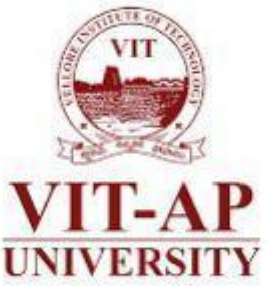
A
Project Report
On
Language Classification



Guided by

Dr K.S. Arikumar

Submitted by

Saloni Reddy 20BCI7167
Sundar Raj Godina 20BCD7183

Table of Contents

Abstract

As people have conquered other places, expanded demographically, or converted others to new religions, languages have moved across space. Language detection is a natural language processing task where we need to identify the language of a text or document. Language identification techniques commonly assume that every document is written in one of a closed set of known languages for which there is training data and is thus formulated as the task of selecting the most likely language from the set of training languages. Using machine learning for language identification was a difficult task a few years ago because there was not a lot of data on languages, but with the availability of data with ease, several powerful machine learning models are already available for language identification.

Automatic language detection is the first step toward achieving a variety of tasks like detecting the source language for machine translation, improving the search relevancy by personalizing the search results according to the query language, providing uniform search box for a multilingual dictionary, tagging data stream from Twitter with appropriate language etc. In this report, we address the problem of detecting documents that contain text from more than one language (multilingual documents).

The data must be evaluated and then pre-processed as necessary. We propose a method that concurrently detects that a document is multilingual and estimates the proportion of the document that is written in each language. To produce an accurate result, we have used 2 algorithms, naive bayes classifier and decision tree algorithm. We have used confusion matrices and accuracy metrics to compare the two classifier models and see which one can predict more accurately.

Problem Statement

The task of identifying the language in which a given document is written has been studied extensively over the past decades, and several classes of methods are available. Briefly, lexically inspired solutions are not practical on embedded devices, syntactically inspired techniques are restricted to long documents where plenty of evidence is available, and generative statistical models based on character n-grams suffer from the conditional independence assumptions. Today, a majority of the data being generated by our users contains only a relatively small number of characters (e.g., text messaging, social network posts). For such short strings of only 10-50 characters, past n-gram approaches based on cumulative frequency addition tend to fall short of accuracy requirements [3], so a more robust sequence classifier is required.

Given an utterance that belongs to the vocabulary of a language, the language identification systems should identify the language irrespective of gender and accents or pronunciations using an acoustic model, which extracts suitable features from speech samples across different languages and different speakers in each language. The language set considered in our project includes English, French, Japanese, Hindi and Kannada among others. The system should evolve over a time with better accuracy and use continuous learning mechanism by incorporating machine learning techniques. The prime objectives of this project are to experiment with different classification methods to see which yields the highest accuracy.

The main objectives can be summarised in the following 3 points:

- To devise a system: confined to identify language of utterances from English, French, Hindi, Kannada, and Japanese and other languages included in the dataset being used.
- The system should not be restricted to a limited vocabulary, in other words the detection is independent of the content of speech.
- The system should not depend on any prosodic features like rhythm, stress, and punctuation. The system performance should not be affected by the nature and gender of the speaker.

Proposed Solution

To get started with applying the modes, we must first evaluate and then pre-process the chosen dataset as necessary. So, the common steps are to load all the required libraries then load the languages data which contains the “Text” which is sentence and target feature “Language” and “Language” contains 22 different languages. And after that check the shape of the data. And it displayed 22000 rows and 2 columns of the data set. This is a very balanced dataset with no missing values, so we can say this dataset is completely ready to be used to train a machine learning model. Then remove the stop words and remove all the characters except the alphabets then lower the sentence and remove the stop words then append into the corpus list. Corpus list is our final list which contains the sentences after pre-processing on which we apply our machine learning models.

Now we to prepare the data by splitting the data into training and testing. Training data is 80% and testing data is 20%. Our model is trained using the 80% data and for the evaluation, we use 20% data. And then check the shape of the data sets.

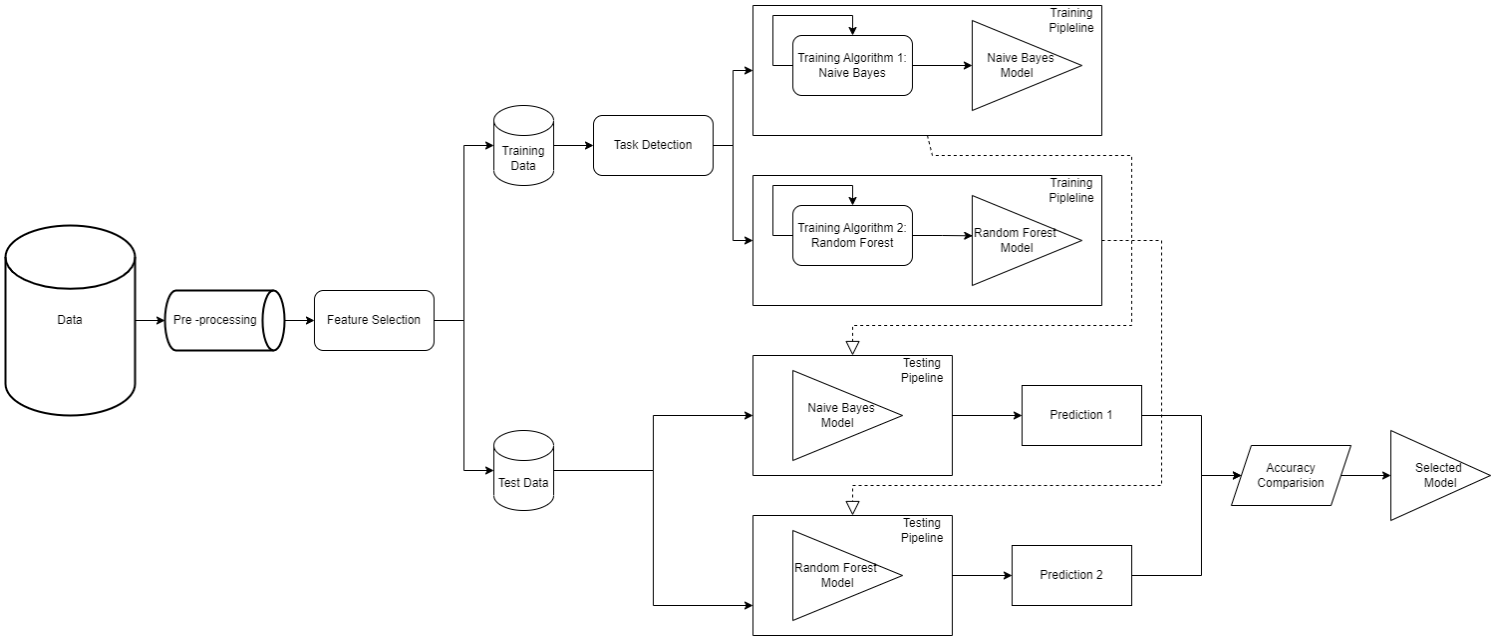
To produce an accurate result, we compare the output from 2 different algorithms, namely

1. Naive Bayes Classifier: This model performs primarily based on the probability. The output based on 20% testing data is basically used to evaluate the model. Then we evaluate the model using the actual testing data and the predicted output data and record the accuracy measured.

2. Decision Tree Algorithm: This model performs primarily based on majority of several decision tree iteration. The output based on 20% testing data is basically used to evaluate the model. Then we evaluate the model using the actual testing data and the predicted output data and record the accuracy measured.

Proposed System Model

We start by defining an Archidecture Diagram for out proposed solution:



Now, we proceed with the codes for implementing the discussed strategy.

IMPORTING NECESSARY PACKAGES:

```
import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.naive_bayes import MultinomialNB

import matplotlib as plt
```

STORING DATASET INTO DATAFRAME:

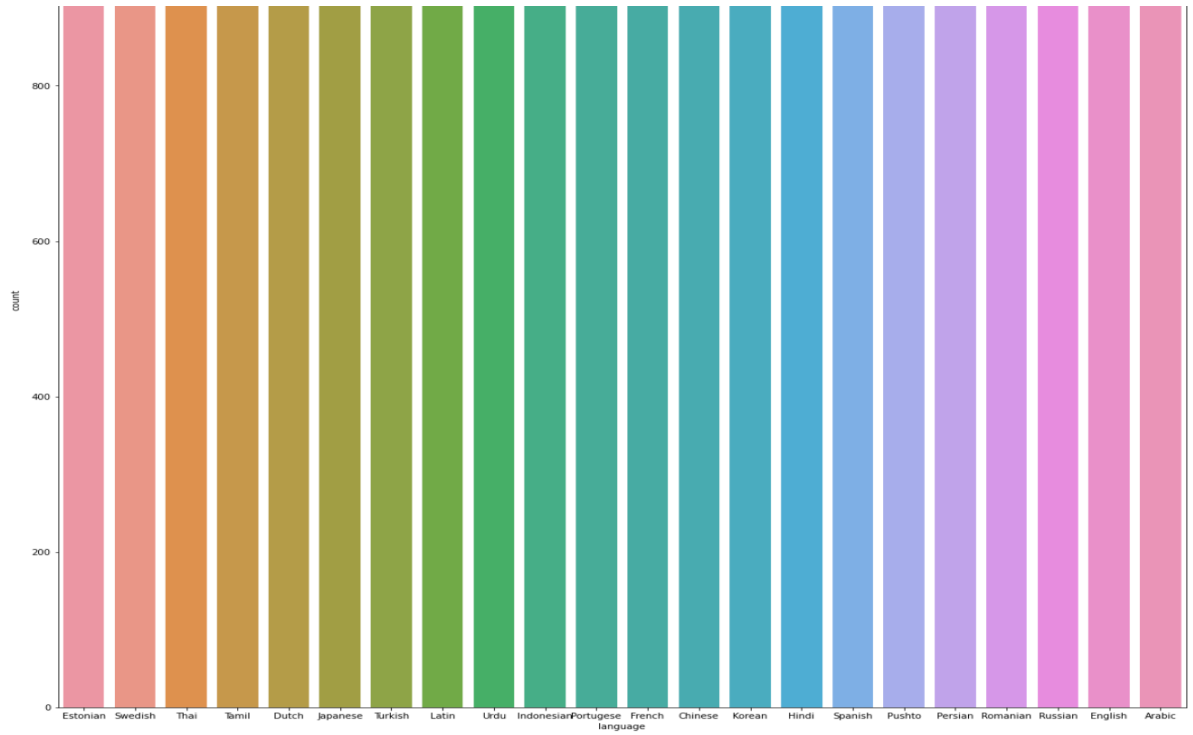
```
data= pd.read_csv("dataset.csv")

data.Text.head(20),data.language.head(20)

data.shape
```

(22000, 2)

```
x=data.Text
y=data.language
```

Visualize that how much language and how much sentences in dataset

```
plt.figure(figsize=(20,20))
sns.countplot(data['language'])
```



Remove stopwards from sentence

We process the given data and remove unwanted data.

```
ps = PorterStemmer()

corpus=[]

for i in range(len(data['Text'])):

    rev = re.sub("^[a-zA-Z]",' ', data['Text'][i])

    rev = rev.lower()

    rev = rev.split()

    rev = [ps.stem(word) for word in rev if set(stopwords.words())]
```

```
    rev = ' '.join(rev)

    corpus.append(rev)

    print(f"{i}")
```

Convert sentences into vector

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=10000)

X = cv.fit_transform(corpus).toarray()

X.shape
```

```
Out[401]: (22000, 10000)
```

LabelEncoding (Convert language name into 1,2,3....etc

```
from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()

y = label.fit_transform(data['language'])

label.classes_
```

```
: array(['Arabic', 'Chinese', 'Dutch', 'English', 'Estonian', 'French',
        'Hindi', 'Indonesian', 'Japanese', 'Korean', 'Latin', 'Persian',
        'Portugese', 'Pushto', 'Romanian', 'Russian', 'Spanish', 'Swedish',
        'Tamil', 'Thai', 'Turkish', 'Urdu'], dtype=object)
```

```
data1 = pd.DataFrame(np.c_[corpus,y],columns=['Sentence','Language'])

data1.head()
```

After data processing this is final dataframe we get.

SPLITTING DATA INTO TRAINING AND TESTING:

```
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 = 0)
```

CREATING NAIVE BAYES CLASSIFIER:

```
from sklearn.naive_bayes import MultinomialNB

classifier = MultinomialNB().fit(X_train,y_train)

pred = classifier.predict(X_test)
```

CREATING RANDOM FOREST CLASSIFIER

```
rf=RandomForestClassifier()

rf.fit(x_train_cv,y_train)

pred = rf.predict(X_test)

print(pred)
```

Result Analysis

EVALUATING NAIVE BAYES CLASSIFIER USING ACCURACY AND CONFUSION MATRIX:

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

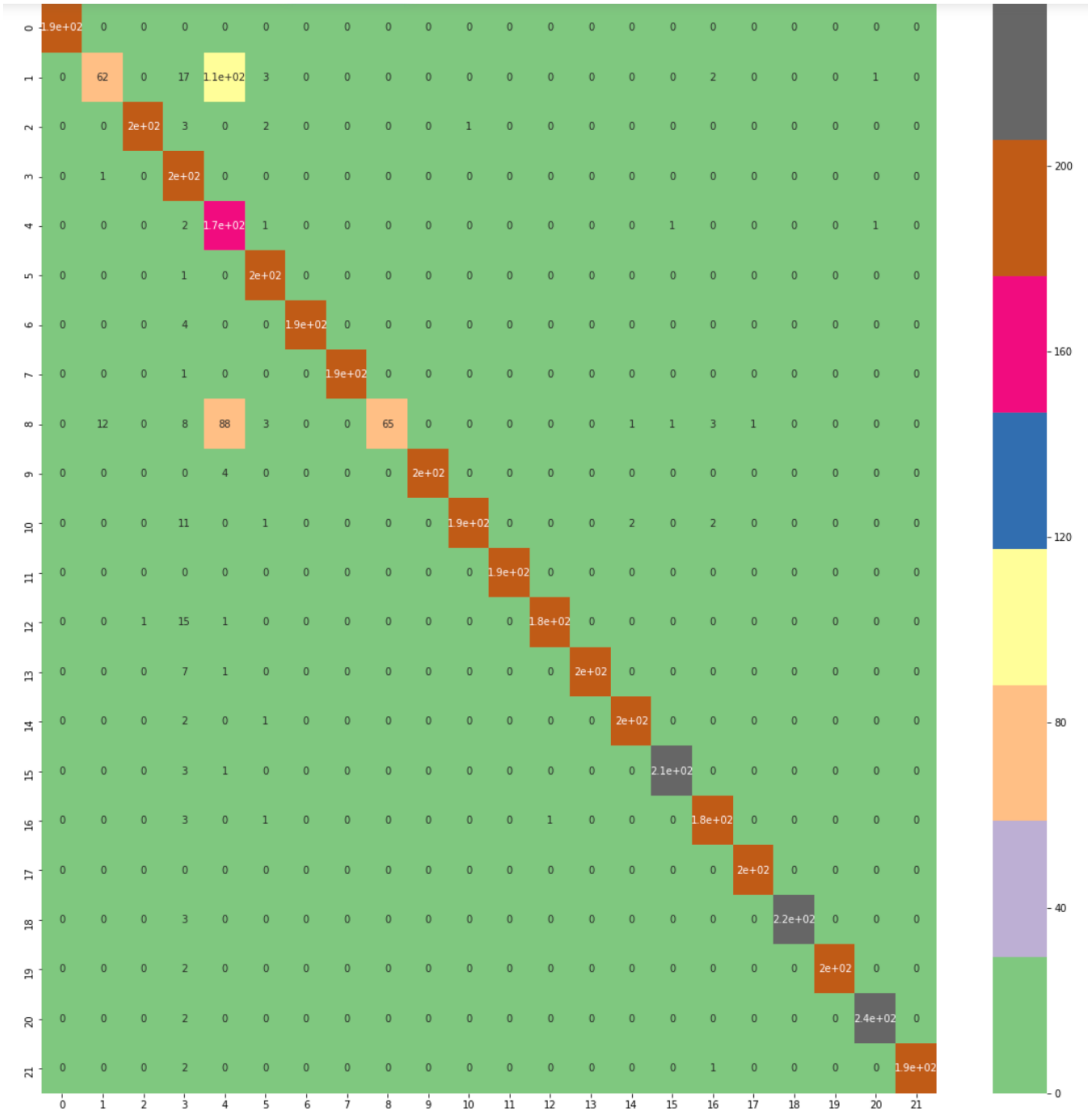
```
print(accuracy_score(y_test,pred))
```

```
0.923409090909091
```

```
confusion_matrix(y_test,pred)
```

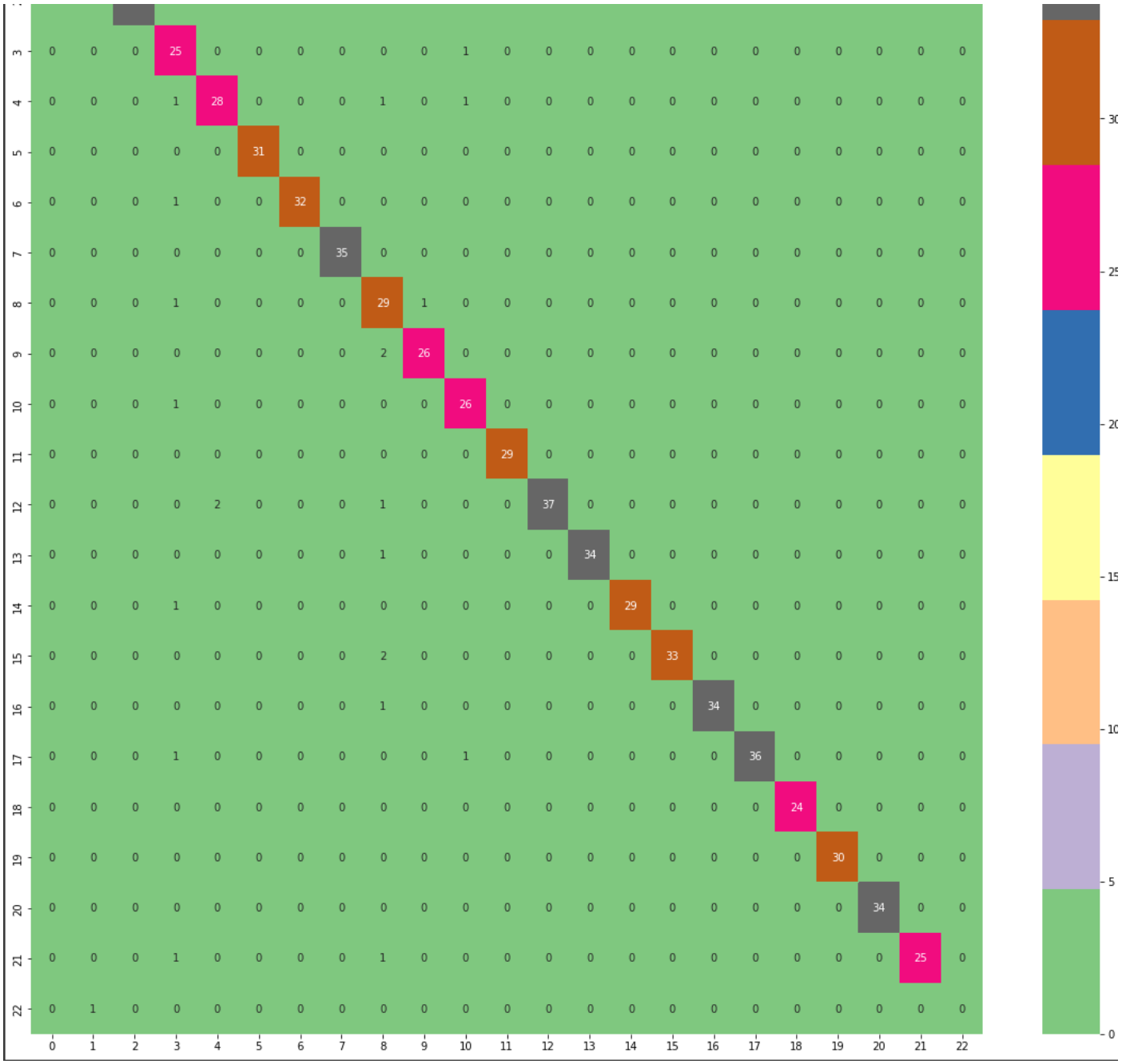
```
plt.figure(figsize=(20,20))
```

```
sns.heatmap(confusion_matrix(y_test,pred),annot=True,cmap=plt.cm.Accent)
```



EVALUATING RANDOM FOREST CLASSIFIER USING ACCURACY AND CONFUSION MATRIX:

```
from sklearn.metrics import accuracy_score,confusion_matrix

print(accuracy_score(y_test,pred))

print(confusion_matrix(y_test,pred))

plt.figure(figsize=(20,20))

sns.heatmap(confusion_matrix(y_test,pred),annot=True,cmap=plt.cm.Accent)
```



Conclusion

The training and disjoint test sets are a mixture of newswire and short conversational texts. The final classification for each language group is captured in the confusion matrices as shown in the above figures. The diagonal represents accuracy, while off-diagonal cells show the confusability with other languages.

On evaluating the models using the actual testing data and the predicted output data we come to know that the accuracy score of Naïve Bayes is 92% while the score from Random Forest is 94% as predicted in the confusion matrix as well.

With this, we have enough evidence to say that random forest is a more accurate predictor since accuracy score of naive bayes is lower than random forest (0.9234<0.9405).