Université Abdelmalek Essâadi de Tanger
Faculté des Sciences et Techniques de Tanger
Département Génie Informatique

Projet de fin de module

NoSQL

Filière : Mobiquité et Big Data

Réaliser par :

- AHARMOUCH Mohamed Hamza
- AMAYOU ANAS

Année universitaire : 2020/2021

Summary

I. General introduction to our project.

II. LITERATURE REVIEW.

III. project's Design & Architecture.

IV. Implementation:

1. Scraping.
2. NLP & machine learning models.
3. web app dev : Django, angular, graphql , Elasticsearch & mongodb.

V. Conclusion .

VI. References .

GitHub Link: https://github.com/anashamza01/NoSQL_Project

I. General introduction to our project.

In our modern life where the internet is ubiquitous, everyone relies on various online resources for news. Along with the increase in the use of social media platforms.

news spread rapidly among millions of users within a very short span of time. The spread of fake news has far-reaching consequences like COVID-19 news.

Moreover, spammers use appealing news headlines to generate revenue using advertisements via click-baits.

In this paper, we aim to perform binary classification of various news articles available online about COVID-19 with the help of concepts pertaining to Machine Learning, Natural Language Processing.

We aim to provide the user with the ability to classify the news as fake or real and also check the authenticity of the website publishing the news.

As an increasing amount of our lives is spent interacting online through social media platforms, more and more people tend to hunt out and consume news from social media instead of traditional news organizations. The explanations for this alteration in consumption behaviors are inherent within the nature of those social media platforms: it's often more timely and fewer expensive to consume news.

- **Our Project:**

In this project, we will use some machine learning techniques that will help us to detect easily some COVID-19 news are fake or not, by implementing this project we will practice what we had learned in this semester like the NOSQL data and classification algorithms, and also some research and storages technologies like Elasticsearch and MongoDB.

II. LITERATURE REVIEW.

In this part, we will explore some similar researches and projects are already implemented, just to have an overview of the other methodologies used to create a system to detect fake news with NLP and Machine Learning algorithms.

Mykhailo Granik et. Al. in their paper shows a simple approach for fake news detection using naïve Bayes classifier. This approach was implemented as a software system and tested against a data set of Facebook news posts. They were collected from three large Facebook pages each from the right and from the left, as well as three large mainstream political news pages (Politico, CNN, ABC News). They achieved classification accuracy of approximately 74%. Classification accuracy for fake news is slightly worse. This may be caused by the skewness of the dataset: only 4.9% of it is fake news. Himank Gupta et. Al. [10] gave a framework based on different machine learning approach that deals with various problems including accuracy shortage, time lag (BotMaker) and high processing time to handle thousands of tweets in 1 sec. Firstly, they have collected 400,000 tweets from Hspam14 dataset. Then they further characterize the 150,000 spam tweets and 250,000 non- spam tweets. They also derived some lightweight features along with the Top-30 words that are providing highest information gain.

Cody Buntain et. Al. develops a method for automating fake news detection on Twitter by learning to predict accuracy assessments in two credibility-focused Twitter datasets: CREDBANK, a crowd sourced dataset of accuracy assessments for events in Twitter, and PHEME, a dataset of potential rumors in Twitter and journalistic assessments of their accuracies. They apply this method to Twitter content sourced from BuzzFeed's fake news dataset. A feature analysis identifies features that are most predictive for crowd sourced and journalistic accuracy assessments, results of which are consistent with prior work. They rely on identifying highly retweeted threads of conversation and use the features of these threads to classify stories, limiting this work's applicability only to the set of popular tweets.

Also, in **Kaggle** forum, there are a lot of projects about detecting fake news, but, most of this project deals with ENGLISH data, in our project we will use FRENCH as a language of our news, so it will be a little bit different.

III. project's Design & Architecture.

In this chapter, we will discuss our project's design, and use the technologies that we will use to implement it.

- **Subject of the project:** The implementation of a single page application type web application, for processing data from a NoSql database.

This paper explains the system which is developed in three parts:

1. The first part is Scraping data (F5rench data) from websites to work on it, we used scrapy library to do this step.

2. The second part is static which works on machine learning classifier. We studied and trained the model with 2 different classifiers (Naives Bayes & SVM) and chose the best classifier for final execution.

3. The third part is dynamic which takes the keyword/text from user and searches for the truth probability of the news and sentiment analysis.

In this project, we have used Python and its Sci-kit libraries. Python has a huge set of libraries and extensions, which can be easily used in Machine Learning. Sci-Kit Learn library is the best source for machine learning algorithms where nearly all types of machine learning algorithms are readily available for Python, thus easy and quick evaluation of ML algorithms is possible.

We have used Django for the web based deployment of the model, provides client side implementation using Angular .

- **Project's Design & Architecture :**

The architecture of fake news detection system is quite simple and is done keeping in mind the basic machine learning process flow. The system design is shown below and self-explanatory. The main processes in the design are :

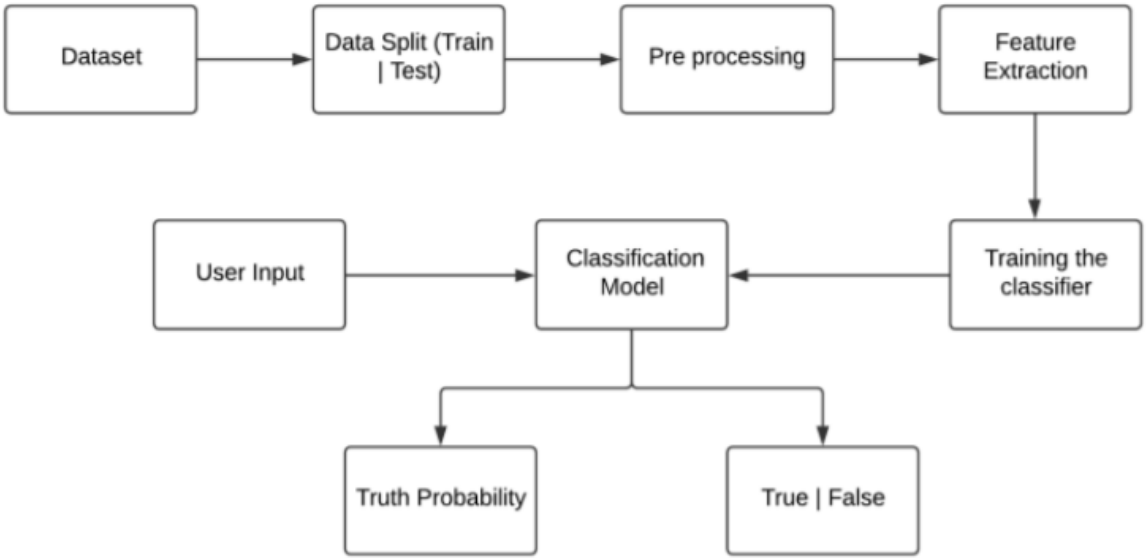


Figure 1 : Project's Design

For the sentiment analysis, we can use either Machine Learning techniques (NB/SVM) to predict the sentiment, or using ANN : Artificial Neural Network .In our project we used the ANN to predict the sentiment, it's more complicated than ML , but it return better result . The figure below show our sentiment analysis design system:

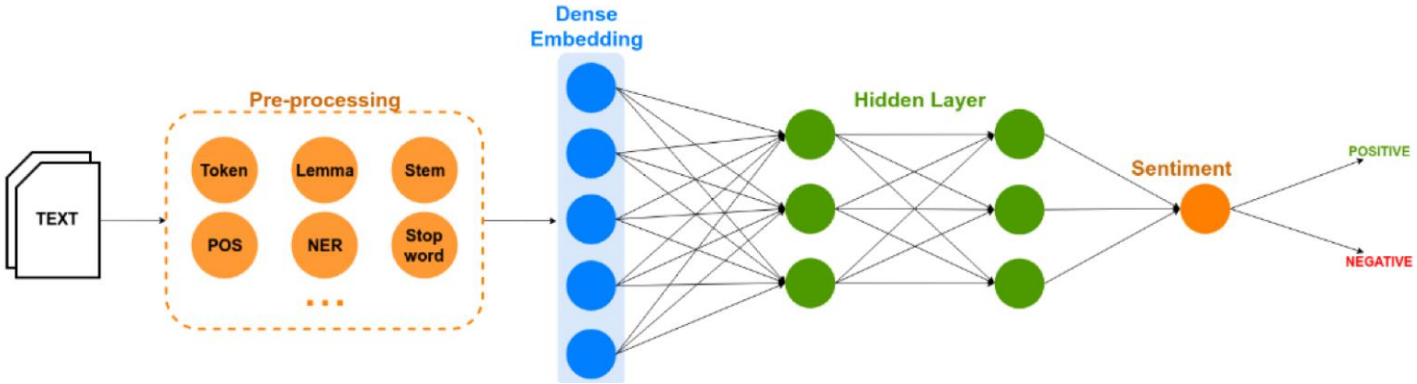


Figure 2 : Sentiment Analysis with ANN

IV. Implementation

➢ Scraping :

We can get online news from different sources like social media websites, search engine, homepage of news agency websites or the factchecking websites. On the Internet, there are a few publicly available datasets for Fake news.

These datasets have been widely used in different research papers for determining the veracity of news. Online news can be collected from different sources, such as news agency homepages, search engines, and social media websites. However, manually determining the veracity of news is a challenging task, usually requiring annotators with domain expertise who performs careful analysis of claims and additional evidence, context, and reports from authoritative sources.

Generally, news data with annotations can be gathered in the following ways: Expert journalists, Fact-checking websites, Industry detectors, and Crowd sourced workers.

Data gathered must be preprocessed- that is, cleaned, transformed and integrated before it can undergo training process.

So, we use Scrapy library to perform web scraping from different web sites (French websites)

as the figure below explains :

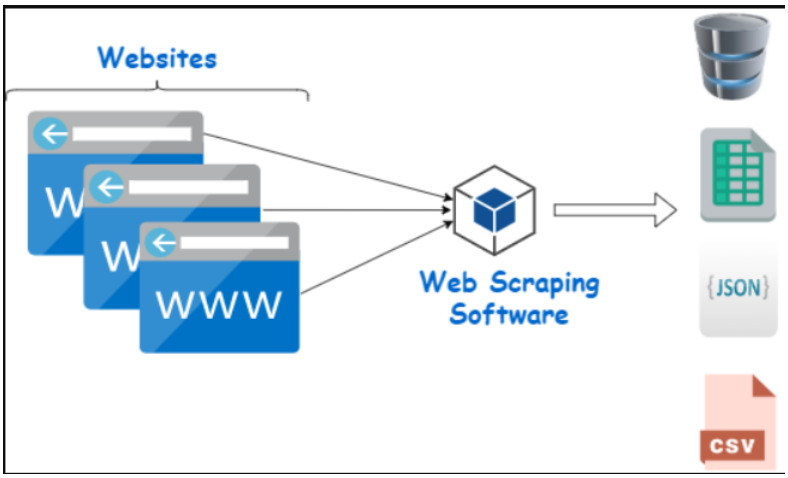


Figure 3: web scraping

➢ Methods used

An open source and collaborative framework for extracting the data you need from websites.In a fast, simple, yet extensible way.
Maintained by Zyte (formerly Scrapinghub) and many other contributors
Scrapy project architecture is built around "spiders", which are self-contained crawlers that are given a set of instructions. Following the spirit of other don't repeat yourself frameworks, such as Django,[3] it makes it easier to build and scale large crawling projects by allowing developers to reuse their code. Scrapy also provides a web-crawling shell, which can be used by developers to test their assumptions on a site's behavior.

➢ Scrapping Code:

In this particular part we have chosen to work with news in French, so we based our self on website that contain news in French and especially news about Corona Virus.

And about the fake news that we scraped for training our model in machine Learning we faced a little difficulty because there aren't enough resources about fake news but we managed to get some fake news for our model.

Here are websites that we scraped data from:

- https://www.lemonde.fr/coronavirus-2019-ncov
- https://www.europe1.fr/dossiers/coronavirus/
- https://www.legorafi.fr/?s=covid

For functions that have been used for this job we worked with spiders which are self-contained crawlers which helps us a lot during this phase.

Here are our scraping spiders:

```python
import random

import scrapy
from Scraping_Project.items import NewsItem
from scrapy.loader import ItemLoader


class NewsSpider(scrapy.Spider):
    name = 'news4'

    start_urls = [
        'https://www.lemonde.fr/coronavirus-2019-ncov/' #ICI on met l'URL du website
    ]

    #the parse methode
    def parse(self, response):

        for new in response.xpath("//div[@class='thread']"):
            choices = list(range(700,1500))
            random.shuffle(choices)
            num = choices.pop()

            l= ItemLoader(item=NewsItem(), selector=new)
            l.add_value('id', num)
            l.add_xpath('title',".//section/a/h3//text()")
            l.add_xpath('details',".//section/a/p//text()")

            yield l.load_item()
```

This our function for scraping real news where we worked with items and ItemLoader while scrapping we will explain why in the next part here you will see that we give a random number to an id, because this data it will be stocked in MongoDB exactly inside a collection that it was created by Django and as we know that Django create with default an id that he distributes to objects so to stocked this scraped data inside MongoDB we had to give it a unique id.

```python
class NewsSpider(scrapy.Spider):
    name = 'fakenews1'

    start_urls = [
        'https://www.legorafi.fr/?s=covid#' #ICI on met l'URL du website
    ]

    #the parse methode
    def parse(self, response):

        for new in response.xpath("//ul/li[@class='mvp-blog-story-wrap left relative infinite-post']"):
            choices = list(range(700,1500))
            random.shuffle(choices)
            num = choices.pop()

            l= ItemLoader(item=NewsItem(), selector=new)
            l.add_value('id', num)
            l.add_xpath('title',".//a/div/div[@class='mvp-blog-story-in']/div/h2//text()")
            l.add_xpath('details',".//a/div/div[@class='mvp-blog-story-in']/div/p//text()")

            yield l.load_item()


        #on crée la variable pour prendre aussi les autres pages
        next_page = response.xpath("//div[@class='mvp-inf-more-wrap left relative']/div/div/a/@href").getall()
        for page in next_page:
            if page is not None:
                next_page_link = response.urljoin(page)
                yield scrapy.Request(url=next_page_link, callback=self.parse)
```

We used the same method to scrap real news from other websites, in this example we have scraped fake news the same way as precedent example, the only different is that in this web site we scraped data from next pages also using a variable where we stock the next web URL so we will parse it with same method because this URL represent the same website but different page with deferent contents.

➢ ItemLoader and Pipeline

So, as we know we needed to stock our data inside MongoDB for this scrapy offer a pipeline that helps with the import of Pymongo library to stock the data where we want for this the first step was to active the pipeline from settings:

```
ITEM_PIPELINES = {
    'Scraping_Project.pipelines.ScrapingProjectPipeline': 300,
}
```

Then the next step was to configure the pipeline, by defining in with database data will be stocked and which collection etc.

```
import pymongo
import random

class ScrapingProjectPipeline(object):

    def __init__(self):
        self.conn = pymongo.MongoClient(
            'localhost',
            27017
        )
        db = self.conn['News_DB']
        self.collection = db['api_fakenews']


    def process_item(self, item, spider):
        if item.get('details') is not None:
            self.collection.insert(item)

        return item
```

As we know the data being scrapped is always dirty so we have to clean it before storing it inside the database for this job we use itemloader from scrapy where we define our cleaning function and then we call it inside the specific itemloader that we want

```
import scrapy
from scrapy.loader.processors import MapCompose, TakeFirst
from w3lib.html import remove_tags
import random



def clean_data(text):
    for ch in ['\\','`','«', '_','{','}','[',']','(',')','>','#','+','-','.','!','$','\'','»',',',':','"','”','xa0']:
        if ch in text:
            text = text.replace(ch,"")
    return text
```

```python
class NewsItem(scrapy.Item):
    # define the fields for your item here like:
    title = scrapy.Field(
        input_processor= MapCompose(clean_data,remove_whitespace),
        output_processor= TakeFirst()
    )
    _id = scrapy.Field()
    id = scrapy.Field(
        input_processor=MapCompose(clean_dataNumbers),
        output_processor=TakeFirst()
    )
    details = scrapy.Field(
        input_processor=MapCompose(clean_data, remove_whitespace),
        output_processor=TakeFirst()
    )
```

Here as we see the itemloader take the scraped data as input applies cleaning function and then returns cleaned data to pipeline who will stock it inside the data base.

Here is an example of the cleaned data that have been stocked inside MongoDB from Scrapy:

```
db.getCollection('api_news').find({})
```

api_news 0.007 sec. 0 50

Key	Value	Type
(1) ObjectId("60c92051f890f56736a4b121")	{ 4 fields }	Object
_id	ObjectId("60c92051f890f56736a4b121")	ObjectId
id	1386	Int32
title	Les dernières restrictions levées dans l'Etat de New York où plus de 70 % des a...	String
details	New York est le premier grand Etat américain à annoncer avoir franchi ce seuil...	String
(2) ObjectId("60c92051f890f56736a4b122")	{ 4 fields }	Object
_id	ObjectId("60c92051f890f56736a4b122")	ObjectId
id	882	Int32
title	Covid19 la courbe des vaccinations s'aplatit déjà chez les plus âgés	String
details	EN UN GRAPHIQUE – La couverture vaccinale des plus de 65 ans tend vers un ...	String
(3) ObjectId("60c92051f890f56736a4b123")	{ 4 fields }	Object
_id	ObjectId("60c92051f890f56736a4b123")	ObjectId
id	1436	Int32
title	Vaccination Le consentement des adolescents doit être véritablement éclairé	String
details	La professeure Alexandra Benachi membre du Comité consultatif national d'ét...	String
(4) ObjectId("60c92051f890f56736a4b124")	{ 4 fields }	Object
_id	ObjectId("60c92051f890f56736a4b124")	ObjectId
id	703	Int32
title	Le gouvernement rend possible un délai de trois semaines entre deux injectio...	String
details	Le gouvernement a pris cette décision afin de ne pas freiner l'accès à la premi...	String
(5) ObjectId("60c92051f890f56736a4b125")	{ 4 fields }	Object
_id	ObjectId("60c92051f890f56736a4b125")	ObjectId
id	917	Int32
title	Covid19 le grand bond en arrière de l'économie indienne	String
details	En Inde la pandémie aurait causé près d'un million de morts selon les scientifi...	String
(6) ObjectId("60c92051f890f56736a4b126")	{ 4 fields }	Object
(7) ObjectId("60c92051f890f56736a4b127")	{ 4 fields }	Object
(8) ObjectId("60c92051f890f56736a4b128")	{ 4 fields }	Object
(9) ObjectId("60c92051f890f56736a4b129")	{ 4 fields }	Object

We had also the possibility to stock the data inside json file so here is an example of the file:

```
{"title": "Covid-19 : 108.069 morts, la baisse se poursuit dans les services de r\u00e9animation", "details": "Le nombre de malades du Covid
{"title": "Covid-19 : un essai lanc\u00e9 en France sur l'interchangeabilit\u00e9 des vaccins \u00e0 ARN messager", "details": "Une dose de
{"title": "Covid : malgr\u00e9 son retard initial, la France fait partie des bons \u00e9l\u00e8ves de la vaccination", "details": "Apr\u00e8
{"title": "Covid-19 : vaccinodrome cherche vaccinateurs, le job d\u2019\u00e9t\u00e9 qui attire", "details": "Apr\u00e8s une campagne de rec
{"title": "Covid-19 : la baisse des hospitalisations se poursuit", "details": "Le nombre de malades du\u00a0Covid-19\u00a0\u00e0 l'h\u00f4pi
{"title": "Statut honorifique pour les soignants : un \"message symbolique\" qui ne suffit pas", "details": "Apr\u00e8s l\u2019annonce d\u20
{"title": "Macron soulag\u00e9 mais prudent sur les r\u00e9ouvertures : \"C\u2019est pas la bamboche du jour au lendemain\"", "details": "En
{"title": "S\u00e9rie de cas contact dans les m\u00e9dias apr\u00e8s une \u00e9mission de Laurent Ruquier", "details": "France 2 diffusait s
{"title": "L'Education nationale contrainte de revoir les notices des autotests pour... une erreur de traduction", "details": "En raison d'u
{"title": "Vaccins : Macron appelle \"les Anglo-saxons\" \u00e0 arr\u00eater de \"bloquer\" les exportations", "details": "Le pr\u00e9sident
{"title": "D\u00e9confinement : le couvre-feu sera lev\u00e9 progressivement ", "details": "Le pr\u00e9sident de la R\u00e9publique a d\u00e
{"title": "Covid : des \u00e9pid\u00e9miologistes poussent pour l'a\u00e9ration afin de r\u00e9duire les contaminations", "details": "Alors
{"title": "Vaccination : Alain Fischer \u00e9carte un \u00e9largissement et assure qu'il \"n'y a pas de g\u00e2chis\" ", "details": "Sur Eur
{"title": "Vaccination : des QR codes certifi\u00e9s d\u00e8s la premi\u00e8re dose, une \u00e9tape vers le pass sanitaire", "details": "D\u
{"title": "Etat d'urgence et pass sanitaire: l'Assembl\u00e9e vote le texte, apr\u00e8s le couac dans la majorit\u00e9", "details": "L'Assem
{"title": "Covid-19 : un tiers des communes envisage d'augmenter la taxe fonci\u00e8re en 2021", "details": "Selon une engu\u00eate de l'Ass
{"title": "D\u00e9confinement : des aides sur-mesure pour les restaurateurs afin de faciliter la reprise", "details": "Avec le d\u00e9confin
{"title": "Covid : comment la Commission europ\u00e9enne veut permettre le retour des touristes \u00e9trangers", "details": "L'Union europ\u
{"title": "Des consultations gratuites chez le psy bient\u00f4t disponibles pour les jeunes de 3 \u00e0 17 ans", "details": "\u00c0 partir d
{"title": "Vaccination des plus de 18 ans fragiles sans certificat : V\u00e9ran d\u00e9fend \"le choix du pragmatisme\"", "details": "Depuis
{"title": "La fin du masque \u00e0 l'ext\u00e9rieur ? Olivier V\u00e9ran \"esp\u00e8re sinc\u00e8rement que ce sera cet \u00e9t\u00e9\"", "d
{"title": "Embauches de contractuels dans la justice : \"On ne va pas dans le bon sens\"", "details": "Divorces, jugements aux prud'hommes,
{"title": "Covid-19 : 105.387 morts en France, la pression hospitali\u00e8re reste forte", "details": "Les lyc\u00e9ens et coll\u00e9giens d
{"title": "\"On ne m'a pas demand\u00e9 d'ordonnance\" : tous les vaccin\u00e9s sont-ils vraiment \u00e9ligibles ?", "details": "Europe 1 s
{"title": "\"\u00c7a nous change la vie !\" : les premiers vacanciers arrivent dans les campings d\u00e9confin\u00e9s", "details": "Apr\u00e
{"title": "R\u00e9ouverture des th\u00e9\u00e2tres : le secteur \"plut\u00f4t rassur\u00e9, mais un peu inquiet\"", "details": "Sur Europe 1
{"title": "DIRECT - Le porte-parole du gouvernement Gabriel Attal est l'invit\u00e9 du Grand Rendez-vous", "details": "\"Le Grand Rendez-vou
{"title": "Coronavirus : 104.706 morts au total, baisse des hospitalisations et des r\u00e9animations", "details": "Selon le dernier bilan d
{"title": "Amiens : quatre hommes bient\u00f4t jug\u00e9s pour le vol de 50.000 masques", "details": "Le 24 juin prochain, quatre hommes d'u
```


- **NLP, machine learning models & sentiment analysis:**

This is the must lovely part of this project!, implementing predictions using machine learnings algorithms.

- Here are the all the steps used :

A) Pre-processing : Data Social media data is highly unstructured – majority of them are informal communication with typos.Quest for increased performance and reliability has made it imperative to develop techniques for utilization of resources to make informed decisions . To achieve better insights, it is necessary to clean the data before it can be used for predictive modelling. For this purpose, basic pre-processing was done on the News training data. This step was comprised of Data Cleaning: While reading data, we get data in the structured or unstructured format

B) Tokenization : Tokenizing separates text into units such as sentences or words. It gives structure to previously unstructured text.

c) Remove stopwords : Stopwords are common words that will likely appear in any text. They don't tell us much about our data so we remove them.

D) Vectorizing Data: Vectorizing is the process of encoding text as integers i.e. numeric form to create feature vectors so that machine learning algorithms can understand our data.

1. **Bag-Of-Words Bag of Words (BoW)** : or CountVectorizer describes the presence of words within the text data. It gives a result of 1 if present in the sentence and 0 if not present. It, therefore, creates a bag of words with a document-matrix count in each text document.
2. **TF-IDF** : It computes "relative frequency" that a word appears in a document compared to its frequency across all documents TF-IDF weight represents the relative importance of a term in the document and entire corpus .TF stands for Term Frequency: It calculates how frequently a term appears in a document. Since, every document size varies, a term may appear more in a long sized document that a short one. Thus, the length of the document often divides Term frequency. Note: Used for search engine scoring, text summarization, document clustering.

$$TF(t,d) = \frac{Number\ of\ times\ t\ occurs\ in\ document\ 'd'}{Total\ word\ count\ of\ document\ 'd'}$$

IDF stands for Inverse Document Frequency: A word is not of much use if it is present in all the documents.

$$IDF(t,d) = \frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it})$$

TF-IDF is applied on the body text, so the relative count of each word in the sentences is stored in the document matrix.

$$TFIDF(t,d) = TF(t,d) * IDF(t)$$

- Data Cleaning and NLP code :
    1. Importing libraries :

```
In [68]: import pandas as pd
         import re
         import numpy as np
         import os
         import pandas as pd
         import numpy as np
         from nltk.tokenize import word_tokenize
         from nltk import pos_tag
         from nltk.corpus import stopwords
         from nltk.stem import WordNetLemmatizer
         from sklearn.preprocessing import LabelEncoder
         from collections import defaultdict
         from nltk.corpus import wordnet as wn
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn import model_selection, naive_bayes, svm
         from sklearn.metrics import accuracy_score
```

2. Basic Data cleaning :

In [70]:
```
#title_0=df['title'][0]
df['title']=df['title'].astype('str')
df['details']=df['details'].astype('str') #date
#df['date']=df['date'].astype('str')

def clean_data(text):
    for ch in ['','\ufeff','\u200b','\"','.',',','%','\\','`','«', '_','{','}','[
        if ch in text:
            text = text.replace(ch,"")
    return text



def clean_scraped_data():
    for j in range(0,2867):
        df['title'][j]=clean_data(df['title'][j])
        df['details'][j]=clean_data(df['details'][j])
        #df['date'][j]=clean_data(df['date'][j])


    return df


df=clean_scraped_data()
df
```

In [78]:
```
# Step - a : Remove blank rows
data['text'].dropna(inplace=True)
```

In [79]:
```
# Step - b : Change all the text to lower case.
data['text'] = [entry.lower() for entry in data['text']]
```

3. Word tokenization and removing stop words:

In [80]:
```
# Step - c : Tokenization : In this each entry in the corpus will be broken into set of words
data['text']= [word_tokenize(entry) for entry in data['text']]
```

In [82]:
```
from spacy.lang.fr.stop_words import STOP_WORDS as fr_stop


final_stopwords_list = list(fr_stop)
```

In [83]:
```
import spacy
from spacy.lang.fr.examples import sentences
import fr_core_news_md
from spacy.lang.en import English
##-----------
nlp = spacy.load('fr_core_news_md')
##-----------
for index,entry in enumerate(data['text']):
    # Declaring Empty List to store the words that follow the rules for this step
    Final_words = []
    # Initializing WordNetLemmatizer()
    word_Lemmatized = WordNetLemmatizer()
    # pos_tag function below will provide the 'tag' i.e if the word is Noun(N) or Verb(V)
    for word, tag in pos_tag(entry):
        # Below condition is to check for Stop words and consider only alphabets
        #
        if word not in final_stopwords_list and word.isalpha():
            #word_Final = word_Lemmatized.lemmatize(word)
            word_Final = word
            #steaming
            doc = nlp(str(word_Final))
            for token in doc:
                word_Final =  token.lemma_

            #word_Final=stemming(word_Final)
            Final_words.append(word_Final)
    # The final processed set of words for each iteration will be stored in 'text_final'
    data.loc[index,'text_final'] = str(Final_words)
```

4. Data after all this previous steps :

In [96]:
```
data
```

Out[96]:

	text	label	text_final
0	[covid, morts, la, baisse, se, poursuit, dans,...	1	['covid', 'mort', 'baisse', 'poursuivre', 'ser...
1	[covid, un, essai, lancé, en, france, sur, lin...	1	['covid', 'essai', 'lancer', 'france', 'linter...
2	[covid, malgré, son, retard, initial, la, fran...	1	['covid', 'malgré', 'retard', 'initial', 'fran...
3	[covid, vaccinodrome, cherche, vaccinateurs, l...	0	['covid', 'vaccinodrome', 'cherche', 'vaccinat...
4	[covid, la, baisse, des, hospitalisations, se,...	1	['covid', 'baisse', 'hospitalisation', 'poursu...
...	...	...	...
2863	[vaccin, en, france, ``, il, est, très, diffic...	0	['vaccin', 'france', 'très', 'difficile', 'sav...
2864	[succès, inattendu, pour, les, stations, de, s...	0	['succès', 'inattendu', 'station', 'ski', 'mal...
2865	[covid, comment, les, français, concernés, par...	1	['covid', 'français', 'concerner', 'couvrefeu'...
2866	[covid, vaccination, à, grande, échelle, à, pé...	0	['covid', 'vaccination', 'grand', 'échell', 'p...
2867	[covid-19, :, le, vaccin, de, moderna, va, êtr...	0	['vaccin', 'moderna', 'autoriser', 'heure', 'j...

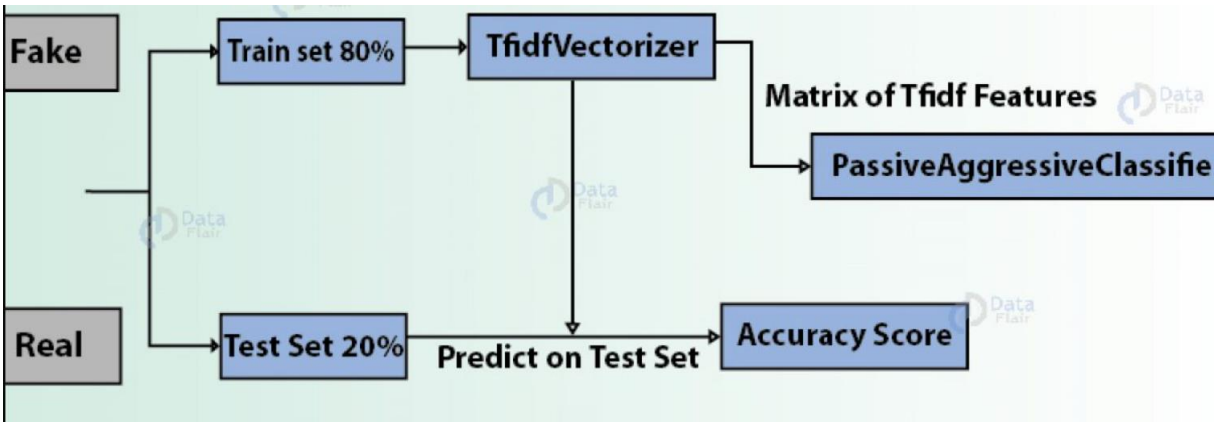
2868 rows × 3 columns

→Example of one phrase after data cleaning :

In [161]:
```
data['text_final'][0]
```

Out[161]:
```
"['covid', 'mort', 'baisse', 'poursuivre', 'service', 'réanimation', 'nombre', 'malade', 'covid', 'service', 'réanimation', 'po
ursuivre', 'baisse', 'mardi', 'total', 'hospitalisation', 'mois', 'chiffre', 'santé', 'public', 'france', 'service', 'soin', 'c
ritique', 'compter', 'patient', 'contre', 'veille']"
```

→The figure bellow explain how we will use this NLP techniques in machine learning models :



5. Prepare Train and Test Datasets :

Prepare Train and Test Data sets

```
In [86]: Train_X, Test_X, Train_Y, Test_Y = model_selection.train_test_split(data['text_final'],data['label'],test_size=0.2,random_state=4

         Train_X.shape, Test_X.shape, Train_Y.shape, Test_Y.shape

Out[86]: ((2294,), (574,), (2294,), (574,))

In [87]: Encoder = LabelEncoder()
         Train_Y = Encoder.fit_transform(Train_Y)
         Test_Y = Encoder.fit_transform(Test_Y)

In [88]: Tfidf_vect = TfidfVectorizer(max_features=5000)
         Tfidf_vect.fit(data['text_final'])
         Train_X_Tfidf = Tfidf_vect.transform(Train_X)
         Test_X_Tfidf = Tfidf_vect.transform(Test_X)
```

- Example of TF-IDF Result:

```
In [90]: print(Train_X_Tfidf)

  (0, 4369)     0.2684725683709742
  (0, 4271)     0.4467818163118111
  (0, 4270)     0.2752994322700048
  (0, 3359)     0.2532902983337102
  (0, 2981)     0.12926935646812251
  (0, 2961)     0.19220338802648398
  (0, 2894)     0.14947325010551163
  (0, 2856)     0.15855065045583674
  (0, 2790)     0.15822074448770806
  (0, 2292)     0.12261977297878596
  (0, 2286)     0.2627005371881062
  (0, 2248)     0.14195931949558302
  (0, 1952)     0.29442683042802387
  (0, 1875)     0.2532902983337102
  (0, 1862)     0.08884703838697798
  (0, 1693)     0.19302636808137755
  (0, 1164)     0.1283706552849778
  (0, 941)      0.25615163689085435
  (0, 925)      0.11617938945809977
```

- **Training the Machine Learning models :**

a) **Naive Bayes classifier :**

Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong independence assumptions between the features. They are among the simplest Bayesian network models, but coupled with kernel density estimation, they can achieve higher accuracy levels.

```
In [91]: # fit the training dataset on the NB classifier
         Naive = naive_bayes.MultinomialNB()
         Naive.fit(Train_X_Tfidf,Train_Y)
         # predict the labels on validation dataset
         predictions_NB = Naive.predict(Test_X_Tfidf)
         # Use accuracy_score function to get the accuracy

         print("Naive Bayes Accuracy Score -> ",accuracy_score(predictions_NB, Test_Y)*1)
         predictions_NB
```

→ The result it's an array of 0 and 1 : FAKE / REAL :

```
Out[91]: array([1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
                1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
                0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
                1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
                0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1,
                0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
                1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
                0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
                1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
                1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
                1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
                1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1,
                0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
                1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,

                Training time: 0.213s
                Prediction time (train): 0.318s
                Prediction time (test): 0.307s

                Train set score: 0.9110723626852659
                Test set score: 0.9110723626852659
```

b) Support vector machine classifier :

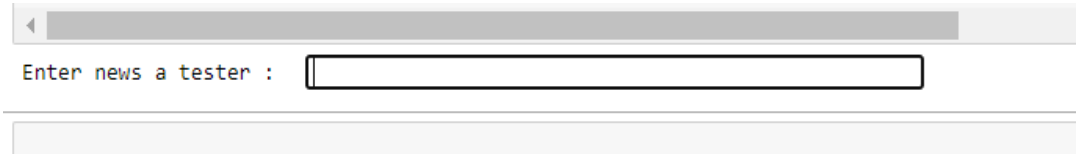
Support vector machines (SVMs) are particular linear classifiers which are based on the margin maximization principle. They perform structural risk minimization, which improves the complexity of the classifier with the aim of achieving excellent generalization performance.

SVM Algorithm Classifier

```
In [128]: # fit the training dataset on the classifier
          SVM = svm.SVC(C=2.0, kernel='linear', degree=3, gamma='auto')
          SVM.fit(Train_X_Tfidf,Train_Y)
          # predict the labels on validation dataset
          predictions_SVM = SVM.predict(Test_X_Tfidf)
          # Use accuracy_score function to get the accuracy
          print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*1)
```

c) Testing this two algorithms in Jupiter notebook :

- **User input :**



➢ **Prediction Result :**

```
Enter news a tester :  Covid  231.800 doses administrées, 109 nouveaux cas ce lundi 14 juin 2021 COMPENSATION : LA CHARGE EXPLO
SE À FIN MAI 2021 TAN-TAN: 35 MIGRANTS SUBSAHARIENS SECOURUS  Plus de 2.300 personnes ont reçu la première dose. Le cumul attei
nt 9,36 millions de personnes. En parallèle, 229.565 personnes ont reçu la deuxième dose, soit 7,32 millions au total. 18 nouve
aux cas sévères ont été enregistrés ce lundi 14 juin.

****  Naive Bayes  :    Real ||  ********  SVM : Real
```

d) Sentiment Analysis :

Artificial Neural Networks (ANNs) are popular models that have shown great promise in many NLP tasks. ANN's make use of sequential information such as text. ... If you want your neural network to learn the meaning (or sentiment in our case) the network must know which words came in which order.

➢ **Sentiment Analysis Code :**

Deep learning

In [27]:
```
base_model = models.Sequential()
base_model.add(layers.Dense(64, activation='relu', input_shape=(NB_WORDS
base_model.add(layers.Dense(64, activation='relu'))
base_model.add(layers.Dense(3, activation='softmax'))
base_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                640064
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 644,419
Trainable params: 644,419
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 32)                320032
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 99
=================================================================
Total params: 320,131
Trainable params: 320,131
Non-trainable params: 0
_________________________________________________________________
```

In [28]:
```
def deep_model(model):
    model.compile(optimizer='rmsprop'
                  , loss='categorical_crossentropy'
                  , metrics=['accuracy'])

    history = model.fit(X_train_rest
                        , y_train_rest
                        , epochs=NB_START_EPOCHS
                        , batch_size=BATCH_SIZE
                        , validation_data=(X_valid, y_valid)
                        , verbose=0)

    return history
```

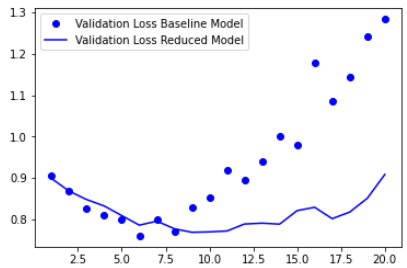
In [29]:
```
base_history = deep_model(base_model)
```

In [30]:
```
%matplotlib inline
from matplotlib import pyplot as plt

def eval_metric(history, metric_name):
    metric = history.history[metric_name]
    val_metric = history.history['val_' + metric_name]

    e = range(1, NB_START_EPOCHS + 1)

    plt.plot(e, metric, 'bo', label='Train ' + metric_name)
    plt.plot(e, val_metric, 'b', label='Validation ' + metric_name)
    plt.legend()
    plt.show()
```

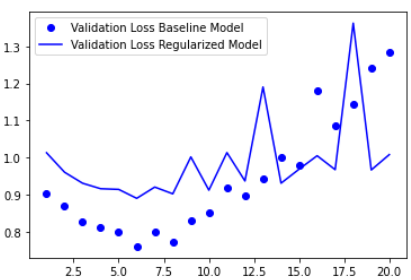
In [31]:
```
eval_metric(base_history, 'loss')
```

➔ **show prediction evaluation :**

In [36]:
```
compare_loss_with_baseline(reduced_history, 'Reduced Model')
```

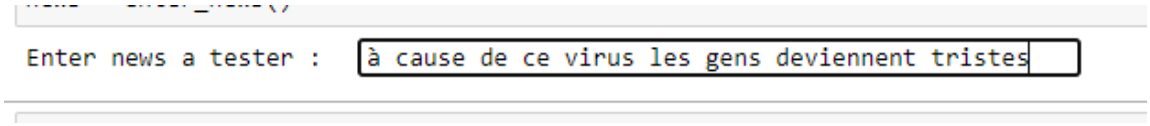


In [39]:
```
compare_loss_with_baseline(reg_history, 'Regularized Model')
```

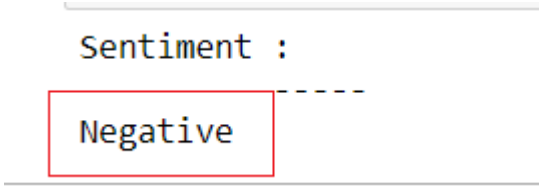


e) Sentiment Analysis Test :

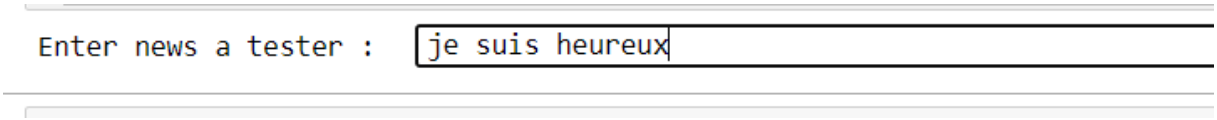
- **User Input : example negative**



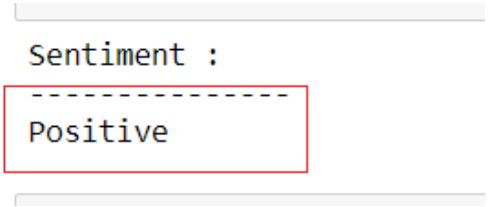
- **Output :**



- **User Input : example Positive :**



- **Output :**



V. web app dev: Django, angular, GraphQL, Elasticsearch & MongoDB.

In this project we decided to base our application single page on Django as a Backend, Angular as a Frontend, MongoDB as our Database and Elasticsearch as a search engine and finally GraphQL to connect Django with Angular.

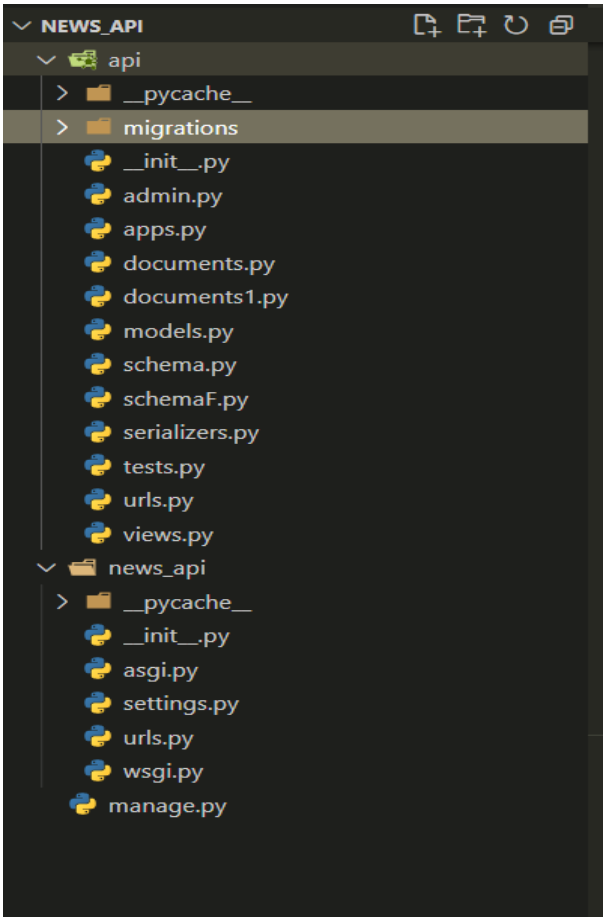
➢ Django:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

To start with Django, we firstly installed it and then we create a new application

```
D:\Autres\VS Tests Folder\Django_Angular_Project\news_api>python -m django --version
3.2.4
```

Here are the application folders and files:



The files inside the " news_api " folder are the project files they are created once when we first start project

The files inside " api " folder are the app file they are created when we create an app inside Django project.

To adapt Django with our Project we had to do some modification inside settings like adding new libraries and creating the models for our app then creating the schema for this app that will be manipulated by GraphQL and also creating the GraphQL vue etc.

here are the apps installed as mentioned before :

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'api',
    'graphene_django',
    'django_filters',
    'django_elasticsearch_dsl',
    'django_elasticsearch_dsl_drf',
    'rest_framework',
    'corsheaders',

]
```

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'News_DB',
    }
}
```

```
ELASTICSEARCH_DSL={
    'default': {
        'hosts': 'localhost:9200'
    },
}
```

These two parameters allowed us to use Django with Elasticsearch and Django with MongoDB.

Here is an example for our models:

```
class News(models.Model):
    #id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=180)
    details = models.TextField()
    #date = models.CharField(max_length=80)

    def __str__(self):
        return self.title

    class Meta:
        ordering = ('id',)


class FakeNews(models.Model):
    #id = models.AutoField(primary_key=True)
    title = models.CharField(max_length=180)
    details = models.TextField()
    #date = models.CharField(max_length=80)

    def __str__(self):
        return self.title

    class Meta:
        ordering = ('id',)
```

In this picture we have two models one for the real news and the other for the fake ones, this model will be used by schema to create query method and mutation so eventually those queries and mutations will be used by GraphQL.

Now let give a look to a schema file:

```python
class NewsType(DjangoObjectType):
    class Meta:
        model = News
        fields = "__all__"   #we can chose all field by using this fields = "__all__" or ("id","title","details","date")


class FakeNewsType(DjangoObjectType):
    class Meta:
        model = FakeNews
        fields = "__all__"   #we can chose all field by using this fields = "__all__" or ("id","title","details","date")
```

```python
class Query(graphene.ObjectType):
    all_news = graphene.List(NewsType)
    all_fake_news = graphene.List(FakeNewsType)
    news = graphene.Field(NewsType, news_id=graphene.Int(required=True))
    fake_news = graphene.Field(FakeNewsType, fake_news_id=graphene.Int(required=True))
    all_post_documents = ElasticsearchConnectionField(Searchnews)



    def resolve_all_news(self, info, **kwargs):
        return News.objects.all()

    def resolve_all_fake_news(self, info, **kwargs):
        return FakeNews.objects.all()

    def resolve_news(self, info, news_id):
        return News.objects.get(pk=news_id)

    def resolve_fake_news(self, info, fake_news_id):
        return FakeNews.objects.get(pk=fake_news_id)
```

Here is the schema's query as mentioned these queries will be used by GraphQL to access data in MongoDB and get back this data to display it, in the schema we have also mutation to do other jobs as adding news or updating deleting etc. and also, we could use this mutation as function to do a certain specific job.

An example for mutations:

```python
class CreateNews(graphene.Mutation):
    class Arguments:
        News_data = NewsInput(required=True)

    News = graphene.Field(NewsType)

    @staticmethod
    def mutate(root, info, News_data):
        News_instance = News(
            #id=News_data.id,
            title=News_data.title,
            details=News_data.details,
            #date=News_data.date
        )
        News_instance.save()
        return CreateNews(News=News_instance)
```

```python
class Mutation(graphene.ObjectType):
    detect_news = DetectNews.Field()
    analyse_data = AnalyseData.Field()
    clean_data = CleanData.Field()
    create_News = CreateNews.Field()
    create_Fake_News = CreateFakeNews.Field()
    update_News = UpdateNews.Field()
    update_fake_news = UpdateFakeNews.Field()
    delete_News = DeleteNews.Field()
    delete_fake_news = DeleteFakeNews.Field()


schema = graphene.Schema(query=Query, mutation=Mutation)
```

- **GraphQL**

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

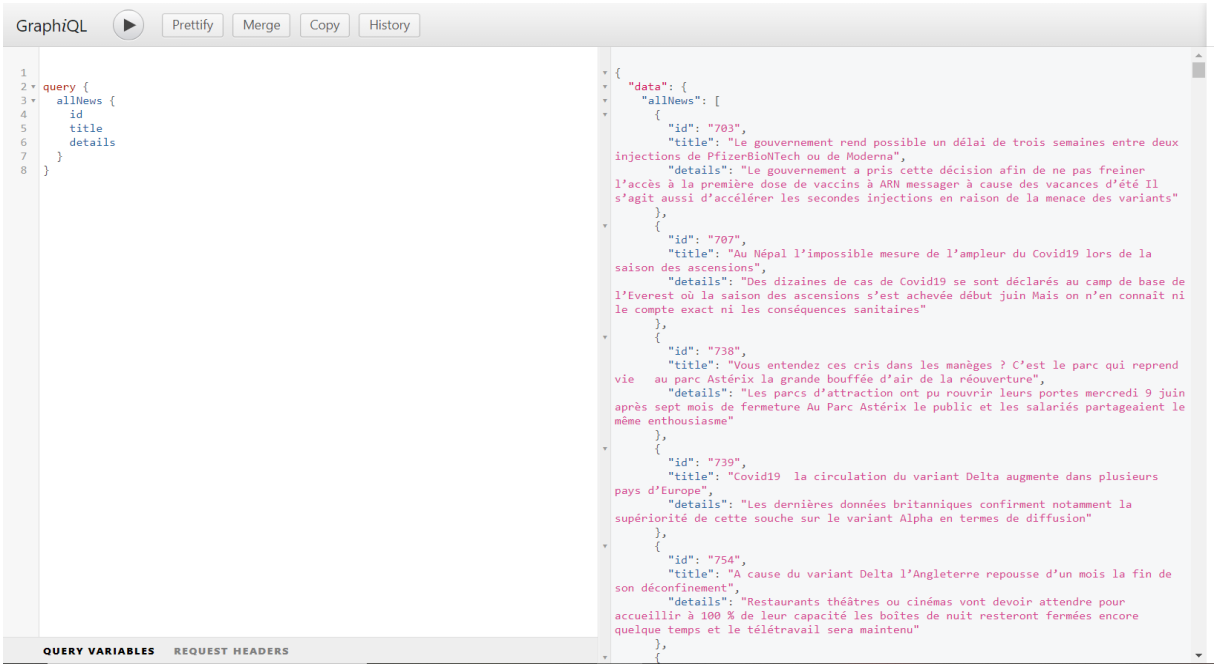
So, to use GraphQL with Django we had simply to install graphene-django library, create a view and call the schema method created before and that's it.

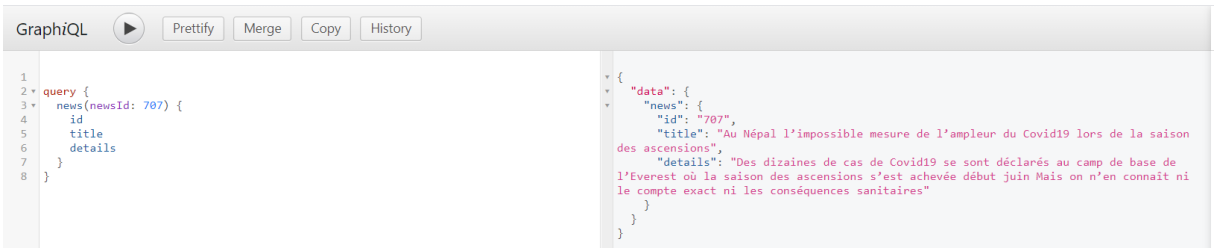
```
'django.contrib.staticfiles',
'api',
'graphene_django',
```

```
from django.urls import path
from graphene_django.views import GraphQLView
from api.schema import schema


urlpatterns = [
    path("graphql", GraphQLView.as_view(graphiql=True, schema=schema)),
    #path("graphql", GraphQLView.as_view(graphiql=True, schema=schemaF)),
]
```

```
GRAPHENE = {
    "SCHEMA": "api.schema.schema"
}
```

Here are some examples of GraphQL Use:





➢ Elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease.

As we know for this project, we had to adapt Elasicsearch with GraphQL so we will use the search from Frontend for this we had to install two libraries called graphene elastic and django-elasticsearch-dsl, then create an indexing for our database use commands line:

“. /manage.py search_index –rebuild “

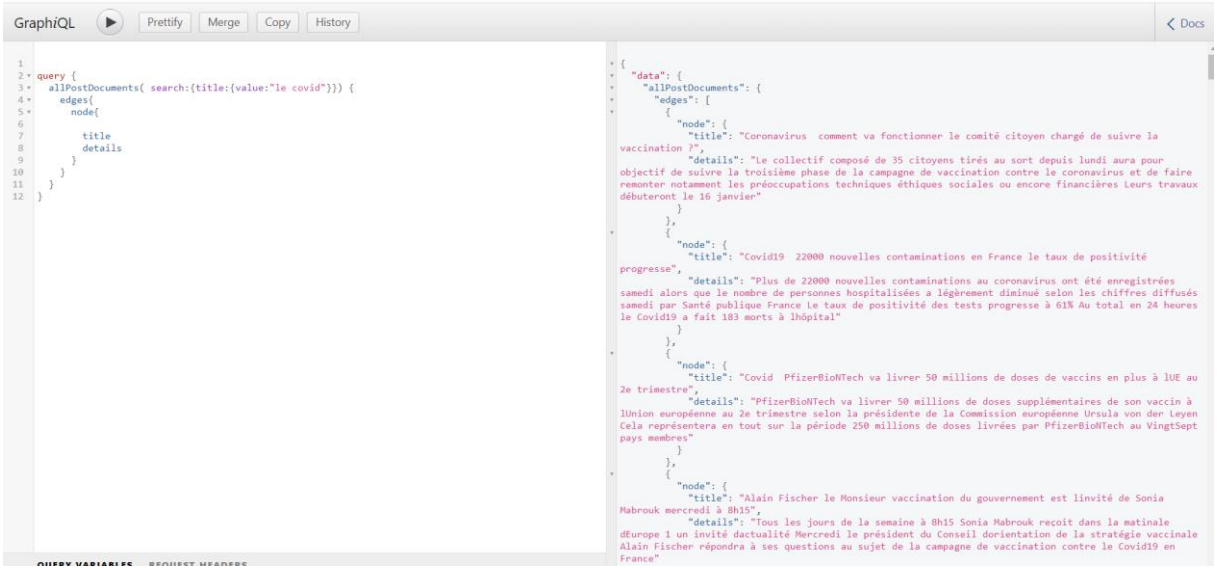
And right before creating the index we had to define elasticsearch_DSL index class in a file colled “document.py”

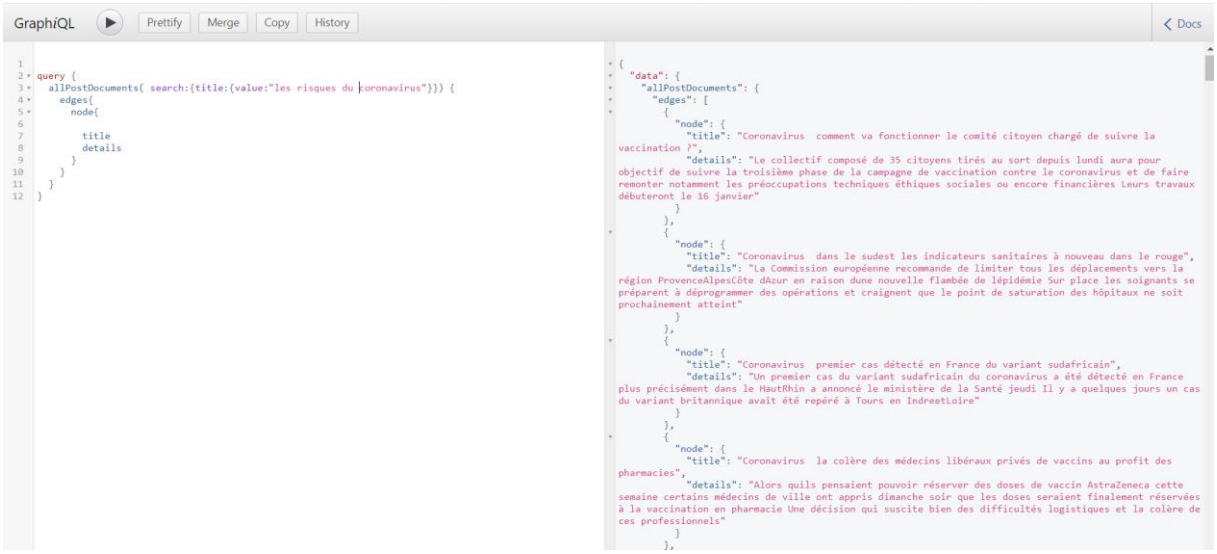
```
from django_elasticsearch_dsl import (Document, fields, Index)
from api.models import News
from django_elasticsearch_dsl.registries import registry
from elasticsearch_dsl import analyzer



news_index = Index('news')
news_index.settings(
    number_of_shards=1,
    number_of_replicas=1
)


@registry.register_document
@news_index.doc_type
class newsSearch(Document):


    title = fields.TextField(
        fields={
            "raw": fields.TextField(analyzer='keyword')
            }
    )

    details = fields.TextField(
        fields={
            "raw": fields.TextField(analyzer='keyword')
            }
    )
    id = fields.IntegerField(attr="id")
```

After this we define the query in schema file to exploit the elasticsearch_DSL library

```python
class Searchnews(ElasticsearchObjectType):

    class Meta(object):
        document = newsSearch
        interfaces = [relay.Node]
        filter_backends = [
            FilteringFilterBackend,
            SearchFilterBackend,
            HighlightFilterBackend,
            OrderingFilterBackend,
            DefaultOrderingFilterBackend,
        ]

        # For `FilteringFilterBackend` backend
        filter_fields = {

            'title': {
                'field': 'title.raw',
                # Available lookups
                'lookups': [
                    LOOKUP_FILTER_TERM,
                    LOOKUP_FILTER_TERMS,
                    LOOKUP_FILTER_PREFIX,
                    LOOKUP_FILTER_WILDCARD,
                    LOOKUP_QUERY_IN,
                    LOOKUP_QUERY_EXCLUDE,
                ],
                # Default lookup
                'default_lookup': LOOKUP_FILTER_TERM,
            },
```

Here is a use example:



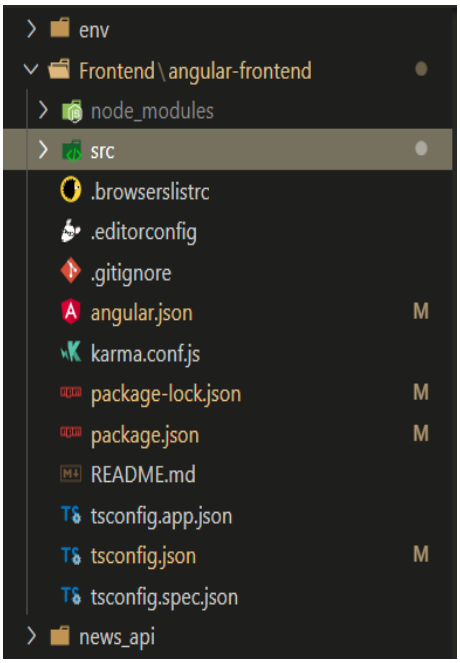


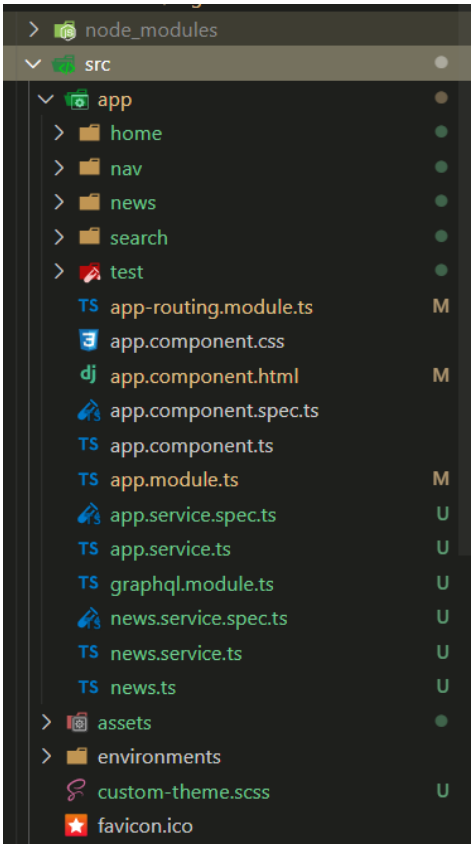
➢ Angular

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps.

These Angular docs help you learn and use the Angular framework and development platform, from your first application to optimizing complex single-page apps for enterprises. Tutorials and guides include downloadable examples to accelerate your projects.

To start with angular, we firstly install an angular client then we create a workspace and we initiate the application

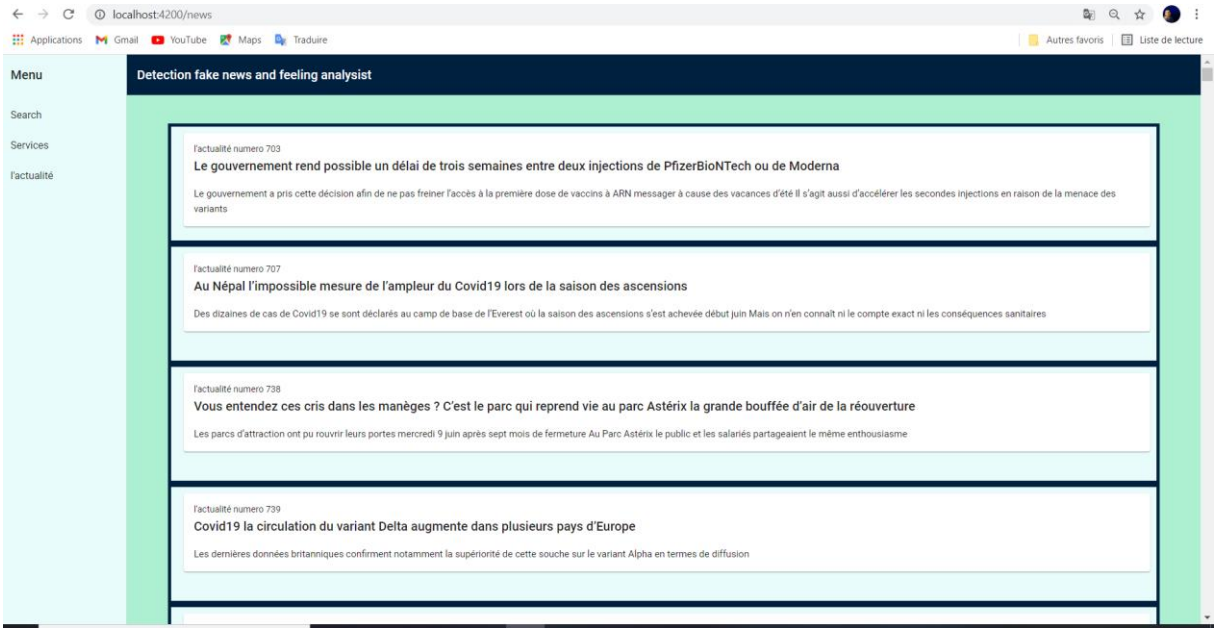




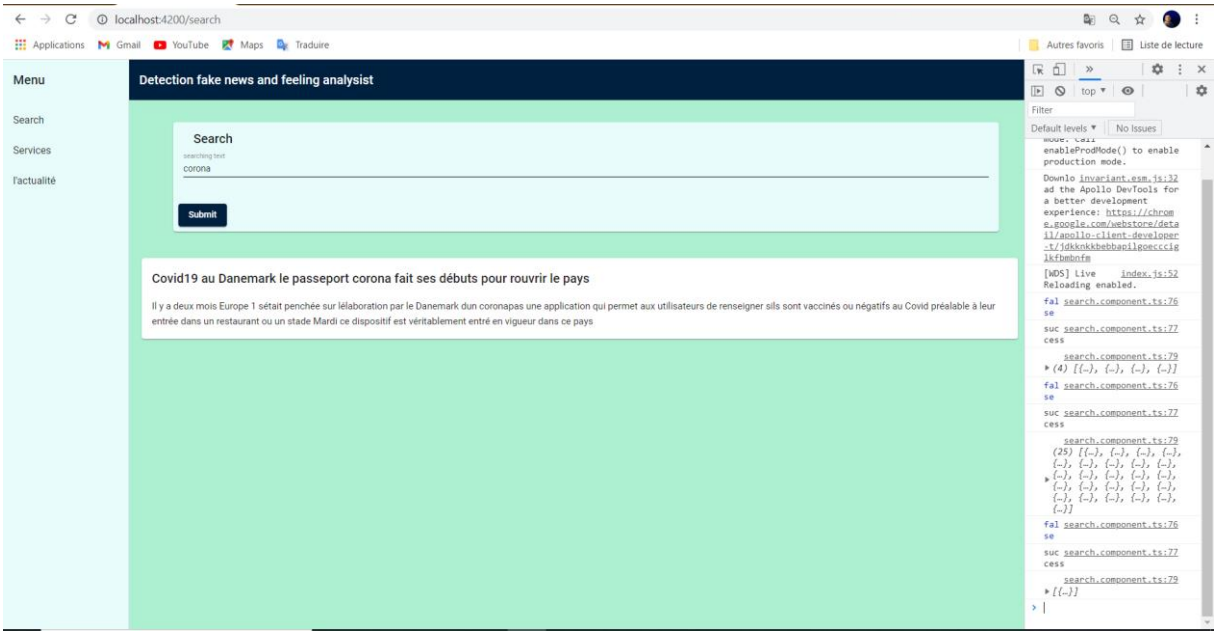
Here we have angular components, assets, environment and node modules etc.…

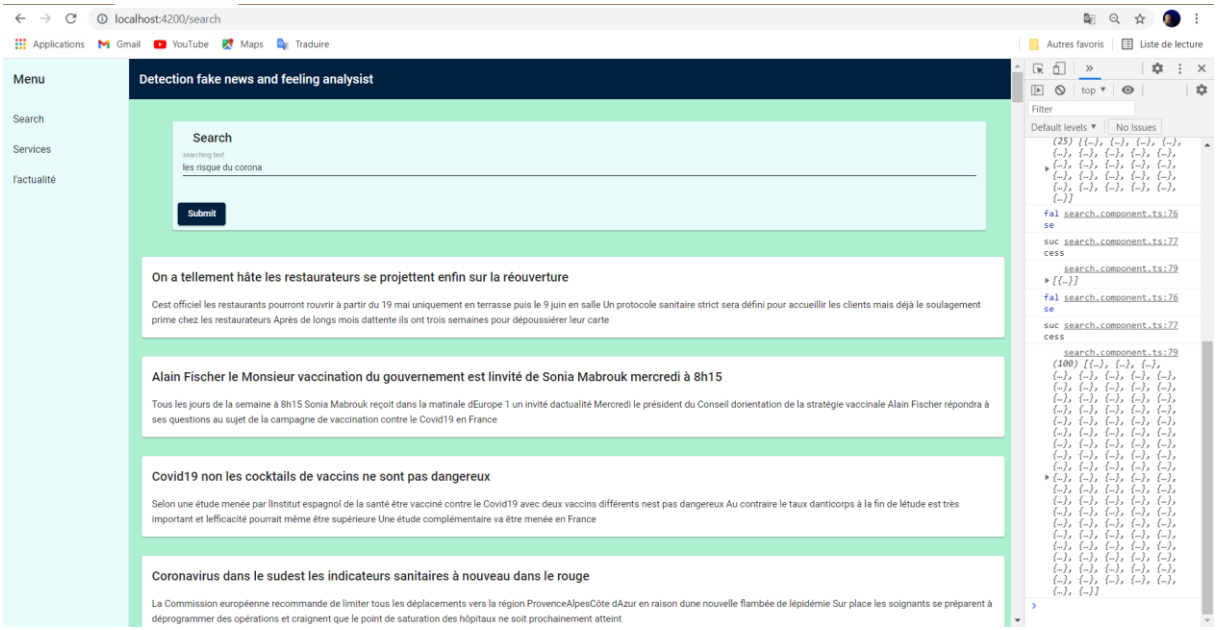
For our project we used 4 components home, nav, news and search

The news component contains all the news that are in the database these news gets sent from MongoDB to Django then to angular using GraphQL and Apollo methods.

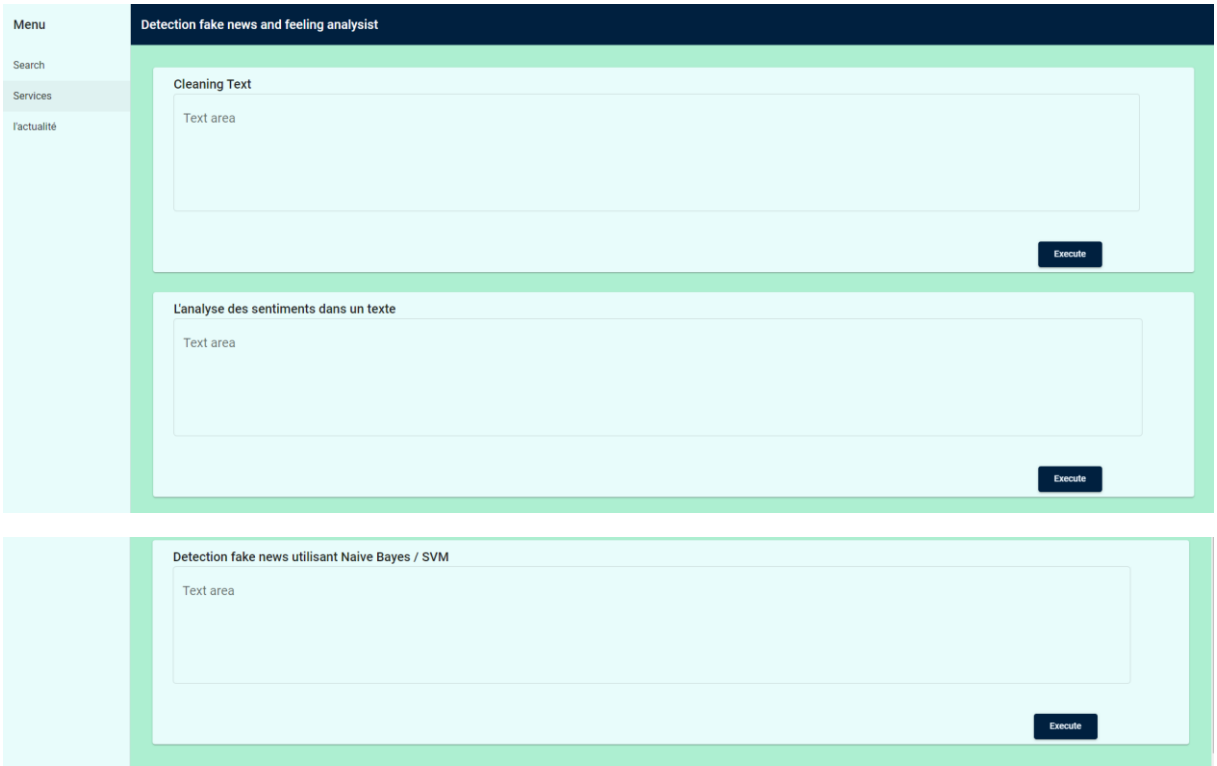


The search component contains a search area where user enter an input then the input gets merged with GraphQL query and sent to backend where Django execute an elastic search in the database about the input, so eventually the news related to the search will be displayed in the frontend here is an example illustrating:





The home component contains text areas where the user enters his text in the service he wants and after clicking on execute the result will come from the back end using GraphQL and apollo methods and it will be displayed in the interface



Now let’s see an example of apollo methods used for these jobs

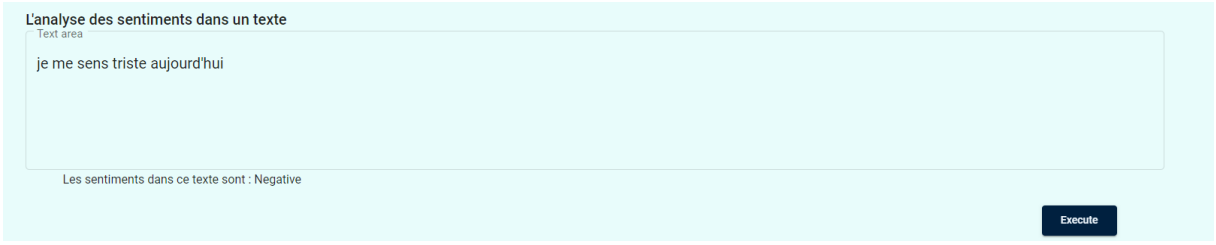
Here is the GraphQL query that being used in the job

```
const GET_Sentiment = gql`mutation AnalysingData($fel: String!){
  analyseData(textInput: $fel){
    data{
      text
    }
  }
}`
```

Here is the apollo method:

```
onclick_Sentiment(){
  this.apollo.mutate({
    mutation: GET_Sentiment,
    variables:{
      fel : this.Sentiment_Textarea,
    },
  }).subscribe(

      ({ data }) => {

      console.log("success");
      console.log(data)
      this.Sentiment_return = Array.of(data);
      console.log(this.Sentiment_return);
      //this.filteredNews = this.newss;
  });

}
```

With the same methods we have detection news and cleaning text working now let's see these services from a user place:



VI. Conclusion

In this paper, we describe and release a fake news detection dataset containing 3000 fake and real news related to COVID-19.

We collect these posts from various social media and fact checking web-sites, and verify the veracity of each news.

Also, we used the machine learning models: NB and SVM classifiers to predict either a news is real or not, and Deep Learning for sentiment analysis.

Future work could be targeted towards collecting more data, enriching the data by providing the reason for being real/fake along with the labels, collecting multilingual data.

It was a good project that we practiced what we learned in machine learning and NLP and also NoSQL database like MongoDB & Elasticsearch.

VII. References

- https://www.kaggle.com/c/fake-detection
- https://www.researchgate.net/publication/345554743_Fighting_an_Infodemic_COVID-19_Fake_News_Dataset
- https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning
- https://towardsdatascience.com/fake-news-detection-with-machine-learning-using-python-3347d9899ad1
- https://www.hindawi.com/journals/complexity/2020/8885861/
- https://www.kaggle.com/c/fakenewskdd2020
- https://www.kaggle.com/c/nlp-getting-started/discussion/123454
- https://www.youtube.com/watch?v=GS_ylghUtLQ
- https://www.youtube.com/watch?v=5X27excCyXk&t=1587s
- https://arxiv.org/pdf/2101.00180.pdf
- https://ieeexplore.ieee.org/document/8843612
- https://www.sciencedirect.com/science/article/pii/S2667096820300070
- https://graphene-elastic.readthedocs.io/en/latest/
- https://pypi.org/project/graphene-elastic/
- https://apollo-angular.com/docs/

GitHub Link: https://github.com/anashamza01/NoSQL_Project