research.

A. Frequent Terms in Conversation

#brexit	@1987_laura	although	assume
#euref	^	among	attacks
#ff	above	amount	attempt
#lufc	absolute	analysis	attention
#pakistan	abt	andy	audience
#periscope	abuse	angry	august
#rio2016	accept	anniversary	average
. ...	access	announced	avoid
00	according	annoying	award
09	account	anymore	awards
1/2	ace	anyway	aware
17	act	anywhere	awful
19	actual	apart	bag
2/2	ad	apparently	balls
22	added	appear	band
24	address	appears	based
31	afraid	apply	basically
33	age	appreciate	battle
3rd	ages	appreciated	bc
4a	agreed	approach	bear
60	ah	april	becoming
70	ain't	area	beer
80	airport	aren't	benefit
89	al	argument	bet
90	alive	army	beyond
:-(	allow	arrived	bigger
:P	allowed	arsenal	bike
;-)	alone	asking	birmingham
<3	alright	asks	blame

bless
block
blood
board
booked
boring
boris
born
boss
bottle
bottom
bought
brain
breaking
brexit
bristol
britain
bro
broken
brother
brought
brown
btw
built
bunch
buying
cabinet
calling
calls
camera
cameron
cancer
candidate
cant
card
cards
career
cars
cash
caught
cause
cc
celebrate
central
certain
certainly
chair
changed
changes
changing
channel
character
charge
charity
chat
cheap
cheers
cheese
chelsea
chicken
chief
china
choice
choose
chris
church
claim
claims
classic
clean
clear
clearly
click
clicking
clinton
closed
coach
code
collection
colour
column
comment
comments
common
companies
competition
complete
completely
congrats
congratulations
consider
considering
contact
continue
contract
control
conversation
copy
corbyn
corner
correct
cos
cost
couldn't
council
count
countries
court
cover
coverage
crap
cream
created
credit
crime
crisis
cross
crowd
cry
crying
culture
current
currently
customer
customers
cute
da
dad
damn
dan
dance
dangerous
dark
daughter
dave
decent
decide
decided
decision
deep
defence
delicious
delighted
delivery

demand
democracy
deserve
despite
development
die
died
difference
difficult
direct
disappointed
discuss
discussion
dm
dogs
donald
donation
dont
door
double
doubt
dr
draw
dress
drinking
drinks
driver
driving
drop
dropped
drunk
dude
earlier
earth
easier
economic
economy
ed
effort
eh
either
election
em
emails
en
ended
enjoyed
enjoying
enter
entire
episode
er
especially
et
etc
europe
european
everywhere
evidence
evil
exactly
example
except
excuse
expect
expected
explain
extra
eye
f
fab
fabulous
failed
faith
fake
families
fancy
farage
fascinating
fat
father
fb
fear
feature
feb
feed
feels
feet
fell
felt
female
ffs
field
fighting
figure
films
finding
fine
fingers
finish
fit
fix
flat
flight
floor
flying
focus
folk
folks
foot
force
forces
foreign
forever
forgot
form
former
france
freedom
french
fresh
fully
funding
further
g
gave
general
genius
genuinely
george
german
germany
gig
gives
glass
goals
gotta
govt
grand
greatest
ground
groups
grow
growing

growth
gym
h
hahaha
hai
hall
hands
happening
happens
harry
hashtag
hasn't
hat
hav
he'd
he'll
heading
hearing
held
helped
helping
helps
higher
hill
hillary
himself
hmm
hold
holding
holy
honest
honestly
hopefully
hoping
horrible
hospital
housing
how's
however
husband
ice
ignore
ill
image
imagine
immigration
impact
impressed
improve
include
including
incredible
indeed
india
indian
industry
information
inspired
instagram
interest
involved
iphone
iraq
ireland
isis
issues
it'll
itself
jan
jeremy
jesus
jo
johnson
joining
joke
journalists
journey
joy
justice
k
keeping
keeps
kept
kick
kid
kill
killed
killing
kinda
king
kit
kitchen
klopp
knew
known
knows
l
lack
lad
ladies
lads
lady
land
language
large
laugh
laughing
le
leader
leaders
leadership
leading
leaving
leeds
legal
lets
letter
lfc
lie
lies
liked
likely
likes
lines
literally
ll
loads
longer
looked
lord
losing
loss
loves
low
lucky
machine
mad
magic
mail
main
major
majority
managed
mass

massive
mate
matters
meant
member
members
memories
memory
mental
mention
mentioned
mess
message
met
michael
middle
miles
military
min
mine
minister
mins
mistake
mix
model
modern
moments
mostly
mother
moved
movement
moving
mp
mps
mr
mrs
murder
muslim
muslims
na
names
nation
neither
network
nhs
nick
nights
nobody
none
nope
normal
note
notice
noticed
numbers
obvious
obviously
october
odd
official
often
oil
okay
olympic
omg
ones
onto
ooh
oops
opening
opinion
opportunity
opposition
option
options
ordered
original
otherwise
owen
p
paid
pain
pakistan
panel
paper
paris
parliament
particularly
parties
pass
passed
paul
paying
peace
per
perhaps
peter
pic
pick
picked
pics
pictures
pitch
pizza
places
plane
planning
platform
played
pleased
pleasure
plenty
pls
plus
podcast
points
policy
political
politicians
politics
poll
pop
popular
position
positive
possible
possibly
potential
powerful
ppl
pr
practice
prefer
present
pressure
previous
price
prime
private
pro
problems
process
product
programme

progress
proper
properly
ps
pub
putting
q
queen
quickly
quiet
quote
race
racist
raise
raised
ran
random
rate
rd
reach
reaction
realise
realised
reality
reasons
received
recent
recently
reckon
recommend
referendum
related
relationship
release
released
religious
remain
reminder
reminds
reply
reports
respect
response
restaurant
result
return
retweet
rich
ride
ridiculous
rights
ring
rip
rise
risk
rock
role
roll
route
row
rubbish
rule
rules
russia
sa
sadly
sam
scared
scene
schools
science
score
scotland
scottish
screen
sea
search
seat
seats
secret
security
seem
self
sell
selling
sending
sense
sent
serious
several
sex
shall
shame
sharing
shirt
shocking
shoes
shoot
shopping
shouldn't
shout
showing
shut
sick
signed
signing
silly
similar
simply
singing
sir
sister
sit
sitting
situation
six
size
skills
slightly
slow
smart
smile
smith
snow
society
sold
solution
somewhere
son
songs
sooo
sort
sorted
soul
source
speak
speaking
speech
spending
spoke
sport
sports
spot
spread

squad
stage
standing
stars
statement
states
station
staying
steve
stick
stopped
store
straight
strange
strategy
stuck
student
studio
study
stupid
suggest
supporters
supporting
suppose
supposed
surely
surprise
surprised
suspect
syria
ta
table
talks
target
tbh
teams
telling
tells
ten
term
terms
terrible
text
thats
themselves
theresa
they'd
they'll
they've
thinks
third
tho
thx
ticket
til
till
tim
tiny
tired
title
tom
tonight's
tony
tories
tory
total
totally
tough
towards
track
trade
traffic
treat
tried
trouble
truly
truth
tuesday
tune
turned
turning
turns
tweeted
tweeting
twice
ty
type
u0626
u0640
u064e
u0650
u067e
u06be
ud83e
ukip
un
unless
useful
usual
usually
value
van
ve
version
victims
videos
views
violence
voice
voted
voters
votes
voting
wake
wales
walking
wall
wanna
warm
waste
watched
we'd
wear
wearing
web
wednesday
weird
weren't
whatever
whether
whilst
whose
wife
window
winner
winners
winning
wins
winter
within
women's
wondering
worked
workers

works
worried
worry
worse
wouldn't
written
wrote
wtf
xmas
y
ya
yay
yep
you'd
yours
youth
youtube
yr
yrs
yup

List of Figures

List of Tables

Bibliography

[BGL10] D. Boyd, S. Golder, and G. Lotan. "Tweet, tweet, retweet: Conversational aspects of retweeting on twitter." In: *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE. 2010, pp. 1–10.

[Cha+10] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi. "Measuring User Influence in Twitter: The Million Follower Fallacy." In: *ICWSM* 10.10-17 (2010), p. 30.

[Kos13] B. Koster. "Modeling Influence in Social Networks with Topic Models." MA thesis. Technische Universität München, 2013.

[KWM11] E. Kouloumpis, T. Wilson, and J. D. Moore. "Twitter sentiment analysis: The good the bad and the omg!" In: *Icwsm* 11 (2011), pp. 538–541.

[Nak+16] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov. "SemEval-2016 task 4: Sentiment analysis in Twitter." In: *Proceedings of the 10th international workshop on semantic evaluation (SemEval 2016), San Diego, US (forthcoming)*. 2016.

[PP10] A. Pak and P. Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." In: *LREc*. Vol. 10. 2010, pp. 1320–1326.

[Tha] M. H. V. Thakkar. "Twitter Sentiment Analysis using Hybrid Naıve Bayes." In: ().

[Yua16] B. Yuan. "SENTIMENT ANALYSIS OF TWITTER DATA." PhD thesis. Rensselaer Polytechnic Institute, 2016.