Language Detection Model

Introduction

- Objective:- Train a simple ML model that detects the language of a given text.
- Dataset Used:- Hugging Face – Wikipedia Language.
- Approach:-
  1. Loaded the dataset of ten langauges from Hugging Face (Wikipedia language)
     Languages used –
     - English
     - French
     - German
     - Hindi
     - Telugu
     - Tamil
     - Punjabi
     - Marathi
     - Bengali
     - Russian
  2. Performed preprocessing tasks on the downloaded dataset.
  3. Applied vectorization to convert text data into numerical features.
  4. Trained the model using the Multinomial Naive Bayes classification algorithm.
  5. Evaluated and tested the trained model for performance and accuracy.

Data Preprocessing

Step 1:

```
from datasets import load_dataset
languages = ["en", "fr", "de", "hi", "te", "ta", "pa", "mr", "bn", "ru"]

datasets = []

min_samples = float('inf')

for lang in languages:
    print(lang + " language is downloading")
    ds = load_dataset("wikimedia/wikipedia", "20231101." + lang)["train"]

    row_count = len(ds)
    print(lang + " dataset has " + str(row_count) + " rows")

    min_samples = min(min_samples, row_count)
    datasets.append(ds)

print("\nSmallest dataset has " + str(min_samples) + " rows, we will balance according to this")
```

- I have implemented a balancing strategy by determining the smallest dataset size (*min_samples*) among the specified languages.
- For each language dataset, it checks the number of rows (*len(ds)*) and updates (*min_samples)* with the smallest dataset size.
- This step is preparing for balancing the dataset, ensuring all languages have an equal number of samples.

Step 2:

```
import re

def extract_language_from_url(url):
    match = re.match("https://" + "([a-z]+)" + ".wikipedia.org", url)
    if match:
        return match.group(1)
    else:
        return None
def add_language_label(row):
    row["label"] = extract_language_from_url(row["url"])
    return row

final_dataset = final_dataset.map(add_language_label)
```

- The function (*extract_language_from_url(url)*) uses regex (*re.match*) to extract the language code (e.g., en, fr, de) from the Wikipedia URLs.
- The function (*add_language_label(row)*) takes a row from the dataset and adds a new column (*label*), which stores the extracted language.
- final_dataset.map(add_language_label) applies the add_language_label function to all rows, effectively adding the "label" column to (*final_dataset*) which is the dataset obtained after balancing and merging all individual datasets, making it ready for training.

Step 3:

```
final_dataset = final_dataset.remove_columns(['url', 'title', 'id'])

print(final_dataset[64000])
```

- Since, I only required text and label. Therefore, I remove the url, title, and id from the dataset.
- Keeping only relevant columns reduces memory usage and speeds up training.

print(final_dataset[64000]):-

- This prints the data at index 64000(randomly taken) to check if the columns were successfully removed and only the required columns remain after preprocessing.

Step 4:

```
def remove_newlines(row):
    row["text"] = row["text"].replace("\n", "")
    return row
final_dataset = final_dataset.map(remove_newlines)
```

- The function (*remove_newlines(row))* takes a row from the dataset as input, modifies the "text" field by replacing all newline characters (\n) with an empty string (""), and returns the modified row.
- The function (*final_dataset.map(remove_newlines))* applies the (*remove_newlines)* function to every row in (*final_dataset)*.This ensures that all newline characters are removed from the "text" column for the entire dataset.

Model Selection & Training

```
from sklearn.feature_extraction.text import TfidfVectorizer


vectorizer = TfidfVectorizer(analyzer='char_wb', ngram_range=(2, 4), max_features=10000)

X = vectorizer.fit_transform(final_dataset['text'])

y = final_dataset['label']

print("TF-IDF Feature Matrix Shape:", X.shape)
```

```
TF-IDF Feature Matrix Shape: (514230, 10000)
```

- I have used the **Naive Bayes** model for language classification
- The text data was vectorized using TF-IDF with character-level n-grams (2 to 4) to extract features from text.

  (max_features=10000):-
- Limits the number of features to the **10,000 most important** ones, preventing memory issues.0
- The *(fit_transform())* method converts the text column of (*final_dataset*) into a numerical TF-IDF feature matrix. The variable (x) is a sparse matrix where each row corresponds to a text sample, and each column represents an *n-gram* feature. Additionally, the label column is extracted as the target variable (y), which denotes the language of the text.

```
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)

print("Training Data Shape:", X_train.shape)
print("Testing Data Shape:", X_test.shape)
```

```
Training Data Shape: (411384, 10000)
Testing Data Shape: (102846, 10000)
```

- Splitting dataset into training and testing data.
- Allocates 20% of the data for testing and 80% for training.

```
from sklearn.naive_bayes import MultinomialNB
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
print("Model Training Complete")
```

```
Model Training Complete
```

- Trained the model using Multinomial Naive-Bayes classification

Model Evaluation

Metrics Used:

- Accuracy Score to measure overall correctness.
- Classification Report including precision, recall, and F1-score.

Performance Analysis:

```
from sklearn.metrics import accuracy_score, classification_report

y_pred = nb_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy: %.4f" % accuracy)

print(classification_report(y_test, y_pred))
```

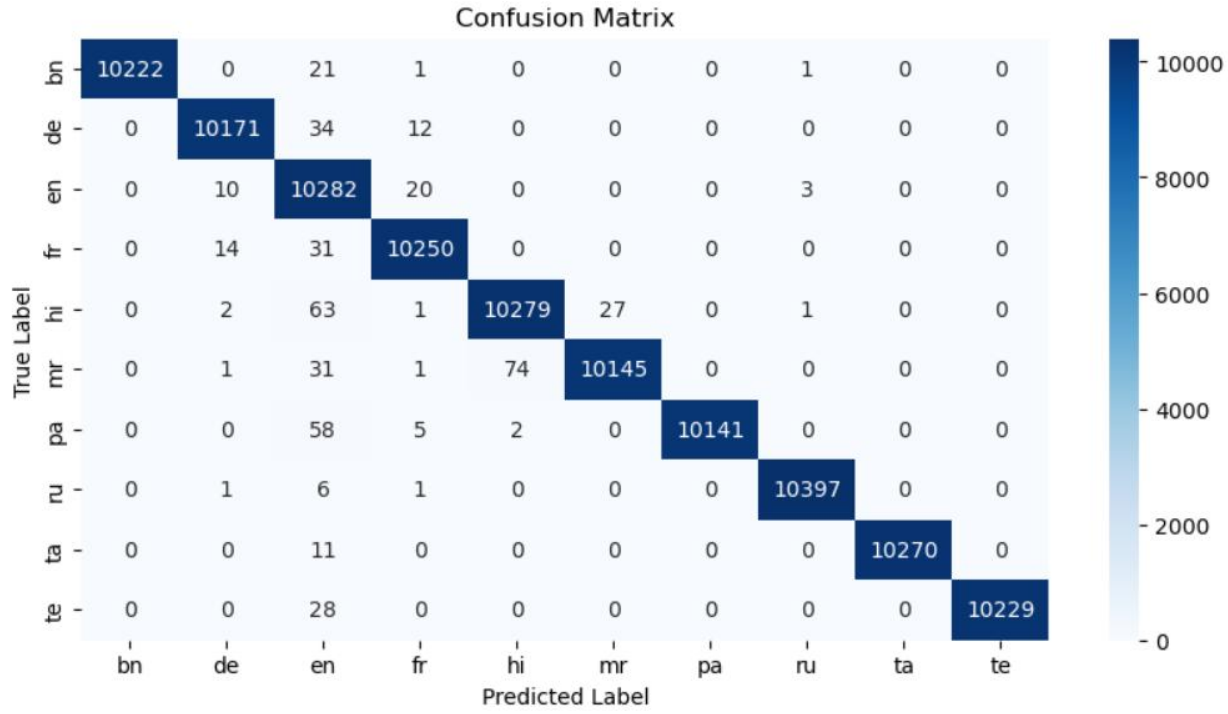
```
Model Accuracy: 0.9955
              precision    recall  f1-score   support

          bn       1.00      1.00      1.00     10245
          de       1.00      1.00      1.00     10217
          en       0.97      1.00      0.98     10315
          fr       1.00      1.00      1.00     10295
          hi       0.99      0.99      0.99     10373
          mr       1.00      0.99      0.99     10252
          pa       1.00      0.99      1.00     10206
          ru       1.00      1.00      1.00     10405
          ta       1.00      1.00      1.00     10281
          te       1.00      1.00      1.00     10257

    accuracy                           1.00    102846
   macro avg       1.00      1.00      1.00    102846
weighted avg       1.00      1.00      1.00    102846
```

- The accuracy score of my trained model is 99.55%, indicating highly effective classification performance.

Results & Visualizations



Observations:

- The diagonal values are high, meaning most predictions are correct.
- Low Misclassification Rate.
- The confusion matrix states that each language class has approximately 10,000 samples, meaning the dataset was well-balanced.
- Since most values outside the main diagonal are nearly zero, it shows that my trained model performs well at correctly identifying different languages.

Conclusion & Next Steps

Summary of Findings:

- The Naive Bayes model achieved an accuracy of **99.55%**, indicating strong classification performance.

Language Prediction Results:

```python
# Example test cases
test_texts = [
    "Hello, how are you?",   # English
    "Bonjour, comment ça va?",  # French
    "नमस्ते, आप कैसे हैं?",  # Hindi
    "Wie geht es Ihnen?",  # German
    "స్వాగతం, మీరు ఎలా ఉన్నారు?", #Telgu
    "ਸਤਿ ਸ੍ਰੀ ਅਕਾਲ ਤੁਹਾਡਾ ਕੀ ਹਾਲ ਹੈ", #Punjabi
    "வணக்கம், எப்படி இருக்கிறீர்கள்?", #Tamil
    "कसे आहात?", #Marathi
    "হ্যালো, কেমন আছো?", #Bengali
    "меня зовут Рия" #Russian
]

for text in test_texts:
    predicted_lang = predict_language(text)
    print("Text: " + text + "\nPredicted Language: " + predicted_lang + "\n")
```

```
Text: Hello, how are you?
Predicted Language: en

Text: Bonjour, comment ça va?
Predicted Language: fr

Text: नमस्ते, आप कैसे हैं?
Predicted Language: hi

Text: Wie geht es Ihnen?
Predicted Language: de

Text: స్వాగతం, మీరు ఎలా ఉన్నారు?
Predicted Language: te

Text: ਸਤਿ ਸ੍ਰੀ ਅਕਾਲ ਤੁਹਾਡਾ ਕੀ ਹਾਲ ਹੈ
Predicted Language: pa

Text: வணக்கம், எப்படி இருக்கிறீர்கள்?
Predicted Language: ta

Text: कसे आहात?
Predicted Language: mr

Text: হ্যালো, কেমন আছো?
Predicted Language: bn

Text: меня зовут Рия
Predicted Language: ru
```

- The model was tested on multiple sentences in different languages.
- It accurately predicted the language for each input text, including English, French, Hindi, German, Telugu, Punjabi, Tamil, Marathi, Bengali, and Russian.

Future Improvements:

- Improve the model to detect and classify texts containing multiple languages.
  Example: *"Mujhe coffee pasand hai, but only with almond milk."* (Hindi + English)

  Approaches for Improvement: Can use Transformer Models(In Deep Learning). These are Pre-trained multilingual models can learn cross-language patterns and handle mixed-language texts better.

- Reducing Misclassifications by Analyzing the confusion matrix errors and fine-tune the model using additional preprocessing techniques.

- Adapting to Noisy & Informal Text by improving robustness against spelling variations, abbreviations, and slang often found in user-generated content.

Github Link for the source Code: https://github.com/RiyaBhanot/language-detection-model