PROJECT REPORT: NLP BASED LANGUAGE IDENTIFICATION

Abdul Moyeed, Sai Priya Reddy Godi, Vaibhavi Buddhiraju

May 23, 2023

1. Introduction

The purpose of this project is to develop a language identification model using machine learning techniques. The objective is to accurately classify the language of a given text sample. Language detection plays a crucial role in various applications, such as text analysis, natural language processing, and multilingual content processing. The ability to automatically identify the language of a given text can enable efficient language-specific processing and enhance user experiences. In this project, we explore different models for language detection and evaluate their performance based on accuracy. The project utilizes a dataset that contains text data in multiple languages. By leveraging this dataset, we aim to train a model that can effectively identify and classify the language of a given text sample. The model will be built using machine learning algorithms and will be evaluated based on its ability to correctly predict the language of unseen text samples. The task of language detection involves classifying text data into different language categories. It is a challenging problem due to the diverse nature of languages and the presence of common words or phrases across multiple languages. To tackle this challenge, we employ different models and evaluate their effectiveness in correctly classifying text samples into their respective language categories.

2. Dataset

The dataset used for this project is obtained from Kaggle:

Dataset : https://www.kaggle.com/datasets/basilb2s/language-detection?datasetId=1150837&sortBy=dateRun&tab=profile

It contains 10,337 rows and 2 columns, with each row representing a text sample and its associated language label.

3. Methodology

3.1. Data Pre-processing

In the data pre-processing stage, we performed several steps to prepare the dataset for further analysis and modelling. The following steps were executed:

Imported necessary libraries:

The required libraries, including pandas, seaborn, NumPy, matplotlib.pyplot, nltk, spacy, sklearn, keras, and TensorFlow, were imported. These libraries provide functions and tools for data manipulation, visualization, and machine learning.

Loaded the dataset:

The dataset was loaded using the pandas library's read_csv() function. The dataset file, "Language Detection.csv," was provided as input to the function. The loaded dataset was stored in the data variable.

Explored the structure of the dataset:

To gain an understanding of the dataset, several functions were used:

head(): This function displays the first few rows of the dataset, giving a glimpse of the data.

shape: This function returns the dimensions of the dataset, providing the number of rows and columns.

info(): This function provides information about the dataset, including the data types of each column and the presence of any missing values.

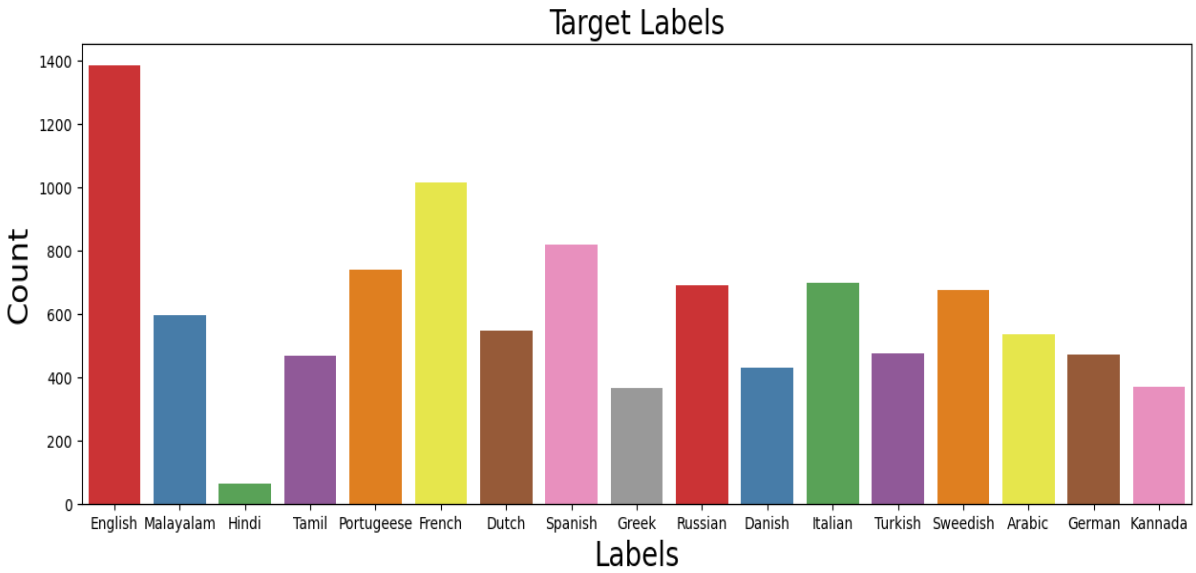
describe(): This function generates descriptive statistics for the dataset, such as count, mean, standard deviation, minimum, and maximum values.

isna().sum(): This function calculates the sum of missing values in each column. It helps identify if any columns contain missing data.

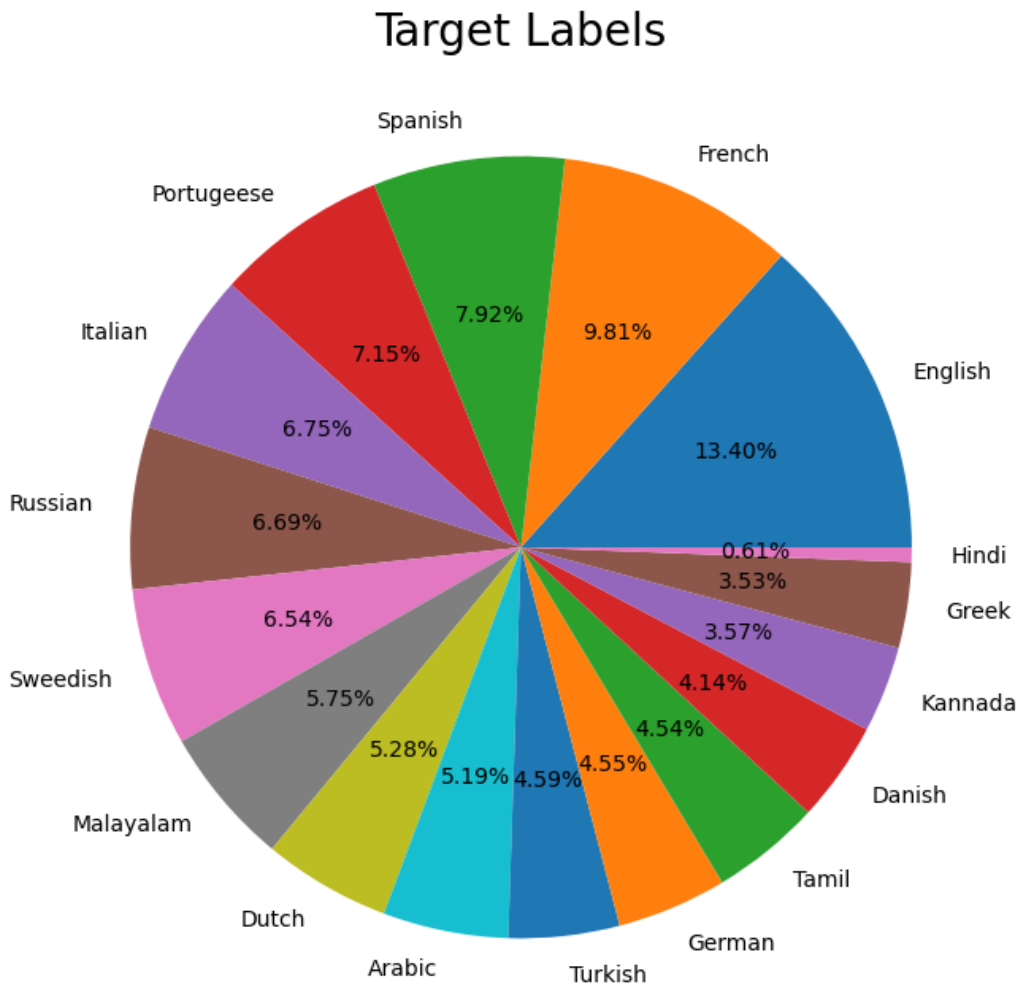
Visualized the distribution of target labels:

To understand the distribution of the target labels in the dataset, visualizations were created:

Count plots: The seaborn library's countplot() function was used to create a bar plot showing the count of each language label. This plot helps identify the distribution and relative frequencies of the target labels.



Pie chart: The matplotlib.pyplot library's pie() function was used to create a pie chart representing the percentage distribution of different target labels. This chart provides a visual representation of the label proportions.



Checked and corrected target label inconsistencies:

In this step, the target labels were examined for any inconsistencies or misspellings. Specifically, the label 'Portugeese' was identified and corrected to 'Portuguese' to ensure consistency and accuracy in the dataset.

By performing these data pre-processing steps, we gained insights into the dataset's structure, ensured data integrity, and prepared the data for subsequent analysis and modelling.

3.2. Text Cleaning and Visualization

The text cleaning and visualization steps involved the following:

Defined functions for tokenization, stop word removal, part-of-speech tagging, stemming, lemmatization, and named entity recognition: In this project, several functions were defined to perform different text processing tasks:

word_token This function tokenizes a sentence into individual words using the word_tokenize function from the nltk library.

remove_stop_word : This function removes stop words, punctuation, and digits from a sentence. It returns a list of words that are not considered stop words or irrelevant characters.

Pos : This function performs part-of-speech tagging on a sentence using the nltk.pos_tag function. It displays the word, its part of speech tag, and the meaning of the tag using the spacy.explain function.

Stemming : This function applies stemming to a sentence using the SnowballStemmer from the nltk library. It stems each word in the sentence and displays the original word followed by its stemmed version.

Lemmatizer : This function performs lemmatization on a sentence using the WordNetLemmatizer from the nltk library. It lemmatizes each word in the sentence and displays the original word followed by its lemmatized form.

NER_word : This function performs named entity recognition (NER) on a sentence using the en_core_web_sm model from the spacy library. It identifies named entities in the sentence and displays the entity text, the entity label, and the meaning of the label using spacy.explain.

Visualization : This function performs dependency parsing visualization on a sentence using the en_core_web_sm model from the spacy library. It displays the dependency parse tree and named entities in the sentence using the displacy.render function.

3.3. Feature Extraction

In the feature extraction phase, the TfidfVectorizer from the sklearn library was used to convert the cleaned text data into numerical features. The following steps were performed:

Imported the TfidfVectorizer from sklearn.feature_extraction.text.

Created an instance of the TfidfVectorizer: The TfidfVectorizer is a text feature extraction technique that assigns weights to words based on their term frequency-inverse document frequency (TF-IDF) values. It takes several parameters to customize the feature extraction process, such as ngram_range (to consider word combinations), max_features (to limit the number of features), and stop_words (to specify additional stop words).

Created a pipeline: A pipeline was created using the Pipeline class from sklearn.pipeline. The pipeline allows chaining multiple steps together, ensuring consistent and efficient feature extraction.

Fit and transform the data: The X_clean, which contains the cleaned text data, was passed to the fit_transform function of the pipeline. This step transformed the text data into numerical features using the TfidfVectorizer. The transformed features were stored in the X_input variable.

Created a DataFrame of feature names: To understand the extracted features, the feature names were obtained using the get_feature_names function of the TfidfVectorizer. These feature names represent the unique words present in the text data.

The resulting DataFrame has 39571 rows and 1 column, representing the feature names extracted from the TfidfVectorizer.

[27]:

	Feature Names
0	00
1	000
2	000ക
3	001
4	001097666virtual
...	...
39566	ൽപ
39567	ൽവർ
39568	ശക
39569	ശപ
39570	ശമ

39571 rows × 1 columns

Displayed a subset of the features: To get an idea of the extracted features, a subset of feature names was displayed. This subset was randomly selected using the np.random.randint function and printed to the console.By utilizing the TfidfVectorizer, the cleaned text data was transformed into numerical features that capture the importance of words in distinguishing different languages. These features will serve as input to the machine learning model for training and classification purposes.

3.4. Label Encoding

Created a dictionary to map each target label to a unique code. Encoded the target labels using the dictionary, resulting in numerical labels. By creating a dictionary to map target labels to numerical codes and encoding the target labels accordingly, we converted the categorical labels into numerical representations. This enables the machine learning model to work with the encoded labels for training and evaluation purposes.

3.5. Data Split

In order to evaluate the performance of the machine learning model, we split the dataset into training and testing sets. This was done using the train_test_split function from the sklearn.model_selection module.

4. Model Training and Evaluation

In this project, we aimed to perform language detection using various machine learning models and evaluated their accuracy. Let's discuss the models and their performance.

Naive Bayes Classifier:

We utilized the Multinomial Naive Bayes classifier to train and predict language labels. The model was trained on the training data After training, we made predictions on the test data and obtained naiveb_predictions. To evaluate the performance of the model, we calculated the accuracy score by comparing the predicted labels with the true labels (y_test). The accuracy achieved by the Naive Bayes classifier was approximately 0.9536, indicating an accuracy of around 95.36%.

Random Forest Classifier:

We also employed a Random Forest Classifierwith specific hyperparameters: criterion = 'gini', n_estimators = 400, max_depth = 600, and random_state = 33. Similar to the Naive Bayes classifier, we trained the Random Forest Classifier on the training data Next, we made predictions on the test data and obtained y_pred. The accuracy achieved by the Random Forest Classifier was approximately 0.9371, indicating an accuracy of around 93.71%.

LSTM (Long Short-Term Memory) Model:

We explored the use of an LSTM model, which is a type of recurrent neural network (RNN) suitable for sequence data. The text data was tokenized using the Tokenizer and converted into sequences. These sequences were then padded to have the same length using the pad_sequences function. The labels were converted into one-hot encoding to represent them as categorical variables. The LSTM model was constructed with an embedding layer, an LSTM layer, and a dense layer with softmax activation for multi-class classification. During training, we also validated the model on the test data. After training, we evaluated the LSTM model by predicting labels for the test data and comparing them with the true labels. The LSTM model achieved a perfect accuracy of 1.0, indicating that it correctly classified all the test samples.

CONCLUSION

Comparing the three models Based on the accuracy results, the LSTM model emerged as the best-suited model for language detection in this project, as it achieved the highest accuracy among the three models. However, it's important to consider other factors such as computational complexity, scalability, and interpretability when choosing a model for real-world applications. By exploring and comparing these different models, we gained insights into their performance and learned how to leverage their strengths for accurate language classification. The choice of the model depends on the specific requirements of the application and the trade-off between accuracy and other factors.

REFERENCES:

[1] https://towardsdatascience.com/4-nlp-libraries-for-automatic-language-identification-of-text-data-in-python-cbc6bf664774

[2]http://yichang-cs.com/yahoo/WWW16_Abusivedetection.pdf

[3] [Baldwin and Kim, 2010] Timothy Baldwin and Su Nam Kim. Multiword expressions. In Handbook of Natural Language Processing. CRC Press, Taylor and Francis Group, 2 edition, 2010.

[4] F.M. Plaza-del-Arco et al.Comparing pre-trained language models for Spanish hate speech detection Expert Syst. Appl.(2021)

[5] P. Nanglia et al.A hybrid algorithm for lung cancer classification using SVM and neural networks

[6] ICT Express(2021)H. Hettiarachchi et al.Embed2Detect temporally clustered embedded words for event detection in social mediaMach. Learn.(2021)

[7] [Robinson et al., 2007]Stuart Robinson, Greg Aumann, and Steven Bird. Managing fieldwork data with toolbox and the natural language toolkit. Language Documentation and Conservation, 1:44–57, 2007