Université Polytechnique HAUTS-DE-FRANCE

UNIVERSITY OF OULU

MACHINE LEARNING ON TWITTER USING PYTHON

IUT département informatique MAUBEUGE

Région Hauts-de-France

Erasmus+

THANKS TO :

First of all, I would like to thank the two universities that gave me the opportunity to go on an internship abroad: the University of Valenciennes, and the University of Oulu.

I would also like to thank the Erasmus + programme and the Mermoz programme, which obviously allowed me to do this internship first, but also to ensure my financial coverage, so as not to cause financial problems.

Finally, my greatest thanks go to my current training supervisor in Finland, Mr Mourad Oussalah, but also my training supervisor in France, Mr Philippe Polet, who allowed me to follow and help me during this internship.

PRESENTATION OF THE UNIVERSITY

History :

The University of Oulu (Finnish: Oulun yliopisto) is one of the largest universities in Finland, located in the city of Oulu. It was founded on July 8, 1958. The university has now more 16,000 students and 3,000 staff. The university is often ranked as one of the better universities in Finland and in the top-400 worldwide.

The different campuses :

There are four campus areas:

- The main campus is located in Linnanmaa, about 5 kilometres (3 mi) north of Oulu city centre. It includes five faculties, Infotech Oulu, Thule Institute, Giellagas Institute, Martti Ahtisaari Institute, Centre for Wireless Communications (CWC), Centre for Advanced Steel Research (CASR), Center of Microscopy and Nanotechnology, Botanical Gardens and Museum, Zoological Museum, Geological Museum and two science libraries (Pegasus and Tellus).

- Faculty of Medicine has its own campus area and it is fully integrated in the regional hospital 2 kilometres (1.2 mi) southeast from the city center, Kontinkangas, Oulu. Clinical Research Centre (CRC), Welltech Oulu and the Medical Library are also located there.

- The third campus area is located in Kajaani, about 185 kilometres (115 mi) southeast of Oulu.

- Department of Architecture which is located in the center of Oulu

Geolocalisation :

The university is located in Linnanmaa in the north of Oulu, close to two big lakes, lots of shops, student housings and it's served by lots of busses which go to the city center.

Opening hours :

The university is opened to public on weekdays from 7am to 4pm (FI time zone) but opened for students and staff 24/7. But on Saturdays, only one cafeteria is opened and they are all closed on Sunday.

PRESENTATION OF THE INTERNSHIP

Introduction :

Following the last Finnish elections in mid-April, it was interesting to monitor social networks in order to collect as much data as possible in order to get people's opinions, but also how the various political parties, as well as their presidents and vice-presidents, reacted. Indeed, it is by doing this that we can analyse the behaviour of a given person or organisation and thus identify the main information and feelings.

My project :

Natural language processing analysis on the Finnish elections.

My work :

Collect thousands of election tweets and tweets from Finnish political parties in a database. Analyze them through pre-processing with NLTK, using the Stanford algorithm, Lemmatization, LDA. And finally analyze them according to feelings.

Means used :

Computer with IDLE development interface to work with Python language, as well as multiple libraries.

HOW, WITH PROGRAMMING TOOLS, IS IT POSSIBLE TO INTERPRET THE RESULTS OF THE FINNISH ELECTIONS?

LET'S TALK ABOUT NLTK

What is NLTK ?

Natural Language Toolkit (NLTK) is a Python software library for automatic language processing. Thanks to this library it is possible to break down each sentence in order to analyze each word and to extract different information. This is one of the basics of machine learning.

NLTK is therefore in the form of a library to import under Python (for my part I use the IDLE interface) :

```
import nltk
```

When I arrived at Oulu University, I first had to familiarize myself with the Python language, which was facilitated by my two previous years of study. I then started to learn how to use the NLTK library thanks to this site which allowed me to use it 100%: **https://www.nltk.org/book/.**

First, let's talk about what **tokenization** is :
It is a very simple algorithm that allows you to separate each word of a sentence and to take out a characteristic. Here under the example below I use tokenization as well as the "pos-tag" which allows me to return for each word its attribute (NN = name, NNP = proper name, etc...) :

```
import nltk
from nltk.tokenize import *

document = "Bonjour et bienvenue sur NLTK"

mots = nltk.sent_tokenize(document)

for mot in mots:
    print(nltk.pos_tag(nltk.word_tokenize(mot)))
```

```
[('Bonjour', 'NNP'), ('et', 'CC'), ('bienvenue', 'NN'), ('sur', 'NN'), ('NLTK',
'NNP')]
>>>
```

As said before, this is one of the basics of machine learning and we will use tokenization to run other algorithms later on.

Stopwords :

Let's now talk about a function that is widely used with tokenization and that we will use for the final project. It's about stopwords.

It is in fact a list of words whose language is specified (any language), very simple words that we do not find useful in our future data collection (such as "the", "as", etc.), and that we want to eliminate so as not to hinder the functioning of algorithms. The stopword list will therefore compare the given text with the recorded words and eliminate them during tokenization :

```
from nltk.tokenize import *
from nltk.corpus import *

data = "Hello my name is Theo and i use Python"
stopWords = set(stopwords.words('english'))

phrases = sent_tokenize(data)
words = word_tokenize(data)
wordsfiltered = []


for w in words:
    if w not in stopWords:
        wordsfiltered.append(w)

print(wordsfiltered)
```

```
['Hello', 'name', 'Theo', 'use', 'Python']
>>>
```

We will now discuss two last algorithms before we start the final project.

Porter-Stemmer algorithm:

A very simple algorithm that we will not use in the project but which allows on a list of words that starts in the same way to extract this part.

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

words = ["habitant","habitation","habite","habiter"]
ps = PorterStemmer()

for word in words:
    print(ps.stem(word))
```

```
========= RESTART: C:\Users\TheoLC\Desktop\python\porter_stemmer1.py =========
habit
habit
habit
habit
```

We can therefore see that on the list of given words the program returns the part of the identical word.

Stanford algorithm :

It is also a very simple algorithm since it will decompose a sentence with tokenization, and will take out the proper names by putting them in a category (city names, first names, etc.). It is an extremely useful algorithm and demonstrates the intelligence of the Python language and the NLTK library :

```
import os
java_path = "C:/Program Files (x86)/Java/jre1.8.0_211/bin/java.exe"
os.environ['JAVAHOME'] = java_path

import nltk

nltk.internals.config_java("C:/Program Files (x86)/Java/jre1.8.0_211/bin/java.exe")

from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize

st = StanfordNERTagger('stanford/classifiers/english.all.3class.distsim.crf.ser.gz','stanford/stanford-ner.jar',encoding='utf-8')
text = input('Entrer la phrase que vous voulez, si possible composé de prenoms, verbes, pays etc \n')
tokenized_text = word_tokenize(text) # On divise la phrase morceau par morceau, en tokens
classified_text = st.tag(tokenized_text)

print(classified_text)
```

```
[('hello', 'O'), ('my', 'O'), ('name', 'O'), ('is', 'O'), ('Theo', 'PERSON'), ('
and', 'O'), ('i', 'O'), ('live', 'O'), ('in', 'O'), ('Oulu', 'LOCATION')]
```

As can be seen, "Theo" has been placed in the category "Person", and "Oulu" has been placed in the category "Location".

It is one of the most intelligent algorithms because even if the code becomes a little more complex because of the different direction paths to configure and the folder to download, it is very practical.

Then you will see that I will use the Stanford algorithm for the project and that it will be quite useful to me.

PROJECT

At the start :

Let us now turn to the project, which I would like to remind you of, is to collect as many tweets as possible in connection with the Finnish elections, from the various political parties, to find a healthy way to store them, and finally to analyse them in different ways.

At the beginning, I had to learn to use a library that allowed me to run this project: TWEEPY.
This API allows you to link a confirmed Twitter developer account to Python to extract data from it.

For that I had to create a Twitter account and then validate it as a developer account, and thus get 4 tokens, which with Tweepy allowed me to connect to my Twitter account (To be entered between " ".) :

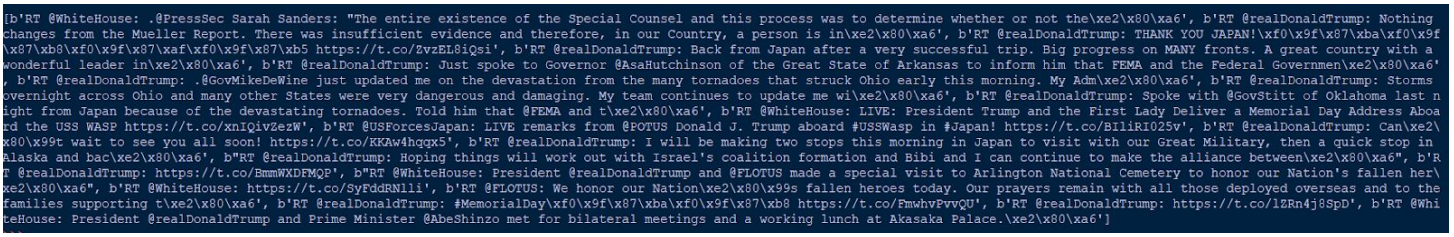
```
import tweepy

auth = tweepy.OAuthHandler("", "")

auth.set_access_token("", "")

api = tweepy.API(auth, wait_on_rate_limit = True)
```

Then I started to learn the features like being able to send tweets (Be careful not to send them too fast otherwise Twitter interprets us as a bot). But also features much more useful in our project since they will allow us to, for example, retrieve tweets from a given user. Or tweets on a given subject :

Below, a Python function to retrieve the last 200 tweets from a selected user :

```
def get_tweets(username):

    api = tweepy.API(auth)

    number_of_tweets = 200
    tweets = api.user_timeline(screen_name=username)

    tmp = []

    tweets_for_csv = [tweet.text.encode("utf-8") for tweet in tweets]

    for j in tweets_for_csv:

        tmp.append(j)
        print("\n")
    print(tmp)
```

This was one of my first programs so the rendering is not necessarily excellent and moreover, the encoding is not well applied, and will be shown in the final project.



We can also find the use of the Stanford algorithm in collaboration with Tweepy :

```
search_word = input("entrer le mot que vous chercher : ")

date_since = "2019-04-01"

tweets = tweepy.Cursor(api.search, q = search_word, lang = "en", since = date_since).items(1)

tweets_text = [[tweet.text] for tweet in tweets]

data_final = str(tweets_text)

tokenized_text = word_tokenize(data_final) # On divise la phrase morceau par morceau, en tokens
classified_text = st.tag(tokenized_text)

print(classified_text)
```

```
entrer le mot que vous chercher : finnish elections
[('[', 'O'), ('[', 'O'), ("'", 'O'), ('@', 'O'), ('spectator', 'O'), ('Not', 'O'
), ('just', 'O'), ('Sweden', 'LOCATION'), (',', 'O'), ('Finland', 'LOCATION'), (
'too', 'O'), ('.', 'O'), ('Finnish', 'O'), ('Greens', 'O'), ('came', 'O'), ('sec
ond', 'O'), ('in', 'O'), ('EU', 'LOCATION'), ('elections', 'O'), ('.', 'O'), ('I
n', 'O'), ('Helsinki', 'ORGANIZATION'), ('City', 'ORGANIZATION'), ('Council', 'O
RGANIZATION'), ('they…', 'O'), ('https', 'O'), (':', 'O'), ('t.coSQL5kyITYH', 'O
'), ("'", 'O'), (']', 'O'), (']', 'O')]
>>>
```

It is also time to present the algorithm we will use to analyze our tweets. This algorithm is called the LDA and allows, on a more or less large list of sentences, to highlight the most used words, regardless of the language (it is of course advisable to use stopwords beforehand in order not to distort the calculations).

It is with this presentation that we now move to the heart of the matter :

To begin with, I was looking to simply retrieve the tweets related to the Finnish elections, in order to build a semblance of a database. For this reason, in my first version, I had the idea to use a translation API like Google's to translate "finnish elections" into French, and into Finnish in order to expand my search field and collect more tweets.

The program then returned the number of tweets found on the Finnish elections as planned.

I then translated each French and Finnish tweet into English in order to have a uniform and readable database. For that I also had to use decoding / encoding, so that the special characters (Finnish for example) are all interpreted correctly.

In my first version, I first saved the tweets in a .json file, which served as a provisional database.

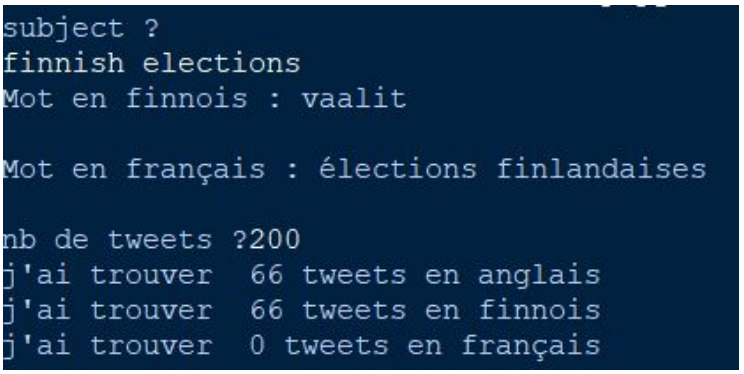
This first version worked almost without any problems.

But :
- The database was not large enough to be analyzed later.
- There were still encoding errors in the rendering of the JSON file.
- The JSON format, and CSV (which I tested later) was not really perfect for a database.
- And finally the most important problem: All translation APIs have a limit of characters to be translated (usually 15000 characters). This was not a problem at first because I only contained 200 tweets more or less. But then I will exceed that limit.

So I had to, after an interview with Mr. Oussalah (whom I saw every week to follow the scope of my work), decided to rethink everything from the beginning.

Below is the first version of my program, so it only works with a small database :

```
search_word = input("subject ? \n")

search_word = TextBlob(search_word)

search_word_finnish = translator.translate(str(search_word), dest='fi')

search_word_french = translator.translate(str(search_word), dest='fr')

print("Mot en finnois : " + str(search_word_finnish.text) + " \n")
print("Mot en français : " + str(search_word_french.text) + " \n")


searched_tweets = []

taille = input("nb de tweets ?")

new_tweets_en = api.search(search_word, count=int(taille)/3)
new_tweets_fi = api.search(search_word_finnish.text, count=int(taille)/3)
new_tweets_fr = api.search(search_word_french.text, count=int(taille)/3)


print("j'ai trouver ", len(new_tweets_en), "tweets en anglais")
print("j'ai trouver ", len(new_tweets_fi), "tweets en finnois")
print("j'ai trouver ", len(new_tweets_fr), "tweets en français")

if not new_tweets_en and not new_tweets_fr and not new_tweets_fi:
    print("pas de tweets trouves")



for tweet in new_tweets_fi:
    tweet.text = translator.translate(str(tweet.text), src='fi', dest='en')

for tweet in new_tweets_fr:
    tweet.text = translator.translate(str(tweet.text), src='fr', dest='en')



new_tweets = new_tweets_en + new_tweets_fr + new_tweets_fi
searched_tweets.extend(new_tweets)


with open("%s_tweets.json" % search_word, 'a', encoding='utf-8') as f:
    for tweet in new_tweets:
        json.dump(tweet._json, f, indent=4, ensure_ascii=False)
```



To get a good version of my program that works I had to solve each problem as it went along.

To begin with, I put the idea of translation in brackets because it became impossible later on and I will explain why.

First problem :

One of the first problems solved is encoding. Indeed, the Finnish characters as well as the smileys were not correctly interpreted by the program and therefore came out with strange characters like "/xc2", etc.
So I did some research on the internet before finding a perfect library that via a function to decode and encode each character, regardless of its language, to output it in any type of file.

This library is called **unidecode** and really allows miracles :
https://pypi.org/project/Unidecode/

The encoding problem was solved, which then allowed me to address one of the big problems:

Second problem :

How to efficiently expand the database?

The solution to this problem was given to me by Mr. Oussalah, whom I thank, because he asked me to collect not only the tweets related to the Finnish elections, but also the tweets of each Finnish political party, which I then extended on my own initiative, to the president of each party, as well as the vice-presidents, and secretaries.

This solution has allowed me to go from a database of 200 tweets to more than 6000 tweets, which is a considerable step forward.

Third problem :

The problem of searching for tweets having been solved, I now needed, to accompany this, a support in order to be able to store everything. The CSV and JSON format did not suit me, so I started searching again, and decided to use SQLite.

It is a library to download and import into a Python program, and which allows to create a real database (as a db file) which makes the storage of tweets more concrete because I can create different tables for each type of tweets.

However, to be able to use SQLite correctly, you must first "connect" to the db file you want to create, as shown below :

```
def main():

    database = "C:\\Users\TheoLC\AppData\Local\Programs\Python\Python37\lib\sqli

    try:
        conn = sqlite3.connect(database)
        print("Connecté a la base de données ! \n")

    except Error as e:
        print(e)
```

As you can see it is necessary to indicate the direction path of your db file.

Once the database is created, it is necessary to assign to a variable, the object conn.Cursor as below in order to be able to execute our queries as the creation of tables but also the insertion, and the selection :

```
c = conn.cursor()
c.execute(create_table_sql)
print("Table créée")
```

For each created table, the information to be stored is therefore the tweet id, the username that tweeted, and of course the content of the tweet.

Last problem :

The storage problem having been solved, all I had left was the translation problem.

Problem that remained unresolved, because as explained before, each translation API has a character limit which prevents me from translating all tweets.
So I thought about the problem and rather than translate all the tweets into English, I preferred to translate the few tweets that were not into Finnish in order to have a uniform database in the same language. After explaining this alternative to Mr. Oussalah, he validated the use of this technique. Thus, each tweet recorded in the database is in Finnish.

Below is an example of the recorded tweets. The table concerned is the table listing the tweets of the official @Demarit account, which is the Finnish Social Democratic Party (which explains why I did not have to record the username as well) :

id_tweet	content
1132980455941300200	RT @AnttiRinnepj: Lähetimme Säätytalolta yhdessä terveiset maailmanmestareille. Onnea, @le…
1132968797151858700	Paljon onnea SDP:n mepeille @EeroHeinaluoma'lle ja @miapetrakumpula'lle! EU🌹 #EUvaalit2…
1132966679229685800	Onnea #Leijonat! 🌹FI https://t.co/ffdJlcnst0
1132900766434435100	RT @SDPhelsinki: Kiitos Mippe ja Eero! Upea tulos 👏 #EUvaalit2019 @miapetrakumpula @E…
1132900727838388200	RT @SatuTaavitsaine: Oli ikimuistoinen eurovaali-ilta! Jännitin @Demarit tulosta, omaa tulosta j…
1132747077241516000	RT @AnttiRinnepj: Uskomatonta, upeaa Suomi ja @leijonat! Suomi on jääkiekon maailmanmes…
1132746192276852700	RT @TyttiTup: Mahtavaa, #leijonat! 🏒 Suomi voitti, Eurooppa voitti, tämä ilta painuu historiaan…
1132746180188938200	RT @TuulaHaatainen: Suomi on maailman mestari!!Mahtavaa Leijonat upeaa!! Isosti onnea!!! #…
1132746011108098000	RT @EeroHeinaluoma: Tästähän tuli mahtava viikonloppu! Kiitos huikeasta tuesta eurovaaleiss…
1132744574923956200	Mahtava tulos ja upea ilta! Kiitos erityisesti loistaville ehdokkaille, kaikille kampanjaa tehneille s…
1132743325247782900	RT @eveheinaluoma: Meidän uudet mepit!🌹Eero ja Mippe! EU Aivan huikea tulos! Onnea isälle!…
1132717959045099500	RT @TopiVihtori: Hyvä meidän joukkue ja keihäänkärjet @EeroHeinaluoma ja @miapetrakump…
1132638846628319200	Vielä ehdit äänestää europarlamenttivaaleissa! Äänestyspaikat sulkeutuvat kello 20. Meillä jokai…
1132630053702033400	RT @iidavallin: We don't brexit. We fix it. Vielä neljä tuntia aikaa äänestää! EUFI #eurovaalit2019…
1132629993895407600	RT @TyttiTup: Käy äänestämässä tänään! #ThisTimeImVoting #tälläkertaaäänestän #samasuu…
1132600873203511300	RT @TimmermansEU: Final morning of the campaign in Vienna. Now it's in your hands 👍 Don'…
1132600497460981800	RT @Tommikael: Oheisella videolla @pekkashemeikka ja Jutta Urpilainen keskustelevat EU:sta…

And here below is a list of tweets related to the Finnish elections, and therefore of any user (and so here you can see that each user name is quoted) :

id_tweet	username	content
1129678590403067900	tarupasa	Onko nyt jotkut vaalit? Mikseivät ne näy? Asun kuitenkin Suomen neljä…
1129695195266789400	homsedeurovali2	RT @tarupasa: Onko nyt jotkut vaalit? Mikseivät ne näy? Asun kuitenki…
1129769523769430000	Herr_Meenzer	Saksalaiset neuvovat pikkuveljeään: uudet vaalit. Kurz tulee julkisuute…
1129773860100935700	ElinaRavantti	Itävallan kansallismielisten populistien mukaan edessä uudet vaalit. Liit…
1129797698356424700	Etela_Helsinki	RT @AuraSalla: Korkean teknologian maassamme Kemin sellutehdas …
1130080996496478200	Teppo_Virta	👍.... EU on Suomen paras keino torjua ilmastonmuutosta ja paras kei…
1130358375789846500	SANLFKAF	Tiesitkö, että enemmistö Suomen äänioikeutetuista on naisia? Naiset …
1130371010119852000	TerhiHeinila	RT @SANLFKAF: Tiesitkö, että enemmistö Suomen äänioikeutetuista …
1130389035275423700	BrigantiaT	RT @SANLFKAF: Tiesitkö, että enemmistö Suomen äänioikeutetuista …
1130425394912550900	VirpiKoskinen	RT @SANLFKAF: Tiesitkö, että enemmistö Suomen äänioikeutetuista …
1130531019545874400	TimoVKorhonen	Suomen kannalta huipputärkeät vaalit! https://t.co/K056ygVmfM
1130553644804399100	JariRuotsalain6	@valtioneuvosto @oikeusmin Menettää eduskunta vaalit merkityksens…
1130661227640832000	AskoKeranen	@Europarl_FI Päivä päivältä EU vaalit kiinnostaa minua vähemmän ja …
1130823974768775200	keskusta	.@juhasipila #EUvaalit2019 ovat tärkeät. Vaalit ratkeavat vasta sillä, ku…
1130825944468459500	MattilaJoni	RT @keskusta: .@juhasipila #EUvaalit2019 ovat tärkeät. Vaalit ratkeav…
1130828505632186400	SMTalala	RT @keskusta: .@juhasipila #EUvaalit2019 ovat tärkeät. Vaalit ratkeav…
1130831051629187100	337Tara	RT @keskusta: .@juhasipila #EUvaalit2019 ovat tärkeät. Vaalit ratkeav…

Before presenting my code as I go along, I will first tell you about the different analyses performed on the tweets collected, as the project progresses. You will see that multiple techniques have been used and some will not have been retained.

First of all, when I still used the JSON file to save the tweets, I obviously did the tokenization so that I could then use other algorithms. Each tweet was recorded in the file and separated from all its words.

Then through SQLite I decided to tokenize the tweets without even saving them so as not to waste time and to be able to process the large database more quickly.

Algorithms such as pos-tag, lemmatization, or even carrier-stemmer didn't seem to me to be really useful for analyzing anything relevant. So I left them out and decided to use the Stanford algorithm first to observe which proper names (names of people, names of parties) were most commonly tweeted. The program worked very well, with the difference that for each iteration of the Stanford algorithm, one Java window was executed to use it (example below). This obviously posed a huge time problem since to process a database of more than 6000 tweets would have taken hours or even several days.

Moreover, after careful consideration, it appears that the information collected was not necessarily the most relevant.

What algorithms?

Let's now talk about the algorithms used in the final version of my program:

The first is therefore the LDA that we mentioned earlier.

Indeed, unlike Stanford's algorithm, LDA will analyze the entire list of tweets very quickly and all at once, to bring out the most used words. The information was therefore quickly revealed but also very relevant following the use of stopwords to eliminate link words.

The second algorithm is the analysis of feelings.

This algorithm is very simple to use because for each language a list of "positive" words and a list of "negative" words are used to analyze the list of tweets and define if each tweet is "optimistic" or the opposite. You will then be able to see how we were able to draw up a scale of optimism.

Final code :

It is now time to present you my code in order to analyze it and explain these functions :
First of all, it is necessary to first (as shown above), establish the direction path of the future database and connect to it :

```
database = "C:\\Users\TheoLC\AppData\Local\Programs\Python\Python37\lib\sqlite3\pythonsqlite.db"

try:
    conn = sqlite3.connect(database)
    print("Connecté a la base de données ! \n")

except Error as e:
    print(e)
```

Once connected you must identify yourself to the Twitter API and enter your tokens

```
auth = tweepy.OAuthHandler("", "")

auth.set_access_token("-", "")

api = tweepy.API(auth)

translate_urls = ["translate.google.com", "translate.google.co.kr",
                  "translate.google.at", "translate.google.de",
                  "translate.google.ru", "translate.google.ch",
                  "translate.google.fr", "translate.google.es"]

translator = Translator(service_urls = translate_urls)
```

:
The search for tweets begins by collecting the last tweets posted in connection with the Finnish elections using keywords :

Here is what this portion of the code tells us :

```
Recherche de tweets sur les futures élections finlandaises
Mot en finnois : Suomen vaalit

j'ai trouver  181 tweets en finnois sur les élections finlandaises
```

```
print("Recherche de tweets sur les futures élections finlandaises")

search_word = TextBlob("élections finlandaises")

search_word2 = TextBlob("presidentin äänestys")

search_word3 = TextBlob("presidentin valinta")

tBlob("finnish electi
 = translator.transl                    dest='fi')
bis : " + str(search_                   ' \n")
```

Now for the future, it is necessary to make a slight political point in order to introduce you to the different Finnish parties so that you can understand the rest of the programme.

In Finland there are 9 main political parties as follows:
- Social Democratic Party
- Finns Party
- National Coalition Party
- Centre Party
- Green League
- Left Alliance
- Swedish's People Party
- Christian Democrats
- Blue Reform

In the rest of my program and especially just now, I will give the main information of each twitter account of each party, such as the number of followers. Below is an example of the code and what the program returns :

```
print("recherche des principaux partis finlandais : \n")

parti1 = api.get_user('Demarit')

print("Parti trouvé : " + parti1.name + "\n")
print("Localisation de " + parti1.name + " : " + str(parti1.location) + "\n")
print("Nombre d'abonnés de " + parti1.name + " : " + str(parti1.followers_count) + "\n")
```

```
recherche des principaux partis finlandais :

Parti trouvé : Sosialidemokraatit

Localisation de Sosialidemokraatit : Finland

Nombre d'abonnés de Sosialidemokraatit : 21065
```

I don't show the entire search code because it's the same for each party.

It is therefore time to collect massively for each political party, the most recent tweets. So I set a limit of 400 tweets maximum per party, and I decide to include retweets because they can sometimes say a lot about someone, something :

```
parti1_tweets = api.user_timeline(screen_name='Demarit', count=400, include_rts = True)
parti2_tweets = api.user_timeline(screen_name='keskusta', count=400, include_rts = True)
parti3_tweets = api.user_timeline(screen_name='kokoomus', count=400, include_rts = True)
parti4_tweets = api.user_timeline(screen_name='KDpuolue', count=400, include_rts = True)
parti5_tweets = api.user_timeline(screen_name='persut', count=400, include_rts = True)
parti6_tweets = api.user_timeline(screen_name='SiniTulevaisuus', count=400, include_rts = True)
parti7_tweets = api.user_timeline(screen_name='sfprkp', count=400, include_rts = True)
parti8_tweets = api.user_timeline(screen_name='vasemmisto', count=400, include_rts = True)
parti9_tweets = api.user_timeline(screen_name='vihreat', count=400, include_rts = True)
```

Tweets are therefore collected for each party in a different dictionary.

To continue in the search, and as explained above, I went to look for the president of each party in order to get their Twitter account and check their tweets (Same process as for political parties, I post its number of followers and collect its tweets) :

```
print("Recherche des leaders de chaque parti : \n")

leader_parti1 = api.get_user('AnttiRinnepj')

print("Nom du leader du parti Demarit : " + leader_parti1.name + "\n")
print("Nombre d'abonnés : " + str(leader_parti1.followers_count) + "\n")
```

```
Recherche des leaders de chaque parti :

Nom du leader du parti Demarit : Antti Rinne

Nombre d'abonnés : 28963
```

```
leader_parti1_tweets = api.user_timeline(screen_name='AnttiRinnepj', count=400, include_rts = True)
leader_parti2_tweets = api.user_timeline(screen_name='juhasipila', count=400, include_rts = True)
leader_parti3_tweets = api.user_timeline(screen_name='PetteriOrpo', count=400, include_rts = True)
leader_parti4_tweets = api.user_timeline(screen_name='SariEssayah', count=400, include_rts = True)
leader_parti5_tweets = api.user_timeline(screen_name='Halla_aho', count=400, include_rts = True)
leader_parti6_tweets = api.user_timeline(screen_name='Simon_Elo', count=400, include_rts = True)
leader_parti7_tweets = api.user_timeline(screen_name='anna_maja', count=400, include_rts = True)
leader_parti8_tweets = api.user_timeline(screen_name='liandersson', count=400, include_rts = True)
leader_parti9_tweets = api.user_timeline(screen_name='Haavisto', count=400, include_rts = True)
```

To further expand the database I have launched tweets collections on the known vice presidents of each party, for some of them there are up to 3 vice presidents, for others, none are known as shown below :
As you can see the database is therefore quite large and will allow you to have big data to analyze.

```
viceleader_parti1_tweets = api.user_timeline(screen_name='AnttiLindtman', count=400, include_rts = True)
viceleader2_parti1_tweets = api.user_timeline(screen_name='MarinSanna', count=400, include_rts = True)
viceleader3_parti1_tweets = api.user_timeline(screen_name='KristaKiuru', count=400, include_rts = True)

viceleaders_partiDemarit_tweets = viceleader_parti1_tweets + viceleader2_parti1_tweets + viceleader3_parti1_tweets

viceleader_parti2_tweets = api.user_timeline(screen_name='JuhaRehula', count=400, include_rts = True)
viceleader2_parti2_tweets = api.user_timeline(screen_name='AnnikaSaarikko', count=400, include_rts = True)
viceleader3_parti2_tweets = api.user_timeline(screen_name='AnuVehvilainen', count=400, include_rts = True)

viceleaders_partiKeskusta_tweets = viceleader_parti2_tweets + viceleader2_parti2_tweets + viceleader3_parti2_tweets

viceleader_parti3_tweets = api.user_timeline(screen_name='AMVirolainen', count=400, include_rts = True)
viceleader2_parti3_tweets = api.user_timeline(screen_name='sannigrahn', count=400, include_rts = True)
viceleader3_parti3_tweets = api.user_timeline(screen_name='JanneSankelo', count=400, include_rts = True)

viceleaders_partiKokoomus_tweets = viceleader_parti3_tweets + viceleader2_parti3_tweets + viceleader3_parti3_tweets

viceleader_parti5_tweets = api.user_timeline(screen_name='LauraHuhtasaari', count=400, include_rts = True)

viceleader_parti8_tweets = api.user_timeline(screen_name='joonasleppan', count=400, include_rts = True)
```

Since I had to restart the program several times, it seemed necessary to me to delete the content of each table in order to be able to reinsert it afterwards :

```
print("suppression du contenu des tables")
c = conn.cursor()
c.execute("DELETE FROM tweets")
c.execute("DELETE FROM tweets_by_Keskusta")
c.execute("DELETE FROM tweets_by_Kokoomus")
c.execute("DELETE FROM tweets_by_Demarit")
c.execute("DELETE FROM tweets_by_KDpuolue")
c.execute("DELETE FROM tweets_by_Persut")
c.execute("DELETE FROM tweets_by_SiniTulevaisuus")
c.execute("DELETE FROM tweets_by_sfprkp")
c.execute("DELETE FROM tweets_by_vasemmisto")
c.execute("DELETE FROM tweets_by_vihreat")
c.execute("DELETE FROM tweets_by_leader_Keskusta")
c.execute("DELETE FROM tweets_by_leader_Kokoomus")
c.execute("DELETE FROM tweets_by_leader_Demarit")
c.execute("DELETE FROM tweets_by_leader_KDpuolue")
c.execute("DELETE FROM tweets_by_leader_Persut")
c.execute("DELETE FROM tweets_by_leader_SiniTulevaisuus")
c.execute("DELETE FROM tweets_by_leader_sfprkp")
c.execute("DELETE FROM tweets_by_leader_vasemmisto")
c.execute("DELETE FROM tweets_by_leader_vihreat")
c.execute("DELETE FROM tweets_viceleaders_Demarit")
c.execute("DELETE FROM tweets_viceleaders_Keskusta")
c.execute("DELETE FROM tweets_viceleaders_Kokoomus")
c.execute("DELETE FROM tweets_viceleaders_parti5")
c.execute("DELETE FROM tweets_viceleaders_parti8")

conn.commit()
```

As you can see there are many tables present. It is now time to insert all the tweets collected and for the moment stored in dictionaries (which will allow us to browse them for each tweet). And as the screenshot will show you, at each loop that allows

me to insert a tweet into the database, the content of the tweet will be decrypted by "unidecode" as I explained earlier.

```
print("INSERTION DES TWEETS EN RAPPORT AVEC LES ELECTIONS FINLANDAISES \n \n")

for tweet in searched_tweets:
    unidecode(tweet.text)
    c.execute("INSERT OR IGNORE INTO tweets (id_tweet, username , content) VALUES (?, ?, ?);", (tweet.id, tweet.author.screen_name, tweet.text))
    conn.commit()
print("Requete acceptée pour la table tweets")

###################################

print("INSERTION DES TWEETS DES PARTIS \n \n")

for tweet in parti2_tweets:
    unidecode(tweet.text)
    c.execute("INSERT OR IGNORE INTO tweets_by_Keskusta (id_tweet, content) VALUES (?, ?);", (tweet.id, tweet.text))
    conn.commit()
print("Requete acceptée pour le parti Keskusta")
```

The same insertion is therefore repeated for political parties, the presidents of each party, as well as the vice-presidents. I use "INSERT OR IGNORE" to avoid having the tweet id duplicated and thus cause errors since it is the primary key.

You can also see that between the tweets of random users, and the tweets of political parties, the insertion is different as already mentioned, since for the first case, it is necessary to insert the user's name.

At this stage of the program, our database is complete, the insertions went well, moreover the tweets are all decoded / encoded in order to have no errors. All that remains now is to analyze this database.

To begin, we must first classify the content of each party's tweets in new dictionaries. In other words, a dictionary will contain the tweets of the party, the

president and his vice-presidents. We therefore have a total of 10 final dictionaries to process.

```
tweets_parti1 = []
tweets_parti1.extend(parti1_tweets)
tweets_parti1.extend(leader_parti1_tweets)
tweets_parti1.extend(viceleaders_partiDemarit_tweets)

tweets_parti2 = []
tweets_parti2.extend(parti2_tweets)
tweets_parti2.extend(leader_parti2_tweets)
tweets_parti2.extend(viceleaders_partiKeskusta_tweets)

tweets_parti3 = []
tweets_parti3.extend(parti3_tweets)
tweets_parti3.extend(leader_parti3_tweets)
tweets_parti3.extend(viceleaders_partiKokoomus_tweets)

tweets_parti4 = []
tweets_parti4.extend(parti4_tweets)
tweets_parti4.extend(leader_parti4_tweets)

tweets_parti5 = []
tweets_parti5.extend(parti5_tweets)
tweets_parti5.extend(leader_parti5_tweets)
tweets_parti5.extend(viceleader_parti5_tweets)

tweets_parti6 = []
tweets_parti6.extend(parti6_tweets)
tweets_parti6.extend(leader_parti6_tweets)

tweets_parti7 = []
tweets_parti7.extend(parti7_tweets)
tweets_parti7.extend(leader_parti7_tweets)

tweets_parti8 = []
tweets_parti8.extend(parti8_tweets)
tweets_parti8.extend(leader_parti8_tweets)
tweets_parti8.extend(viceleader_parti8_tweets)

tweets_parti9 = []
tweets_parti9.extend(parti9_tweets)
tweets_parti9.extend(leader_parti9_tweets)
```

On the screenshot we can see that only 9 dictionaries are present because the 10th is the dictionary containing the last 200 user tweets, and this one is defined above in the code, its name is "searched_tweets". Also we can see that some dictionaries will be bigger than others since some parties have known vice presidents, while others do not.

Once our dictionaries have been defined, it is time to move on to the analysis of feelings. For this reason I have decided to use the "pandas" or "maptplotlib" modules which will allow me to interpret the results in graphical form in order to obtain a better understanding.

I also define a scale for matplotlib and pandas whose limits will be between -1 and 1, so the closer we get to 1, the more positive the tweet concerned will be

considered by the algorithm as positive. And on the other hand, the closer we get to -1, the more negative the tweet will be.

So, on the abscissa axis we have the polarity, and on the y-axis, the number of tweets. And at the end we use matplotlib to display the graphs.

```
if rep_graph == '0':
    sentiments_objects = [TextBlob(tweet.text) for tweet in searched_tweets]
    sentiments_values = [[tweet.sentiment.polarity, str(tweet)] for tweet in sentiments_objects]
    fig, ax = plt.subplots(figsize=(8,6))
    sentiments_parti0 = pd.DataFrame(sentiments_values, columns=["polarité", "tweet"])
    sentiments_parti0.hist(bins=[-1, -0.75, -0.5, -0.25, 0.0, 0.25, 0.5, 0.75, 1], ax = ax, color="blue")
    plt.title("Sentiments sur les tweets des utilisateurs")
    plt.show()

if rep_graph == '1':
    sentiments_objects = [TextBlob(tweet.text) for tweet in tweets_parti1]
    sentiments_values = [[tweet.sentiment.polarity, str(tweet)] for tweet in sentiments_objects]
    fig, ax = plt.subplots(figsize=(8,6))
    sentiments_parti1 = pd.DataFrame(sentiments_values, columns=["polarité", "tweet"])
    sentiments_parti1.hist(bins=[-1, -0.75, -0.5, -0.25, 0.0, 0.25, 0.5, 0.75, 1], ax = ax, color="blue")
    plt.title("Sentiments sur les tweets du parti Demarit")
    plt.show()
```

Here I present two cases: the analysis of feelings on user tweets, and on party tweets, to show the difference between the two.

Finally, here is the last part of my code which consists in using the LDA on each tweets dictionary to get out the most used words. Of course, the stopword list is used to avoid misleading results.
As you can see, we leave the choice to the user to continue or not, we also define the number of topics to 5.

```
rep = input("Souhaitez vous utiliser LDA ? si oui tapez oui sinon tapez autre chose : \n")

i = 0

total_topics = 5
TEMP_FOLDER = tempfile.gettempdir()
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
```

We also see above that I have defined a special "stoplist" since it includes Finnish stopwords, but also punctuation marks, and also, since we are talking about tweets, the words "rt" and "RT" in order not to distort the data.

```
list1 = ['RT', 'rt']
stoplist = stopwords.words('finnish') + list(punctuation) + list1

print("application du LDA pour les tweets en rapport avec les élections finlandaises")

texts = [[word for word in str(tweet.text).lower().split() if word not in stoplist] for tweet in searched_tweets]

dictionary = corpora.Dictionary(texts)
dictionary.save(os.path.join(TEMP_FOLDER, 'tweets.dict'))

searched_tweets = [dictionary.doc2bow(text) for text in texts]
corpora.MmCorpus.serialize(os.path.join(TEMP_FOLDER, 'tweets.mm'), searched_tweets)

tfidf = models.TfidfModel(searched_tweets)

searched_tweets_tfidf = tfidf[searched_tweets]

lda = models.LdaModel(searched_tweets, id2word=dictionary, num_topics=total_topics)
searched_tweets_lda = lda[searched_tweets_tfidf]

lda.show_topics(total_topics, 5)
```

RESULTS

We now come to the last part of this report, the most important part, the results as well as possible interpretations.

```
recherche des principaux partis finlandais :

Parti trouvé : Sosialidemokraatit

Localisation de Sosialidemokraatit : Finland

Nombre d'abonnés de Sosialidemokraatit : 21108

-------------------------------------------------------------------------------

Parti trouvé : Suomen Keskusta

Localisation de Suomen Keskusta :

Nombre d'abonnés de Suomen Keskusta : 18540

-------------------------------------------------------------------------------

Parti trouvé : Kokoomus

Localisation de Kokoomus : Finland

Nombre d'abonnés de Kokoomus : 27857

-------------------------------------------------------------------------------

Parti trouvé : KD

Localisation de KD : Suomi

Nombre d'abonnés de KD : 6096

-------------------------------------------------------------------------------

Parti trouvé : Perussuomalaiset

Localisation de Perussuomalaiset : Finland

Nombre d'abonnés de Perussuomalaiset : 15089
```

```
Parti trouvé : Sininen tulevaisuus

Localisation de Sininen tulevaisuus : Suomi

Nombre d'abonnés de Sininen tulevaisuus : 2559

_____

Parti trouvé : SFP • RKP

Localisation de SFP • RKP :

Nombre d'abonnés de SFP • RKP : 4992

_____

Parti trouvé : Vasemmistoliitto

Localisation de Vasemmistoliitto : Finland

Nombre d'abonnés de Vasemmistoliitto : 26050

_____

Parti trouvé : Vihreät – De Gröna

Localisation de Vihreät – De Gröna : Finland

Nombre d'abonnés de Vihreät – De Gröna : 37865

_____
```

The first data, which is easily recorded, is the number of subscribers, in order to be able to see which party seems to be the most popular on Twitter, the flagship social network.

So from the data on the two screenshots, we can see that the party with the most followers is @vihreat with 37865 subscribers., which would seem to be the Finnish Green Party since its full name is "The Green League".

This shows first of all that the Finns are more involved in ecology, since on social networks, it seems that the ecologist party is the most popular. And it is clear that this seems to be true because Finns are very environmentally friendly people: very frequent use of bicycles, significant waste sorting... etc.

This information may allow people who have never been to Finland to learn a little more about these people.

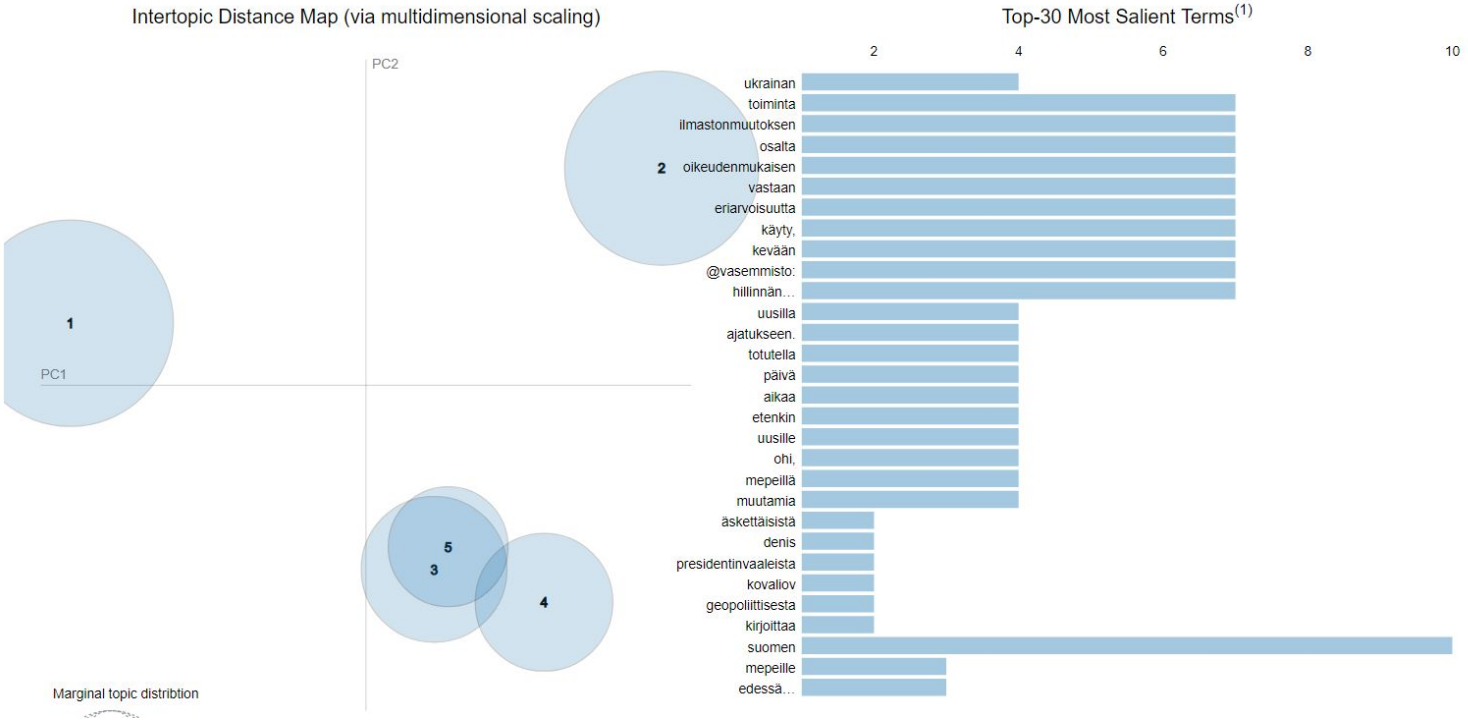
On the other hand, the political party with the fewest followers is @sinitulevaisuusn with 2559 followers, which according to the Internet, is called the "blue reform". It is a right-wing party, considered conservative. It seems important to me to point out that this party was created following the dissolution of the True Finnish' party, which asserted itself as an extreme right-wing party.

```
Recherche des leaders de chaque parti :

Nom du leader du parti Demarit : Antti Rinne

Nombre d'abonnés : 29398

Nom du leader du parti Keskusta : Juha Sipilä

Nombre d'abonnés : 128083

Nom du leader du parti Kokoomus : Petteri Orpo

Nombre d'abonnés : 53287

Nom du leader du parti KDpuolue : Sari Essayah

Nombre d'abonnés : 6095

Nom du leader du parti Persut : Jussi Halla-aho

Nombre d'abonnés : 21653

Nom du leader du parti SiniTulevaisuus : Simon Elo

Nombre d'abonnés : 6623

Nom du leader du parti sfprkp : Anna-Maja Henriksson

Nombre d'abonnés : 7198

Nom du leader du parti vasemmisto : Li Andersson

Nombre d'abonnés : 82259

Nom du leader du parti vihreat : Pekka Haavisto

Nombre d'abonnés : 133131
```

If we take a quick look at the number of subscribers of each party president, we see that the previous interpretations seem to be confirmed since the leader with the largest number of subscribers is the president of the Green League, and that of the "Blue Reform" is one of those with the least.

LDA :

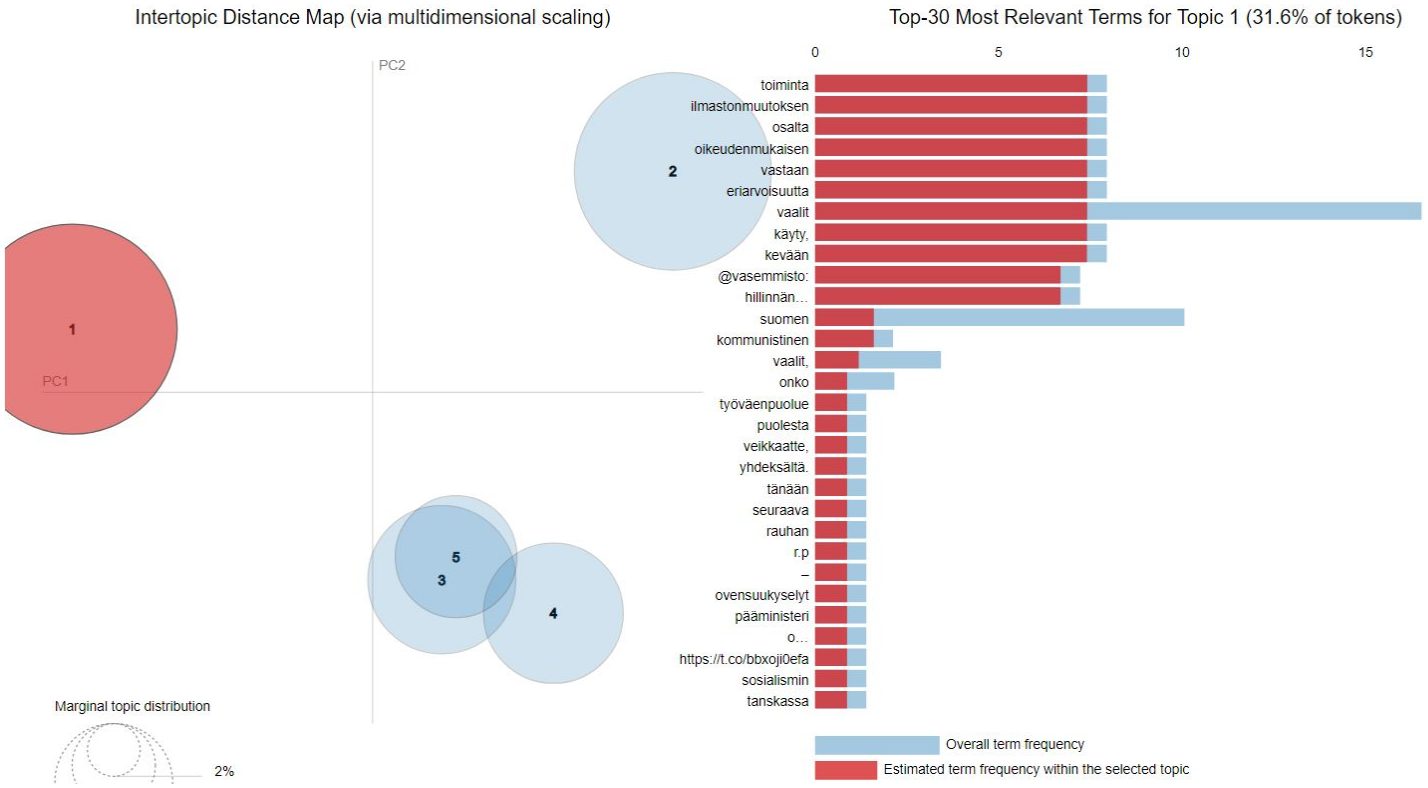
It is now time to move on to more detailed analyses, as it is time to interpret the LDA results on each party, as well as their positive analysis :
Let's start with the LDA on user tweets about the Finnish elections. We can therefore find 5 topics and the 10 most used words.
To make it easier to interpret, and also to make it more visible, we will use pyLDAvis which is a library to import and use to display the LDA results in graphical form via a local web page :



On this interface we can see that on the left are arranged bubbles, which are in fact the number of topics recorded, and the larger the bubble, the more prominent the topic is considered. On the right we can see as a table listing all the most used words and a "gauge" estimating their frequency, while sorting them according to this.

We can therefore see that in general the most recurrent words are therefore "Ukrainian" (because the Ukrainian elections took place at the end of April), but also words like "tolminta" (which means activities) or "ilmastonmuutoksen" (meaning climate change). These early data can therefore suggest that Finnish people are tweeting about elections in other countries, but also about global warming.

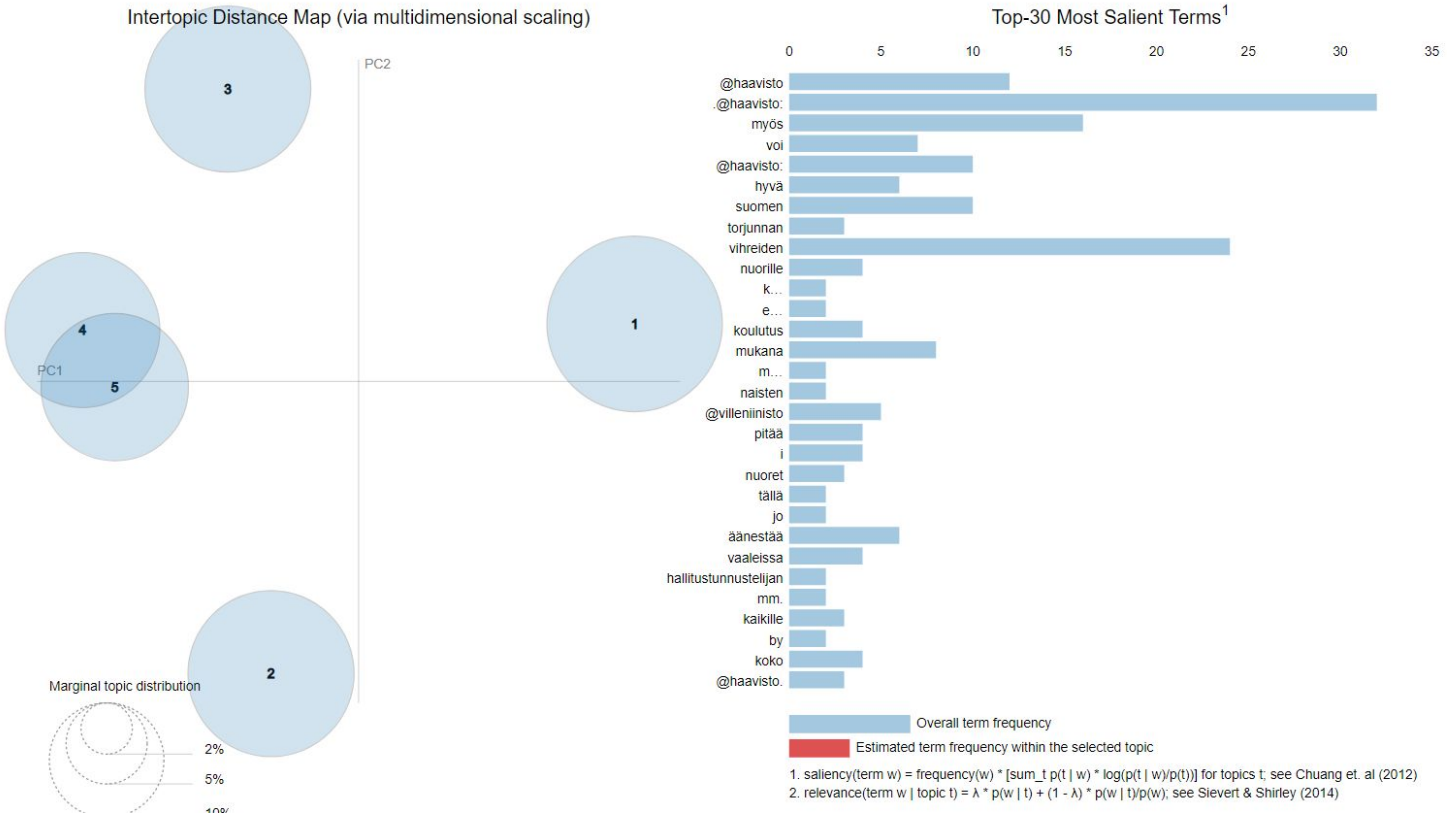
The interest of this interface is that by passing the cursor over one of the topics, we can see the most used words of this topic, as shown in the screenshot below:

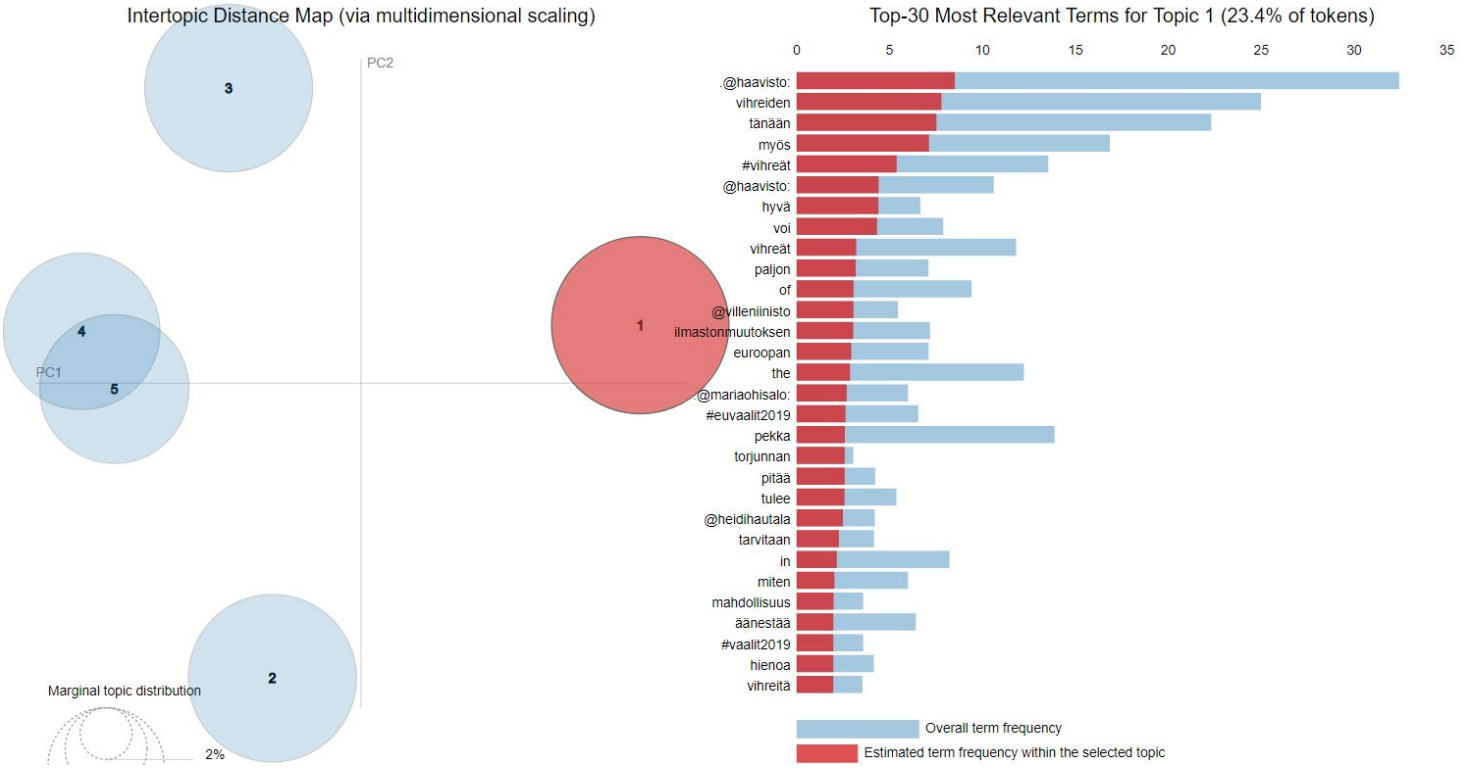


We can therefore have more detailed analysis results on each of the topics and thus better interpret the results. For each of the topics, the graphical user interface takes care of sorting from the most used to the least used term.

As screenshots can show, and this is what is interesting, it is when two or three bubbles cross, it means that terms are in common and that the topics are more or less close. For topics 1 and 2 for example, the bubbles do not touch each other, which means that the terms used for each of these topics are different and that the topics are far apart.

Since I find this important, we will compare with the same interface but this time on the tweets of a political party :

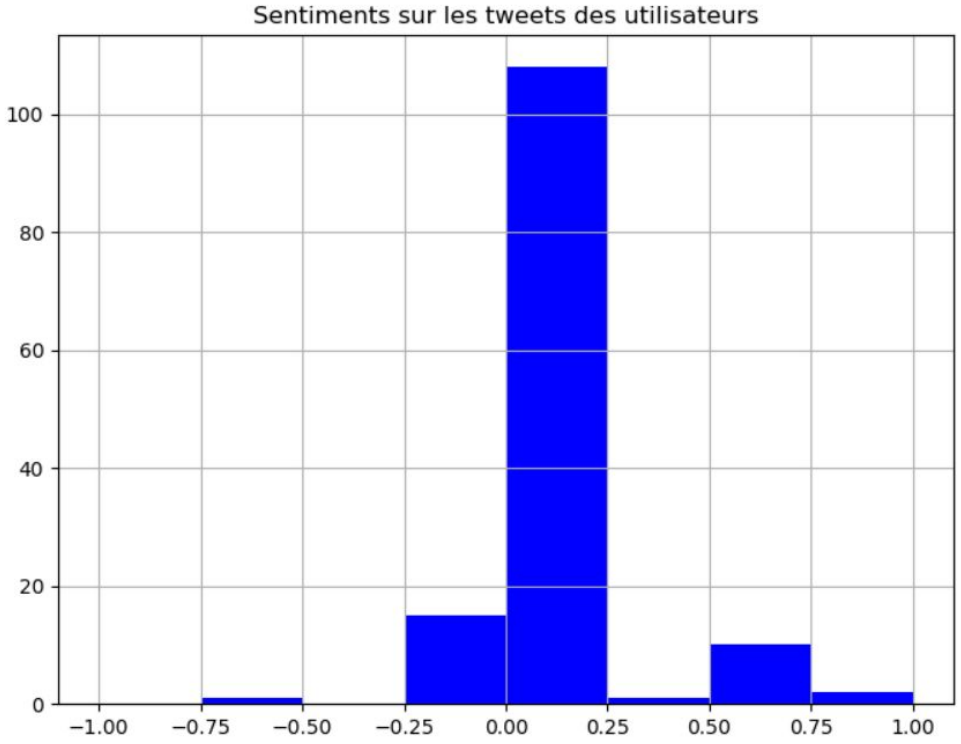



Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
Marginal topic distribution
2%
Top-30 Most Relevant Terms for Topic 1 (23.4% of tokens)
0
5
10
15
20
25
30
35
.@haavisto:
vihreiden
tänään
myös
#vihreät
@haavisto:
hyvä
voi
vihreät
paljon
of
@villeniinisto
ilmastonmuutoksen
euroopan
the
@mariaohisalo:
#euvaalit2019
pekka
torjunnan
pitää
tulee
@heidihautala
tarvitaan
in
miten
mahdollisuus
äänestää
#vaalit2019
hienoa
vihreitä
Overall term frequency
Estimated term frequency within the selected topic

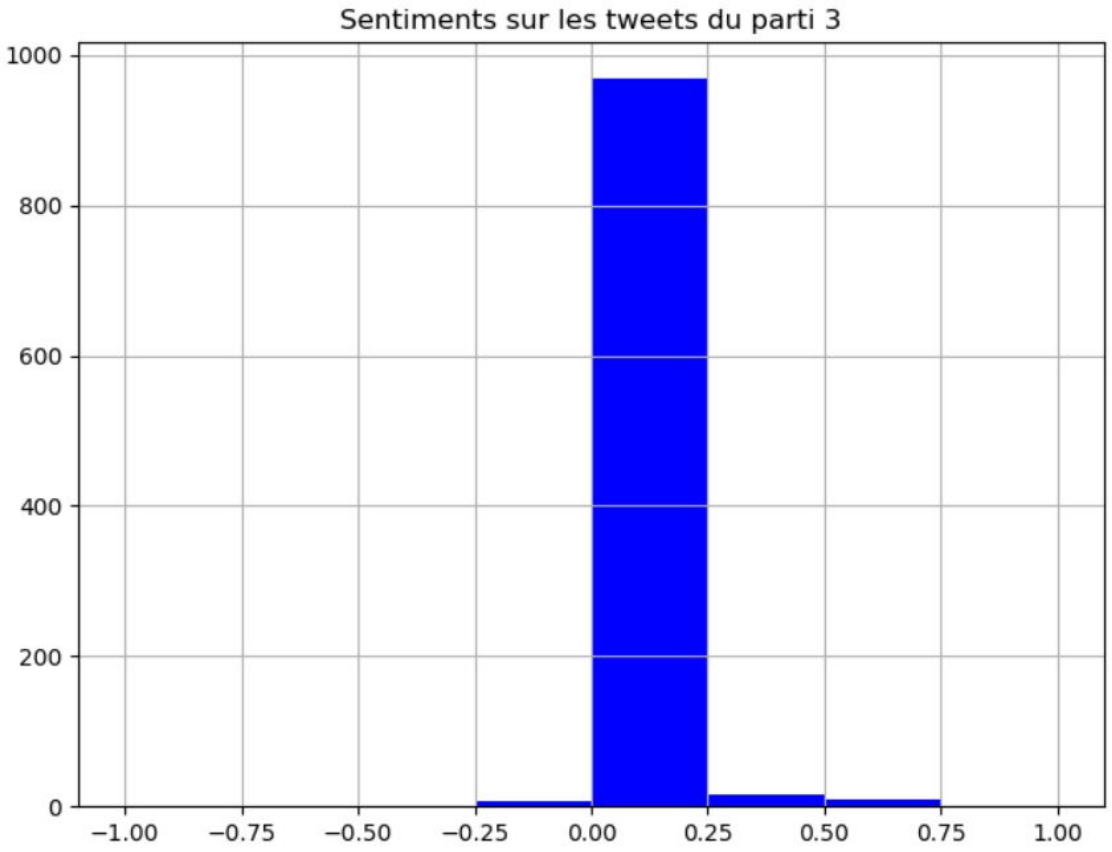
As we can see, tweets are a little less interesting since they are more tweets that promote the party in itself and the terms often come up again, such as the @ of the party, the @ of the party president, or the name of the party. I find that these data are a little less exploitable than the previous ones, even if words related to ecology come up again, which is normal since the party discussed here is the "Green League".

Sentiments analysis :

Let's now move on to the analysis of feelings on tweets dictionaries :



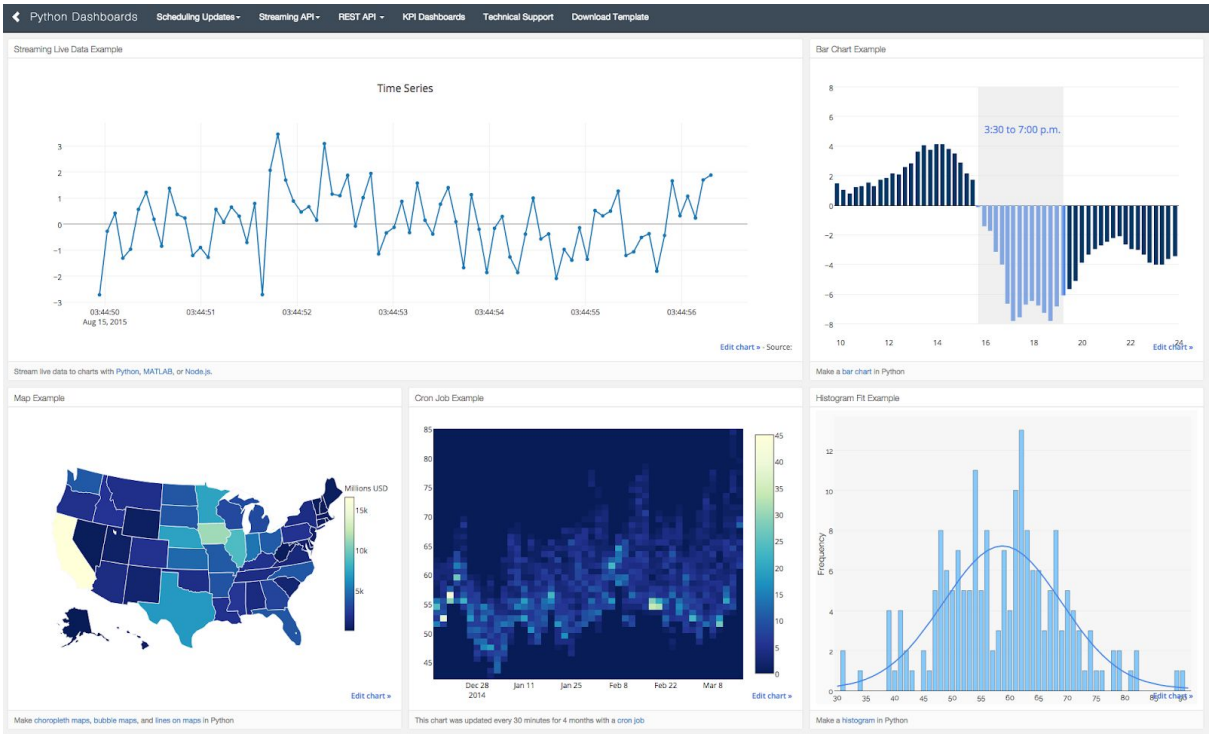
On this first graph, which concerns the users's tweets, there are about 150 tweets and the vast majority of them are between 0 and 0.25, which I would call a positive neutrality, positive words are used but there are enough negative words to compensate, or only a few positive words are used and the rest of the sentences are neutral. The results between -0.25 and 0.25 are considered mostly neutral while between 0.25 and 1 we find that many tweets are positive which would mean that Twitter users are quite comfortable with the elections and approach the subject with a relaxed and happy look. We can therefore start to think that the Finns are happy with the results of the European elections, and this can be understood since the results are truly mixed and not dominated by one, two parties.



On this graph, we analyze the tweets of party 3, which contains many more samples since it includes the tweets of the official party, the party president, and the vice-presidents. We therefore have a database of about 1000 tweets as shown by the y-axis. However, as already explained above, it is more difficult to interpret the results from a political party because their tweets must be neutral in general.

EVOLUTION

Before concluding this internship thesis, we can indeed see the different possibilities for improvement in the future. First of all it would be interesting to make a streaming system to capture the tweets live, and then analyze them instantly. This is possible thanks to the API stream but I think it would be more interesting to use on more discussed topics. In addition, another feature that could be interesting would be to design a real interface for the application as a dashboard in order to make all this graphically pretty to see.
Another possible evolution is that once you have passed the application on an interface such as a web interface, you can integrate a database that can be displayed on the same application in order to save time to see the results.



CONCLUSION

To conclude on this internship abroad, I will first speak from a human point of view:

- First of all, this internship was a huge experience in terms of relationships, because it was exceptional to be able to meet people from all over the world like English, Spanish, German, etc... It allowed me to have a great time and to forge memories that were impossible to forget. I was obviously able to improve my English because 80% of the time I used it. Meeting all these people, living in a country so different from ours, discovering a different culture, all these are the ingredients to be able to remember this experience all my life, and if the opportunity comes up again, I will do it again without any worry.

- Then from a technical point of view, it obviously seems to me to specify again that my level of English has constantly improved during the 3 months of the internship. From a computer point of view, I was able to learn Python, databases, as well as machine learning, which I find very interesting because it is really the part of computing that I find most useful because we are able to analyze a lot of data in a very short time, to also provide probabilities, all kinds of data that can be used by the future to make life easier. Moreover, learning Python was a pleasure to learn, since it was made easier by the fact that I had previously known other languages that were similar to it. The most interesting part was the learning of NLTK, and the different algorithms. I can say that I will continue to use the Python language personally since it is still relevant today, and many projects are possible through it.

REFERENCES

Python : https://www.python.org/

IDLE : https://fr.wikipedia.org/wiki/IDLE_(Python)

NLTK : https://www.nltk.org/book/

Tweepy : https://www.tweepy.org

Gensim : https://radimrehurek.com/gensim/

Stanford : https://www.nltk.org/_modules/nltk/tag/stanford.html

LDA : https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

Pandas : https://pandas.pydata.org/

SQLite3 : https://www.sqlite.org/version3.html

Unidecode : https://pypi.org/project/Unidecode/

pyLDAvis : https://pypi.org/project/pyLDAvis/