Projet Machine Learning et Systèmes multi-agents

Réalisée par :
EL KTIBI El hassane
CHAOUKI Mouad

Encadrée par :
Mr. Lotfi Aachak
Mr. Ennaimi

2ème année Cycle Ingénieur Logiciels et Système Intelligents [2019-2020]

Département Génie Informatique

Introduction générale

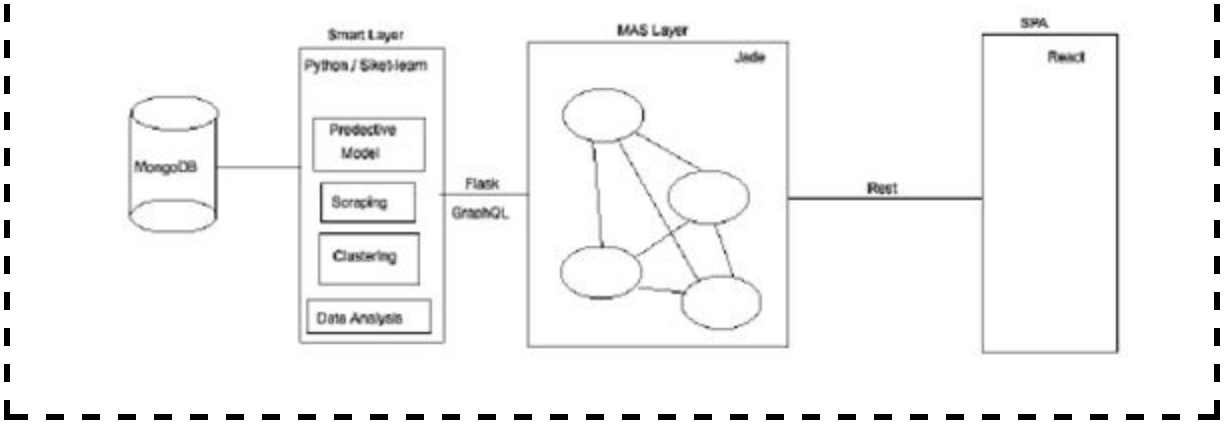
L'informatique est aujourd'hui au centre de toutes les activités de l'homme. Que ce soit un simple achat dans un magasin, le règlement d'une facture, un appel téléphonique, une transaction bancaire ou un partage de données entre filiale d'une entreprise, un système d'information simple ou complexe intervient toujours. Quel que soit le domaine, (télécommunications, commerce, entreprise ou médecine), nous avons toujours besoin d'un système de traitement automatique de l'information. Il y a quatre ans, l'Université de Maroua fut créée et par la même occasion un centre de santé chargé du suivi médical de chaque individu de la communauté universitaire : le Centre Médico-Social.

Objectif de projet

L'objectif principal du projet est la réalisation d'un système multi-agents intelligents basé sur plusieurs algorithmes d'apprentissage supervisé et non supervise, le système sera composé de plusieurs agents intelligents et interactifs, d'où chaque agent effectue une tâche spécifique. En plus d'une application Single page Application qui va interagir avec le système.

Conception de projet

l'architecture de système :



Le projet est constitué de 3 couches distinctes qui peuvent être déployé sur plusieurs machines.

La couche intelligente

C'est une api REST basée sur Flask contenant tous les scripts nécessaires pour scrapper, prétraiter et charger les données dans une base de données mongodb

en plus de tous les modèles que nous avons développés pour faire des clusters, des prédictions, des analyses sentimentales et des visualisations

La couche MAS

C'est une api REST basée sur SpringBoot contenant JADE Framework avec tous nos agents, et leurs comportements et comment ils communiquent entre eux et comment ils servent la couche frontend SPA.

La couche SPA

C'est la partie front qui contient des pages pour chaque agent(analyse des sentiments, clustering, prediction, data visualisation), chaque page communique directement avec l'agent approprié.

Chapitre 1 : La partie Smart Layer

Introduction

Dans cette partie, nous avons développé une API (REST) en utilisant le framework Flask, elle contient tous les scripts nécessaires pour les agents, et aussi pour le grattage, le prétraitement et le chargement des données dans une base de données mongodb.

Nous l'avons appelée couche intelligente car elle contient tous les modèles d'apprentissage machine qui peuvent être consommés par les agents à l'aide de requêtes HTTP.

Environnement de développement

VS code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

Mongodb

MongoDB (de l'anglais humongous qui peut être traduit par « énorme ») est un système de gestion de base de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il est écrit en C++.

Flask

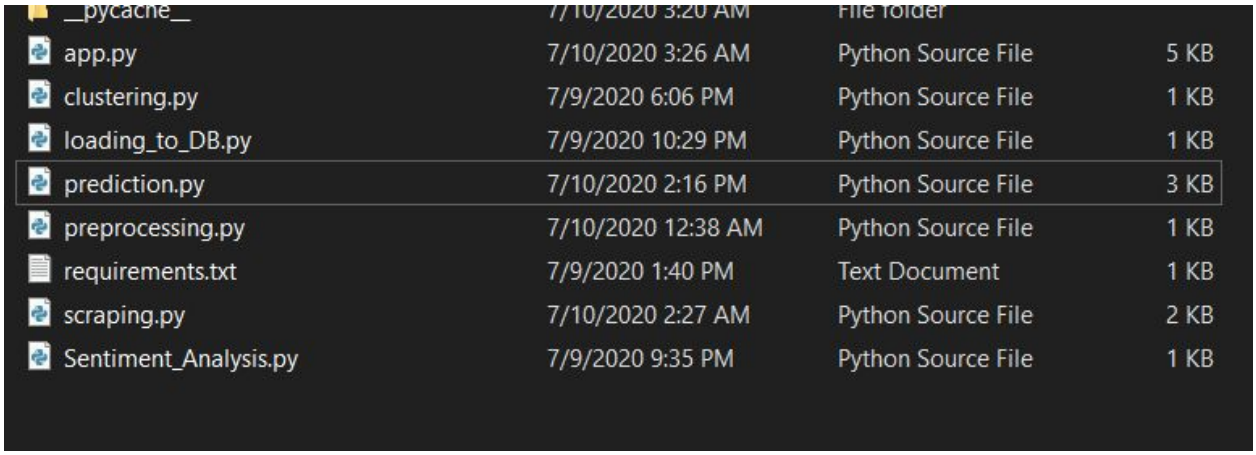
Flask est un framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de templates. Il est distribué sous licence BSD.

Libraries

Pymongo, pandas, numpy, sklearn, matplotlib, BeautifulSoup, flair

Pour exécuter correctement cette couche, assurez-vous de lire le fichier “requirement.txt” et d'installer tous ces libraries.

Architecture du projet



Cette couche est structurée en différents fichiers dont chacun contient des fonctions spécifiques et un fichier principal app.py qui est exécuté et contient notre API.

Description de l'API

@app.route('/api/v1/scrapNews', methods=['GET']) :

nécessite un paramètre "nombre de pages" pour commencer à scraper les titres des actualités du site web source "maroc world news".

@app.route('/api/v1/preprocessing', methods=['GET'])

lance le prétraitement et le nettoyage des données collectées

@app.route('/api/v1/scrapCovData', methods=['GET'])

nécessite un paramètre "type de data", pour commencer à collecter les données correctes de covid19 au maroc selon le type de données que nous voulons.

@app.route('/api/v1/loadToDB', methods=['GET'])

nécessite un paramètre "type", pour charger les données collectées dans la collection mongodb appropriée.

@app.route('/api/v1/startAnalysis', methods=['GET'])

il commence à analyser les titres de l'actualité recueillis et enregistre les résultats.

@app.route('/api/v1/analyzeSentiment', methods=['POST'])

nécessite un paramètre "text", pour commencer à analyser le texte passé en paramètres et renvoie le sentiment, et la confiance de model pour ce résultat.

@app.route('/api/v1/getNews', methods=['GET'])

nécessite les paramètre "number" et 'sentiment' , et renvoie le nombre d'actualités correspondent au sentiment passé en paramètres.

@app.route('/api/v1/statAnalysis', methods=['GET'])

renvoie des statistiques sur les titres de l'actualité, le nombre de titres, le nombre de titres positifs et négatifs.

@app.route('/api/v1/marocData', methods = ["GET"])

renvoie les données covid19 marocaines pour la visualisation.

@app.route('/api/v1/getClusters', methods = ["GET"])

renvoie les régions regroupées “clusters” du maroc en ce qui concerne les données de la covid 19.

@app.route('/api/v1/prediction', methods = ["GET"])

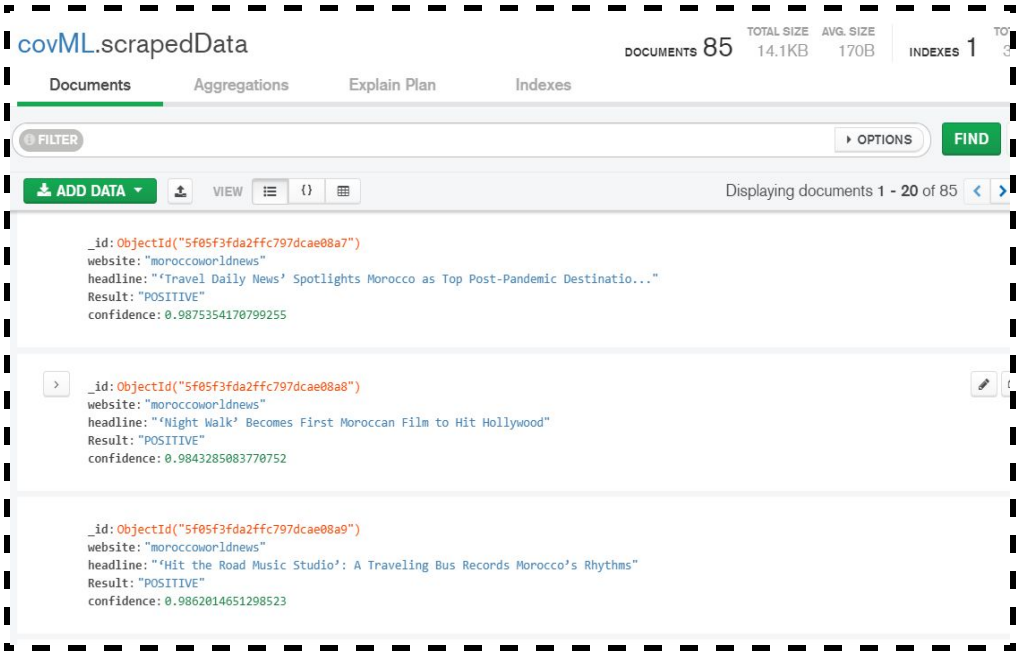
nécessite un paramètre "number", c'est le nombre des cas confirmés pour prédire et renvoie le nombre des mort.

Scraping

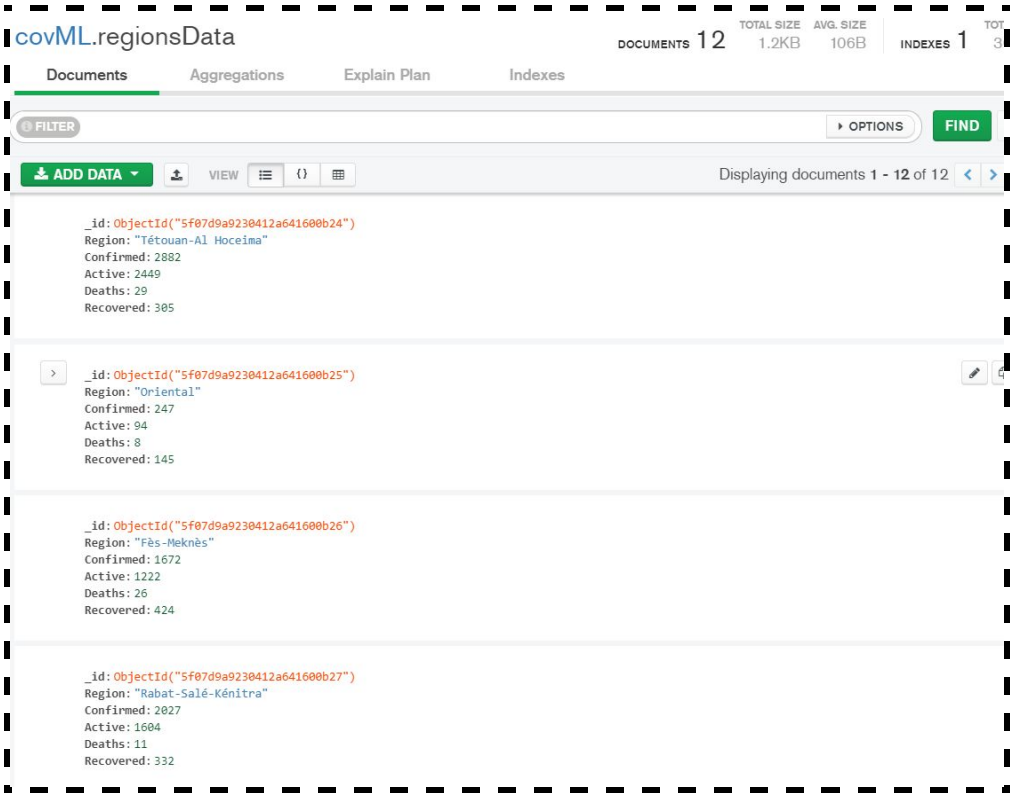
```python
# scraping.py > ...
from bs4 import BeautifulSoup
import requests
import pandas as pd

def scrap_moroccan_news(numOfPages):
  website='moroccoworldnews'
  news_dict=[]
  numOfPages=int(numOfPages)
  for i in range(1, numOfPages+1):
      url="https://www.moroccoworldnews.com/news-2/page/{}".format(i)
      response = requests.get(url)
      soup = BeautifulSoup(response.content,"html.parser")
      for div in soup.findAll("div", {'class':'td-ss-main-sidebar'}):
        div.decompose()
      for div1 in soup.findAll("div", {'class':'td-subcategory-header'}):
        div1.decompose()
      for head in soup.find_all('h3', {'class':'entry-title td-module-title'}):
        headl=head.find('a').get('title')
        news_dict.append({'website':website,'url': url,'headline': headl})

  news_df=pd.DataFrame(news_dict)
  news_df.to_csv("myScrapedData.csv" ,index=False, encoding='utf8')

def covid19TSdataScrapingMA():
    json_url ="https://raw.githubusercontent.com/aboullaite/Covid19-MA/master/stats/MA-times_series.csv"
    df = pd.read_csv(json_url,index_col=0)
    df.reset_index(level=0, inplace=True)
    df = df.rename(columns={'Dates / التواريخ': 'Dates', 'Cases / الحالات': 'Cases', 'Recovered / تعافى': 'Recovered', 'Deaths / الوفيات': 'De
    df.to_csv("MA-times_series.csv", index=False)

def regionDataScraping():
    json_url ="https://raw.githubusercontent.com/aboullaite/Covid19-MA/master/stats/regions.csv"
    df = pd.read_csv(json_url,index_col=0)
    df.reset_index(level=0, inplace=True)
    df = df.rename(columns={'Region / الجهة': 'Region', 'Total Cases / إجمالي الحالات': 'Confirmed', 'Active Cases / الحالات النشطة': 'Active',
    df.to_csv("MA-regions_data.csv", index=False)
```

Comme vous le voyez, nous supprimons trois types de données :

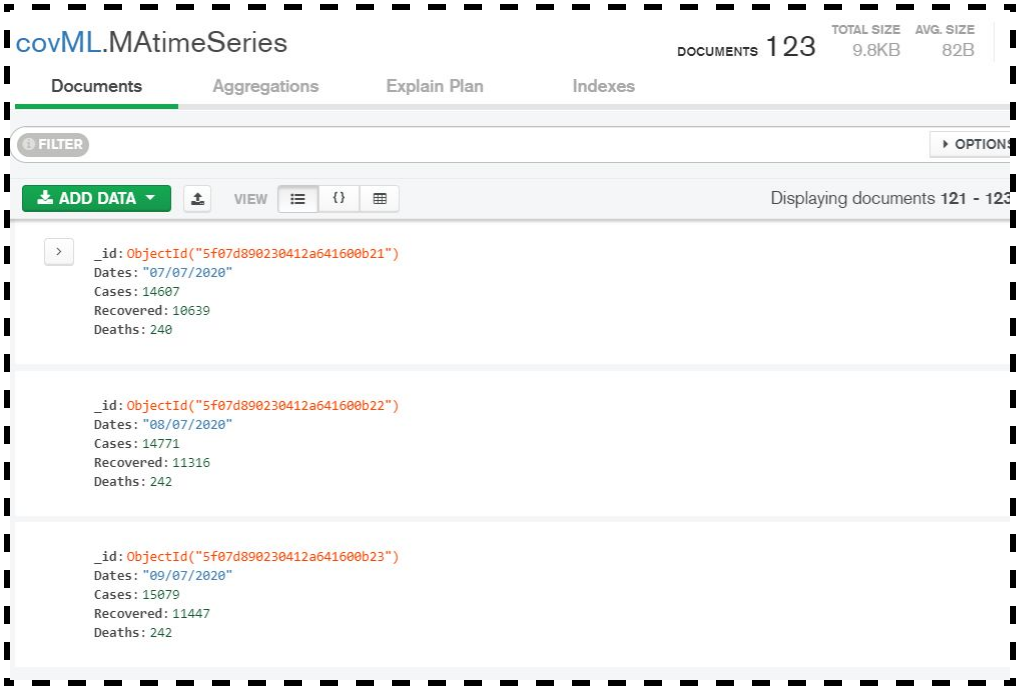
- les titres d'actualités au maroc, source: "moroccoworldnews"



- données de la covid 19 par régions au maroc



- données de la covid 19 pour chaque jour au maroc



note : pour les données marocaines, nous avons utilisé une source ‘updated’ quotidiennement, de sorte que nous n'avons pas besoin de scraper les données manuellement et d'exécuter le script chaque jour.

Preprocessing

preprocessing.py > ...

```python
import pandas as pd

def preprocessing_data():
    news_df=pd.read_csv("myScrapedData.csv")
    news_df=news_df.sort_values('headline', ascending=False)
    news_df=news_df.drop_duplicates(subset='headline')
    news_df=news_df.drop('url', 1)
    news_df.to_csv("myScrapedData.csv",index=False)
```

Voici un exemple de preprocessing et cleaning de data des titres d’actualités: suppression des colonnes non utilisé (colonne: url), suppression des titre répétés

Loading to database

loading_to_DB.py > ...

```python

import pandas as pd
from pymongo import MongoClient

client = MongoClient('localhost', 27017)
db = client.covML
covTS_col = db.MAtimeSeries
news_col =  db.scrapedData
regions_col =  db.regionsData

def load_to_DB(loading_type):
    if loading_type=="sentiment":
        news_df=pd.read_csv("myScrapedData.csv")
        myData = news_df.to_dict('records')
        news_col.delete_many({})
        news_col.insert_many(myData)
    if loading_type=="prediction":
        news_df=pd.read_csv("MA-times_series.csv")
        myData = news_df.to_dict('records')
        covTS_col.delete_many({})
        covTS_col.insert_many(myData)
    if loading_type=="regions":
        news_df=pd.read_csv("MA-regions_data.csv")
        myData = news_df.to_dict('records')
        regions_col.delete_many({})
        regions_col.insert_many(myData)

if __name__ == "__main__":
    load_to_DB("prediction")
```

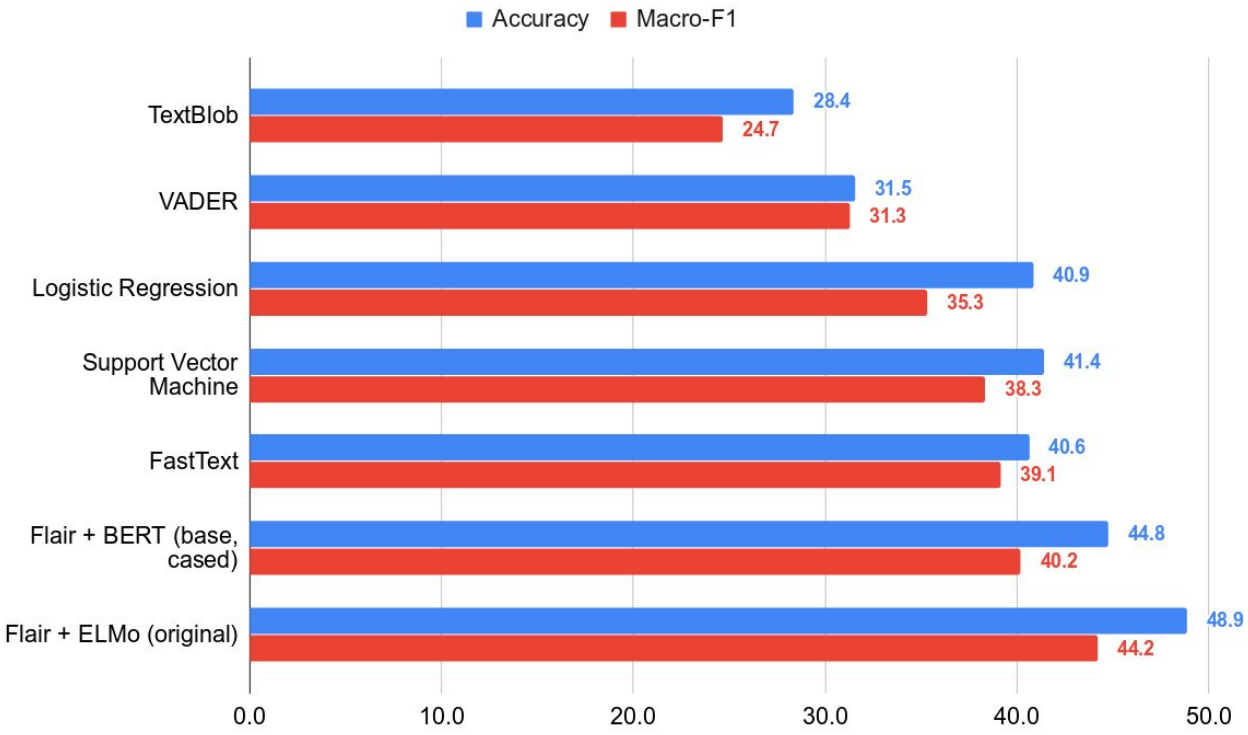
Chargement des données collectées (en fichiers .csv) dans la base de données mongodb appropriée.

Sentiment analysis

```python
Sentiment_Analysis.py > ...
from pymongo import MongoClient
import pandas as pd
import numpy as np
from flair.models import TextClassifier
from flair.data import Sentence
import logging
logging.basicConfig(level=logging.ERROR)

###### Mongodb connection
client = MongoClient('localhost', 27017)
db = client.covML
data_col = db.scrapedData
############################

flair_sentiment = TextClassifier.load('en-sentiment')

##
def analyze_sentiment(headline):
    s = Sentence(headline)
    flair_sentiment.predict(s)
    total_sentiment = s.labels[0].to_dict()
    return total_sentiment

##
def analyseSentiments():
    myNews=data_col.find()
    df =  pd.DataFrame(myNews)
    del df['_id']
    df['Result'] = np.array([analyze_sentiment(headline)['value'] for headline in df['headline']])
    df['confidence'] = np.array([analyze_sentiment(headline)['confidence'] for headline in df['headline']])
    data_col.delete_many({})
    myData = df.to_dict('records')
    data_col.insert_many(myData)
```

pour analyser la polarité des sentiments dans les titres des actualités, nous avons utilisé un modèle préformé “pretrained model” du "Flair" qui est le meilleur modèle que nous puissions trouver accessible à l'utilisation, et il est considéré comme l'état de l'art de l'analyse sentimentale.

Classifier	Accuracy	Macro-F1
TextBlob	28.4	24.7
VADER	31.5	31.3
Logistic Regression	40.9	35.3
Support Vector Machine	41.4	38.3
FastText	40.6	39.1
Flair + BERT (base, cased)	44.8	40.2
Flair + ELMo (original)	48.9	44.2



Clustering

clustering.py > clustering

```python
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans


def clustering():
    df=pd.read_excel('scrap_morocco.xlsx')
    data_kmeans=df[['Confirmed','Deaths']]
    kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
    y_kmeans = kmeans.fit_predict(data_kmeans)
    y_kmeans1=y_kmeans
    y_kmeans1=y_kmeans+1
    cluster = pd.DataFrame(y_kmeans1)
    data_kmeans['cluster'] = cluster
    data_risk= pd.DataFrame()
    data_risk["country"]=df["Region"]
    data_risk["Confirmed"]=y_kmeans1
    lis = list()
    for group in range(1,4):
        countries=data_risk.loc[data_risk['Confirmed']==group]
        listofcountries= list(countries['country'])
        lis.append(listofcountries)
    return lis
```

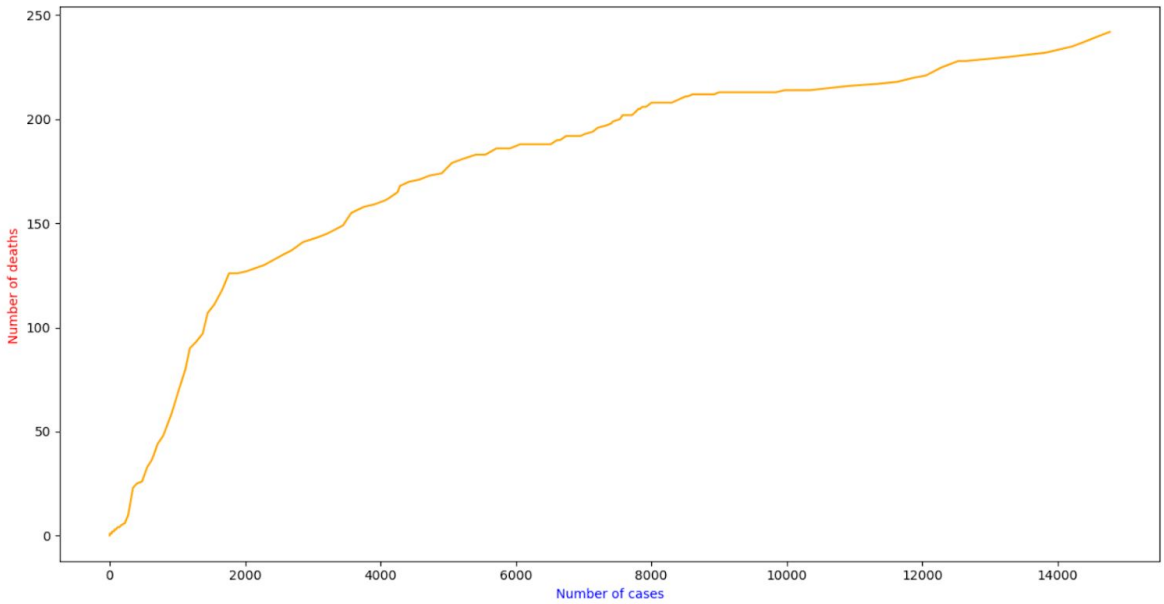
dans ce modèle, nous avons utilisé la bibliothèque sklearn pour entraîner l'algorithme K-means sur des données non étiquetées afin d'obtenir des clusters (régions) basés sur le nombre de cas confirmés et de décès, il est à noter que nous avons utilisé la "elbow method" pour déterminer le nombre de clusters

Prediction

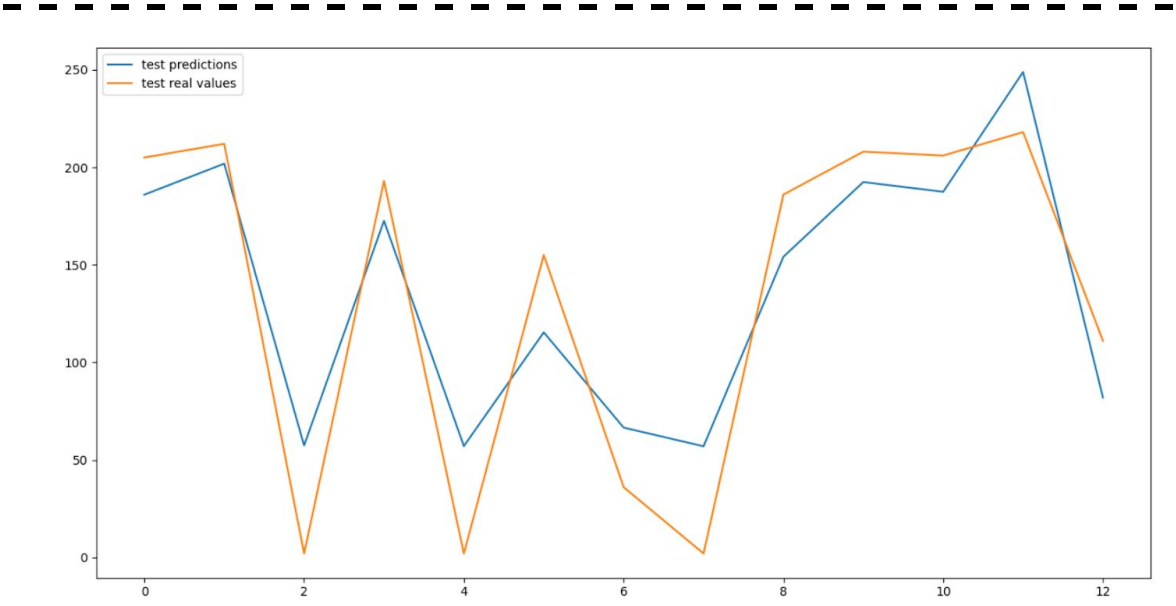
prediction.py > predictDeaths

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from pymongo import MongoClient
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import datetime as dt

###### Mongodb connection
client = MongoClient('localhost', 27017)
db = client.covML
data_col = db.MAtimeSeries
############################

def predictDeaths(val):
    myData=data_col.find()
    df = pd.DataFrame(myData)
    x=df['Cases'].values.reshape(-1,1)
    y=df['Deaths'].values.reshape(-1,1)
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0)
    linear_model=LinearRegression(normalize=True,fit_intercept=True)
    linear_model.fit(x_train,y_train)
    arr = np.array([val])
    arr=arr.reshape(-1, 1)
    linear_pred=linear_model.predict(arr)
    res=int(linear_pred[0])
    return res
```

dans ce modèle, nous avons utilisé la bibliothèque sklearn pour entraîner un algorithme de régression linéaire sur des données étiquetées (apprentissage supervisé) afin d'obtenir des prédictions sur le nombre de décès en fonction du nombre de cas confirmés, il convient de mentionner que ce modèle n'est pas précis, car les données ne sont pas linéaires et il est impossible de prédire un tel phénomène quel que soit le modèle utilisé



Voici la différence entre les nombres prédits et les nombre réels de dataset de test.



Conclusion

Dans cette partie, nous avons appris et appliqué différents concepts et méthodes, dont le scraping, le nettoyage et le prétraitement des données, en plus de certains algorithmes et techniques classiques de ML et de la manière dont tout cela peut être déployé dans une api REST à l'aide de Flask.

Chapitre 2 : La partie Backend-SMA: Spring Boot, JADE

Introduction

Le « backend » est un peu comme la partie immergée d'un iceberg. On ne la voit pas en tant que simple Internaute mais elle représente une très grande partie d'un projet web.

- Le Backend se compose généralement de trois éléments :
- Un serveur (hébergement web)
- Une application (site web, administration)
- Une base de données (sorte de feuille de calcul pour organiser les données)

Environnement de développement et frameworks

Spring

Spring est un framework open source pour construire et définir l'infrastructure d'une application Java, dont il facilite le développement et les tests.

JADE

JADE (Java Agent DEvelopment Framework) est un framework logiciel entièrement implémenté dans le langage Java. Il simplifie la mise en œuvre de systèmes multi-agents grâce à un middleware conforme aux spécifications du FIPA et à un ensemble d'outils graphiques qui prennent en charge les phases de débogage et de déploiement

Postman

Postman est un outil de test API évolutif qui s'intègre rapidement dans le pipeline CI/CD. Il a été lancé en 2012 en tant que projet parallèle par Abhinav Asthana pour simplifier le flux de travail des API dans les tests et le développement. API signifie Application Programming Interface (interface de programmation d'applications), qui permet aux applications logicielles de communiquer entre elles par le biais d'appels API.

Spring Boot

Un Framework ou kit de développement est un espace de travail modulaire, c'est à dire une suite d'outils et de bibliothèques qui facilitent et accélèrent le développement d'un logiciel. Il contient toutes les fonctions de base utiles au développement d'un type de programme, et permet donc de ne pas avoir besoin de réécrire les mêmes fonctions à chaque programme créé. Il en existe dans tous les langages de programmation.

Parmi les Framework Web, nous somme plus particulièrement intéressée aux Framework Spring boot. Dans leur grande majorité, elle est conçue sur le modèle MVC, ce qui permet de structurer les données. Ils imposent un cadre et des normes de développement qui permettent une programmation propre et modulaire. Une autre motivation pour ce choix est le fait que nous les membres de cette équipe nous avons déjà travaillé avec cette Framework. Il est aussi important de mentionner que Maven a été utilisé pour faire la gestion des dépendances du projet. De cette façon, il a été assez facile d'inclure plusieurs modules nécessaires pour l'application web. Ces modules auront permis de grandement réduire le code écrit grâce aux annotations fournies par le module permettant l'intégration avec Jade. De plus, il est assez facile d'implémenter de nouvelles routes avec les annotations fournies par Spring Boot.

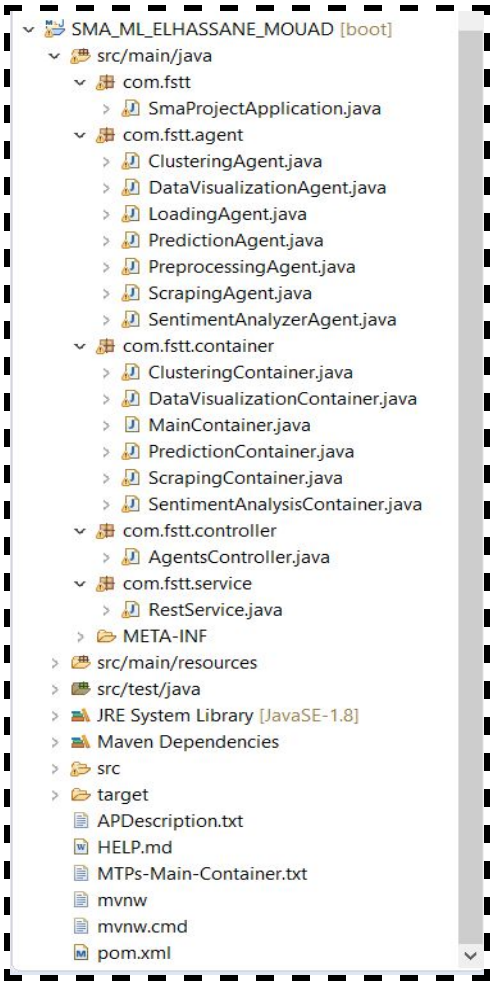
API Rest

Pour rendre accessible des données via un site, il existe les méthodes dites REST(representational state transfer). Il s'agit d'un ensemble de conventions et de

bonnes pratiques à respecter et non d'une technologie entière. L'information de base, dans une architecture REST, est appelée ressource. Toute information qui peut être nommée est une ressource : la description d'un bâtiment, la liste des arrêts de bus ou n'importe quel concept. Dans un système hypermédia, une ressource est tout ce qui peut être référencé par un lien. L'interface entre les composants est simple et uniforme. En HTTP, cette interface est implantée par les verbs GET, PUT, POST, DELETE, . . . qui permettent aux composants de manipuler les ressources de manière simple.

Architecture du projet

Le backend de ce projet se compose de quatre parties, soit la partie service, contrôleur, agents plus l'application main Spring boot.

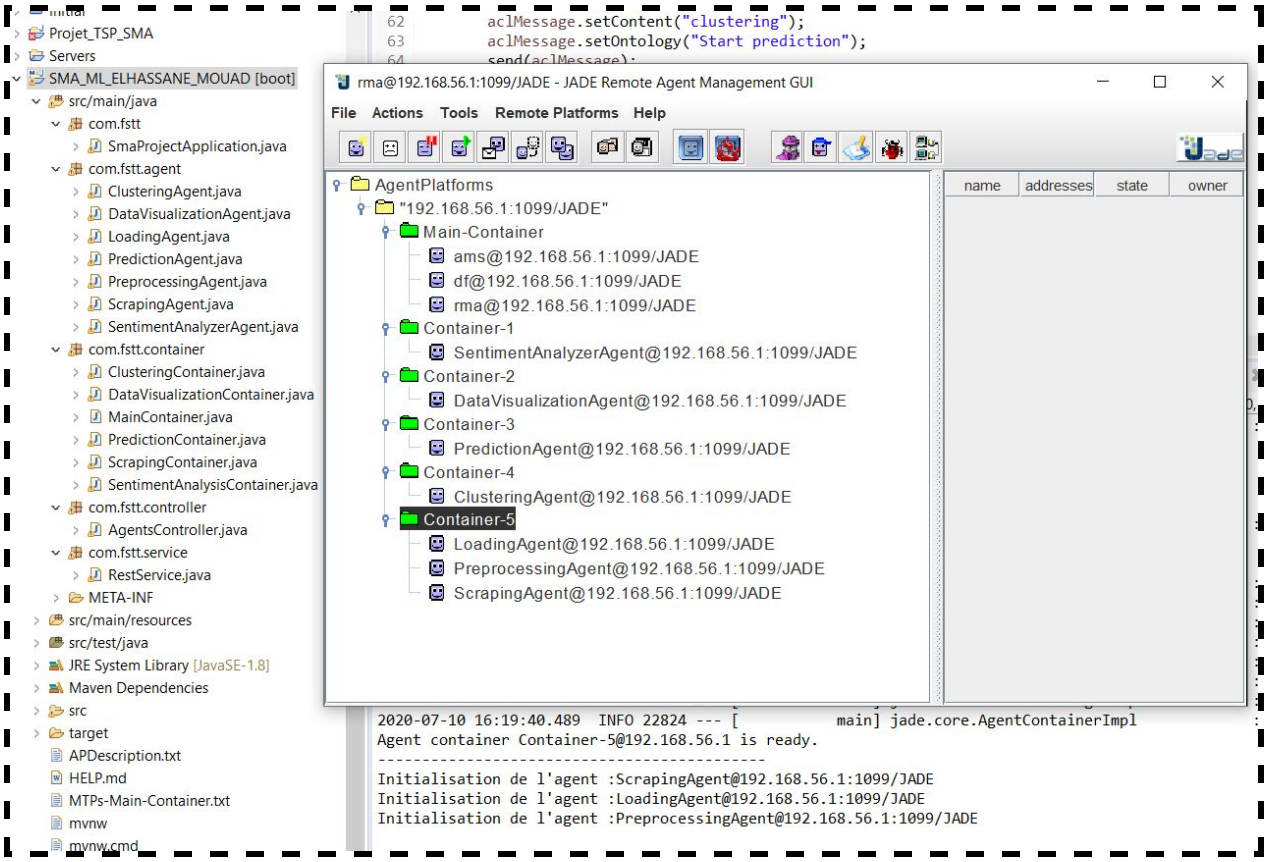


agent: ce package contient tous les agents, les agents interagissent directement avec leurs pages SPA et consomment leur besoin de la couche intelligente de Flask

container : ce package contient tous les containers des agents et le main conainer

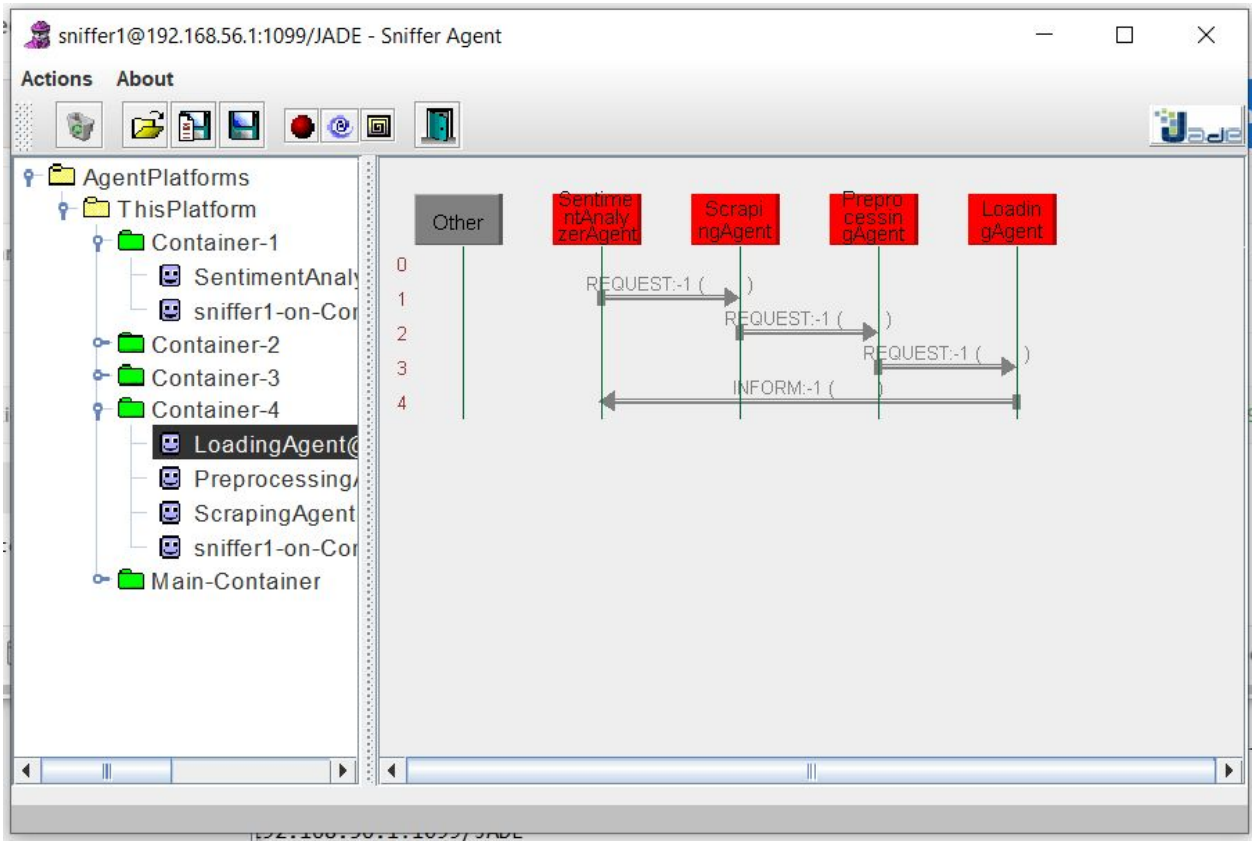
controller : c'est un contrôleur pour démarrer les agent depuis l'extérieure avec une requête http.

service : pour consommer l'api de la couche intelligente de Flask.



Nous avons ajouté les capacités REST à nos agents en utilisant l'annotation @RestController de Springboot alors nos agents sont capables de consommer la couche inteligennt et aussi de communiquer avec la couche SPA de frontend.

SentimentAnalyzerAgent



cet agent envoie une demande à l'agent de scrapping (requête REQUEST), l'agent de scraping collecte les données et envoie une demande à l'agent de prétraitement pour nettoyer les données et les prétraiter, puis l'agent de prétraitement envoie une demande à l'agent de chargement pour que ce dernier charge les données dans la base de données mongodb, puis l'agent de chargement envoie une requête INFORM à SentimentAnalyzerAgent pour qu'il puisse lancer une analyse sentimentale sur les données scrappées.

Voici le code source de l'agent:

```java
@RestController
@CrossOrigin("*")
public class SentimentAnalyzerAgent extends Agent {

    private static final long serialVersionUID = 1L;
    boolean isterminated=false;
    public static SentimentAnalyzerAgent dump;
    @Autowired
    RestService rs;
    @Override
    public void setup() {
        RestService restService=new RestService();
        System.out.println("Initialisation de l'agent :"+this.getAID().getName());
        dump=this;
        addBehaviour(new Behaviour() {
            @Override
            public void action() {
                ACLMessage aclMessage = receive();
                if (aclMessage!=null) {
                    try {
                        restService.get("/api/v1/startAnalysis");
                        System.out.println("Scraping-Preprocessing-Loading-SentimentAnalysis finished ----------");
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }

            @Override
            public boolean done() {
                return isterminated;
            }
        });
    }

    public SentimentAnalyzerAgent getDump(){
        return dump;
    }

    public void sendMessage(String param1){
        ACLMessage aclMessage=new ACLMessage(ACLMessage.REQUEST);
        aclMessage.addReceiver(new AID("ScrapingAgent",AID.ISLOCALNAME));
        try {

            aclMessage.setContentObject((Serializable) param1);
            aclMessage.setContent("analysis");

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        aclMessage.setOntology("Start Scraping");
        send(aclMessage);
    }


    @RequestMapping(value = "/getNews/{sentiment}/{number}", method = RequestMethod.GET)
    public JsonNode getSentiment(@PathVariable("sentiment") String sentiment, @PathVariable("number") String number) throws
        return rs.get("api/v1/getNews?number="+number+"&sentiment="+sentiment+"");
    }

    @RequestMapping(value = "/startScraping/{numOfPages}", method = RequestMethod.GET)
    public String startScraping(@PathVariable("numOfPages") String numOfPages){
        getDump().sendMessage(numOfPages);
        return "success";
    }

    @RequestMapping(value = "/getNewsStatistics", method = RequestMethod.GET)
    public JsonNode getStatistics() throws JsonMappingException, JsonProcessingException{
        return rs.get("/api/v1/statAnalysis");
    }

    @RequestMapping(value = "/analyzeText", method = RequestMethod.POST)
    public JsonNode analyseText(@PathVariable("text") String text) throws JsonMappingException, JsonProcessingException{
        return rs.get("/api/v1/analyzeSentiment?text="+text);
    }
```

ScrapingAgent

```java
@RestController
@CrossOrigin("*")
public class ScrapingAgent extends Agent {
    private static final long serialVersionUID = 1L;
    boolean isterminated=false;

    public void setup() {
        RestService restService=new RestService();
        System.out.println("Initialisation de l'agent :"+this.getAID().getName());
        addBehaviour(new Behaviour() {
            @Override
            public void action() {
                ACLMessage aclMessage = receive();
                if (aclMessage!=null) {
                    try {
                        String type=(String) aclMessage.getContent();

                        if(type.contains("analysis")){
                            String numOfPages= (String) aclMessage.getContentObject();
                            restService.get("/api/v1/scrapNews?numOfPages="+numOfPages);
                            ACLMessage demande = new ACLMessage(ACLMessage.REQUEST);
                            demande.addReceiver(new AID("PreprocessingAgent", AID.ISLOCALNAME));
                            demande.setContent("analysis");
                            System.out.println("scrapped succesfully ****************************");
                            send(demande);
                        }
                        if(type.contains("prediction")) {
                            restService.get("/api/v1/scrapCovData?typeData=prediction");
                            ACLMessage demande = new ACLMessage(ACLMessage.REQUEST);
                            demande.addReceiver(new AID("PreprocessingAgent", AID.ISLOCALNAME));
                            demande.setContent("prediction");
                            System.out.println("scrapped succesfully ****************************");
                            send(demande);
                        }
                        if(type.contains("clustering")) {
                            restService.get("/api/v1/scrapCovData?typeData=regions");
                            ACLMessage demande = new ACLMessage(ACLMessage.REQUEST);
                            demande.addReceiver(new AID("PreprocessingAgent", AID.ISLOCALNAME));
                            demande.setContent("clustering");
                            System.out.println("scrapped succesfully ****************************");
                            send(demande);
                        }
                        if(type.contains("visualization")) {
```

Cet agent lance des requêtes de scraping selon le type de données qu'on veut scraper et envoie une demande à l'agent de prétraitement

PreprocessingAgent

```java
@RestController
@CrossOrigin("*")
public class PreprocessingAgent extends Agent{
    private static final long serialVersionUID = 1L;

    boolean isterminated=false;

    @Override
    public void setup() {
         RestService restService=new RestService();
        System.out.println("Initialisation de l'agent :"+this.getAID().getName());
        addBehaviour(new Behaviour() {
            @Override
            public void action() {
                ACLMessage aclMessage = receive();
                if (aclMessage!=null) {
                    try {
                        String type=(String) aclMessage.getContent();
                        if(type.contains("analysis")){
                            JsonNode result=restService.get("/api/v1/preprocessing");
                            ACLMessage demande = new ACLMessage(ACLMessage.REQUEST);
                            demande.addReceiver(new AID("LoadingAgent", AID.ISLOCALNAME));
                            demande.setContent("Start loading");
                            System.out.println("preprocessed succesfully ****************************");
                            send(demande);
                        }
                        else {
                            ACLMessage demande = new ACLMessage(ACLMessage.REQUEST);
                            demande.addReceiver(new AID("LoadingAgent", AID.ISLOCALNAME));
                            demande.setContent(type);
                            System.out.println("preprocessed succesfully ****************************");
                            send(demande);
                        }

                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
```

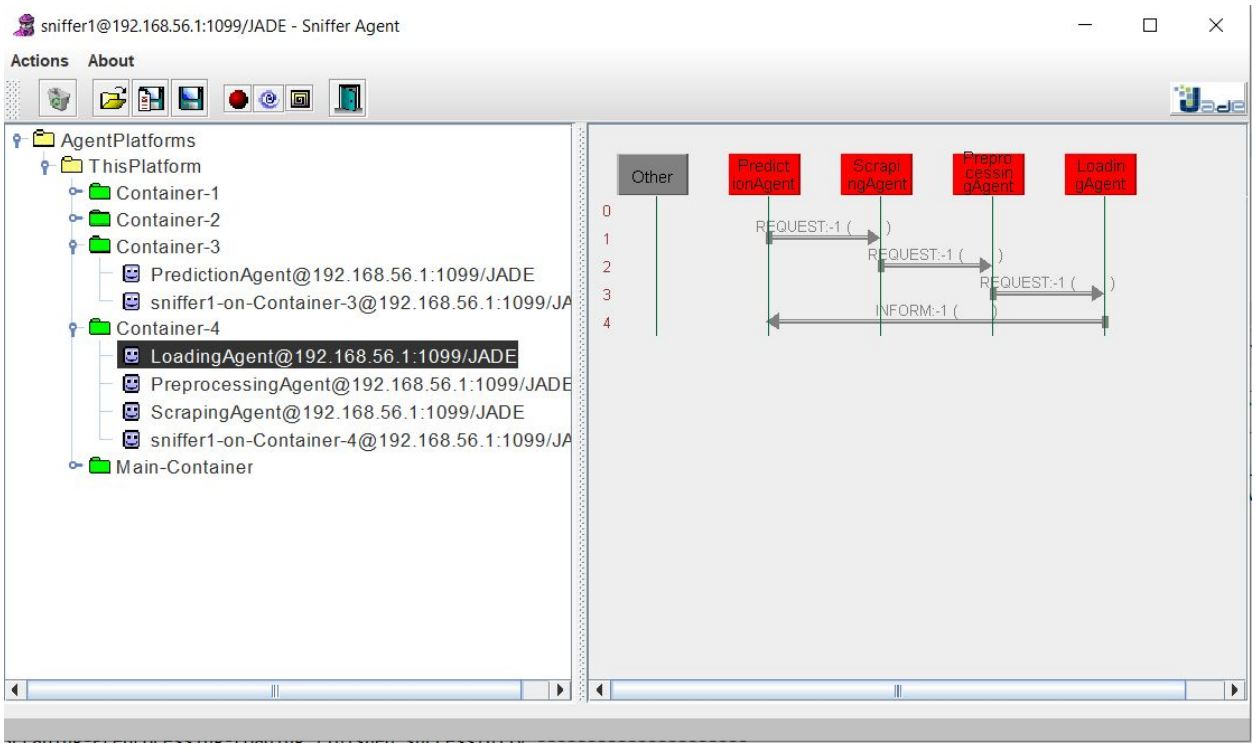
Cet agent lance des requêtes de preprocessing selon le type de données et envoie une demande à l'agent de chargement.

LoadingAgent

```java
public class LoadingAgent extends Agent {
    private static final long serialVersionUID = 1L;

    boolean isterminated=false;

    @Override
    public void setup() {
        System.out.println("Initialisation de l'agent :"+this.getAID().getName());
        addBehaviour(new Behaviour() {
            @Override
            public void action() {
                RestService restService=new RestService();
                ACLMessage aclMessage = receive();
                if (aclMessage!=null) {
                    try {
                        String typeScraping=(String)aclMessage.getContent();
                        System.out.println("typeScraping : "+typeScraping);
                        if(typeScraping.contains("analysis")){
                            restService.get("/api/v1/loadToDB?type=sentiment");
                            ACLMessage demande = new ACLMessage(ACLMessage.INFORM);
                            demande.addReceiver(new AID("SentimentAnalyzerAgent", AID.ISLOCALNAME));
                            demande.setContent("Start analysis");
                            System.out.println("loaded succesfully ****************************");
                            send(demande);
                        }
                        if(typeScraping.contains("prediction")) {
                            restService.get("/api/v1/loadToDB?type=prediction");
                            ACLMessage demande = new ACLMessage(ACLMessage.INFORM);
                            demande.addReceiver(new AID("PredictionAgent", AID.ISLOCALNAME));
                            demande.setContent("Start prediction");
                            System.out.println("loaded succesfully ****************************");
                            send(demande);
                        }
                        if(typeScraping.contains("clustering")) {
                            restService.get("/api/v1/loadToDB?type=regions");
                            ACLMessage demande = new ACLMessage(ACLMessage.INFORM);
                            demande.addReceiver(new AID("ClusteringAgent", AID.ISLOCALNAME));
                            demande.setContent("Start clustering");
                            System.out.println("loaded succesfully ****************************");
                            send(demande);
                        }
                        if(typeScraping.contains("visualization")) {
                            restService.get("/api/v1/loadToDB?type=regions");
                            ACLMessage demande = new ACLMessage(ACLMessage.INFORM);
```

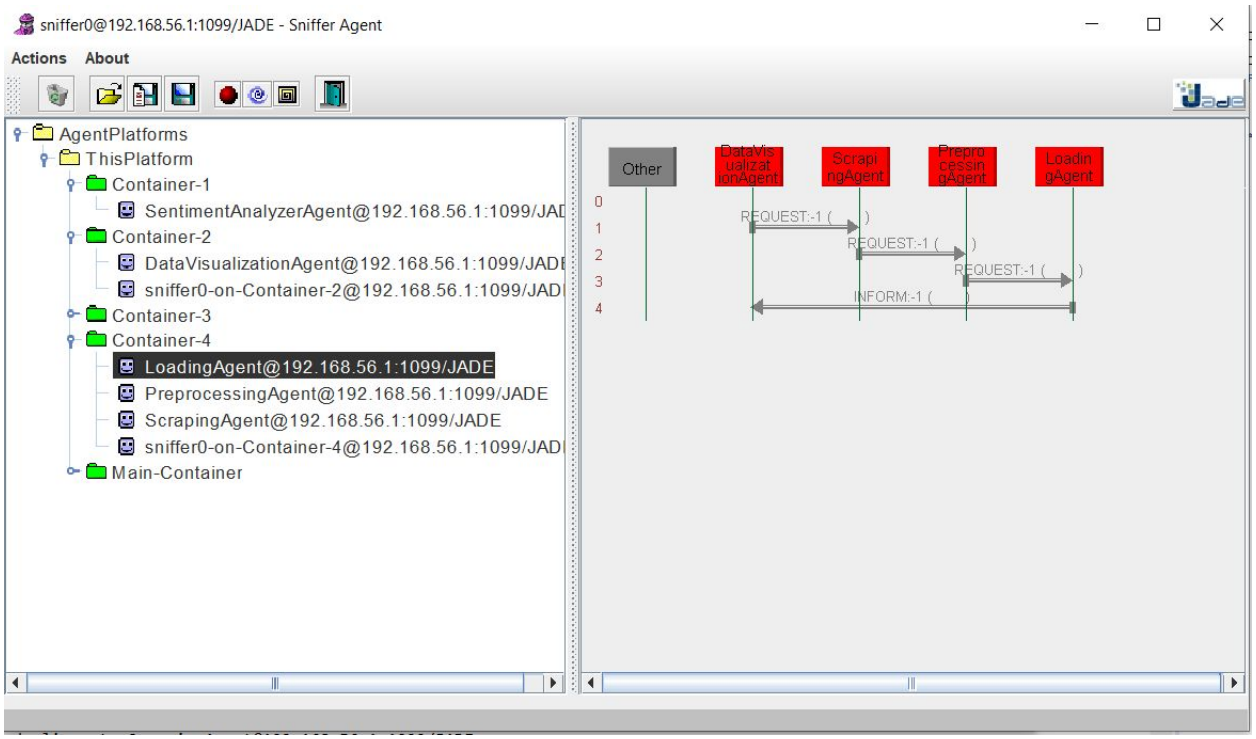
Cet agent lance des requêtes de chargement selon le type de données et envoie une requête INFORM à l'agent approprié.

PredictionAgent



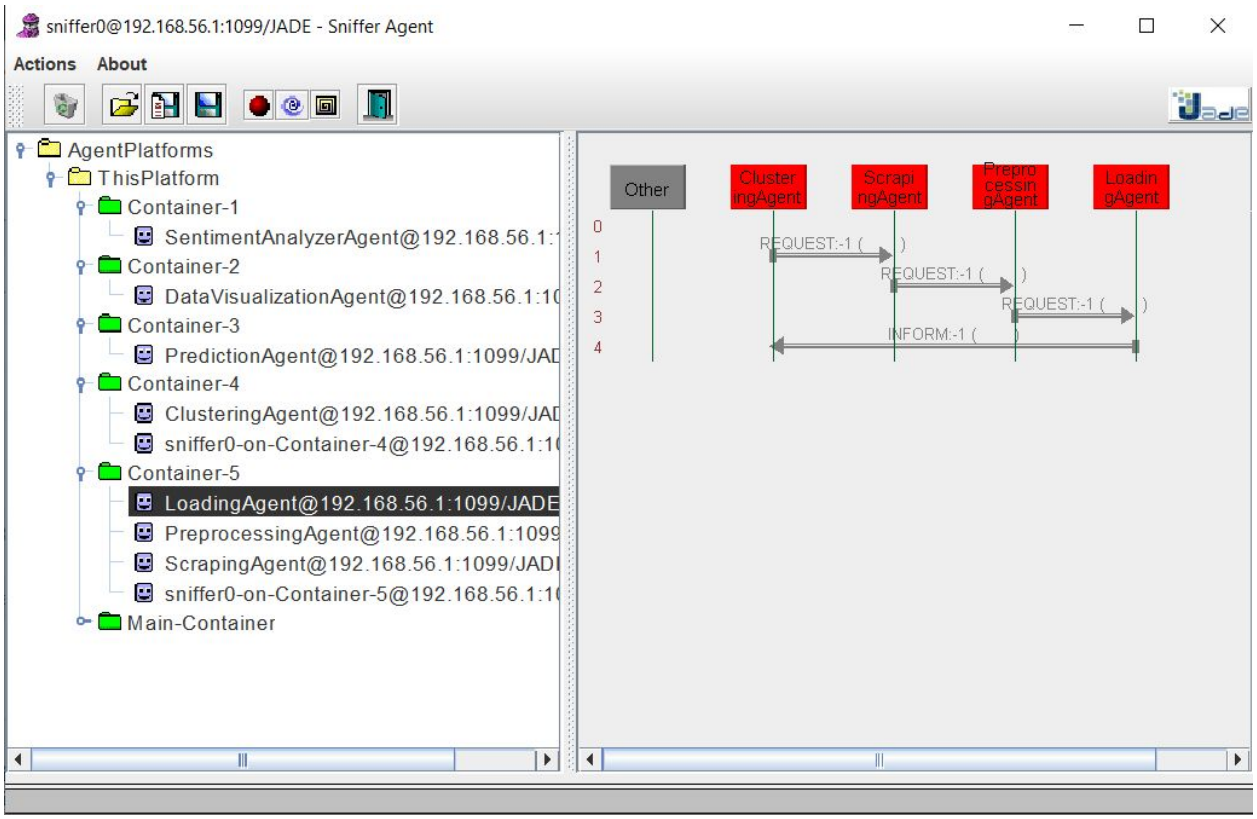
La même manière de travail que celle de l'agent SentimentAnalyzerAgent.

DataVisualizationAgent



La même manière de travail que celle de l'agent SentimentAnalyzerAgent.

ClusteringAgent



La même manière de travail que celle de l'agent SentimentAnalyzerAgent.

Conclusion

Cette section du rapport a présenté la partie du développement Backend du projet, c'est la partie du code qui est exécutée par le serveur, il s'agit du aussi de la partie des systèmes multi agents .

Ainsi on a expliqué dans cette partie le choix de technologies utilisés et ses fonctionnements.

Chapitre 3 : La partie Frontend

Introduction

La partie frontend est la partie du code qui est reçue par le client. Le client c'est notre navigateur Web. Il s'agit des éléments du site web que l'on aperçoit à l'écran et avec lesquels on pourra interagir. Ces éléments sont développés en utilisant trois langages ; HTML, CSS et Javascript.

Le Navigateur web

Notre navigateur est l'outil qui va nous permettre de voir notre application. Les navigateurs les plus connus sont : Chrome, Firefox, Safari, Internet Explorer, etc. On peut le voir comme une sorte de traducteur, c'est-à-dire qu'il va recevoir du code et nous le montrer sous forme visuelle, il va afficher nos pages Web. D'ailleurs, ils n'ont pas tous la même façon de traduire ce code et lorsque l'on codera, on devra faire attention aux spécificités de certains navigateurs web.

Le code client que votre navigateur peut comprendre est composé de 3 langages différents : HTML, CSS et Javascript.

HTML (Hyper Text Markup Language)

HTML – Hyper Text Markup Language – est un langage composé de tags, balises en Français. Il va nous permettre de représenter la structure, le squelette de nos pages Web. Par exemple : un titre, un paragraphe, une image ou une liste.

CSS (Cascading Styles Sheets)

Le langage CSS – Cascading Styles Sheets – est un langage qui va mettre en forme nos pages Web et les décorer. Il va désigner nos éléments HTML à l'aide de sélecteurs et va leur appliquer un style CSS. C'est ce langage CSS qui est responsable des couleurs, des tailles, de la mise en page, etc.

Javascript

Un site Web peut être composé uniquement d'HTML et de CSS, mais si on veut lui insuffler un peu de vie on aura besoin de Javascript, qui lui, est un vrai langage de programmation, avec des boucles, des conditions…II sera responsable de l'interactivité et de la logique qu'il y a derriere nos pages web. Par exemple, si on veut ouvrir un menu en cliquant sur un bouton particulier, on le fera avec du Javascript.

Frameworks

Bootstrap

Il y a aussi Bootstrap qui est un framework HTML, CSS et Javascript, c'est-à-dire une structure qui contient de nombreux composants prêts à l'emploi: boutons, listes déroulantes, menus, etc.

Angular

Angular est un framework JavaScript Open Source basé sur Typescript et développé par Google. Il utilise l'architecture MVM (Modèle Vue Modèle), proche du modèle MVC. Cela va permettre de structurer le code et bien séparer la vue (l'interface) des modèles (fonctionnement).

Il est considéré comme un langage « côté client », ceux-ci permettent de gérer l'interface utilisateur de chaque page (affichage, interactions…) de façon dynamique et viennent en complément aux langages côté serveur.

En général, Angular récupère des données du back-end, les affiche à l'utilisateur via interface contenant des boutons, des formulaires… et ces boutons et ces formulaires permettent de déclencher des modifications côté back-end. On pourrait donc dire que Angular fait l'intermédiaire entre l'utilisateur et le back-end.

Angular Materials

Material est un module d'Angular, c'est un module d'intégration qui permet d'obtenir facilement une interface responsive harmonieuse et unie dans le style 'flat' de Google. La documentation de Angular Material se trouve ici material.angularjs.org/latestest et elle est très bien faite.

Charts.js

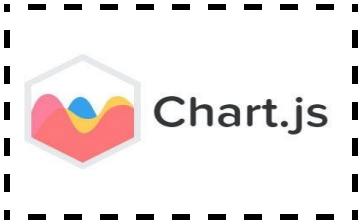


Chart.js, la bibliothèque JavaScript de représentation des données sous forme de graphes statistiques vient d'être publiée sur le site officiel de son créateur Nick Downie. D'une utilisation très simple avec plusieurs options de personnalisation, l'outil offre une prise en charge du HTML5 et ne nécessite pas de dépendance tierce.

Mise en œuvre

La partie front end est divisée en 4 sous-parties selon les membres du groupe :

- Partie Visualisation
- Partie Clusters

- Partie Prédiction
- Partie Scrapping

Partie Visualisation

Cette partie permettra la visualisation totale des données en se basant sur des graphes.

Partie Scrapping

Après un Scrapping des news, on peut vérifier si les news sont positifs ou bien négatifs en utilisant l'analyse des sentiments.

On peut aussi entrer une chaîne de caractères et voir si est-elle dans le sens positif ou bien négatif.

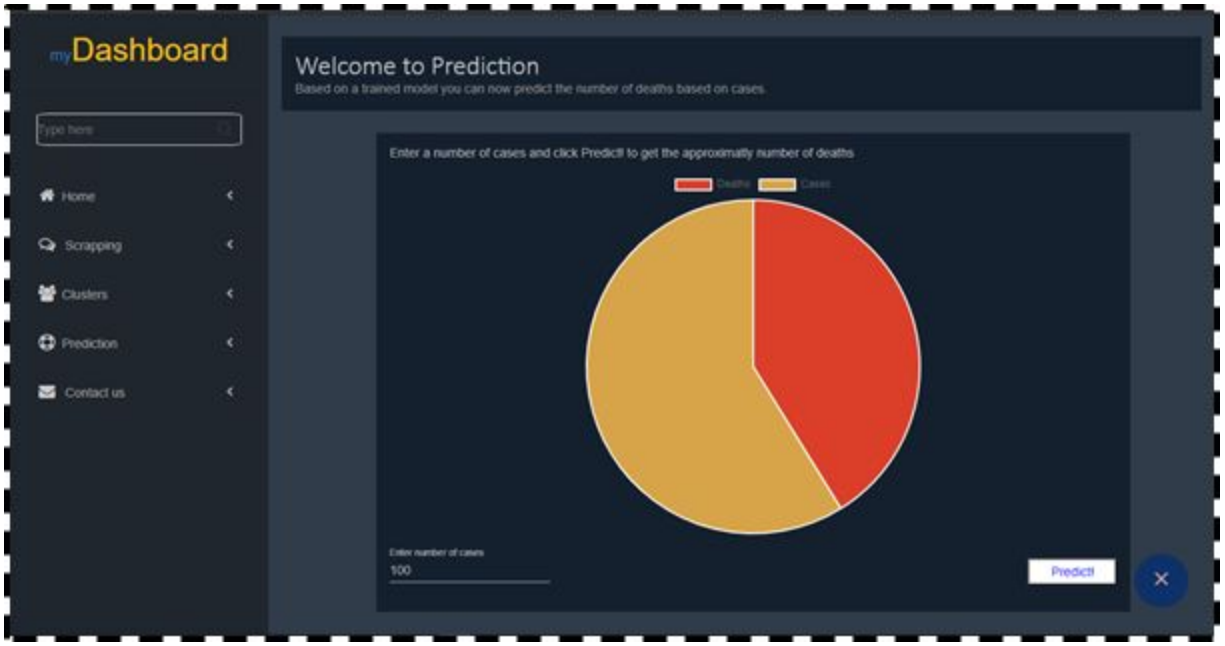
Partie Clusters

Après un clustering des régions du Maroc, on projette leurs statistiques des cas, déçus et rétablies.

D'autre part on affiche les villes qui ont un nombre de déçus et cas en commun se forme d'un graphe de clusters.

Partie Prédiction

Pour cette partie, on peut savoir approximativement le nombre des déçus en se basant sur le nombre des cas entré en paramètre.



Conclusion

Ce chapitre de projet s'intéresse à la réalisation de la partie Frontend, c'est l'ensemble des interfaces que l'utilisateur peut, en utilisant le Framework Angular

qui est basé sur le modèle MVC qui donne lui-même une architecture simplifiée et bien organisée.

Conclusion générale

Dans ce projet, nous avons essayé d'apprendre beaucoup de choses en science des données et en particulier comment prétraiter les données, de les transformer, de les charger de manière utile qui nous aidera à faire des analyses, nous avons également appris à faire des système multi agents dans des environnements compliquées, ainsi on a appliquer et déployer des algorithmes ML en combinaison avec un système multiagent.