

Mémoire de Projet d'ingénierie

Filière : Data engineer

Text Mining Project

Detection du language

Réalisé par

HALMAOUI Hajar

Encadré par :

ELASRI Ikram

Année universitaire :2023/2024

Table de matière

Mémoire de Projet d'ingénierie	1
Filière : Data engineer	1
Introduction.....	4
1. Sélection et compréhension des données.....	4
(i) Sources de Données	4
(ii) Fichiers de Données	4
(iii) Chargement des Données.....	4
2. Compréhension des Données.....	5
(i) Forme des Données :.....	5
(ii) Premières Lignes :.....	5
3. Prétraitement des Données.....	6
i) Suppression des Caractères de Ponctuation.....	6
ii) Prétraitement des Données en Anglais	6
iii) Prétraitement des Données en Tchèque	7
iv) Prétraitement des Données en Polonais	7
v) Transformation des Données en un Seul Ensemble.....	8
vi) Visualisation de la Répartition des Langues	9
4. Division des Données	9
i) Séparation des Variables Indépendantes et Dépendantes	10
ii) Division des Données en Ensembles d'Entraînement et de Test	10
iii) Application du Vectoriseur TF-IDF.....	10
iv) Préparation des Pipelines de Modélisation	11
❖ Pipeline pour MultinomialNB	11
❖ Pipeline pour Logistic Regression	11
v) Évaluation des Modèles	11
• Régression Logistique.....	11
• Multinomial Naive Bayes	12
• Matrice de Confusion.....	12
Conclusion	12
5. Sauvegarde du Modèle Entraîné	13
6. Deployment du modèle	13

(i) Chargement du Modèle Entraîné	13
(ii) Interface Utilisateur avec Streamlit	13
(iii) Prédiction avec le Modèle :	14
(iv) Résultat	14
Conclusion	16

Introduction

Ce projet vise à construire un modèle d'apprentissage automatique capable d'identifier la langue d'un texte donné. Les langues prises en compte sont l'anglais, le tchèque et le polonais. L'ensemble de données utilisé pour entraîner et évaluer le modèle est constitué de phrases du corpus Europarl, un corpus de textes parallèles qui fournit des données multilingues.

1. Sélection et compréhension des données

L'objectif de ce projet est de construire un modèle d'apprentissage automatique capable d'identifier la langue d'un texte donné, en se concentrant sur l'anglais, le tchèque et le polonais. Cette section décrit la sélection des sources de données et fournit une compréhension des ensembles de données choisis.

(i) Sources de Données

Les ensembles de données proviennent du corpus Europarl, qui est un corpus parallèle multilingue. Le corpus Europarl contient les débats du Parlement européen, ce qui en fait une source riche de données textuelles alignées en plusieurs langues.

(ii) Fichiers de Données

- ❖ **Données en anglais** : europarl-v7.bg-en.en
- ❖ **Données en tchèque** : europarl-v7.cs-en.cs
- ❖ **Données en polonais** : europarl-v7.pl-en.pl

Ces fichiers contiennent des phrases en anglais, en tchèque et en polonais respectivement, utilisées pour entraîner et évaluer les modèles d'identification de langue.

(iii) Chargement des Données

Les données des fichiers ont été chargées dans des DataFrames pandas pour une manipulation et un prétraitement plus faciles :

```
# Loading raw english data
english_df = []
with open("C:/Users/PcPack/Downloads/aaaa/europarl-v7.bg-en.en", encoding="utf-8") as file:
    for line in file:
        english_df.append(line.strip())

english_df = pd.DataFrame(english_df, columns=["English"])
english_df.head()
```

```
# Loading raw Czech data
czech_df = []
with open("europarl-v7.cs-en.cs", encoding="utf-8") as file:
    for line in file:
        czech_df.append(line.strip())

czech_df = pd.DataFrame(czech_df, columns=["Czech"])
czech_df.head()
```

```
# Loading raw Czech data
polish_df = []
with open("europarl-v7.pl-en.pl", encoding="utf-8") as file:
    for line in file:
        polish_df.append(line.strip())

polish_df = pd.DataFrame(polish_df, columns=["Polish"])
polish_df.head()
```

2. Compréhension des Données

Après avoir chargé les données, il est crucial de comprendre leur structure et leurs caractéristiques :

- (i) **Forme des Données :** L'inspection de la forme de chaque ensemble de données permet de comprendre le nombre de phrases disponibles dans chaque langue.

```
print(english_df.shape)
print(czech_df.shape)
print(polish_df.shape)
```

```
(406934, 1)
(646605, 1)
(632565, 1)
```

- (ii) **Premières Lignes :** La visualisation des premières lignes de chaque DataFrame donne un aperçu du contenu :

```
english_df.head()
```

Polish

- 0 Działania podjęte w wyniku rezolucji Parlament...
- 1 Składanie dokumentów: patrz protokół
- 2 Oświadczenia pisemne (art. 116 Regulaminu): pa...
- 3 Teksty porozumień przekazane przez Radę: patrz...
- 4 Skład Parlamentu: patrz protokół

```
czech_df.head()
```

Czech

- 0 Následný postup na základě usnesení Parlamentu...
- 1 Předložení dokumentů: viz zápis
- 2 Písemná prohlášení (článek 116 jednacího řádu)...
- 3 Texty smluv dodané Radou: viz zápis
- 4 Složení Parlamentu: viz zápis

```
polish_df.head()
```

Polish

- 0 Działania podjęte w wyniku rezolucji Parlament...
- 1 Składanie dokumentów: patrz protokół
- 2 Oświadczenia pisemne (art. 116 Regulaminu): pa...
- 3 Teksty porozumień przekazane przez Radę: patrz...
- 4 Skład Parlamentu: patrz protokół

3. Prétraitement des Données

Le prétraitement des données est une étape cruciale dans le pipeline d'apprentissage automatique, car il permet de nettoyer et de préparer les données brutes pour l'entraînement des modèles. Voici les étapes détaillées de prétraitement appliquées dans ce projet pour les ensembles de données en anglais, tchèque et polonais.

i) Suppression des Caractères de Ponctuation

Pour commencer, nous avons généré une table de traduction pour supprimer tous les caractères de ponctuation des textes. Cela aide à se concentrer uniquement sur les mots et les caractères pertinents pour l'identification des langues.

```
for char in string.punctuation:
    print(char, end = ' ')
translate_table = dict((ord(char), None) for char in string.punctuation)
```

```
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
```

ii) Prétraitement des Données en Anglais

Nous avons ensuite nettoyé les données en anglais en appliquant plusieurs étapes :

- ❖ Conversion en minuscules : Pour éviter la distinction entre majuscules et minuscules.
- ❖ Suppression des chiffres : Les chiffres ne contribuent pas à l'identification des langues.
- ❖ Suppression de la ponctuation : Utilisation de la table de traduction créée précédemment.

```
# Cleaning the data for english dataset

data_eng = []
lang_eng = []

for i, line in english_df.iterrows():
    line = line['English']
    if len(line) != 0:
        line = line.lower()
        line = re.sub(r"\d+", "", line)
        line = line.translate(translate_table)
        data_eng.append(line)
        lang_eng.append("English")
```

iii) Prétraitement des Données en Tchèque

Les mêmes étapes de prétraitement ont été appliquées aux données en tchèque :

```
# Cleaning the data for Czech dataset

data_czech = []
lang_czech = []

for i, line in czech_df.iterrows():
    line = line['Czech']
    if len(line) != 0:
        line = line.lower()
        line = re.sub(r"\d+", "", line)
        line = line.translate(translate_table)
        data_czech.append(line)
        lang_czech.append("Czech")
```

iv) Prétraitement des Données en Polonais

Enfin, nous avons appliqué les mêmes étapes aux données en polonais :

```
# Cleaning the data for Czech dataset

data_pol = []
lang_pol = []

for i, line in pol_df.iterrows():
    line = line['Polish']
    if len(line) != 0:
        line = line.lower()
        line = re.sub(r"\d+", "", line)
        line = line.translate(translate_table)
        data_pol.append(line)
        lang_pol.append("Polish")
```

v) Transformation des Données en un Seul Ensemble

Après le prétraitement, nous avons combiné les données des trois langues en un seul DataFrame, ce qui nous permet de traiter l'ensemble des textes et de leurs étiquettes de langue respectives de manière uniforme.

```
df = pd.DataFrame({
    "Text" : data_eng+data_pol+data_czech,
    "Language" : lang_eng+lang_pol+lang_czech,
})

print(df.shape)
```

```
(1685821, 2)
```

```
df.head()
```

	Text	Language
0	membership of parliament see minutes	English
1	approval of minutes of previous sitting see mi...	English
2	membership of parliament see minutes	English
3	verification of credentials see minutes	English
4	documents received see minutes	English


```
df['Language'].value_counts()
```

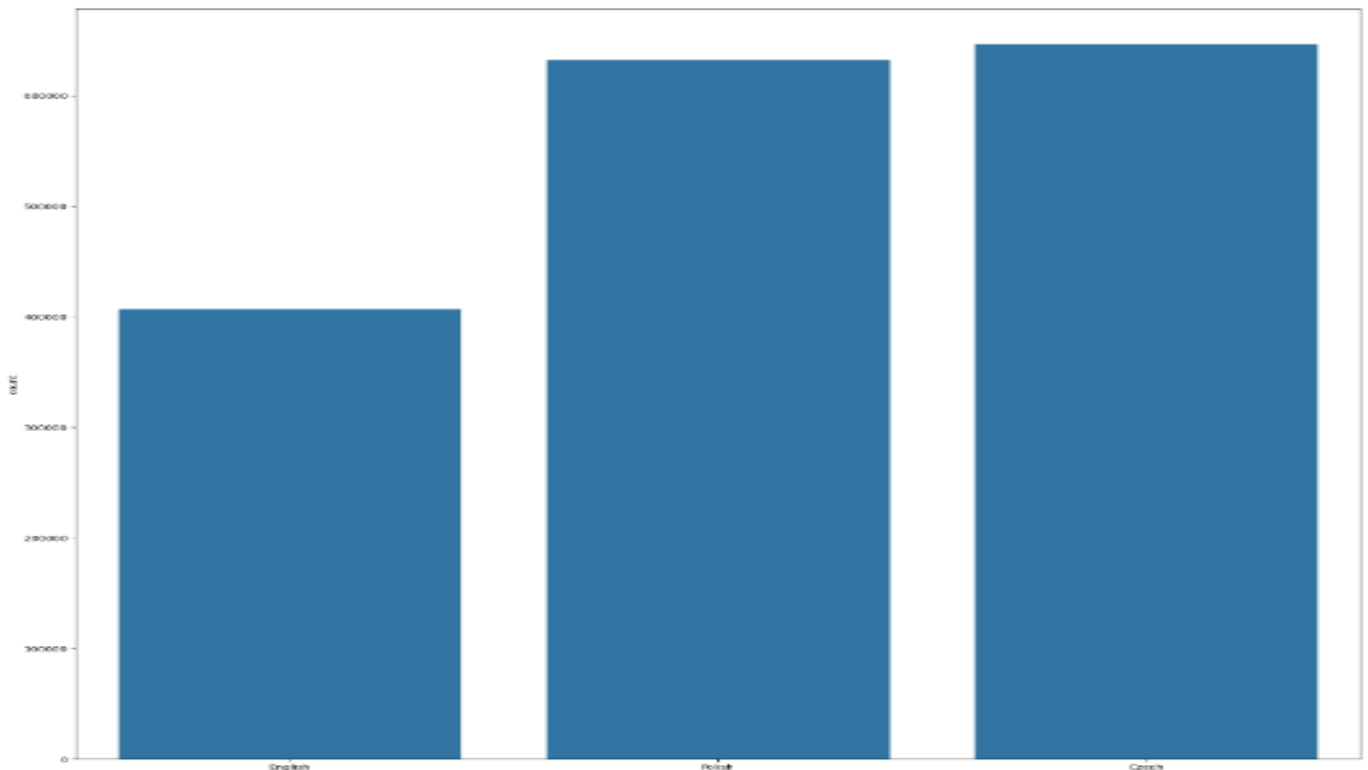
```
Language
Czech      646526
Polish      632416
English     406879
Name: count, dtype: int64
```

vi) Visualisation de la Répartition des Langues

Pour s'assurer que notre ensemble de données est équilibré et pour mieux comprendre sa composition, nous avons visualisé la répartition des langues à l'aide d'un graphique à barres.

```
plt.figure(figsize=(20,20))
sns.countplot(x = df["Language"], data=df)
```

<Axes: xlabel='Language', ylabel='count'>



4. Division des Données

La division des données est une étape essentielle dans le pipeline de machine learning. Elle permet de séparer les données en ensembles d'entraînement et de test, assurant ainsi que les performances du modèle sont évaluées de manière objective sur des données qu'il n'a pas vues auparavant. Voici les étapes détaillées pour diviser les données dans ce projet.

i) Séparation des Variables Indépendantes et Dépendantes

Tout d'abord, nous devons séparer les variables indépendantes (les textes) des variables dépendantes (les étiquettes de langue).

```
x = df.iloc[:,0]
y = df.iloc[:,1]
```

```
y.head()
```

```
0    English
1    English
2    English
3    English
4    English
Name: Language, dtype: object
```

```
x.head()
```

```
0      membership of parliament see minutes
1  approval of minutes of previous sitting see mi...
2      membership of parliament see minutes
3  verification of credentials see minutes
4      documents received see minutes
Name: Text, dtype: object
```

ii) Division des Données en Ensembles d'Entraînement et de Test

Nous utilisons la fonction `train_test_split` de la bibliothèque `scikit-learn` pour diviser les données en ensembles d'entraînement et de test. Nous spécifions une proportion de 20% pour l'ensemble de test et un état aléatoire pour assurer la reproductibilité des résultats.

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

iii) Application du Vectoriseur TF-IDF

Nous utilisons le vectoriseur TF-IDF (Term Frequency-Inverse Document Frequency) pour transformer les textes en représentations numériques. Dans ce cas, nous utilisons des n-grammes de caractères (de 1 à 3 caractères) plutôt que des mots pour capturer des informations linguistiques plus fines.

```
vectorizer = feature_extraction.text.TfidfVectorizer(ngram_range=(1,3), analyzer='char')
```

iv) Préparation des Pipelines de Modélisation

Pour simplifier le processus d'entraînement et de prédiction, nous utilisons des pipelines qui incluent à la fois le vectoriseur TF-IDF et le modèle de classification. Cela permet de chaîner les étapes de transformation et de modélisation en un seul objet.

❖ Pipeline pour MultinomialNB

```
pipe_mnb = pipeline.Pipeline([
    ('vectorizer', vectorizer),
    ('clf', MultinomialNB())
])
```

```
pipe_mnb.fit(x_train, y_train)
```

```
Pipeline(steps=[('vectorizer',
                  TfidfVectorizer(analyzer='char', ngram_range=(1, 3))),
                 ('clf', MultinomialNB())])
```

❖ Pipeline pour Logistic Regression

```
pipe_lr = pipeline.Pipeline([
    ('vectorizer', vectorizer),
    ('lr_clf', LogisticRegression())
])
```

```
pipe_lr.fit(x_train, y_train)
```

```
Pipeline(steps=[('vectorizer',
                  TfidfVectorizer(analyzer='char', ngram_range=(1, 3))),
                 ('lr_clf', LogisticRegression())])
```

v) Évaluation des Modèles

Après avoir entraîné nos modèles de régression logistique et de classification naïve bayésienne multinomiale (MultinomialNB), nous évaluons leurs performances sur l'ensemble de test. Voici les résultats d'évaluation :

• Régression Logistique

❖ Précision : 99.57%

```
lr_predicted = pipe_lr.predict(x_test)
```

```
lr_acc = (metrics.accuracy_score(y_test, lr_predicted))*100
print('The logistic regression has:', lr_acc, '% accuracy')
```

```
The logistic regression has: 99.5723162249937 % accuracy
```

- **Multinomial Naive Bayes**

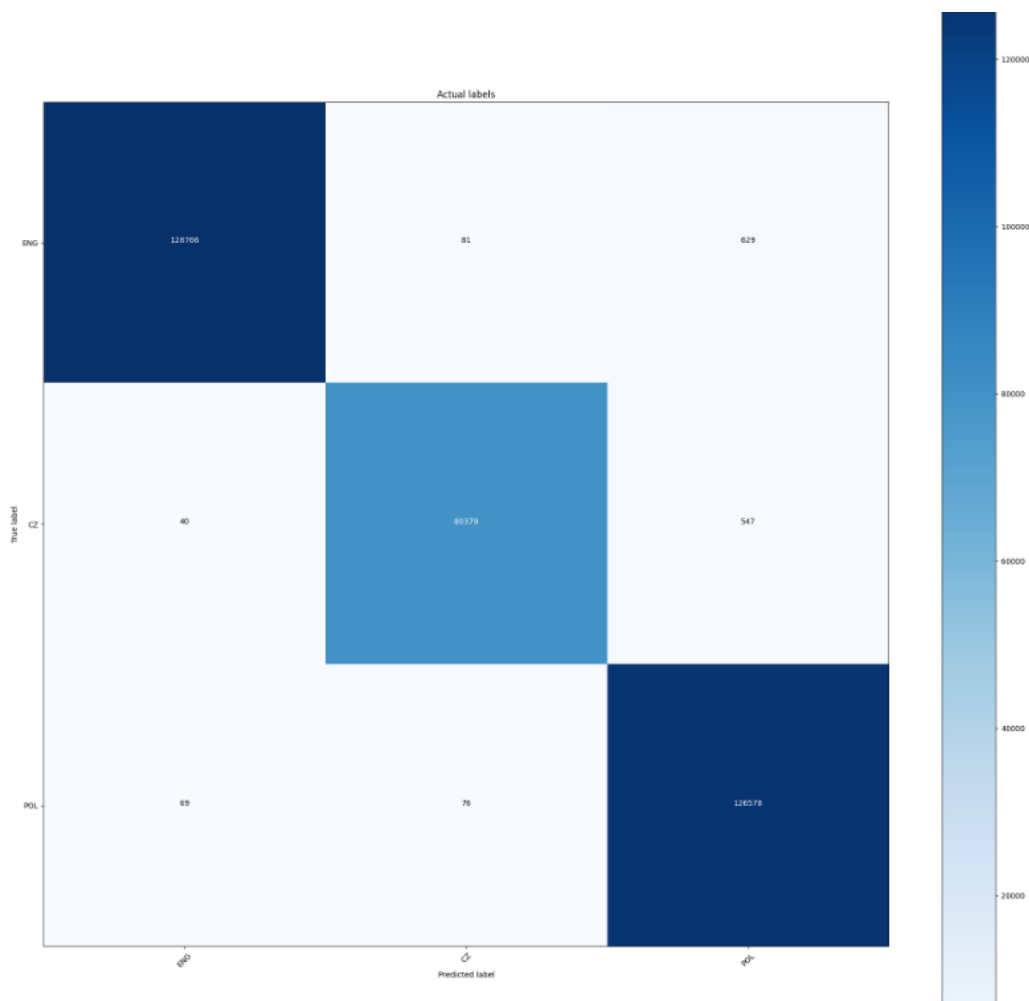
❖ **Précision : 99.52%**

```
mnb_predicted = pipe_mnb.predict(x_test)
```

```
mnb_acc = (metrics.accuracy_score(y_test, mnb_predicted))*100  
print('The MultinomialNB has : ',mnb_acc,'% accuracy')
```

The MultinomialNB has : 99.52397194252073 % accuracy

- **Matrice de Confusion**



Ces résultats montrent une performance très élevée des deux modèles, avec des précisions supérieures à 99%. De plus, la matrice de confusion indique que les modèles ont bien généralisé sur les trois langues (anglais, tchèque, polonais) et ont une capacité élevée à prédire correctement la langue des textes.

Conclusion

Les deux modèles semblent bien fonctionner sur cet ensemble de données, avec des performances très proches. La régression logistique a légèrement surpassé le Multinomial Naive Bayes en termes de précision, mais les deux modèles sont très compétitifs. Il semble que les caractéristiques extraites par le vectoriseur TF-IDF, combinées à des algorithmes

de classification efficaces, permettent d'obtenir des résultats de haute qualité pour cette tâche de classification de la langue.

5. Sauvegarde du Modèle Entraîné

```
import pickle

lrfile = open('lrmodel.pkl', 'wb')
pickle.dump(pipe_lr, lrfile)
lrfile.close()
```

6. Deployment du modèle

Dans cette section de code, nous déployons le modèle de régression logistique entraîné à l'aide de Streamlit, une bibliothèque Python pour la création d'applications web interactives. Voici ce que chaque partie du code fait :

(i) Chargement du Modèle Entraîné

Nous utilisons la fonction `pickle.load()` pour charger le modèle de régression logistique précédemment sauvegardé à partir du fichier 'lrmodel.pkl'. Une fois chargé, le modèle est stocké dans la variable `pipe_lr`.

```
import pickle

# Load the trained model
with open('lrmodel.pkl', 'rb') as lrfile:
    pipe_lr = pickle.load(lrfile)
```

(ii) Interface Utilisateur avec Streamlit

Nous définissons l'interface utilisateur de l'application à l'aide de Streamlit. Cela comprend un titre, une brève description, une barre latérale avec un logo et une section "À Propos", ainsi qu'une zone de texte pour saisir le texte d'entrée.

```
# Title and description in the main section
st.markdown('<h1 style="color: #017bff;">Language Identification Application</h1>', unsafe_allow_html=True)
st.write("Welcome to the Language Identification Model application.")

# Sidebar with logo and description
st.sidebar.image("lo.png", width=100, use_column_width=True)
st.sidebar.title("About")
st.sidebar.info(
    """
    This application employs state-of-the-art machine learning methodologies to predict the language of text
    """
)


# Input text box
input_text = st.text_area("Enter text here:")
```

(iii) Prédiction avec le Modèle :

Lorsque l'utilisateur clique sur le bouton "Predict", le texte d'entrée est récupéré et passé au modèle de régression logistique pour prédire la langue. La prédiction est affichée à l'utilisateur.

```
# Predict button
if st.button("Predict"):
    if input_text:
        prediction = pipe_lr.predict([input_text])[0]
        st.write(f"The predicted language is: <span style='color:green'><b>{prediction}</b></span>", unsafe_a
    else:
        st.write("Please enter some text for prediction.")
```

(iv) Résultat



About

This application employs state-of-the-art machine learning methodologies to predict the language of text inputs. By harnessing the power of Logistic Regression, it achieves exceptional accuracy in identifying languages, making it a valuable tool for various language-related tasks.

Deploy

Language Identification Application

Welcome to the Language Identification Model application.

Enter text here:

Predict

➤ **English:**

Language Identification Application

Welcome to the Language Identification Model application.

Enter text here:

The quick brown fox jumps over the lazy dog.

Predict

The predicted language is: **English**

➤ **Polish:**

Language Identification Application

Welcome to the Language Identification Model application.

Enter text here:

Szybki brązowy lis skacze nad leniwym psem.

Predict

The predicted language is: **Polish**

➤ **Tchèque**

🔗 Language Identification Application

Welcome to the Language Identification Model application.

Enter text here:

Rychlý hnědý pes skáče přes lenivého psa.

Predict

The predicted language is: **Czech**

Étendre la Classification à d'Autres Langues

Bien que l'application actuelle soit conçue pour identifier les langues anglaise, polonaise et tchèque, l'objectif à long terme est d'étendre la classification à un éventail plus large de langues. Cela pourrait inclure des langues telles que le français, l'espagnol, l'allemand et bien d'autres. Toutefois, en raison des limitations de ressources matérielles sur mon ordinateur, telles que la capacité de stockage et la puissance de calcul, il n'est pas actuellement possible de traiter un volume important de données pour toutes les langues souhaitées. Par conséquent, la portée de l'application est actuellement limitée aux langues pour lesquelles des données d'entraînement ont été collectées et où le modèle a été formé avec succès. À l'avenir, des efforts seront déployés pour élargir la gamme de langues prises en charge, dès que les ressources techniques nécessaires seront disponibles.

Conclusion

Le projet de classification automatique de langues a été une exploration enrichissante dans le domaine de l'apprentissage automatique appliqué au traitement du langage naturel, avec une utilisation significative du text mining. En exploitant des techniques avancées telles que la vectorisation TF-IDF et des modèles de classification tels que la régression logistique et la classification naïve bayésienne multinomiale, nous avons pu

construire un système capable d'identifier avec une grande précision la langue d'un texte donné.

L'approche adoptée a intégré le text mining à différentes étapes du processus. Tout d'abord, lors de la prétraitement des données, des techniques de nettoyage avancées ont été utilisées, telles que la suppression de la ponctuation et des chiffres, ainsi que la normalisation de la casse, pour préparer les données brutes pour l'analyse ultérieure. Ensuite, la vectorisation TF-IDF a été appliquée pour représenter les textes sous forme de vecteurs numériques, permettant ainsi aux algorithmes de classification de les traiter de manière efficace. De plus, l'exploration des n-grammes de caractères a permis de capturer les informations de séquence importantes dans les textes, renforçant ainsi les performances du modèle de classification.

Les résultats des évaluations du modèle ont été prometteurs, démontrant l'efficacité de l'approche de text mining adoptée. Les modèles de régression logistique et de classification naïve bayésienne multinomiale ont tous deux atteint des précisions supérieures à 99%, témoignant de leur capacité à généraliser efficacement sur les données de test.

Cependant, malgré les succès rencontrés, ce projet n'est pas sans ses limites. En raison de contraintes matérielles, la portée de l'application est actuellement limitée à un ensemble restreint de langues, bien qu'il existe une intention de l'élargir à l'avenir. De plus, il reste des domaines d'amélioration potentiels, tels que l'exploration de modèles plus complexes et l'augmentation du volume de données d'entraînement pour améliorer la généralisation.

En fin de compte, ce projet a démontré la puissance du text mining dans le domaine du traitement automatique du langage naturel et a fourni une base solide pour des développements futurs. Avec un engagement continu et une exploration plus approfondie, il est possible d'améliorer encore les performances du modèle et de créer une application plus robuste et polyvalente pour identifier les langues dans une variété de contextes et de scénarios d'utilisation.