

English sentiment Analysis

Acquisition de Compétences Professionnelles (ACP)

ETMANOUKTY ISMAIL

EL MAKOUL EL MEHDI

Master 1 BDMA

Promotion 2023-2024

Table des matières

I.	Introduction	3
II.	Collection de données	3
III.	Prétraitement de texte.....	3
IV.	Modèles de classification.....	4
1.	Random Forest Classifier	4
2.	Naïve Bayes classifieur	5
3.	Regression logistique	5
V.	Réalisation	6
1)	Transformer le classificateur en application Web :.....	6
2)	Création de notre application web :	6
3)	Lancement d'applications :	7
4)	Description de l'application :.....	8
VI.	Conclusion	9
VII.	ANNEXE	9

I.Introduction

Ce rapport présente le travail que nous avons réalisé dans le cadre de l'Acquisition de Compétences Professionnelles (ACP). Notre projet consiste en la conception et le développement d'une application web permettant de classifier des textes en fonction du sentiment ressenti.

Le défi auquel nous avons été confrontés était nouveau pour nous. Il a fallu acquérir une maîtrise des concepts et des techniques de traitement du langage naturel (NLP), notamment pour analyser et comprendre les textes des utilisateurs.

II.Collection de données

L'ensemble de données utilisé dans ce mini-projet est composé de 27 481 tweets en anglais, étiquetés comme positifs, négatifs et neutres. Il est disponible sous forme de fichier Excel et se compose de trois colonnes et 27 482 lignes. Pour télécharger la base de données, utilisez le lien suivant :

<https://github.com/Mehdimakoul/ACP/blob/main/Tweets.csv>.

```
[25]
output = '/content/Tweets.csv'

df = pd.read_csv(output, encoding='utf-8', sep=',')

df.head()
```

	textID	text	sentiment
0	cb774db0d1	I'd have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	negative
2	088c60f138	my boss is bullying me...	negative
3	9642c003ef	what interview! leave me alone	negative
4	358bd9e861	Sons of ****, why couldn't they put them on t...	negative

III.Prétraitement de texte

Dans un premier traitement effectué sur la base de données, nous avons supprimé les sentiments neutres pour nous concentrer uniquement sur les sentiments négatifs et positifs. Les mots vides sont des mots très courants dans une langue. Ces mots n'ont pas beaucoup de sens. Ils ne sont donc pas très utiles pour l'analyse des sentiments. Par exemple, considérons la phrase suivante :

"I love this movie. It is so great!"

Les mots "I", "this", "it" et "So" sont des mots vides. Ils ne sont donc pas très utiles pour l'analyse des sentiments. Nous allons donc les supprimer des phrases.

Ensuite, nous avons nettoyé le texte en le mettant en minuscules, en supprimant la ponctuation et les chiffres, et en éliminant les mots vides.

```

import string

def preprocess(txt):
    tokens = txt.split()
    txt = ' '.join(tokens)

    # Remove punctuation
    txt = txt.translate(str.maketrans('', '', string.punctuation))

    # Tokenize again after removing punctuation
    tokens = txt.split()

    # Assuming 'stopwords' is a predefined list of stopwords
    tokens = [token for token in tokens if token not in stopwords]

    return ' '.join(tokens)

original_txt = df['text'][50]
processed_txt = preprocess(df['text'][50])
print(f'The original text was:\n{original_txt}\n-----\n The preprocessed text is: \n{processed_txt}')

```

The original text was:
Well what im working on isn't QUITE ready to post about publicly (still beta testing) but its a cool new script I coded

The preprocessed text is:
Well im working isnt QUITE ready post publicly beta testing cool new script I coded

IV. Modèles de classification

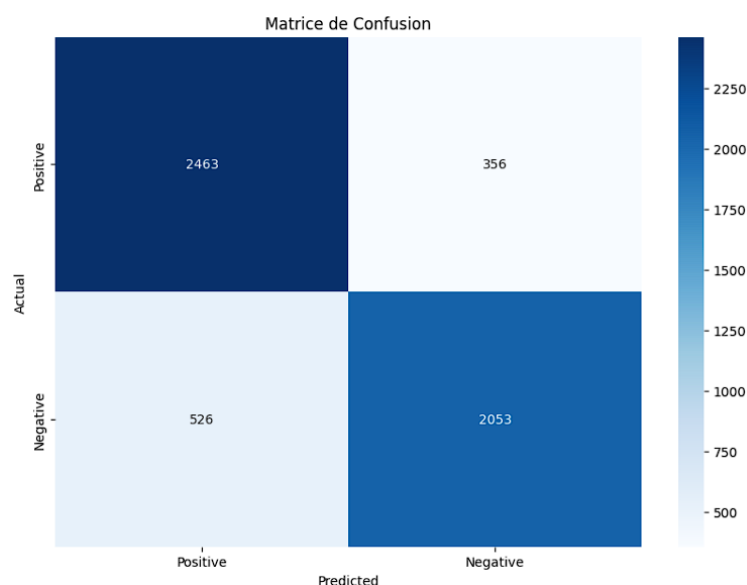
Dans cette section, nous présentons les trois modèles de classification appliqués à notre jeu de données prétraité et pour évaluer les performances du chaque modèle, nous avons utilisé les métriques suivantes : accuracy , rapport de classification, et matrice de confusion.

1. Random Forest Classifier

Accuracy: 0.8366061504260838

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.87	0.85	2819
1	0.85	0.80	0.82	2579
accuracy			0.84	5398
macro avg	0.84	0.83	0.84	5398
weighted avg	0.84	0.84	0.84	5398

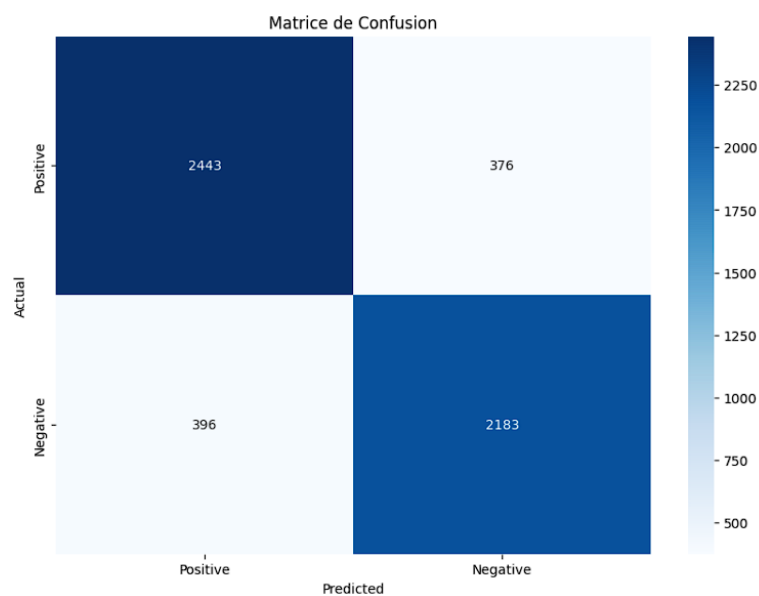


2. Naïve Bayes classifieur

Accuracy: 0.8569840681733976

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.87	0.86	2819
1	0.85	0.85	0.85	2579
accuracy			0.86	5398
macro avg	0.86	0.86	0.86	5398
weighted avg	0.86	0.86	0.86	5398

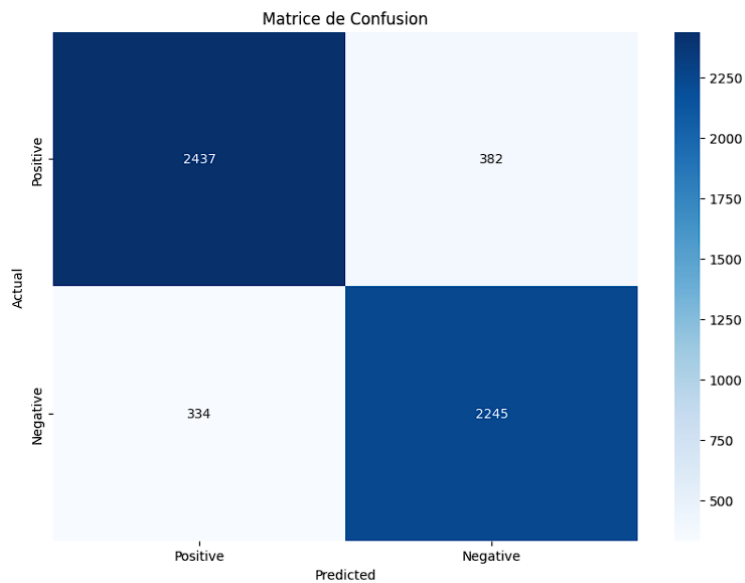


3. Regression logistique

Accuracy: 0.8673582808447573

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.86	0.87	2819
1	0.85	0.87	0.86	2579
accuracy			0.87	5398
macro avg	0.87	0.87	0.87	5398
weighted avg	0.87	0.87	0.87	5398



V.Réalisation

Dans cette section, nous présenterons un aperçu des différentes étapes de développement du backend de notre chatbot, ainsi que le frontend du projet

1) Transformer le classificateur en application Web :

Nous avons utilisé Flask, un framework open-source de développement web en Python. Flask vise à maintenir un noyau simple mais extensible. Il ne comprend pas de système d'authentification, de couche d'abstraction de base de données, ni d'outil de validation de formulaires.



2) Création de notre application web :

Le fichier App.py contient le code principal exécuté par l'interprète Python pour lancer l'application web. Ce fichier inclut également le code de machine learning pour classifier le texte à l'aide de **la classification de Bayes**.

App.py :

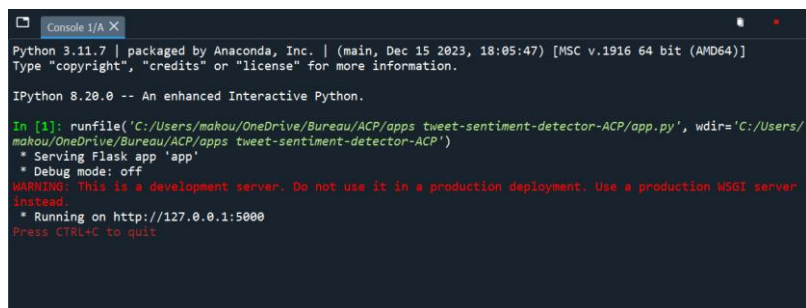
```

from flask import Flask,render_template,url_for,request
import pandas as pd
import pickle
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
app=Flask(__name__)
@app.route('/predict',methods=['POST'])
def predict():
    df = pd.read_excel('TweetsFinal.xlsx')
    df['label'] = df['sentiment'].map({'positive': 0, 'negative': 1})
    X = df["Feed"]
    y = df["label"]
    cv = CountVectorizer()
    X = cv.fit_transform(X) # Fit the Data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
    clf = MultinomialNB()
    clf.fit(X_train,y_train)
    clf.score(X_test,y_test)
    #Alternative Usage of Saved Model
    # joblib.dump(clf, 'NB_spam_model.pkl')
    # NB_spam_model = open('NB_spam_model.pkl','rb')
    # clf = joblib.load(NB_spam_model)
    if request.method=='POST':
        v2=request.form['v2']
        data=[v2]
        vect=cv.transform(data).toarray()
        my_prediction=clf.predict(vect)
        return render_template('result.html',prediction=my_prediction)
@app.route('/')
def home():
    return render_template('home.html')
if __name__=='__main__':
    app.run(debug=False)

```

3) Lancement d'applications :

Pour lancer l'application, nous avons travaillé Spyder. Au préalable, nous avons installé les bibliothèques nécessaires, comme Flask, via Anaconda Prompt.



```

Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.20.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/makou/OneDrive/Bureau/ACP/apps tweet-sentiment-detector-ACP/app.py', wdir='C:/Users/
makou/OneDrive/Bureau/ACP/apps tweet-sentiment-detector-ACP')
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server
instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit

```

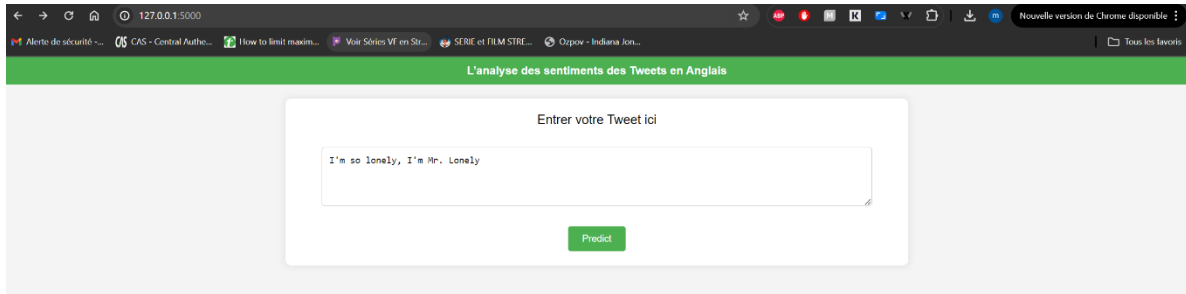
4) Description de l'application :

Maintenant, nous pouvons ouvrir un navigateur Web et naviguer vers <http://127.0.0.1:5000/>,
Nous allons avoir un site web simple comme suit :

Exemple 1:

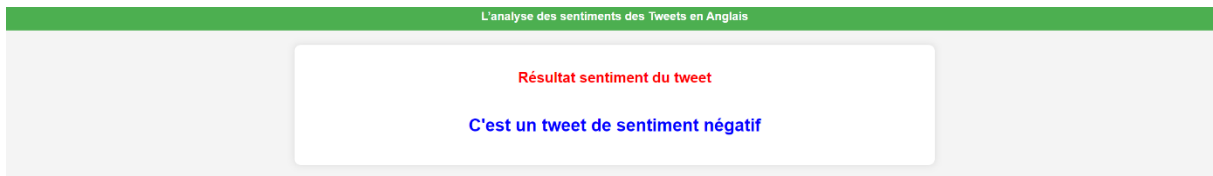
On prends par exemple “I'm so lonely, I'm Mr. Lonely”.

Home.html



The screenshot shows a web browser window with the address bar at 127.0.0.1:5000. The page has a green header with the text "L'analyse des sentiments des Tweets en Anglais". Below the header is a white box with the title "Entrer votre Tweet ici". Inside this box is a text input field containing the text "I'm so lonely, I'm Mr. Lonely". Below the input field is a green button labeled "Predict".

Result.html



The screenshot shows the same web browser window. The page has a green header with the text "L'analyse des sentiments des Tweets en Anglais". Below the header is a white box with the title "Résultat sentiment du tweet" in red. Below the title is a line of blue text that reads "C'est un tweet de sentiment négatif".

Exemple 2:

On prends par exemple “I'm good, yeah, I'm feelin' alright”.

Home.html



The screenshot shows the same web browser window. The page has a green header with the text "L'analyse des sentiments des Tweets en Anglais". Below the header is a white box with the title "Entrer votre Tweet ici". Inside this box is a text input field containing the text "I'm good, yeah, I'm feelin' alright". Below the input field is a green button labeled "Predict".

Result.html



The screenshot shows the same web browser window. The page has a green header with the text "L'analyse des sentiments des Tweets en Anglais". Below the header is a white box with the title "Résultat sentiment du tweet" in red. Below the title is a line of blue text that reads "C'est un tweet de sentiment positif".

VI. Conclusion

En conclusion, ce projet a permis de développer une application web en suivant un processus rigoureux, depuis la collecte des données jusqu'à la mise en ligne de l'application web. Nous avons exploré plusieurs modèles de classification, dont le Random Forest Classifier, le Naïve Bayes Classifier et la Régression Logistique.

Ce projet nous a permis d'acquérir des compétences pratiques en traitement du langage naturel et en apprentissage automatique. Les résultats obtenus sont prometteurs et ouvrent la voie à des améliorations futures pour rendre cette application encore plus performant et polyvalent.

VII. ANNEXE

Lien vers le GitHub :

<https://github.com/Mehdimakoul/ACP/tree/main>