Université de Cergy-Pontoise

Synthesis Project

Emotional rating of events based on tweets

Reporter:
Philippe Gaussier

Technical tutors:
Dimitris Kotzinos
Pierre Andry

Project Management Supervisor:
Tianxiao Liu

Hervé-Madelein Attolou
Loïc Bachelot
David de Castilla

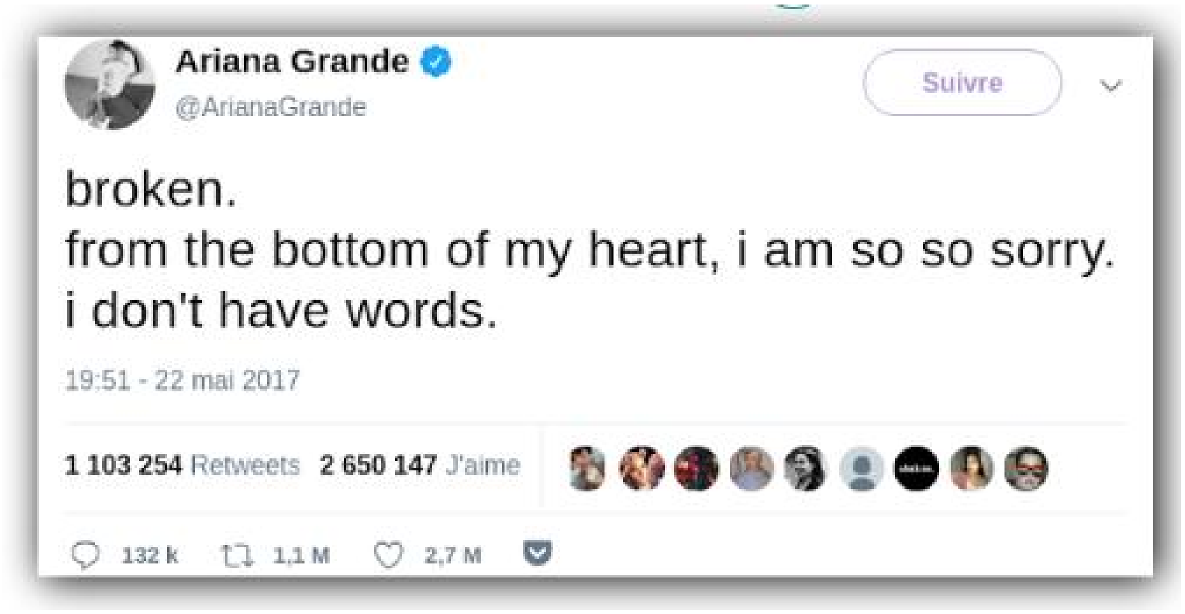
Contents

Chapter 1

Introduction

1.1 Idea of the project

The idea of the project is to get tweets in real-time in order to detect events that are happening in Paris, and to rate these events depending on their positivity or negativity.



Ariana Grande
@ArianaGrande
Suivre

broken.
from the bottom of my heart, i am so so sorry.
i don't have words.

19:51 - 22 mai 2017

1 103 254 Retweets 2 650 147 J'aime

132 k 1,1 M 2,7 M

Figure 1.1: example of tweet from Twitter

Twitter is a microblogging site created in 2006 [1] that is used by over 500 million people worldwide [2]. Researchersd it an invaluable data repository for opinion mining and prediction in a number of fields, including politics [3] and financial markets [4]. Twitter has also served as a resource for predicting and tracking the propagation of natural disasters, epidemics, and terrorist incidents [5].

We will be able to rate the events by analysing the content of the tweets, each tweet will be rated by an artificial intelligence (AI), the rate correspond to the negativity or positivity of the tweet (-1 for very negative, 0 for neutral, and +1 for very positive). The rates of the tweets that are defining the event will then be processed to rate the event.

Then the events will be displayed on a "heat map", with a color for each even. The color intensity depends on the positivity rate of the event.

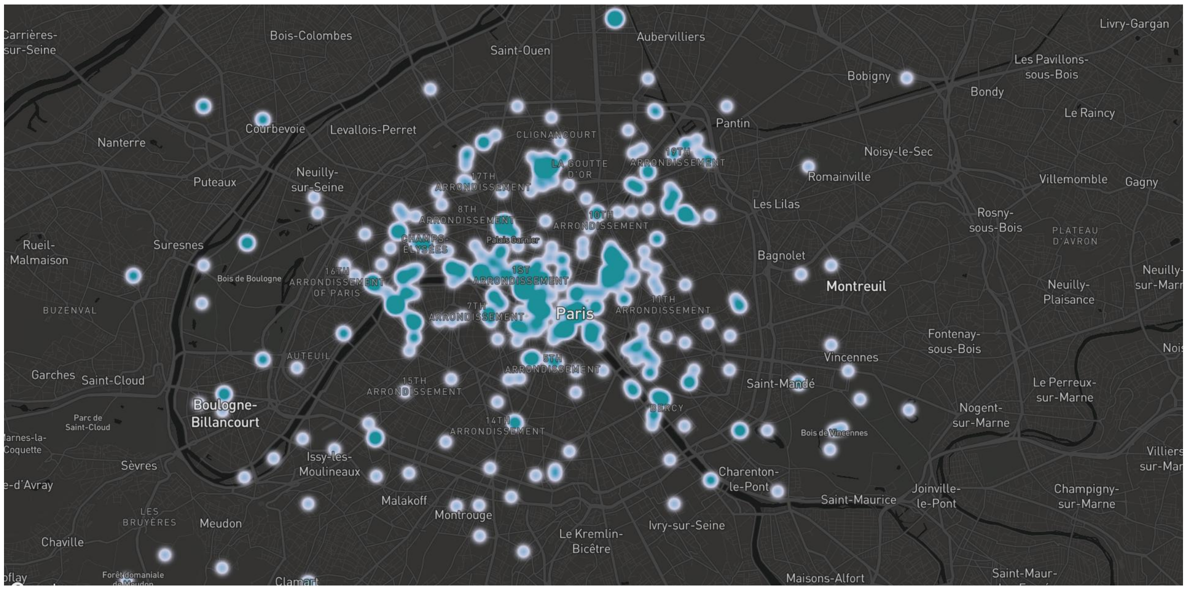


Figure 1.2: example of heatmap of tweets in Paris

1.2 Subject Choice

The choice of the subject was a long process that began before the official release of the subject, because we wanted to make our own subject.

The first subject was to predict the future price of the bitcoin with an AI. We started working on that subject during September and October doing some research on the bitcoin itself and on the different techniques to predict a two dimensional line. We decided to study two different directions : predicting using external parameters like the number of research on google on the bitcoin to predict the future and just using the previous prices.

Then, as we were going in Finland for the second semester, Mr. Pierre Andry proposed us another subject that included to work with a Finnish company named Subsea Energy [6]. This company is building a device named the Stormrider that's able to create energy from the waves. They needed, in order to optimize the production of energy, to predict different parameters of the next wave. For them, knowing these parameters about the next wave would have allow to change the configuration of the Stormrider automatically in order to get more energy from the wave. We started to contact this company, and talk with, but they were not reactive enough, so we decided to work on another subject at the beginning of january. This when we thought about our final subject (in the end of December). We decided not to go back to the bitcoin project because during the time we were working on the wave project we continued to follow the new about the bitcoin and the prices became unpredictable (raise to

20000$ and falls under 10000$ right after).

This project idea came from the tweets analyses during the last football world cup to determine how won based on the tweets.

In 2014, the FIFA world cup took place in Brazil. During the 32-day event and with more that 60 games, Twitter's users generated over 672m tweets [7]. The generated data was used in many project as support to test some sentimental analysis algorithms [8]. Their goal was then to be able either to :

- Analyse the sentiments of the supporters when they are loosing or when they are winning [9]
- Predict the winner of a game only by reading tweets with the related hashtag

Knowing that those projects were successful, we tried to be a little bit original by adding the geographic factor to those project and reversing the relation between emotion and event.

Instead of taking an event and then look for the emotion related to this event we went the other way by gathering and find the emotion then find the related event. By working in that order we thought that this solution could make us able to gather more tweets.

To show all our results, the map was a pretty good solution to showcase a geographically reported data. Moreover as the data is scaled and with two major steps (Negative or Positive) a heat map could be the most adapted type of map to illustrate the result of our dataset.

1.3 Use Case

The purpose of this project could be an analysis or a prediction.

Our goal, at first, was to be able to predict the sentiment in each place of a future event. Using machine learning and by exploiting all the data we collected the goal could be to locate some pattern on the feeling. Simply, we could think about gathering tweets about the usual friday traffic jams or the rush hours in the subway.

Next, by linking the events and the sentiment we can analyze the success or the fail of a specific event, venue or a product. The goal could be the extend this project to use social networks to have an overview about a specific subject. This would be an interesting system to sell for commercial analytics.

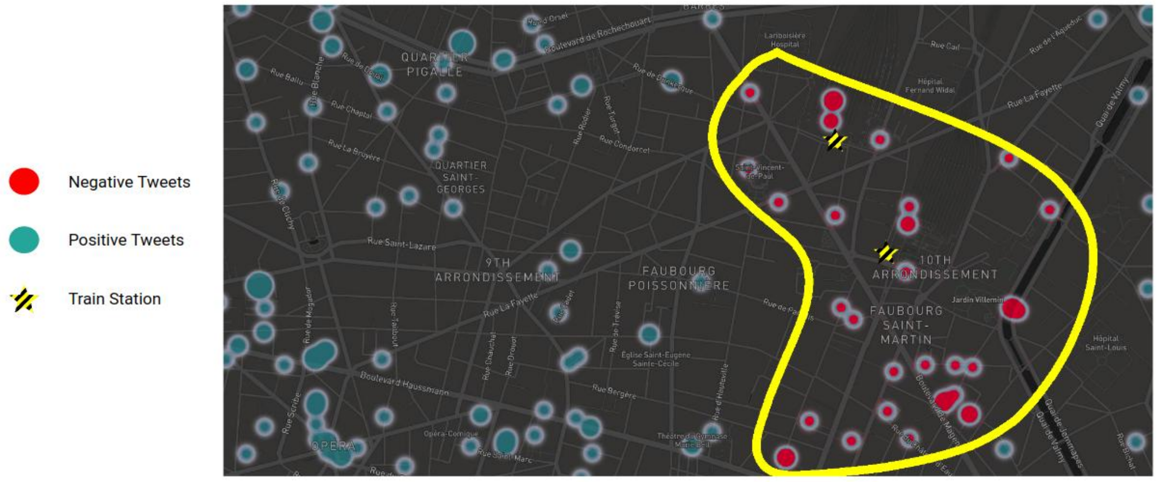


Figure 1.3: expected result in case of a strike

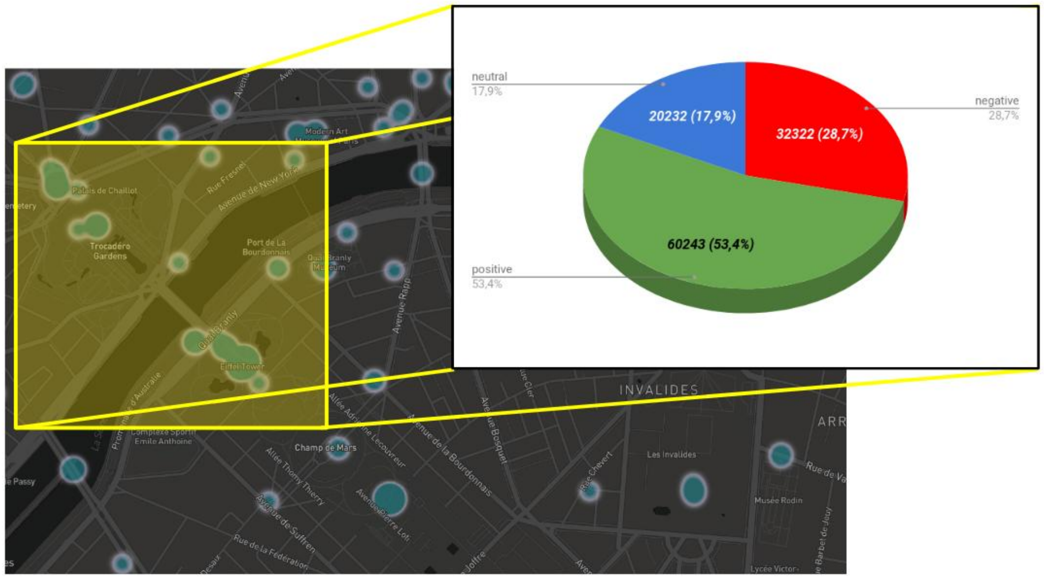


Figure 1.4: mockup of a sentimental analytics overview for a specific event

Chapter 2

Work environment and conception

2.1 Work environment

2.1.1 Programmation languages

In this project, we decided to use mainly python as a script language. We choose this language because each library we are using in this project is available in python :

- RethinkDB has a python driver in order to do request on the database [10]
- Snowball (Stemming library) has a python interface named pystemmer [11]
- Scikit-learn (Machine learning library) is available in python [12]
- Tensorflow a Machine Learning library is available in python [13]

In order to make our web interface, we used nodeJS to create our server. The use of node-js allows us to link the RethinkDB database and the map output, because we used mapbox [14] as a map provider.

2.1.2 Database

In order to store all of our data, we are using a RethinkDB database, there are a few reasons why we chose this technology :

Document based Every element in a table of this database is a JSON object, you can add, request, change or delete those JSON objects that are present in a table.

Real-time A program can be notified when changes are made to the database, this can allow our different modules to communicate and store the data at the same time. When one is writing in the database, another module, which may need the data that has just been stored, receives the data instantly.

Open-source This database technology is free and can be installed on any computer.

Our database is running on an OVH VPS (Virtual Private Server), which has this configuration (VPS Cloud 2):

- 2 vCore(s) 3,1 GHz
- RAM : 4 Go + SWAP 4 Go
- Data storage : 50 Go
- Operating System : Ubuntu 16.04.04 LTS

2.1.3 Map provider

We also had to choose a map provider for our "Map output" module. We had two choices:

- Google Maps API [15]
- Mapbox

What we needed to do is display a map, and add elements to it after its creation. Both of them has a JavaScript API that we could use with nodeJS, and the free plans are almost the same.

We choosed Mapbox because it was the only one that allows us to do this. Google Maps only allows us to display a static map and not the "*heat map*" that we want to produce.

2.2 Conception

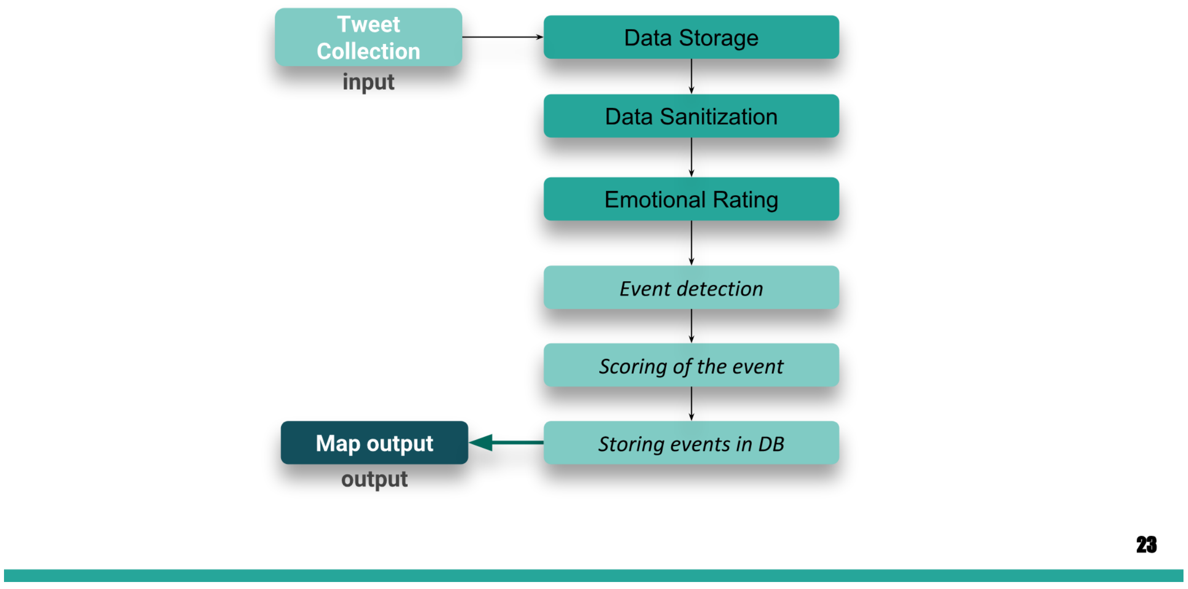


Figure 2.1: This graph represents the different modules and how they work together in the project

- **Tweets Collection**: Using the Twitter API, we are able to a real-time stream of tweets in a geographical area (Paris). Theses tweets are sent to us in a JSON format.
- **Data Storage**: As we receive tweets from the Twitter API, we store all of them in RethinkDB database.
- **Data Sanitization**: In this part we are cleaning the data by only keeping the useful text and the location of the tweet. The text is cleaned using the Porter stemming algorithm which will cut the words to only their stem.
- **Emotional Rating**: This part is about determining whether a tweet is positive, neutral or negative. In order to do that, we give a grade to each tweet between -1 and 1. -1 is a very negative tweet, and 1 is very positive, 0 is neutral.
- **Event Detection**: After grading the tweets, we need to detect events from them, using text, location, hashtags, and user mentions. We also grade the events between -1 and 1 depending on their positivity, this grade is based on the grade of the tweets composing the event.
- **Storing events in the database**: We are also keeping tracks of the events in time, this mean, keeping all the events we detected in the RethinkDB database.
- **Map output**: In the map output, we display the event we collected with a color depending on their grade.

Chapter 3

Tweet collection and sanitization

By doing some research we found many ways to cut the suffix of a french word. In this part we are going to review some possible algorithms that we can apply. We're also going to explain what algorithm we choose, and how we implemented it.

3.1 Data collection

In order to collect the tweets, we used the Twitter API that allows us to connect to a real-time stream of tweets. So we are receiving tweets instantly after they've been created by an user. The Twitter API give to us a lot of informations about each tweet :

Feature name	**Description**
created_at	tweet's creation date and time
id	Twitter tweet's id
text	The tweet text
user	informations about the user
coordinates	Precise coordinates of the tweet (if available)
entities	URL, user mentions, hashtags, and media that are in the tweet
place	informations about the place where the user published the tweet (the city)
possibly_sensitive	if the tweet is sensitive or not
source	Utility used to post the tweet
lang	The lang of the tweets

Figure 3.1: Most important information we get about the tweet. This table only contains the most important elements that we get, you can find the whole list at this address

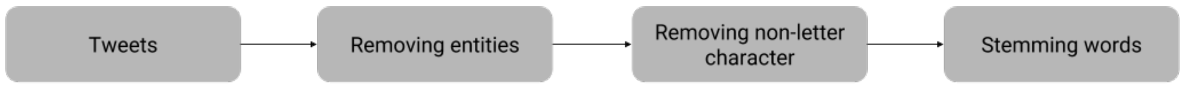


Figure 3.2: This graph represents the data sanitization process in our project

	Total number of tweets	French tweets	Usable tweets for event detection	Average number of tweets per day
Number	325 621	208 788	5811	11 000
%	-	64.12	1.78	-

Figure 3.3: Statistics about the tweets we collected

We launched the data collection on the May 7th for the tweet that have a precise location, and the May 13th for all the tweet. You can find some statistics of the tweets we collected in figure 3.3.

Once we received a tweet, we have to sanitize the text of it, this include :

- Removing hashtags, user mentions, URL
- Removing character that are not letter
- Removing suffixes of words

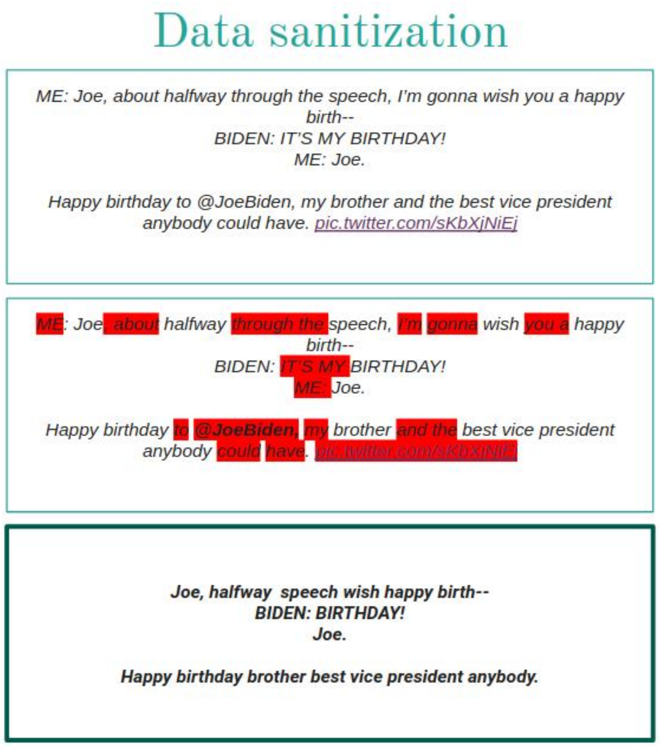

Data sanitization
ME: Joe, about halfway through the speech, I'm gonna wish you a happy birth--
BIDEN: IT'S MY BIRTHDAY!
ME: Joe.
Happy birthday to @JoeBiden, my brother and the best vice president anybody could have. pic.twitter.com/sKbXjNiEj
ME: Joe, about halfway through the speech, I'm gonna wish you a happy birth--
BIDEN: IT'S MY BIRTHDAY!
ME: Joe.
Happy birthday to @JoeBiden, my brother and the best vice president anybody could have. pic.twitter.com/sKbXjNiEj
Joe, halfway speech wish happy birth--
BIDEN: BIRTHDAY!
Joe.
Happy birthday brother best vice president anybody.

Figure 3.4: sanitization of tweets

In order to remove the hashtags, user mentions, and URL, we use the entities that are given by the Twitter API with the tweet, so we don't have to detect them ourselves, then we just have to remove them from the text.

Removing character that are not letter is also an easy part, we can use a built-in function of python, which is *is_alpha*(), that allows us to tell if a character is a letter or not, and we rebuild the text of the tweet without the non-letter character.

3.2 Stemming

The real challenge in the sanitization is to remove suffixes of words. In most of the cases (**at least in french**), the suffix does not change the meaning of a word. This lexical derivation, is able to create new words adapted to the grammatical usage of a word. In our research, we found that a grammatical derivation is not "sentimentaly" different than the stem.

In order to do that, we found in our research that the most used way to do so is called porterisation, and refers to the Porter algorithm [16]. But the tweets that we are using are in french, and the Porter algorithm only applies to english words. So we searched for another solution that applies to french words, and we found a library named snowball [11] that provides a stemmer in a various languages including french. We're using this library in order to remove suffixes of words in our project.

3.2.1 The Porter stemmer algorithm

The Porter Stemmer algorithm is able from an english word to get the stem.

The algorithm consist in 5 differents steps that are applying some rules to cut parts of the word.

A consonant in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. for instance in BOY the consonants are B and Y.
Considering c as a consonant and v as a vowel, we will also write C as a list ccc and V as a vvv, if those list if their length is greater than 0.

By that notation we can say that any word can be written like this:

$[C](VC)^m[V]$.

where the square brackets denote arbitrary presence of their contents and $(VC)^m$ to denote VC repeated m times.

m is the notation of the measure of the word.

- m=0 TR, EE, TREE, Y, BY.
- m=1 TROUBLE, OATS, TREES, IVY.
- m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

Every step a rule is applied. A rule take this form: $(condition)S1 \rightarrow S2$

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2.

For example here is are rules for the step 3 :

- $(m > 0)ICATE \rightarrow IC$ for example : $triplicate \rightarrow triplic$
- $(m > 0)ATIVE \rightarrow$ for example : $formative \rightarrow form$
- $(m > 0)ALIZE \rightarrow AL$ for example : $formalize \rightarrow formal$

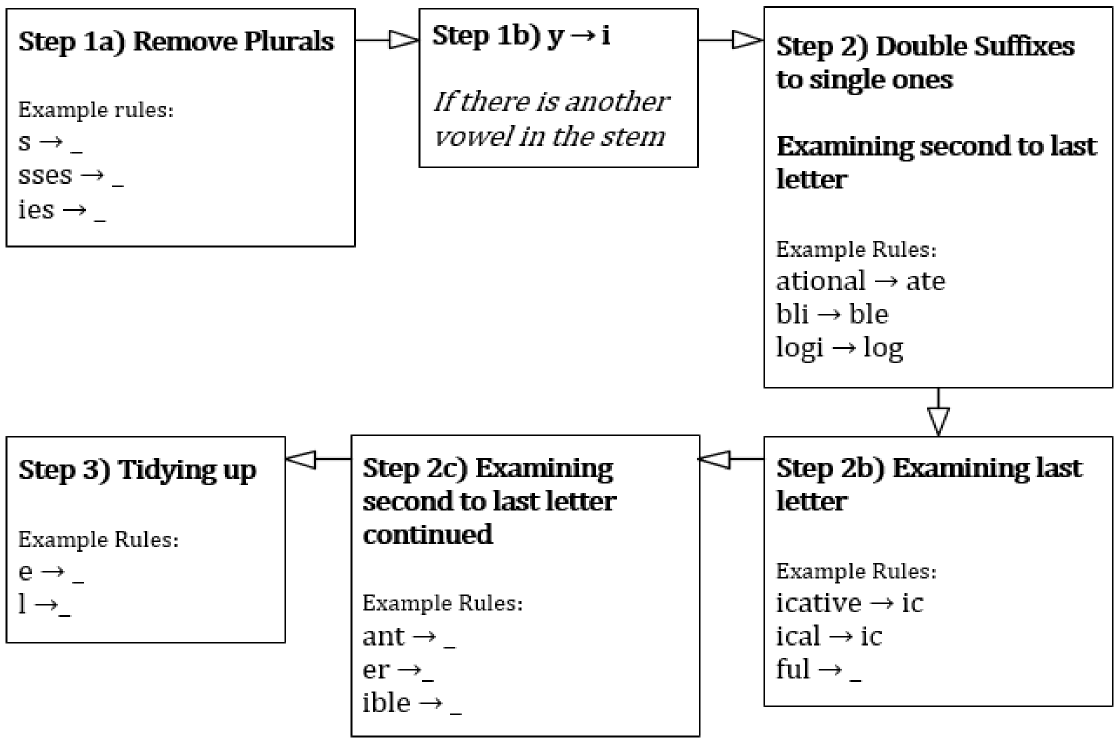


Figure 3.5: simple Porter Stemming algorith representation

- $(m > 0)ICITI \rightarrow IC$ for example : *electriciti* $\rightarrow$ *electric*
- $(m > 0)ICAL \rightarrow IC$ for example : *electrical* $\rightarrow$ *electric*
- $(m > 0)FUL \rightarrow$ for example : $hopeful \rightarrow hope$
- $(m > 0)NESS \rightarrow$ for example : *goodness* $\rightarrow$ *good*

Relation with our project This algorithm is a perfect solution to our project, because it make us abel to cut the suffix of a word. This algorithm is usable via a lot of libraries. Only one issue with this is that it only applies to english words.

Chapter 4

Emotional rating of tweets

By doing some research we found many ways to get the sentiment (or emotion) from a small textual container such as a tweet.

In this part we are going to review some possible algorithms that we can apply. We're also going to explain what algorithm we choose, and how we implemented it.

4.1 Bibliographic research

4.1.1 Emoji Approach

System Review The usage of the emoji is now a common thing in many of the social networks [17] . An emoji, smiley or an emoticon is a simple form icon that can represent many things from a flag to an object or an action.

In this project [18] , some emoticons are classified as negative or positive depending on their cultural meaning. Then, the emoticons have 2 (positive and negative) binary labels that are reported to every tweet.

Positive Emoticons	**Negative Emoticons**
:)	:(
:-)	:-(
:D	;(
=)	=(
;)	:/
¡3	:-/
:p	):
:-p	
(:	

Figure 4.1: Classification of negative and positive emoticons

Relation with our project Some problems are exposed by that solution for our case:

Some tweets are categorized as neutral if they contain emoticon from both categories which is different from a "real" neutral tweet that can just give some informations for example. We thing that in our project the negativity and positivity expressed in the same tweet should be treated as a presence of emotion, thus differently as the absence of emotion.

There is no levels in the positivity or the negativity for a tweet so two positive tweets have the same value to classify an event. In our project we think it would be a good thing to evaluate the "level" of emotion in every tweet to give it more or less weight in the evaluation. Moreover the neutrality could be a really interesting data to have in our system.

4.1.2 POMS-ex, a psychometric instrument

System review In this case [19], the goal is to analyze the link between tweets emotion and big social events in the world.

The emotion classification is made using a psychometric instrument POMS (the Profile of Mood States). This tool measure six individual dimensions of the mood including :

- Tension
- Depression
- Anger
- Vigour
- Fatigue
- Confusion

For each of those dimensions, a grade or score is given by the algorithm. The original algorithm is supposed to be used a human-answered-test to define the current emotional state of a person.The subject has to give a five-point score, based on his actual emotional state, to each 65 adjectives of the list. In this case an extension to the test is used: POMS-ex:

> This version of the instrument is not intended to be administered as a questionnaire to human subjects, but rather to be applicable to large textual corpora. POMS-ex extends the original set of 65 POMS mood adjectives to 793 terms, including synonyms and related word constructs, thus augmenting the possibility of matching terms in large data, such as online textual corpora.

A normalisation algorithm is also applied:

1. Separation of individual terms on white-space boundaries
2. Removal of all non-alphanumeric characters from terms, e.g. commas, dashes, etc.
3. Conversion to lowercase of all remaining characters.

4. Removal of 214 standard stop words, including highly common verb-forms.
5. Porter stemming [16] of all remaining terms in tweet.

$$P(t) \to m \in \mathbb{R}^6 = [\|w \ \cap p_1\|, \| \ w \cap p_2\|, \ldots, w \cap p_6\|]$$

The result of the normalisation is then applied to the POMS algorithm as a function which give us a 6 dimension vector.

Results Here are some results of the algorithm compared to some events:

- November 4 On U.S. election day, Tension skyrockets to over +2 standard deviations. The day after Vigour jumps from baseline levels to +3 standard deviations, while fatigue steadily drops to -2 standard deviations.
- November 27 On U.S. Thanksgiving day, Vigour notably records a sharp increase from baseline levels to +4 standard deviations.

We can see that the algorithm give us a report of the Twitter users feelings in a precise date.

System review For our project this seems like a good thing:

- The algorithm give us **different dimensions of a feelings** for the same tweet/event
- **A precise formula** give us a way to classify all the feeling in a graph (that we will use in a heat map)
- The ability to use **many vectors** can create the possibility to make as **many maps** as vectors (in this cas 6 vectors $\to$ 6 maps)

The main issue with this project is the implementation of POMS-ex algorithm. If the algorithm is well described in the paper and is documented enough to make our own implementation, this could require a big amount of time, and could represent a project on it's own. Sadly we didn't found any library to use in our project to be able to include this feature.

4.1.3 Machine learning techniques

System review In this case [20], tweets are simply classified as positive or negative. The main goal, would be to check the feedback of a product by a query on some concerning tweets. The marketers could also use the tool to search public opinion and analyse customer satisfaction.

This project could replace all the star-rated reviews for some shopping websites (Amazon, BestBuy, Boulanger, Fnac, . . .) or even for some flat renting platforms (AirBnB, Booking,. . .)

The tweets are classified in some queries groups. Each tweet can be in one or many query group.

Emoticons mapped to :)	**Emoticons mapped to :(**
:)	:(
:-)	:-(
:)	: (
:D	
=)	

Figure 4.2: List of emoticons

Sentiment	**Query**	**Tweet**
Positive	jQuery	dcostalis: Jquery is my new friend.
Neutral	San Francisco	schuyler: just landed at San Francisco
Negative	exam	jvici0us: History exam studying ugh.

Figure 4.3: Examples of Tweet

The emoticons are playing a role in the emotional classification. The neural net is trained with tweets excluding those emoticons because they can have a negative impact on the accuracies. Moreover the algorithm is supposed to focus on the non-emoticon features to determine the sentiment.

Then if the dataset contain an emoticon the classifier is not influenced. In this case emoticons are considered as nose. Any tweet containing both positive and negative emotions are removed from the dataset. Those are considered to express more than one feeling or more than one subject.

A feature reduction is made by including 3 properties:

Username: "tag" directly someone in the tweet using his @ (retweets are not part of the dataset

URLs: tweets tweets including some link to a website or any content from the internet

Repeated letters: as an example "I'm huuuungry"

The machine learning method used in this cased are:

- Keyword-based
- Naive Bayes [21]
- Maximum Entropy
- Support Vector Machines [22]

The methods to gather the tweets from the API rely on some queries into the Twitter API. For example the query "google" will return 5 tweet in the data set, "Obama" will return 10 tweets.

The test dataset is built using a API query then the result set is marked as positive or negative.

Features	Keyword	Naive Bayes	MaxEnt	SVM
Unigram	65.2	81.3	80.5	82.2
Bigram	N/A	81.6	79.1	78.8
Unigram + Bigram	N/A	82.7	83.0	81.6
Unigram + POS	N/A	79.9	79.9	81.9

Figure 4.4: Classifier Accuracy

results We can see that the presented result are mostly over 80% which seems like precise for our usage.

Relation with our project The good information from this paper is that they use some machine learning techniques to be able to classify with success the tweets.

In our case this seems like a good option to classify negative or positive samples. The main issue here is that this system is not made to give a level of feeling like we are trying to get. If a scale can't be built a heat map is more difficult to make.

Another problem is that the system doesn't get any neutral tweets. This problem is kind of linked to the scale problem because if you use a scale you can rate it as neutral.

In our dataset, a lot of tweets are for sure neutral (informations, bots, media sharing, ...). Another issue is that the test dataset is marked manually. For our big and heterogeneous dataset that could be harder to make.

4.2 Selected implementations

4.2.1 system review

Dataset creation The creation of a dataset is an important part during the development of an AI. In order to create the dataset to train and test our emotional rating AI, we started to collect tweets with the Twitter API. These tweets are only coming from a geographical area that we defined with two points :

- The bottom-left corner : 48°48'44.0"N 2°14'08.9"E
- The top-right corner : 48°54'16.6"N 2°25'42.3"E

This area is Paris and a little bit more.

We collected almost 300 000 tweets since we started collecting tweets the May 7th 2018. We had some interruptions in our tweet collection because of the free Twitter API limits, but we managed to check if the collection was running everyday and launch it again if it wasn't still running.

After collecting all those tweets, we needed to generate a dataset in order to train our AI, this meant, because we choose this solution, to generate the input vectors and the labels for the tweets

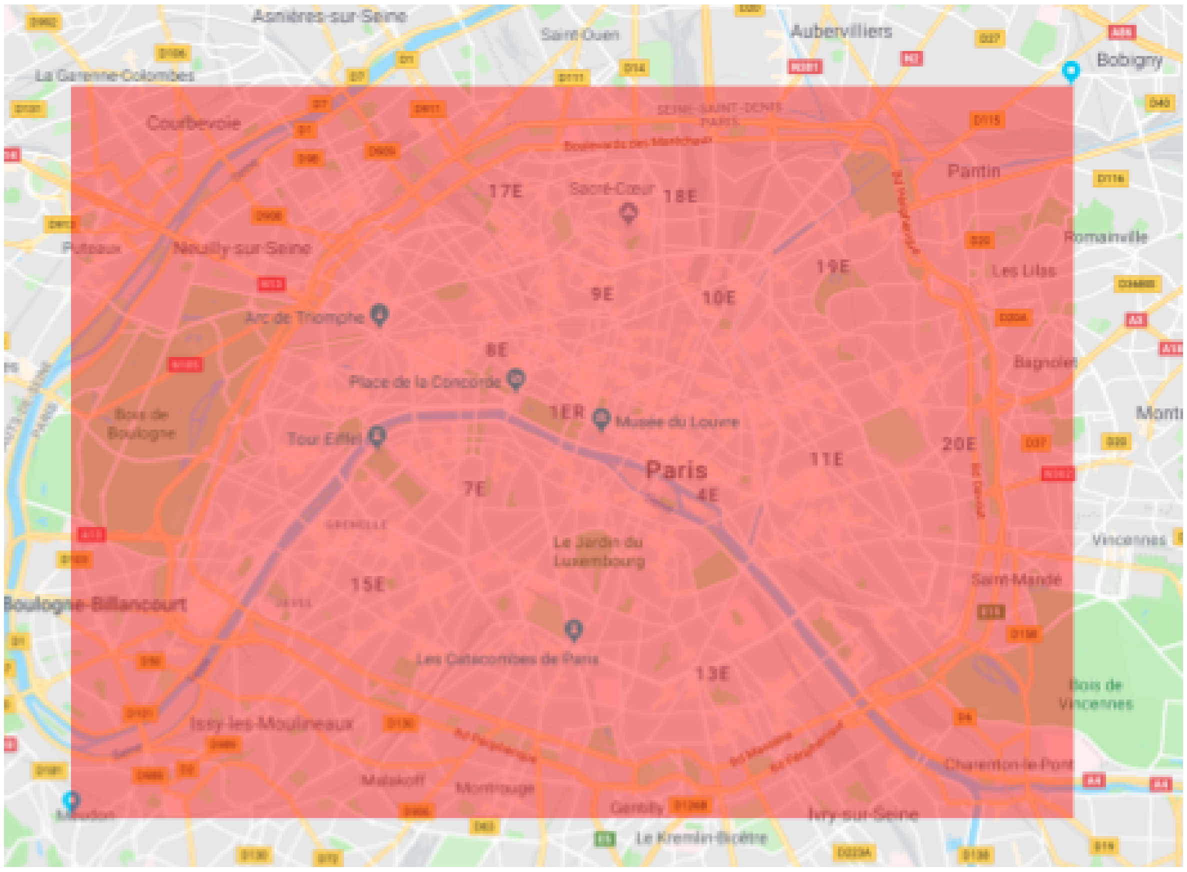


Figure 4.5: The area where we collect tweets

Generate the input vectors The first thing to do in order to generate an input vector is the data sanitisation we explained earlier. This gave us a list of stemmed words for each tweet in the database. What we need to do from this list of stemmed words, is generate an input vector that can be given to an SVM in order to train it, or to predict an output. So we made a list of all the stemmed words we had in the tweets, and the number of occurrence for each word we detected, and we kept a list of the tweets stemmed words. We decided to remove the words with less than 3 occurrences or with less than 2 letters, because we can't get any pattern from words that are not used enough, and word that have less than 2 letters are not really significant to express an emotion.

As we choose a solution with unigrams in input, a word is a component of the input vector. So the size of the input vector depends on the size of the list of words we created before, and that we cleaned from useless words.

The next step is to create the actual vectors. In order to do that for one tweet, we create a vector filled with 0, and we replace 0 by 1 at the indexes that corresponds to index of stemmed words in the list of words.

We generated the input vectors only for the tweets where we could also generate the labels, this filter reduce the size of our datasets

Generate the labels In our system, the labels are the score given to the tweet. This score can be either the positivity, the negativity, the neutrality or a formula using those values.

To generate the label in a first step, we used a list of 900 most common emojis [23] on Twitter then we applied the grades form a list which is similar to the Afinn list.

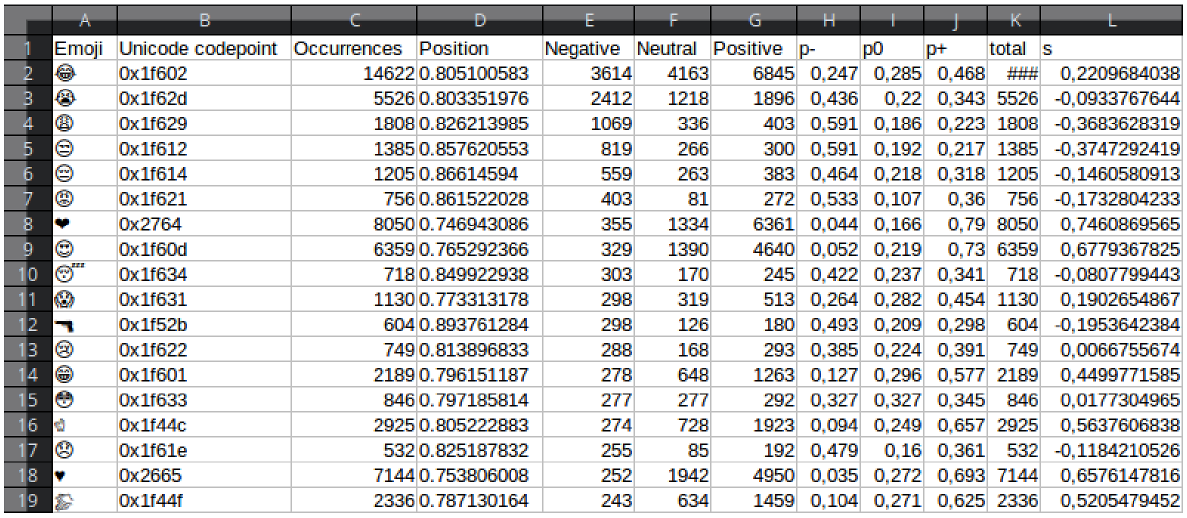
AFINN is a list of words rated for valence with an integer between minus five (negative) and plus five (positive). The words have been manually labeled by Finn Årup Nielsen in 2009-2011. Many sentimental analysis are base on this model [24].

We created our own AFINN-style list based on the emojis. By using the same annotation system but using emojis as inputs instead of words.

We tried to create our own list by categorizing emojis as simply positive or negative from the unicode list.Unfortunately this list wasn't flawless :

- The number of emoji presented in the list wasn't big enough compared to the 1644 (in june 2018) available items (only less than 80 items so less than 5%)
- This list only made us able to create a segregation between positive and negative tweets. The fact is that a scale would be a better idea to create to use the values in a formula to rate the emotion. Even if an emotion is similar in two cases, the intensity of this emotion could vary.

As a base we used the data from a previous study on about sentiment of emojis. In this study, the dataset is also composed of tweets (over 1.5 million but only approximately 65,000 of those contained an emoji), every tweet is manually annotated as positive, negative or neutral.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Emoji	Unicode codepoint	Occurrences	Position	Negative	Neutral	Positive	p-	p0	p+	total	s
2	😂	0x1f602	14622	0.805100583	3614	4163	6845	0,247	0,285	0,468	###	0,2209684038
3	😭	0x1f62d	5526	0.803351976	2412	1218	1896	0,436	0,22	0,343	5526	-0,0933767644
4	😩	0x1f629	1808	0.826213985	1069	336	403	0,591	0,186	0,223	1808	-0,3683628319
5	😒	0x1f612	1385	0.857620553	819	266	300	0,591	0,192	0,217	1385	-0,3747292419
6	😔	0x1f614	1205	0.86614594	559	263	383	0,464	0,218	0,318	1205	-0,1460580913
7	😡	0x1f621	756	0.861522028	403	81	272	0,533	0,107	0,36	756	-0,1732804233
8	❤	0x2764	8050	0.746943086	355	1334	6361	0,044	0,166	0,79	8050	0,7460869565
9	😍	0x1f60d	6359	0.765292366	329	1390	4640	0,052	0,219	0,73	6359	0,6779367825
10	😴	0x1f634	718	0.849922938	303	170	245	0,422	0,237	0,341	718	-0,0807799443
11	😱	0x1f631	1130	0.773313178	298	319	513	0,264	0,282	0,454	1130	0,1902654867
12	🔫	0x1f52b	604	0.893761284	298	126	180	0,493	0,209	0,298	604	-0,1953642384
13	😢	0x1f622	749	0.813896833	288	168	293	0,385	0,224	0,391	749	0,0066755674
14	😁	0x1f601	2189	0.796151187	278	648	1263	0,127	0,296	0,577	2189	0,4499771585
15	😳	0x1f633	846	0.797185814	277	277	292	0,327	0,327	0,345	846	0,0177304965
16	👌	0x1f44c	2925	0.805222883	274	728	1923	0,094	0,249	0,657	2925	0,5637606838
17	😞	0x1f61e	532	0.825187832	255	85	192	0,479	0,16	0,361	532	-0,1184210526
18	♥	0x2665	7144	0.753806008	252	1942	4950	0,035	0,272	0,693	7144	0,6576147816
19	👏	0x1f44f	2336	0.787130164	243	634	1459	0,104	0,271	0,625	2336	0,5205479452

Figure 4.6: sample of our emoji dataset

In this array we have :

- As informations,
  - Emoji

- Unicode codepoint
- Unicode name

- As values,
  - Occurrences: number of times the symbol is present a in the dataset
  - Position: average position of the symbol in each tweet

  As emotional values:

  - Negative : number of occurrences of the symbol in tweets annotated as Negative
  - Neutral : number of occurrences of the symbol in tweets annotated as Neutral
  - Positive : number of occurrences of the symbol in tweets annotated as Positive

- $p_- = occurrences/negative$
- $p_0 = occurrences/neutral$
- $p_+ = occurrences/positive$

$$s = -1 * p_- + 0 * p_0 + 1 * p_+ = p_+ - p_-$$

We used s as our emotional grade for the emoji.

- s has the range: -1 ¡ s ¡ +1 so the grade is positive in case of a positive emoji and negative in case of a negative one.
- s will be what we are calling the sentiment grade or score of the emoji.

Now to evaluate every tweet we have to compute a grade from the emoji and/or the text. To achieve this we used different cases :

- AFINN list + EMOJI list
- AFINN list
- EMOJI list

Then from the used list we get an array of grades from every detected item (word or emoji) We used different formulas to get the grade of one tweet from the dataset:

- Average
- Distance from 0

In case of the Average, we get a grade that mainly exploit the fact many sentiments can be expressed in one tweet and that we should consider everyone of them.

The Distance from 0 is also exploiting the fact many sentiments can be expressed in one tweet but in this case only the strongest one is taken in consideration.

This get us 3 ways of gathering grades from the items in the tweet and 2 ways of computing the final grade of the tweet.

We tested every possible combination to create the initial dataset to feed the neural network that will be able to rate every tweet coming into the system.

After generating the dataset, we needed to save it. In order to do this, we used the numpy function numpy.save() that allows us to save an entire python list. We can load a python list with another numpy.load().

Before saving, we needed to shuffle the inputs and the label at the same time (to keep the correspondence between them), and to generate a training dataset and a testing dataset. We decided to do this separation like this :

- 75% of the data for the training dataset
- 25% of the data for the test dataset

Having a training and a testing dataset is important, and allows us to evaluate the overfitting of the SVM after the training.

This way, we are training and testing our SVM with the training dataset, and testing it again with the test dataset.

Dataset name	**Size of input vector**	**Size of training data**	**Size of test data**
Unicode Emoji list	4319	17911	5970
Emotional Emoji, average score	8417	29534	9845
Emotional Emoji, furthest score	8417	29534	9845

Figure 4.7: Classifier Accuracy

Our datasets

Creating the machine learning algorithm For the *Unicode Emoji list* dataset, we decided to train a multiclass SVM. The labels are -1 for negative, 0 for neutral, and +1 for positive. In order to implement this multiclass SVM, we used the library scikit-learn [12] (a library specialised in machine learning algorithm). This library allows us to easily create and train a multiclass SVM, and set different parameters to it.

For the two *Emotional Emoji* datasets, it's not a classification problem, so we are not using a SVM, but the Tensorflow's [13] DNNRegressor. The DNN regressor is a deep neural network, with tensorflow, we can set the number of hidden layers, and the number of neurons in the hidden layers.

Performance evaluation and comparison between the algorithms In order to test our algorithm, we manage to create our dataset with a training dataset and a test dataset. It's important to test a machine learning algorithm with datas that are differents from the training data. This way, we can detect overfitting problems due to the training.

With tensorflow and scikit-learn, we can process the mean accuracy score, we're going to use this measurement to compare our machine learning algorithms.

Chapter 5

Event detection

The project is based on the detection and the rating of spatio-temporal events. It means that our events need a location and a duration. Once the events are detected they need to be validated and rated using the emotional rating of the tweets in the event.

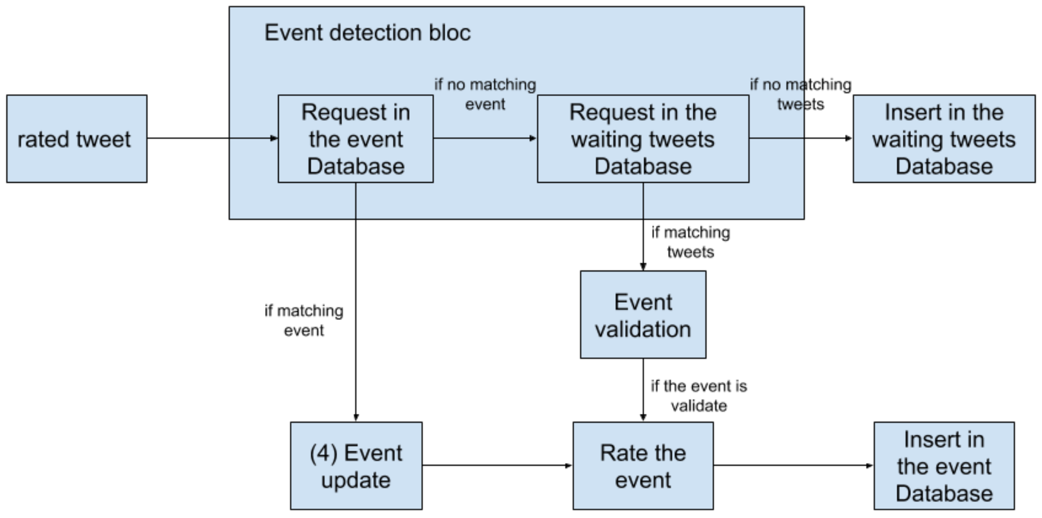


Figure 5.1: Architecture of the event detection and rating module

5.1 Bibliographic research

Doing our research on the event detection we found a lot of paper related to the detection based on hashtags and trending topics, but only a few to detect spatio-temporal events. The main problem studying theses papers was the number of tweets we were getting compared to the number of tweets needed for the techniques. We focused our work on the paper : Geo-spatial Event Detection in the Twitter Stream from Maximilian Walther and Michael Kaisser [25].

System Review The system is splited in 4 modules:

- TweetFetcher
- ClusterCreator
- ClusterUpdater
- ClusterScorer

The TweetFetcher gets new tweets from Twitter's public APIs and add it in a queue to be written in a database.

The ClusterCreator is trying to creat cluster by linking tweets:

> For each new incoming tweet, this module checks whether there were more than x other tweets issued in the last y minutes in a radius of z meters. For the experiments in this paper, we used the settings x = 3, y = 30, z = 200. Whenever a new cluster can be created this way, it is written to the table EventCandidates. [25]

The ClusterUpdater updates existing clusters when a tweet can directly be added to it or if events overlap.

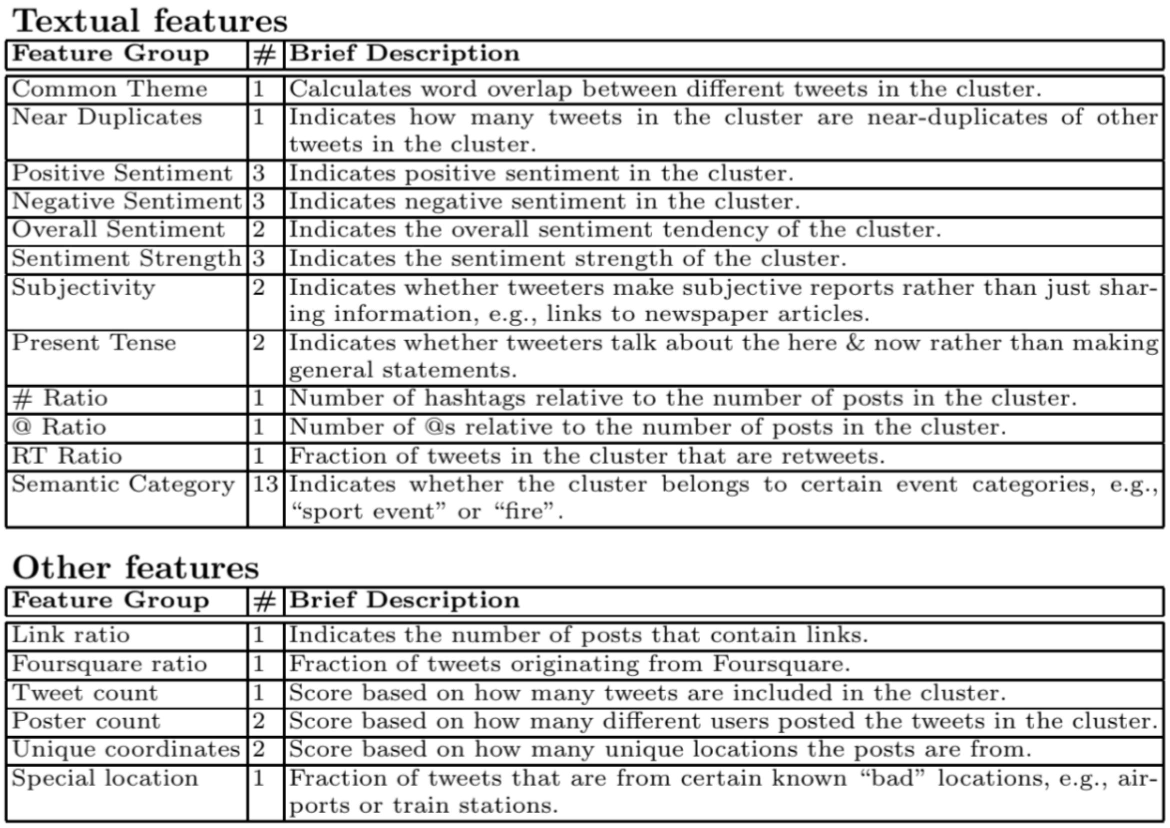
The ClusterScorer is giving a score to the cluster, depending on the confidence level.

Relation with our subject For our project we can't apply directly what is in this paper because they say the get 3 000 000 tweets every 24 hours. By we used this paper to build an efficient event structure.

Here is the tab of the features used in this paper:

Results Using theses features and different machine learning algorithms they get the following results:

- Naive bayes: precision 0.783
- Multilayer Perceptron: precision 0.829
- Pruned C4.5 decision tree: precision 0.858
- ZeroR: precision 0.319



Textual features

Feature Group	#	Brief Description
Common Theme	1	Calculates word overlap between different tweets in the cluster.
Near Duplicates	1	Indicates how many tweets in the cluster are near-duplicates of other tweets in the cluster.
Positive Sentiment	3	Indicates positive sentiment in the cluster.
Negative Sentiment	3	Indicates negative sentiment in the cluster.
Overall Sentiment	2	Indicates the overall sentiment tendency of the cluster.
Sentiment Strength	3	Indicates the sentiment strength of the cluster.
Subjectivity	2	Indicates whether tweeters make subjective reports rather than just sharing information, e.g., links to newspaper articles.
Present Tense	2	Indicates whether tweeters talk about the here & now rather than making general statements.
# Ratio	1	Number of hashtags relative to the number of posts in the cluster.
@ Ratio	1	Number of @s relative to the number of posts in the cluster.
RT Ratio	1	Fraction of tweets in the cluster that are retweets.
Semantic Category	13	Indicates whether the cluster belongs to certain event categories, e.g., “sport event” or “fire”.

Other features

Feature Group	#	Brief Description
Link ratio	1	Indicates the number of posts that contain links.
Foursquare ratio	1	Fraction of tweets originating from Foursquare.
Tweet count	1	Score based on how many tweets are included in the cluster.
Poster count	2	Score based on how many different users posted the tweets in the cluster.
Unique coordinates	2	Score based on how many unique locations the posts are from.
Special location	1	Fraction of tweets that are from certain known “bad” locations, e.g., airports or train stations.

Figure 5.2: features event

5.2 Our implementation

5.2.1 Event definition

Our definition of an event is something that people can talk about during 48h and in a circle with a radius of 1km. The group of tweets in this area and in this period should talk about the same subject.

A event is defined by different parameters to allow us to verify it.

Commontheme(w1, ..., wn) = $\frac{1}{cm}\sum_{i=0}^{n} f(n)$ with f(n) = $word\_count[n] * number\_of\_tweets$ and if word_count[n] = total_words c a constant set to 3, m the number of words in the list.

5.2.2 Get matching event

First, when a new tweet is coming, the system tries to link the tweet with a existing event.

```
circle1 = r.circle([tweet["coordinates"]["coordinates"][0], tweet["coordinates"]["coordinate
events = r.table('events').get_intersecting(circle1, index='coordinates').filter(
   r.row['startDate'].to_epoch_time() > (tweet["created_at"] - r.epoch_time(172800))).run()
```

Feature name	description
commontheme	score base on the words in tweets
hashtags	list of features related to the hashtags
avgNumberOfHashtag	average number of hashtag per tweet
hashtagList	list of the hashtags
mostUsedHashtag	statistics on the most used hashtag
frequencyOnHashtags	number of appearance / number of different hashtag
frequencyOnTweets	number of appearance / number of tweets
numberOfAppearence	number of time the hashtag is in tweets
text	text of the most used hashtag
numberOfHashtag	number of different hashtag
id	id of the event
mentions	features related to the mentions
avgNumberOfMention	Average number of mention
mentionList	list of mention
mostUsedMention	statistics on the most used mention
frequencyOnMentions	number of appearance / number of different mentions
frequencyOnTweets	number of appearance / number of tweets
numberOfAppearence	number of time the mention is in tweets
text	text of the most used mention
numberOfMention	number of different mention
numberOfTweets	total tweets in the event
startDate	date of the first tweet in the event
users	statistics on the users
maxTweetPerUser	number of tweet for the user that tweeted the most
numberOfUser	number of different user
ratioMaxTweetPerUser	maxTweetPerUser / number of tweets
tweetPerUserAvg	Average number of tweet per user
userList	List of the users
wordsCount	number of apearence in the tweets of the word in the “wordsList”
wordsList	List of stemmed words in the tweet after cleaning

Figure 5.3: Structure of an event

This request gets all of the events ordered by proximity that can match with the current tweet. It is based on the location, and on the time. The event needs to be under a kilometer away from the tweet. And the start of the event should be less than 48h before.

```
if "extended_tweet" in tweet.keys():
   clean = clean_tweet(tweet["extended_tweet"]["full_text"], tweet["extended_tweet"]["entiti
else:
   clean = clean_tweet(tweet["text"], tweet["entities"])
stem_words = stemTweet(clean)
if get_tweet_ranking_score(listEvents[0]["wordsList"], listEvents[0]["wordsCount"], stem_wor
   update_event(listEvents[0], tweet)
```

Then this part is the verification of the matching with the tweet. First it “cleans” the tweet as explained in the tweet satitization part. Then this is used to get a score of the matching between the tweet and the event based on the words used.

```
def get_tweet_ranking_score(wordslist, wordscount, tweetstemedtext):
   tweetscore = 0
   for tmpword in tweetstemedtext:
       if tmpword in wordslist:
           tweetscore = tweetscore + wordscount[wordslist.index(tmpword)]
   return tweetscore/len(tweetstemedtext)
```

The tweet score is just a sum of the appearance of the words in the tweet in the words of the event. If the matching is good enough (limit fixed at a score of 2), the tweet is added to the event and the event parameters are updated.

5.2.3 Detect a new event

If no event in the database is matching the tweet, then the program tries to create a new one.

```
collection2 = r.table('waitingTweets').get_intersecting(circle1, index='coordinates').run()
```

This request gets all of the potentially matching tweets based on the location. The request is not considering the time because another script is continuously running the this waitingTweets database to delete the too old tweets (48 hours).

```
listTweets = list(collection2)
if len(listTweets) > 5:
```

We choose 5 tweets as the limit base of the reference paper we used for the event detection. In the paper they use 3 tweets as the limit but for us and because we want to detect bigger events we increased that number. Then, if the list contains more than 5 tweets, the list of tweets is considered as a potential event, and the tweets_list_to_event function is call.

```
listTweets.append(tweet)
# create new event with the tweet list
tweets_list_to_event(listTweets)
```

This function process algorithms on the list to extract the event parameters listed in the general event description. Then a first filter on the common theme is applied to avoid having potential event just based on the location and time witch are not related at all. A second one is used to avoid events with only one user talking, by verifying that less than 70% of the tweets are from the same person.

5.2.4 Validation of a new event

To certify that a potential event is actually one, the potential event is going through a multi-layer perceptron. The parameters of the event we used are:

- commontheme
- numberOfHashtag
- frequencyOnHashtags
- frequencyOnTweets (for the most used hashtag)
- NumberOfMention
- frequencyOnMentions
- frequencyOnTweets (for the most used mention)
- numberOfTweets
- numberOfUser
- ratioMaxTweetPerUser

5.3 Dataset and Results

The training dataset has been created from all the tweets we got. Then we filtered the tweets to keep only the ones in French, with a precise location.

Dataset 1: We had 8800 tweets for this one. We detected 201 potential events and 22 were real, about 10,9%. The specificity of this database is that to build the potential events we used the methode "get matching event".

Results of the AI:

Hidden layers	training	test	number of events detected
20, 10	88.8%	89.5%	0 on 0 events detected
15, 5	88.8%	89.5%	0 on 0 events detected
18	88.8%	89.5%	0 on 0 events detected
20, 10, 5	88.8%	89.5%	0 on 0 events detected

Figure 5.4: MLP performances dataset 1

Dataset 2: We have 9355 tweets. On these tweets we detected 225 potential events using filters on the location, the time, the theme and the number of users. The specificity of this one is that the potential events created are not using the "get matching event" part to link relevant tweets to it. The dataset 2 is composed of 14 events and 211 fake ones. So only 6,2% are real events.

The results of the training using different architectures of MLP are the following ones for the tests on dataset 2

Hidden layers	training	test	number of events detected
20, 10	95.3%	94.6%	5 on 6 events detected
15, 5	94%	93.3%	2 on 3 events detected
18	93.3%	93.3%	0 on 0 events detected
20, 10, 5	95.3%	93.3%	5 on 7 events detected

Figure 5.5: MLP performances dataset 2

The AI trained on the dataset 1 is supposed to be used on the events that are in the event database for them to be displayed. And the AI trained on the dataset 2 is supposed to be used only right after the detection of a new event, in the validation event block.

But because of the bad Datasets with only 10.9% and 6.2% of real events these good precisions are mostly due to the fact the the MLP return false for almost all of the events. The MLP trained on the first dataset detected no events during the training. The MLP trained on the second dataset detected 5 true events with the first configuration, 2 with the second, and 5 with the last one.

To try to find solution to this problem we combined the datasets to produce a more flexible AI. Dataset 3: 426 potential events with 36 real ones (8.5%)

Tests on dataset 2

Hidden layers	training	test	number of events detected
20, 10	93.3%	93.3%	1 on 2 events detected
15, 5	94%	93.3%	1 on 1 events detected
18	93.3%	93.3%	0 on 0 events detected
20, 10, 5	93.3%	93.3%	0 on 0 events detected

Figure 5.6: MLP performances dataset 3

According to this results we can say that this is still not working with a precision high enough to detect the events.

The last dataset (dataset 4) we tried is the dataset number 2 but taking the 22 true event and then 22 false events.

The results with this AI are the following while testing on the dataset number 2, so 22 real events:

Hidden layers	training	test	number of events detected
20, 10	64.6%	53.3%	12 on 97 events detected
15, 5	38%	28%	12 on 156 events detected
18	31.3%	36%	8 on 152 events detected
20, 10, 5	42%	52%	10 on 128 events detected

Figure 5.7: MLP performances dataset 4

According to these results we can see that for the 3 first AI they are too selective. This is due to the small number of real events compare to the number of fake events. The solution 4 is based on a dataset that avoid this problem, but there the problem is the really small amount of examples (44). The results are as expected, a AI that is more flexible and detect more events than what it should. A other problem possible but that we can't verify with these datasets is the choice of the features that are not different enough between a event and a fake one.

5.4 Improvements

As we saw in the previous parts the main problem we have is the density of tweets related to events and that we can use based on the location and the language. We go from around 360 000 tweets to only 500 actually in the real events (this number is not exact because tweets not related to the event can be included and tweets related can be forgotten). This is only 0.14% of the tweets.
Between the submission of the report and the oral, we will try to find a other dataset of event more complete to train an other AI and verify that the problem is coming from the number of data.
It is really difficult to detect the events we were hoping for (party in a bar, car accidents etc. . .). So we had to change and try to detect bigger events. For that we changed the size of the area that define an event and also the time window. But the events we are able to detect now are big events like concerts.
So the trend on Twitter related to these events is not that much time and location related but more topic related. For a huge football match, people are talking about the composition of the team days before the game and they are not sending the tweet from the stadium. That is why to improve our version, we need to combine what we have with a topic related technique to link not localized or not localized related and not time related tweets to the event.
When we were building our events datasets we noticed that one of the most important feature of a potential event to certify it is a real one are the hashtags and the mentions used. For example for the event in example (game of throne live) we have the hashtag gameOfThrones-Live and the mention AccorH_Arena.
But we don't have enough tweets to filter those without hashtag or mention informations.

Chapter 6

Project Management

The team is composed of Hervé-Madelein Attolou, Loïc Bachelot and David de Castilla. We are students in M1, and the three of us did the second semester together in Finland as exchange students.

6.1 Life cycle

Our project is based on 5 important features :

- Tweet collection
- Data sanitization
- Emotional rating of tweets
- Event detection
- Map output

In order to develop this different features, we decided to adopt Feature Driven Development life cycle, we chose this life cycle because our project has the particularity of having to do a bibliographic search before the actual development of the project (Figure 6.1).

The first thing we do before developing a feature is a research, we're looking for all the different solutions that are available in order to achieve our goal. For some features like the Tweet collection we didn't had a choice so the research ended with the beginning of the development. But for the more complicated tasks like the Emotional rating and the Event detection, the research never stopped.

6.2 Task repartition

Tab of the task repartition

6.3 Parallel working

6.4 Self learning Organization

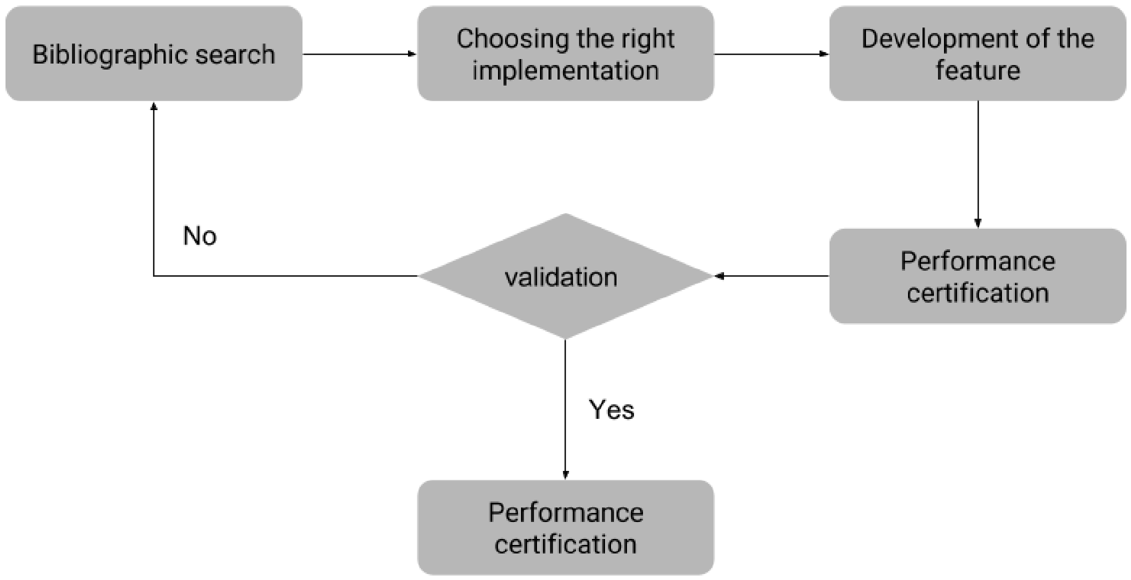

Bibliographic search
Choosing the right implementation
Development of the feature
No
validation
Performance certification
Yes
Performance certification

Figure 6.1: This graph represents the process of developing a feature in our project

Task	Planned date	Realization date
Tweet collection		
Data sanitization		
Emotional rating		
Event detection		
Map output		

Figure 6.2: List of the task and the planning

Task	**Hervé**	**Loïc**	**David**
Tweet collection			
Script API Tweeter			X
Statistiques		X	X
Data sanitization			
Stemming	X		X
Remove entities	X		
Emotional rating			
Dataset creation:			
- Emoji Unicode			X
- Emoji List average	X		
- Emoji List distance	X		
Machine learning			X
Event detection		X	
Map output	X		X

Figure 6.3: List of the task and the repartition

Chapter 7

Conclusion

Bibliography

[1] "twitter turns six, twitter blog," http://blog.twitter.com/2012/03/Twitter-turns-six.html.

[2] "I. lunden, analyst: Twitter passed 500m users in june 2012, 140m of them in us; jakarta biggest tweeting city, techcrunch," http://techcrunch.com/2012/07/30/analyst-twitter-passed-500m-users-in-june-2012-140m-of-them-in-us-jakarta-biggest-tweeting-city.

[3] N. A. Diakopoulos and D. A. Shamma, "Characterizing debate performance via aggregated twitter sentiment," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1195–1198. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753504

[4] A. Mittal and A. Goel, "Stock prediction using twitter sentiment analysis."

[5] S. Doan, B.-K. H. Vo, and N. Collier, "An analysis of twitter messages in the 2011 tohoku earthquake," in *Electronic Healthcare*, P. Kostkova, M. Szomszor, and D. Fowler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 58–66.

[6] "Subsea energy," http://www.subsea-energy.com/ .

[7] "World cup was biggest event yet for twitter with 672m tweets," https://www.theguardian.com/technology/2014/jul/15/twitter-world-cup-tweets-germany-brazil.

[8] Y. Yu and X. Wang, "World cup 2014 in the twitter world: A big data analysis of sentiments in us sports fans' tweets," *Computers in Human Behavior*, vol. 48, pp. 392–400, 2015.

[9] "tweetping worldcup analysis," https://www.youtube.com/watch?v=mgjZVohenRw.

[10] "rethinkdb," https://www.rethinkdb.com/.

[11] "pystemmer," https://github.com/snowballstem/pystemmer.

[12] "Scikit-learn."

[13] "Ttensorflow, an open source machine learning framework for everyone," https://www.tensorflow.org/.

[14] "mapbox," https://www.mapbox.com/.

[15] “google maps,” https://developers.google.com/maps.

[16] “porter stemmer,” http://www.tartarus.org/ martin/PorterStemmer.

[17] “Emoji statistics and trends,” https://emojipedia.org/stats/.

[18] K. Z. Bertrand, M. Bialik, K. Virdee, A. Gros, and Y. Bar-Yam, “Sentiment in new york city: A high resolution spatial and temporal view,” *arXiv preprint arXiv:1308.5010*, 2013.

[19] J. Bollen, H. Mao, and A. Pepe, “Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena.” *Icwsm*, vol. 11, pp. 450–453, 2011.

[20] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” vol. 150, 01 2009.

[21] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing.* Cambridge, MA: MIT Press, 1999.

[22] T. Joachims, “Making large-scale support vector machine learning practical,” in *Advances in Kernel Methods: Support Vector Machines*, A. S. B. Schölkopf, Ed. MIT Press, Cambridge, MA, 1998. [Online]. Available: citeseer.nj.nec.com/joachims98making.html

[23] “Real time most used emojis,” http://emojitracker.com/.

[24] “Afinn lexic source,” http://darenr.github.io/afinn.

[25] M. Walther and M. Kaisser, “Geo-spatial event detection in the twitter stream,” in *European conference on information retrieval.* Springer, 2013, pp. 356–367.