Devoir-projet :

Applied Data Science

Année universitaire:2023/2024

—

Abdessamad AHCHOUCH
Mounia NAOUA
Naoufal CHABAA
El Mahdi JAMRANI
Achraf MZIMIZ

—

Encadré par : Zakaria KERKAOU

Contents

Introduction :

L'analyse des sentiments, également connue sous le nom de "mining d'opinion", est une technique essentielle dans le domaine de la science des données et du traitement du langage naturel (NLP). Elle permet d'extraire et de quantifier les opinions et les sentiments exprimés dans des textes. Dans le cadre de ce projet, nous allons mettre en pratique ces compétences pour développer un modèle capable de classifier les commentaires du site web Hespress en trois catégories de tonalité : positif, négatif ou neutre.

Contexte :

Les commentaires en ligne, tels que ceux présents sur les articles de presse ou les plateformes de médias sociaux, représentent une source précieuse d'opinions et de feedbacks des utilisateurs. Analyser ces commentaires peut fournir des insights significatifs pour diverses applications, comme améliorer le service client, surveiller la réputation en ligne, ou encore comprendre les besoins et les attentes des lecteurs. Hespress, étant l'un des sites d'information les plus visités au Maroc, offre une multitude de commentaires variés qui sont idéaux pour une telle analyse.

Problématique :

Avec l'augmentation exponentielle du volume de commentaires en ligne, il devient crucial pour les entreprises et les organisations de comprendre rapidement et efficacement les opinions exprimées par les utilisateurs. Les sites d'information, tels que Hespress, reçoivent quotidiennement des milliers de commentaires sur leurs articles. La tâche manuelle de tri et de classification de ces commentaires selon leur tonalité est non seulement fastidieuse, mais aussi imprécise et non scalable. La problématique est donc la suivante : **comment automatiser de manière efficace et précise la classification des commentaires en ligne en fonction de leur tonalité (positive, négative ou neutre) ?**

Description du Projet :

Le projet consiste à développer un modèle d'analyse de sentiment capable de classifier automatiquement les commentaires du site web Hespress. Le processus comprend plusieurs étapes clés :

1. **Collecte des Données:**
    - Télécharger ou scraper un ensemble de données de commentaires à partir du site Hespress.
    - Stocker ces données dans un format compatible avec Pandas pour une manipulation facile.

2. **Prétraitement des Données :**
    - Nettoyer les données en supprimant les caractères spéciaux, les stopwords (mots vides) et en normalisant le texte.
    - Utiliser Pandas pour effectuer ces opérations de nettoyage et préparer les données pour l'analyse.

3. **Exploration des Données :**
    - Utiliser des outils de visualisation comme Matplotlib et Seaborn pour analyser la distribution des sentiments dans les commentaires.
    - Identifier les mots les plus fréquents et les relations entre les différents mots et sentiments.

4. **Entraînement du Modèle :**
    - Utiliser NLTK pour tokeniser le texte et construire un vocabulaire.
    - Entraîner un modèle de machine learning, comme un classificateur Naive Bayes, en utilisant Scikit-learn.

5. **Évaluation du Modèle :**
    - Évaluer les performances du modèle en utilisant des métriques telles que la précision, le rappel et le score F1 sur un ensemble de données de test.

6. **Application du Modèle :**
    - Déployer le modèle pour prédire les sentiments des nouveaux commentaires de produits ou d'articles.

Fonctionnement :

Le fonctionnement du projet peut être divisé en plusieurs phases détaillées ci-dessous :

1. Collecte des Données

- Scraping : Utiliser des outils comme BeautifulSoup ou Scrapy pour extraire les commentaires du site Hespress.
- Stockage: Sauvegarder les commentaires extraits dans un fichier CSV ou une base de données compatible avec Pandas.

2. Prétraitement des Données

- Nettoyage du Texte: Utiliser Pandas pour supprimer les caractères spéciaux, les stopwords et normaliser le texte (mise en minuscule, suppression des ponctuations, etc.).
- Tokenisation : Utiliser NLTK pour diviser le texte en tokens (mots individuels).

3. Exploration des Données

- Analyse Exploratoire: Utiliser Matplotlib et Seaborn pour créer des graphiques et des visualisations qui montrent la distribution des sentiments, les mots les plus fréquents, etc.
- Statistiques Descriptives : Utiliser Pandas pour calculer des statistiques descriptives sur les données.

4. Entraînement du Modèle

- Vectorisation : Convertir le texte tokenisé en vecteurs de caractéristiques en utilisant des techniques comme TF-IDF (Term Frequency-Inverse Document Frequency).
- Séparation des Données : Diviser les données en ensembles d'entraînement et de test.
- Entraînement : Utiliser Scikit-learn pour entraîner un modèle de machine learning, par exemple, un classificateur Naive Bayes.

5. Évaluation du Modèle

- Prédictions : Utiliser le modèle pour prédire les sentiments des commentaires dans l'ensemble de test.
- Métriques de Performance : Calculer des métriques telles que la précision, le rappel et le score F1 pour évaluer les performances du modèle.

6. Application du Modèle

- Déploiement : Déployer le modèle entraîné pour analyser les sentiments des nouveaux commentaires en temps réel.

En suivant ces étapes, nous pourrons développer un système automatisé de classification des sentiments des commentaires, capable de traiter de grands volumes de données de manière efficace et précise.

Réalisation :

Collecte de données :

Scraper les commentaires d'un article du site web Hespress, extraire les votes associés à chaque commentaire, trier les commentaires par nombre de votes positifs, et enfin stocker les données dans un DataFrame Pandas pour une analyse ultérieure.

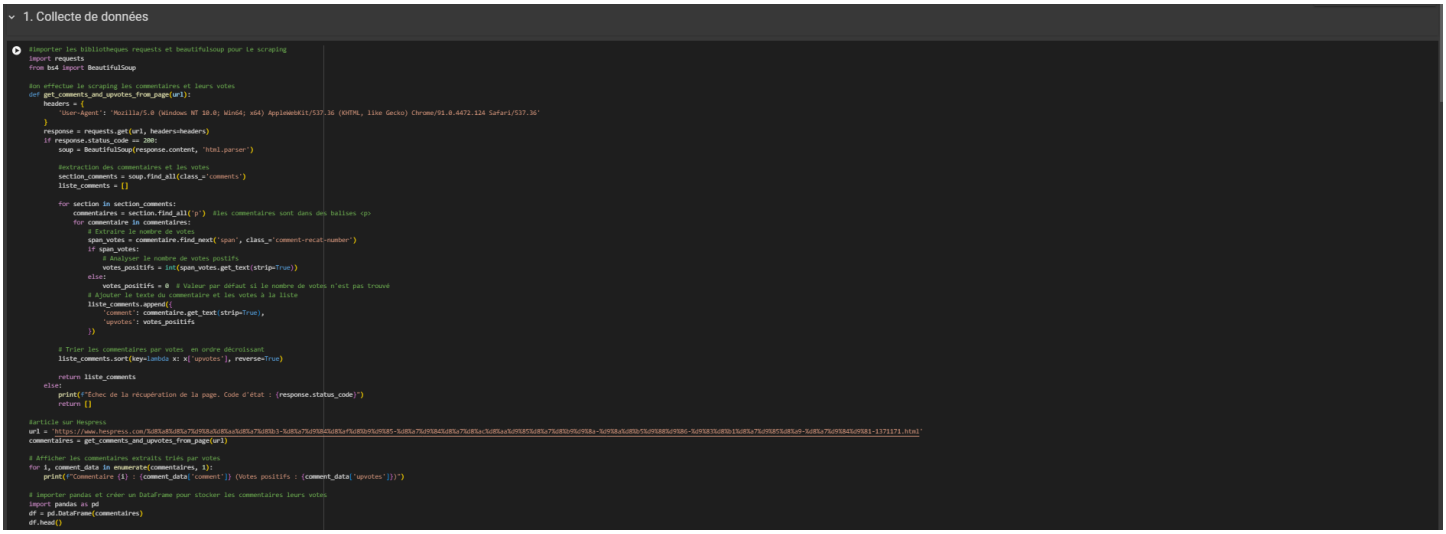

Figure 1: Code collecte de données

	comment	upvotes
0	...ذكرتني بوزيرة حينما قالت إن من له مدخول 20 دره	231
1	...نعم داشي لي غادي ياخدوه فالدعم غادي يخسروه فال	208
2	...عطيتوا للناس دعم شهر وتم توقيفه اشمن دعم هذا ك	182
3	.....الدعم وارتفاع الأسعار لا يلتقيان. اسي الوزير	159
4	... ونسي المتقاعدون الذين اصبحوا من افقر الفئات..	144

Figure 2: Résultat collecte de données

Prétraitement des données :

Prétraiter des commentaires en langue arabe en vue d'une analyse ultérieure. Ce processus de prétraitement inclut la tokenisation du texte, la suppression de la ponctuation, et la réassemblage des tokens en texte.

2. Prétraitement des données

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import ISRIStemmer
import string  # Ajouter cette ligne pour utiliser string.punctuation

# Fonction de prétraitement pour le texte arabe
def preprocess_arabic_text(text):
    # Tokenisation du texte: Le texte est divisé en motsen utulisant fct
    tokens = word_tokenize(text)

    # Suppression de la ponctuation: Tous les mots qui sont des caractères de ponctuation, y compris le point, sont supprimés.
    tokens = [word for word in tokens if word not in string.punctuation]

    # Réassemblage des tokens en texte: Les mots restants sont réassemblés en une chaîne de texte
    preprocessed_text = ' '.join(tokens)
    return preprocessed_text
# Prétraiter les commentaires:La fonction de prétraitement est appliquée à la colonne des commentaires d'un DataFrame df, et les résultats sont stockés dans une nouvelle colonne appel
df['preprocessed_comment'] = df['comment'].apply(preprocess_arabic_text)

# Afficher le DataFrame prétraité
print(df.head())
```

Figure 3:Code prétraitement des données

```
                                                comment  upvotes  \
0  230       ذكرتني بوزيرة حينما قالت إن من له مدخول 20 دره...
1  207       نعم داشي لي غادي ياخدوه فالدعم غادي يخسروه فال...
2  181       عطيتوا للناس دعم شهر وتم توقيفه اشمن دعم هذا ك...
3    158       الدعم وارتفاع الأسعار لا يلتقيان. اسي الوزير.....
4  ..143       ونسي المتقاعدون الذين اصبحوا من افقر الفئات ...

                                  preprocessed_comment
0  ذكرتني بوزيرة حينما قالت إن من له مدخول 20 دره...
1  نعم داشي لي غادي ياخدوه فالدعم غادي يخسروه فال...
2  عطيتوا للناس دعم شهر وتم توقيفه اشمن دعم هذا ك...
3      الدعم وارتفاع الأسعار لا يلتقيان اسي الوزير.....
4  .. ونسي المتقاعدون الذين اصبحوا من افقر الفئات...
```

Figure 4:Résultat prétraitement des données

Exploration des données :

Effectuer une exploration et une visualisation des données sur les commentaires d'un article, en se basant sur le nombre de votes positifs pour déterminer le sentiment (négatif, neutre, ou positif). Il crée des visualisations pour illustrer la distribution des sentiments et les mots les plus fréquents associés à chaque sentiment.

3. Exploration des données

```python
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

# Défini les plages d'upvotes
bins = [-float('inf'), -10, 10, float('inf')]
labels = ['négatif', 'neutre', 'positif']

# Crée une nouvelle colonne 'sentiment' basée sur les plages d'upvotes
df['sentiment'] = pd.cut(df['upvotes'], bins=bins, labels=labels)

# Compte le nombre de commentaires dans chaque plage de sentiment et les classer selon les indexes
sentiment_counts = df['sentiment'].value_counts().sort_index()

# Crée un graphe montrant la fréquence des commentaires selon les sentiments
plt.figure(figsize=(18, 6))

plt.subplot(1, 2, 1)
sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values)
plt.xlabel('Sentiment')
plt.ylabel('Nombre de Commentaires')
plt.title('Fréquence des Commentaires selon les Sentiments')
plt.gca().spines[['top', 'right']].set_visible(False) # Cache les bordures supérieures et droites du graphique

# Calcule les mots les plus fréquents selon les sentiments
top_words_per_sentiment = {}
for sentiment in labels:
    # Filtrer les commentaires, Joindre les commentaires et Séparer en une liste de mots individuels
    words = ' '.join(df[df['sentiment'] == sentiment]['preprocessed_comment']).split()
    word_counts = Counter(words)
    top_words_per_sentiment[sentiment] = dict(word_counts.most_common(10))

# Crée un graphe montrant les 10 mots les plus fréquents pour chaque sentiment
plt.subplot(1, 2, 2)
for sentiment, top_words in top_words_per_sentiment.items():
    sns.barplot(x=list(top_words.keys()), y=list(top_words.values()), label=sentiment)
plt.xlabel('Mots')
plt.ylabel('Fréquence')
plt.title('Les 10 mots les plus fréquents par Sentiment')
# Rotation des étiquettes sur l'axe des x pour une meilleure lisibilité
plt.xticks(rotation=45)
plt.legend(title='Sentiment')
plt.gca().spines[['top', 'right']].set_visible(False)
# Pour éviter que les graphiques se chevauchent
plt.tight_layout()
```

Figure 5: Code exploration des données

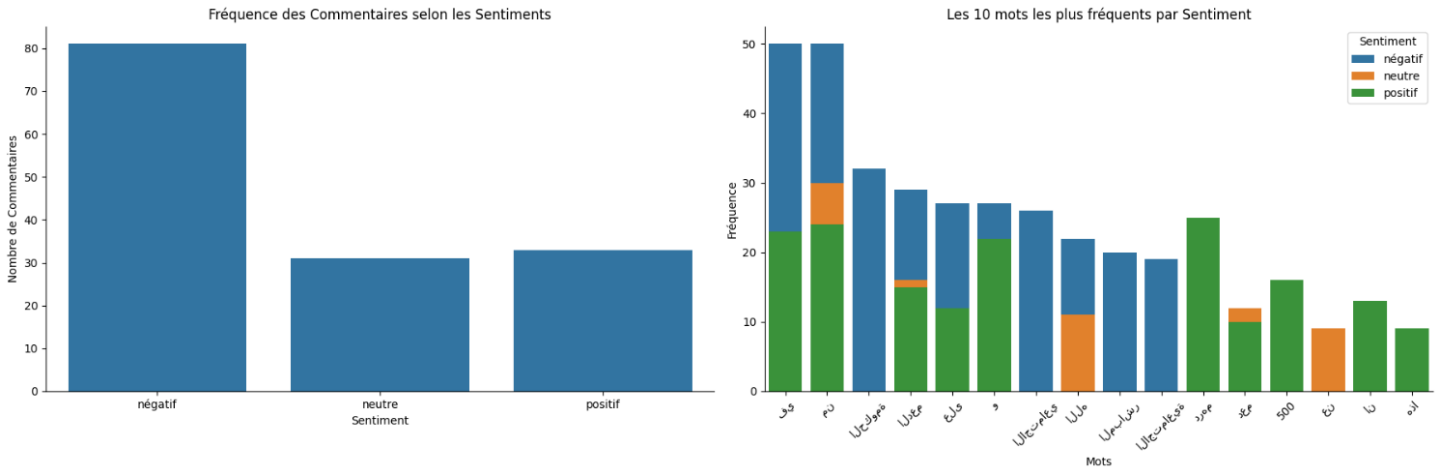

Figure 6:Résultat exploration des données

Entraînement du modèle :

Construire et entraîner un modèle de classification de sentiments en utilisant les commentaires extraits d'un site web.

4. Entraînement du modèle

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from collections import Counter

# Télécharger les ressources NLTK nécessaires
nltk.download('punkt')
nltk.download('stopwords')

# Définir les plages d'upvotes
bins = [-float('inf'), -10, 10, float('inf')]
labels = ['négatif', 'neutre', 'positif']

# Créer une nouvelle colonne 'sentiment' basée sur les plages d'upvotes
df['sentiment'] = pd.cut(df['upvotes'], bins=bins, labels=labels)

# Créer un dictionnaire pour stocker les mots les plus fréquents par sentiment
top_words_per_sentiment = {}

# Calculer les mots les plus fréquents selon les sentiments
for sentiment in labels:
    words = ' '.join(df[df['sentiment'] == sentiment]['filtered_comment']).split()
    word_counts = Counter(words)
    top_words_per_sentiment[sentiment] = [word for word, _ in word_counts.most_common(100)]

# Combinez les mots les plus fréquents de chaque sentiment
top_words = set()
for words in top_words_per_sentiment.values():
    top_words.update(words)

# Créer une nouvelle colonne 'filtered_comment_top' contenant uniquement les mots les plus fréquents par sentiment
df['filtered_comment_top'] = df['filtered_comment'].apply(lambda comment: ' '.join(word for word in word_tokenize(comment) if word in top_words))

# Vectorisation des commentaires en utilisant TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)
X_tfidf = tfidf_vectorizer.fit_transform(df['filtered_comment_top'])  # Utiliser toutes les données

# Entraînement du classificateur Naive Bayes
classifier = MultinomialNB()
classifier.fit(X_tfidf, df['sentiment'])  # Utiliser toutes les étiquettes
```

Figure 7: Code entraînement du modèle

Evaluation du modèle :

Poursuivre le processus d'entraînement d'un modèle de classification des sentiments en évaluant les performances du modèle sur l'ensemble de données utilisé.

5. Evaluation du modèle :

```python
# Prédiction sur l'ensemble de données
y_pred = classifier.predict(X_tfidf)

# Évaluation du modèle
accuracy = accuracy_score(df['sentiment'], y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(df['sentiment'], y_pred))
```

Figure 8: Code évaluation du modèle

```
Accuracy: 0.8482758620689655
Classification Report:
              precision    recall  f1-score   support

      neutre       1.00      0.52      0.68        31
     négatif       0.80      0.99      0.88        81
     positif       0.93      0.82      0.87        33

    accuracy                           0.85       145
   macro avg       0.91      0.77      0.81       145
weighted avg       0.87      0.85      0.84       145
```

Figure 9: Résultat évaluation du modèle

Application du modèle :

Ajouter une fonctionnalité de prédiction pour de nouveaux commentaires.

```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Liste des nouveaux commentaires à prédire
new_comments = ["الحكومة ماضية في الاتجاه  ", " 500 اعيش بكرامة", "كفاكم ان تغطو الشمس بالغربالو اعطوا الناس الفقراء "]

# Prétraitement des nouveaux commentaires
preprocessed_new_comments = [preprocess_arabic_text(comment) for comment in new_comments]

# Vectorisation des nouveaux commentaires
X_new_comments_tfidf = tfidf_vectorizer.transform(preprocessed_new_comments)

# Prédiction des upvotes des nouveaux commentaires
new_comments_upvotes = classifier.predict(X_new_comments_tfidf)

# Affichage des prédictions
for comment, upvote in zip(new_comments, new_comments_upvotes):
    print(f"Commentaire : {comment} | Upvotes prédits : {upvote}")
```

Figure 10: Code application du modèle

```
Commentaire : الحكومة ماضية في الاتجاه    | Upvotes prédits : négatif
Commentaire :  500 اعيش بكرامة | Upvotes prédits : positif
Commentaire : كفاكم ان تغطو الشمس بالغربالو اعطوا الناس الفقراء  | Upvotes prédits : neutre
```

Figure 11:Résultat application du modèle

Conclusion :

Ce projet d'analyse de sentiment des commentaires sur le site web Hespress a permis de mettre en œuvre diverses compétences en data science, traitement du langage naturel et apprentissage automatique. En utilisant des bibliothèques telles que Pandas, NLTK et Scikit-learn, nous avons réussi à créer un modèle de classification capable de prédire la tonalité des commentaires (positif, neutre ou négatif) de manière automatisée et précise.

Référence :

https://colab.research.google.com/drive/1_A7zfikzhU7sifbWHdHIDX_BUDTr6w3_?usp=sharing&authuser=4