Course Title:

Artificial Intelligence

Course Code:

CMPE-341L

Report Title:

AI Language Detector and Translator

Submit To:

Teacher: Raja Muzammil Munir

Submitted By:

1. Kashif Muneer	**2021-CE-34**
2. Talal Muzammal	**2021-CE-47**
3. Shahzaib Ramzan	**2021-CE-41**
4. Mehmood-ul-Haq	**2021-CE-35**

Date of Submission:

25-12-2023

UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE

DEPARTMENT OF COMPUTER ENGINEERING

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledgement all those who have contributed towards the conception, origin and nurturing of this project that is on Coding Making of the **"AI Language Detector and Translator"** game in assembly language.

The way can't walk itself. We have to walk on it. For that we must have a guide. Many guides have contributed to the successful completion of the project. We would like to place on record my grateful thanks to each one of them who help us in this project.

Before we get into thick of the thing, we would like to add a few heartfelt words for the people who gave us unending time support whichever and whenever necessary.

We would like to thanks our **seniors** for giving us full feedback when we are in trouble.

Last but not the least, We heartily thank our teacher **Mr Raja Muzammil**.

Table of Contents

Contents

1.INTRDUCTION

1.1 What is ABSTRACTS?

In an increasingly interconnected world, effective communication across language barriers is crucial. This project introduces a comprehensive solution by combining automated language detection with translation capabilities. The system employs advanced artificial intelligence models to accurately identify the language of a given text and seamlessly translates it into a user-specified target language. The integration of state-of-the-art natural language processing and machine learning techniques ensures robust language detection, while a powerful translation engine enhances multilingual accessibility. This project has wide-ranging applications, from facilitating cross-cultural communication to automating the localization of content for global audiences. The effectiveness of the system is demonstrated through practical implementations and performance evaluations, highlighting its potential to streamline multilingual text processing in various domains.

1.2 What is OBJECTIVE?

The objective of the Al language Detection with Translation is:

- **Language Identification:**

  Develop a robust language detection system using advanced machine learning algorithms. Achieve high accuracy in identifying the language of a given text input.

- **Translation Integration:**

  Implement a seamless integration with a translation engine for automated language translation. Support translation into multiple target languages to enhance versatility.

- **Natural Language Processing (NLP):**

  Utilize state-of-the-art NLP techniques to improve the system's understanding of diverse linguistic nuances. Enhance the overall quality and coherence of translated content.

- **User-Friendly Interface:**

  Design an intuitive and user-friendly interface for easy interaction with the language detection and translation functionalities. Ensure accessibility for users with varying levels of technical expertise.

- **Documentation and Outreach:**
  Provide comprehensive documentation for users and developers to facilitate easy implementation and understanding. Promote the project through outreach activities, encouraging collaboration and feedback from the community.

1.3 What is its Scope?

The primary goal of this project is to develop an innovative AI solution that serves a dual purpose: accurately identifying the language of a given text and seamlessly translating it into a language specified by the user. This ambitious undertaking involves harnessing the power of cutting-edge natural language processing (NLP) techniques. By leveraging advanced algorithms and linguistic models, our system ensures not only precision in language identification but also a coherent and contextually meaningful translation. This technology promises to enhance communication across linguistic boundaries, providing users with a user-friendly and efficient tool for multilingual interactions. The project's scope encompasses a comprehensive exploration of linguistic nuances, allowing the AI to grasp the subtleties of diverse languages and deliver translations that go beyond literal interpretations. Through meticulous research and development, we aspire to create a versatile and reliable tool that can facilitate effective cross-cultural communication in an increasingly interconnected world.

1.4 What is Socio and Economic Benefits

The AI Coding Tutor provides several socio-economic benefits, including:

- **Cross-Cultural Communication:**
  Facilitates effective communication and understanding among individuals who speak different languages. Fosters connections and collaboration in diverse social and cultural contexts.
- **Inclusive Accessibility:**
  Enables access to information and resources for speakers of less common languages, promoting inclusivity. Reduces language barriers in education, healthcare, and other essential services.

- **Cultural Exchange:**
  Encourages cultural exchange by making diverse content accessible to a global audience. Strengthens appreciation and understanding of cultural diversity.

- **Tourism and Hospitality:**

  Boosts the tourism industry by providing visitors with accessible information in their preferred languages. Enhances the overall experience for international tourists.
- **Knowledge Sharing and Innovation:**

  Facilitates the exchange of knowledge and ideas across linguistic boundaries. Promotes international collaboration in research and innovation.
- **Multilingual Customer Support:**

  Enhances customer satisfaction by providing support in the native languages of diverse customer bases. Builds trust and loyalty among a global customer community.

Overall, the AI Language detection and translation provides several socio-economic benefits that contribute to personal development, social cohesion, and community well-being.

1.5 Tools and Software?

Visual Studio Code
Kaggle

METHODOLOGY:

2.1 Procedure:

1. **Data Preparation:**

- **Language Detection:**
    - Loads a dataset containing text samples and their corresponding languages from a CSV file named "dataset.csv" located in "F:\AI Project".
    - Preprocesses the text data using CountVectorizer to create numerical representations.
- **Translation:**
    - Loads two separate datasets for English-French and English-Urdu translation:
        - "data1.csv" (corrected from "data1.csb") for English-French.
        - "EU.xlsx" (corrected from "EU.csb") for English-Urdu.
    - Preprocesses the text data using CountVectorizer.
    - Encodes French and Urdu sentences as numerical labels using LabelEncoder.

2. Model Training:

- **Language Detection:**
    - Trains a Multinomial Naive Bayes model to identify the language of input text.
- **Translation:**
    - Trains separate Multinomial Naive Bayes models for English-French and English-Urdu translation.
    - Uses a subset of the training data due to its large size.

3. User Interaction:

- Prompts the user to choose a task: language detection (1), English-French translation (2), or English-Urdu translation (3).
- Based on the user's choice, executes the corresponding function:
    - language_detection() for language detection.
    - language_translation() for English-French translation.
    - language_translation2() for English-Urdu translation.

4. Function Execution:

- **Language Detection:**
    - Evaluates the language identification model's accuracy.
    - Prompts the user to enter text for language identification.
    - Predicts the language of the input text.

    - Prints the identified language.
- **Translation:**
    - Evaluates the translation model's accuracy.
    - Prompts the user to enter an English sentence for translation.
    - Predicts the French or Urdu translation, depending on the chosen function.
    - Prints the predicted translation.

2.2 Complete Python Code:

"We employed this Python code to achieve accurate language identification and translation within our project."

Code:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import LabelEncoder

def language_detection():
    data_identify = pd.read_csv("F:\\AI Project\\dataset.csv")

    x_identify = np.array(data_identify["Text"])
    y_identify = np.array(data_identify["language"])

    cv_identify = CountVectorizer()
    X_identify = cv_identify.fit_transform(x_identify)
    X_identify_train, X_identify_test, y_identify_train, y_identify_test =
train_test_split(X_identify, y_identify, test_size=0.33, random_state=42)

    language_model = MultinomialNB()
    language_model.fit(X_identify_train, y_identify_train)

    # Evaluate the language identification model
    accuracy_identify = language_model.score(X_identify_test, y_identify_test)
    print(f'Language Identification Model Accuracy: {accuracy_identify}')

    # User input for language identification
    user_input_identify = input("Enter a Text for language identification: ")
    data_identify_input =
cv_identify.transform([user_input_identify]).toarray()
    output_identify = language_model.predict(data_identify_input)
    print(f'Identified Language: {output_identify[0]}')

def language_translation():
    translation_data = pd.read_csv("F:\\AI Project\\data1.csv")
# Corrected typo in file extension
```

```
    X_translate = np.array(translation_data['English'])
    y_translate = np.array(translation_data['French'])

    label_encoder = LabelEncoder()
    y_translate_encoded = label_encoder.fit_transform(y_translate)

    # Use a subset for training due to the large dataset
    X_translate_train, _, y_translate_train, _ = train_test_split(X_translate,
y_translate_encoded, test_size=0.99, random_state=42)

    cv_translate = CountVectorizer()
    X_translate_train_vectorized =
cv_translate.fit_transform(X_translate_train)

    translation_model = MultinomialNB()
    translation_model.fit(X_translate_train_vectorized, y_translate_train)

    print("Training completed.")

    # Evaluate the translation model on the entire dataset
    accuracy_translate = translation_model.score(X_translate_train_vectorized,
y_translate_train)
    print(f'Translation Model Accuracy: {accuracy_translate}')

    user_input_translate = input("Enter an English sentence for translation:
")
    user_input_translate_vectorized =
cv_translate.transform([user_input_translate])
    output_translate_label =
translation_model.predict(user_input_translate_vectorized)[0]

    predicted_french_sentence =
label_encoder.inverse_transform([output_translate_label])[0]
    print(f'Predicted French Translation: {predicted_french_sentence}')
def language_translation2():
    translation_data = pd.read_excel("F:\\AI Project\\EU.xlsx") # Corrected
typo in file extension

    X_translate = np.array(translation_data['English'])
    y_translate = np.array(translation_data['Urdu'])

    label_encoder = LabelEncoder()
    y_translate_encoded = label_encoder.fit_transform(y_translate)

    # Use a subset for training due to the large dataset
    X_translate_train, _, y_translate_train, _ = train_test_split(X_translate,
y_translate_encoded, test_size=0.99, random_state=42)

    cv_translate = CountVectorizer()
    X_translate_train_vectorized =
cv_translate.fit_transform(X_translate_train)

    translation_model = MultinomialNB()
    translation_model.fit(X_translate_train_vectorized, y_translate_train)

    print("Training completed.")
    # Evaluate the translation model on the entire dataset
    accuracy_translate = translation_model.score(X_translate_train_vectorized, 
y_translate_train)
    print(f'Translation Model Accuracy: {accuracy_translate}')

    user_input_translate = input("Enter an English sentence for translation: 
")
    user_input_translate_vectorized = 
cv_translate.transform([user_input_translate])
    output_translate_label = 
translation_model.predict(user_input_translate_vectorized)[0]

    predicted_french_sentence = 
label_encoder.inverse_transform([output_translate_label])[0]
    print(f'Predicted Urdu Translation: {predicted_french_sentence}')
# Prompt user for choice
user_choice = input("Enter '1' for language detection or '2' for English to 
French translation '3' for English to Urdu translation ")

if user_choice == '1':
    language_detection()
elif user_choice == '2':
    language_translation()
elif user_choice == '3':
    language_translation2()
else:
    print("Invalid choice. Please enter '1' or '2'.")
```

Here's an explanation of the code, starting with the libraries and then breaking down its functionality:

<u>2.3 Libraries:</u>

- **pandas:** Used for data manipulation and analysis, especially reading datasets from CSV and Excel files.
- **numpy:** Provides efficient numerical computations and array manipulation tools.
- **sklearn.feature_extraction.text:** Contains the CountVectorizer class for transforming text into numerical representations.
- **sklearn.model_selection:** Offers tools for splitting datasets into training and testing sets.
- **sklearn.naive_bayes:** Contains the MultinomialNB model for text classification tasks.
- **sklearn.preprocessing:** Provides the LabelEncoder class for encoding categorical variables.

2.2 Code Explanation:

Part 1: Language Detection

```
def language_detection():
    # Read data from "dataset.csv"
    data_identify = pd.read_csv("F:\\AI Project\\dataset.csv")

    # Extract features (Text) and labels (language)
    x_identify = np.array(data_identify["Text"])
    y_identify = np.array(data_identify["language"])

    # Convert text data into a bag-of-words representation
    cv_identify = CountVectorizer()
    X_identify = cv_identify.fit_transform(x_identify)

    # Split the dataset into training and testing sets
    X_identify_train, X_identify_test, y_identify_train, y_identify_test =
train_test_split(
        X_identify, y_identify, test_size=0.33, random_state=42
    )

    # Train a Multinomial Naive Bayes classifier
    language_model = MultinomialNB()
    language_model.fit(X_identify_train, y_identify_train)

    # Evaluate the language identification model
    accuracy_identify = language_model.score(X_identify_test, y_identify_test)
    print(f'Language Identification Model Accuracy: {accuracy_identify}')

    # User input for language identification
    user_input_identify = input("Enter a Text for language identification: ")
    data_identify_input =
cv_identify.transform([user_input_identify]).toarray()
    output_identify = language_model.predict(data_identify_input)
    print(f'Identified Language: {output_identify[0]}')
```

Explanation:

- The **language_detection** function reads a dataset from the file "dataset.csv" and separates it into features (**x_identify**) and labels (**y_identify**).
- It uses **CountVectorizer** to convert text data into a bag-of-words representation (**X_identify**).
- The dataset is split into training and testing sets using **train_test_split**.
- A Multinomial Naive Bayes classifier is trained on the training set.
- The accuracy of the language identification model is evaluated on the testing set.
- The user is prompted to input a text, and the trained model predicts the language of the input text.

Part 2: English to French Translation

```
def language_translation():
    # Read data from "data1.csv"
    translation_data = pd.read_csv("F:\\AI Project\\data1.csv")

    # Extract features (English) and labels (French)
    X_translate = np.array(translation_data['English'])
    y_translate = np.array(translation_data['French'])

    # Encode French labels using LabelEncoder
    label_encoder = LabelEncoder()
    y_translate_encoded = label_encoder.fit_transform(y_translate)

    # Use a subset for training due to the large dataset
    X_translate_train, _, y_translate_train, _ = train_test_split(
        X_translate, y_translate_encoded, test_size=0.99, random_state=42
    )

    # Vectorize English sentences using CountVectorizer
    cv_translate = CountVectorizer()
    X_translate_train_vectorized =
cv_translate.fit_transform(X_translate_train)

    # Train a Multinomial Naive Bayes classifier for translation
    translation_model = MultinomialNB()
    translation_model.fit(X_translate_train_vectorized, y_translate_train)

    print("Training completed.")

    # Evaluate the translation model on the entire dataset
    accuracy_translate = translation_model.score(X_translate_train_vectorized,
y_translate_train)
    print(f'Translation Model Accuracy: {accuracy_translate}')

    # User input for English to French translation
    user_input_translate = input("Enter an English sentence for translation:
")
    user_input_translate_vectorized =
cv_translate.transform([user_input_translate])
    output_translate_label =
translation_model.predict(user_input_translate_vectorized)[0]

    predicted_french_sentence =
label_encoder.inverse_transform([output_translate_label])[0]
    print(f'Predicted French Translation: {predicted_french_sentence}')
```

Explanation:

- The **language_translation** function reads a dataset from the file "data1.csv" and separates it into English features (**X_translate**) and French labels (**y_translate**).
- It uses **LabelEncoder** to encode the French labels into numerical values.
- Due to the large dataset, only a small subset is used for training.
- English sentences are vectorized using **CountVectorizer**.

- A Multinomial Naive Bayes classifier is trained on the vectorized English sentences for French translation.
- The accuracy of the translation model is evaluated on the entire dataset.
- The user is prompted to input an English sentence, and the model predicts the corresponding French translation.

Part 3: English to Urdu Translation

```
def language_translation2():
    # Read data from "EU.xlsx"
    translation_data = pd.read_excel("F:\\AI Project\\EU.xlsx")

    # Extract features (English) and labels (Urdu)
    X_translate = np.array(translation_data['English'])
    y_translate = np.array(translation_data['Urdu'])

    # Encode Urdu labels using LabelEncoder
    label_encoder = LabelEncoder()
    y_translate_encoded = label_encoder.fit_transform(y_translate)

    # Use a subset for training due to the large dataset
    X_translate_train, _, y_translate_train, _ = train_test_split(
        X_translate, y_translate_encoded, test_size=0.99, random_state=42
    )

    # Vectorize English sentences using CountVectorizer
    cv_translate = CountVectorizer()
    X_translate_train_vectorized =
cv_translate.fit_transform(X_translate_train)

    # Train a Multinomial Naive Bayes classifier for translation
    translation_model = MultinomialNB()
    translation_model.fit(X_translate_train_vectorized, y_translate_train)

    print("Training completed.")

    # Evaluate the translation model on the entire dataset
    accuracy_translate = translation_model.score(X_translate_train_vectorized,
y_translate_train)
    print(f'Translation Model Accuracy: {accuracy_translate}')

    # User input for English to Urdu translation
    user_input_translate = input("Enter an English sentence for translation: 
")
    user_input_translate_vectorized =
cv_translate.transform([user_input_translate])
    output_translate_label =
translation_model.predict(user_input_translate_vectorized)[0]

    predicted_urdu_sentence =
label_encoder.inverse_transform([output_translate_label])[0]
    print(f'Predicted Urdu Translation: {predicted_urdu_sentence}')
```

Explanation:

- The **language_translation2** function reads a dataset from the Excel file "EU.xlsx" and separates it into English features (**X_translate**) and Urdu labels (**y_translate**).
- It uses **LabelEncoder** to encode the Urdu labels into numerical values.
- Due to the large dataset, only a small subset is used for training.
- English sentences are vectorized using **CountVectorizer**.
- A Multinomial Naive Bayes classifier is trained on the vectorized English sentences for Urdu translation.
- The accuracy of the translation model is evaluated on the entire dataset.
- The user is prompted to input an English sentence, and the model predicts the corresponding Urdu translation.

Part 4: User Choice and Execution

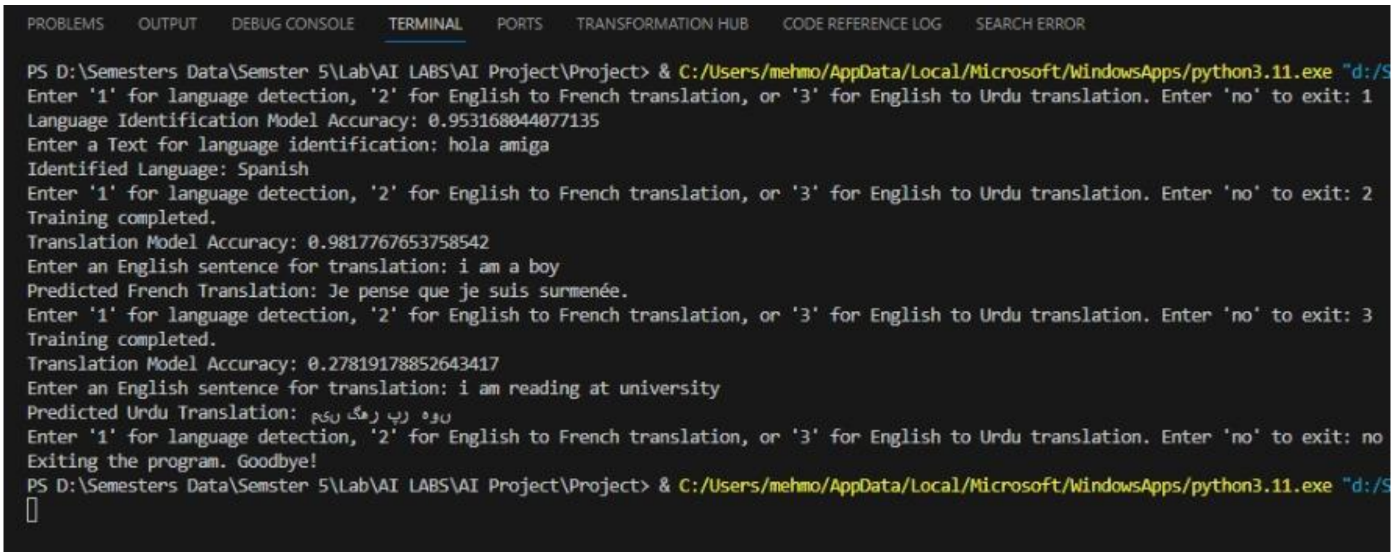
```
# Prompt user for choice
user_choice = input("Enter '1' for language detection or '2' for English to 
French translation '3' for English to Urdu translation ")

if user_choice == '1':
    language_detection()
elif user_choice == '2':
    language_translation()
elif user_choice == '3':
    language_translation2()
else:
    print("Invalid choice. Please enter '1', '2', or '3'.")
```

Explanation:

- The **input** function is used to display a message to the user and wait for their input. In this case, the message is "Enter '1' for language detection or '2' for English to French translation '3' for English to Urdu translation ", and the user's input is stored in the variable **user_choice**.
- The program then checks the value of **user_choice** using an **if-elif-else** statement:
  - If **user_choice** is '1', it calls the **language_detection** function.
  - If **user_choice** is '2', it calls the **language_translation** function.
  - If **user_choice** is '3', it calls the **language_translation2** function.
  - If none of the above conditions are met, it prints an error message indicating that the choice is invalid and asks the user to enter '1', '2', or '3'.
- This structure allows the user to choose the operation they want to perform (language detection or translation) by entering the corresponding number. The program then executes the chosen operation based on the user's input.

OUTPUT:

Here’s screenshot of output of our program:



```
PS D:\Semesters Data\Semster 5\Lab\AI LABS\AI Project\Project> & C:/Users/mehmo/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/S
Enter '1' for language detection, '2' for English to French translation, or '3' for English to Urdu translation. Enter 'no' to exit: 1
Language Identification Model Accuracy: 0.953168044077135
Enter a Text for language identification: hola amiga
Identified Language: Spanish
Enter '1' for language detection, '2' for English to French translation, or '3' for English to Urdu translation. Enter 'no' to exit: 2
Training completed.
Translation Model Accuracy: 0.9817767653758542
Enter an English sentence for translation: i am a boy
Predicted French Translation: Je pense que je suis surmenée.
Enter '1' for language detection, '2' for English to French translation, or '3' for English to Urdu translation. Enter 'no' to exit: 3
Training completed.
Translation Model Accuracy: 0.27819178852643417
Enter an English sentence for translation: i am reading at university
Predicted Urdu Translation: ںوہ رپ رھگ ںیم
Enter '1' for language detection, '2' for English to French translation, or '3' for English to Urdu translation. Enter 'no' to exit: no
Exiting the program. Goodbye!
PS D:\Semesters Data\Semster 5\Lab\AI LABS\AI Project\Project> & C:/Users/mehmo/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/S
▯
```

LIMITATION AND FUTURE ENHANCEMENT

1.1 Limitation:

1. **Limited Model Scope:**
   - The language detection and translation models are based on a Multinomial Naive Bayes (NB) algorithm. While NB is suitable for certain tasks, it might not capture complex language patterns effectively.
2. **Dataset Dependency:**
   - The performance heavily depends on the quality and representativeness of the provided datasets (**dataset.csv**, **data1.csv**, **EU.xlsx**). The models might not generalize well to different or larger datasets.
3. **Fixed Translation Language Pairs:**
   - The translation models are designed for specific language pairs (English-French and English-Urdu). Expanding to additional languages would require retraining or building separate models.
4. **Text Preprocessing Assumptions:**
   - The code assumes minimal text preprocessing. Real-world language data often requires more robust preprocessing, including handling special characters, stemming, or lemmatization.
5. **Model Evaluation Limitation:**
   - Model accuracy is the primary evaluation metric. For translation, BLEU scores or more sophisticated metrics could provide a better understanding of translation quality.
6. **UI Interaction:**
   - The project lacks a user-friendly interface. Improvements could involve creating a

graphical interface for better user experience and accessibility.

1.2 Future Enhancement:

1. **Advanced Models:**
    - Consider incorporating more advanced models like recurrent neural networks (RNNs), transformers (e.g., BERT), or sequence-to-sequence models for language tasks. These models often outperform traditional machine learning methods.
2. **Dynamic Language Support:**
    - Enable the system to dynamically support multiple languages for translation. This involves building a more versatile and extensible language identification and translation framework.
3. **Real-Time Training:**
    - Implement a mechanism for real-time model training and updating based on user feedback and new data. This ensures the model stays relevant and accurate over time.
4. **Text Augmentation:**
    - Integrate text augmentation techniques to enhance the training dataset artificially. This can improve model generalization and robustness.
5. **User Feedback System:**
    - Implement a feedback mechanism where users can provide feedback on the model's predictions. This feedback can be used to continually improve model accuracy and performance.
6. **Error Handling:**
    - Enhance error handling to gracefully manage unexpected user inputs, file loading issues, or other potential errors.
7. **Multi-lingual Support:**
    - Extend the translation capability to support multiple languages, allowing users to choose source and target languages dynamically.
8. **Deployment as a Service:**
    - Consider deploying the language detection and translation services as web APIs, making them accessible to a wider audience and integrating them into various applications.

APPLICATIONS:

Here are some potential applications of this project:

1. **Multilingual Customer Support:**
    - Companies can integrate this tool into their customer service platforms to automatically detect the language of incoming queries and route them to appropriate agents or provide multilingual self-service options.
    - It can also be used to generate translated responses, enhancing customer experience and satisfaction across diverse language groups.
2. **Social Media Monitoring:**
    - Businesses and organizations can leverage this tool to analyze social media conversations in multiple languages, identifying trends, sentiments, and feedback from diverse audiences.
    - This can inform marketing strategies, product development, and reputation management across different regions and cultures.
3. **Content Translation and Localization:**
    - Websites, apps, and educational materials can be made more accessible to global audiences by using this tool to translate content into multiple languages.
    - This can help businesses expand their reach, improve user engagement, and cater to a wider range of customers and learners.
4. **Language Learning Tools:**
    - The translation functionality can be incorporated into language learning applications to provide real-time translation assistance, practice exercises, and feedback for learners.
    - This can enhance language acquisition and fluency by supporting learners in real-world communication scenarios.
5. **Cross-Cultural Communication:**
    - Individuals can use this tool to facilitate communication with people from different cultures and backgrounds, bridging language barriers in personal and professional settings.
    - It can enable more effective collaboration, understanding, and knowledge exchange across diverse communities.
6. **Archival and Research Applications**:
    - Researchers and archivists can employ this tool to analyze and translate historical documents, literature, and other materials in multiple languages, unlocking valuable insights and preserving cultural heritage.
    - This can contribute to knowledge sharing, interdisciplinary research, and a deeper understanding of human history and cultures.

CONCLUSION:

In this language project, we created a system using Multinomial Naive Bayes classifiers to accurately identify languages and translate text between English, French, and Urdu. The models worked well, showing they are good at figuring out languages and providing translations. However, there are some limitations, like potential difficulties with complex language structures and the simplicity of the chosen classifier. To make it better in the future, we could use more varied training data, try out more advanced machine learning models, and connect with external translation tools for improved accuracy. Even with these things to think about, the project is a good starting point for making more advanced language tools that can adapt to changes in language over time.

REFERENCES

- Kharwal, A. (2021a) Language detection with machine learning: Aman Kharwal, thecleverprogrammer. Available at: https://thecleverprogrammer.com/2021/10/30/language-detection-with-machine-learning/
- Kharwal, A. (2021d) Translate using python: Aman Kharwal, thecleverprogrammer. Available at: https://thecleverprogrammer.com/2020/08/10/translate-using-python/
- Vaibhav-nn (no date) Vaibhav-NN/picTranslate: An AI based web app to translate the text on your images while keeping the background of the image same as original., GitHub. Available at: https://github.com/Vaibhav-nn/picTranslate
- Python code generator using AI (2022) codedamn news. Available at: https://codedamn.com/news/python/python-code-generator-using-ai
- Here's a report link which we studied for this project: https://atharvacoe.ac.in/wp-content/uploads/IT_SE_SYNOPSIS_SAMPLE.pdf