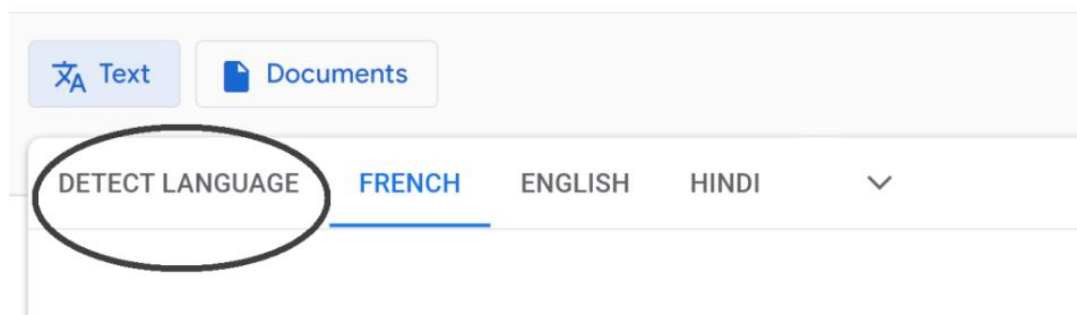


Language Detection with Machine Learning

Language detection is a natural language processing task where we need to identify the language of a text or document. 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. So, if you want to learn how to train a machine learning model for language detection, then this article is for you. In this article, I will walk you through the task of language detection with machine learning using Python.

Language Detection

As a human, you can easily detect the languages you know. For example, I can easily identify Hindi and English, but being an Indian, it is also not possible for me to identify all Indian languages. This is where the language identification task can be used. Google Translate is one of the most popular language translators in the world which is used by so many people around the world. It also includes a machine learning model to detect languages that you can use if you don't know which language you want to translate.



Above image is an one of the example of language detection model.

The most important part of training a language detection model is data. The more data you have about every language, the more accurate your model will perform in real-time. The dataset that I am using is collected from Kaggle, so it will be an appropriate dataset for training a language detection model with machine learning. So in the section below, I will take you through how you can train a language detection model with machine learning using Python.

Language Detection using Python

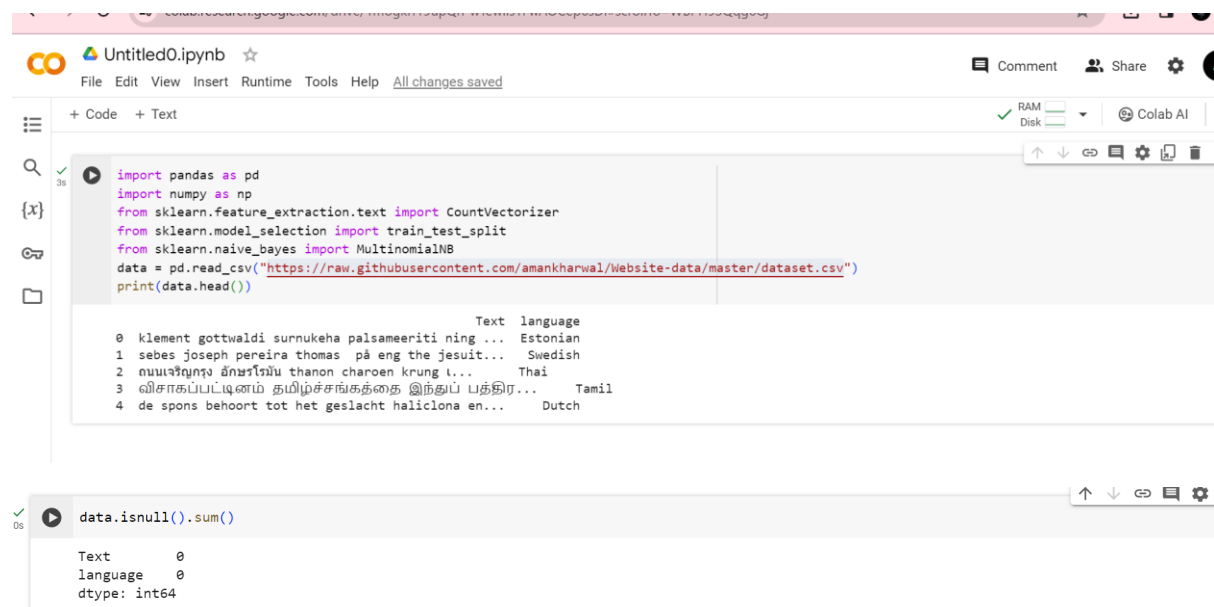
->go to google colab

->at first download the dataset from kaggle

->upload to google colab

-><https://github.com/amankharwal/Website-data/blob/master/dataset.csv>

We have the required dataset there

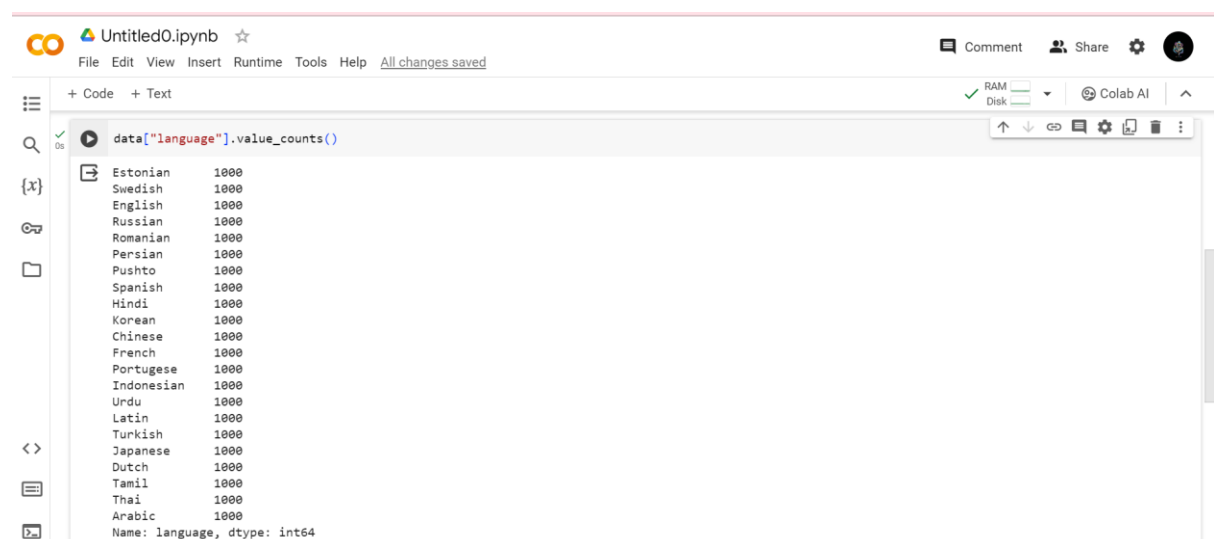


```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/dataset.csv")
print(data.head())
```

	Text	language
0	klement gottwaldi surnukeha palsameeriti ning ...	Estonian
1	sebes joseph pereira thomas pa eng the jesuit...	Swedish
2	นายเจริญฤทธิ์ โนนท์ นายธนกร ชวรงค์ ...	Thai
3	விசாகப்பட்டினம் தமிழ்ச்சங்கத்தை இந்துப் பத்திர...	Tamil
4	de spons behoort tot het geslacht haliclona en...	Dutch

```
data.isnull().sum()
```

Text	0
language	0
dtype:	int64



```
data["language"].value_counts()
```

language	count
Estonian	1000
Swedish	1000
English	1000
Russian	1000
Romanian	1000
Persian	1000
Pushto	1000
Spanish	1000
Hindi	1000
Korean	1000
Chinese	1000
French	1000
Portugese	1000
Indonesian	1000
Urdu	1000
Latin	1000
Turkish	1000
Japanese	1000
Dutch	1000
Tamil	1000
Thai	1000
Arabic	1000

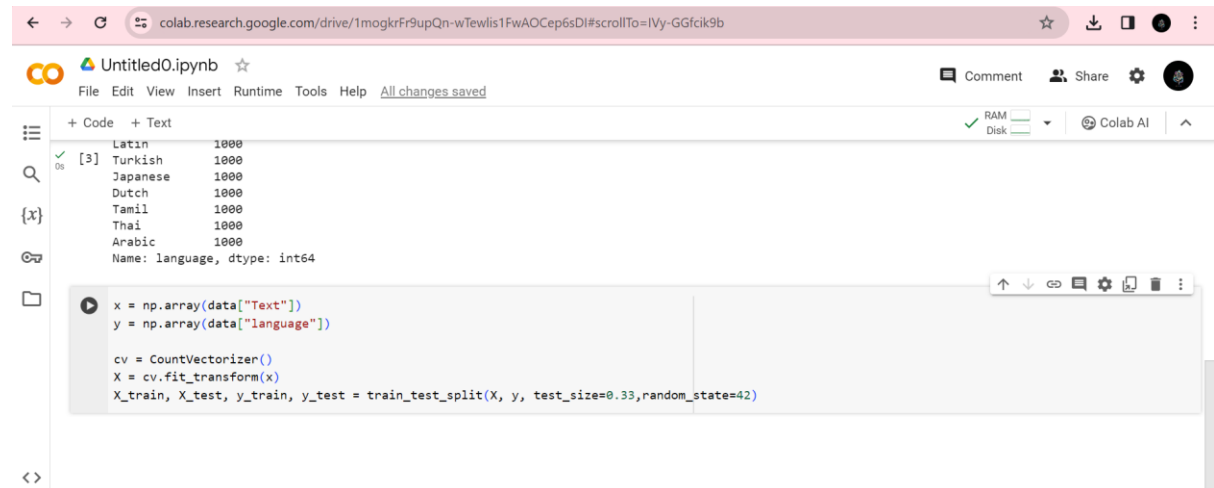
Name: language, dtype: int64

This dataset contains 22 languages with 1000 sentences from each language.

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.

Language Detection Model

Now let's split the data into training and test sets:

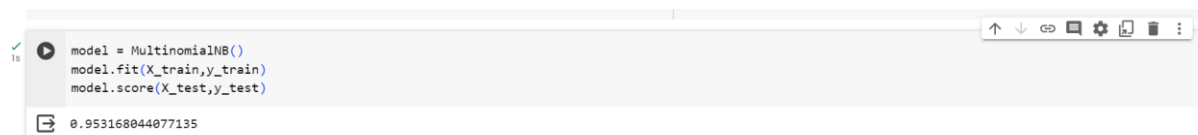


The screenshot shows a Google Colab notebook titled 'Untitled0.ipynb'. The left sidebar displays a file explorer with a table of data: Latin (1000), Turkish (1000), Japanese (1000), Dutch (1000), Tamil (1000), Thai (1000), and Arabic (1000). The main code cell contains the following Python code:

```
x = np.array(data["Text"])
y = np.array(data["language"])

cv = CountVectorizer()
X = cv.fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

As this is a problem of multiclass classification, so I will be using the Multinomial Naïve Bayes algorithm to train the language detection model as this algorithm always performs very well on the problems based on multiclass classification:



The screenshot shows a Google Colab notebook with a code cell containing the following Python code:

```
model = MultinomialNB()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

The output of the code cell is displayed below the code:

```
0.953168044077135
```

User input



The screenshot shows a Google Colab notebook with a code cell containing the following Python code:

```
user = input("Enter a Text: ")
data = cv.transform([user]).toarray()
output = model.predict(data)
print(output)
```

The output of the code cell is displayed below the code:

```
Enter a Text: तुम्हें देखकर मुझे खुशी होती है
['Hindi']
```

So as you can see that the model performs well. One thing to note here is that this model can only detect the languages mentioned in the dataset.

Conclusion:

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. I hope you liked this article on detecting languages with machine learning using Python. Feel free to ask your valuable questions in the comments section below.

Link is available at [github:KAVYA_Sri05](#)