#IMPORTING LIBRARIES

import string #for strings

import re #regular expressions

import pandas as pd #for loading data,reading and manipulating

import matplotlib.pyplot as plt #for plotting

#sklearn is used to build machine learning machine

from sklearn import feature\_extraction

from sklearn import linear\_model

from sklearn.naive\_bayes import MultinomialNB

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn import pipeline

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

import itertools

# LOADING RAW DATA OF ENGLISH,GERMAN,FRENCH FROM EUROPARL\_RAW data

eng\_df = pd.read\_csv("ep-00-01-17.en", "utf-8",engine='python', header=None, names=["English"])

ger\_df=pd.read\_csv("ep-00-01-17.de","utf-8", engine='python',header=None,names=["German"])

fre\_df=pd.read\_csv("ep-00-01-17.fr","utf-8",engine='python',header=None,names=["French"])

# REMOVING SPECIAL CHARECTERS

print ("THE SPECIAL CHARECTERS THAT ARE REMOVED ARE:",end="\n")

for char in string.punctuation:

print(char,end=" ")

translate\_table=dict((ord(char),None) for char in string.punctuation)

#data preprocessing for english raw data

data\_eng=[]

lang\_eng=[]

for i,line in eng\_df.iterrows():

line=line['English']

if len(line) !=0:

line = line.lower()

line = re.sub(r"\d+","",line)

line=line.translate(translate\_table)

data\_eng.append(line)

lang\_eng.append("english")

# data preprocessing for german raw data

data\_ger=[]

lang\_ger=[]

for i,line in ger\_df.iterrows():

line=line['German']

if len(line) !=0:

line = line.lower()

line = re.sub(r"\d+","",line)

line=line.translate(translate\_table)

data\_ger.append(line)

lang\_ger.append("german")

# data preprocessing for french raw data

data\_fre=[]

lang\_fre=[]

for i,line in fre\_df.iterrows():

line=line['French']

if len(line) !=0:

line = line.lower()

line = re.sub(r"\d+","",line)

line=line.translate(translate\_table)

data\_fre.append(line)

lang\_fre.append("french")

df=pd.DataFrame({"Text":data\_eng+data\_ger+data\_fre,

"language":lang\_eng+lang\_ger+lang\_fre})

print(df.shape)

#SPLITTING DATA INTO TRAIN AND TEST SETS(80:20)

print ("ALLOCATION OF TRAIN AND DATA SETS",end="\n") x,y = df.iloc[:,0],df.iloc[:,1]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0)

print(x\_train.shape)

print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

#VECTORIZATION AND MODEL FITTING PIPELINE

vectorizer = feature\_extraction.text.TfidfVectorizer(ngram\_range=(1,3),analyzer='char')

pipe\_lr\_r13 = pipeline.Pipeline([

('vectorizer',vectorizer),

('clf',linear\_model.LogisticRegression())

])

pipe\_lr\_r13.fit(x\_train,y\_train)

#MODEL PREDECTION

y\_predicted = pipe\_lr\_r13.predict(x\_test)

#MODEL EVOLUTION

print("ACCURACY SCORE",end="\n")

acc = (metrics.accuracy\_score(y\_test,y\_predicted))\*100

print(acc,'%')

matrix = metrics.confusion\_matrix(y\_test,y\_predicted)

print('confusion matrix:\n',matrix)

print("OUTPUT FOR THE GIVEN INPUT OF TEXTS:")

#METHOD DEFENITION

def lang\_detect(text):

import numpy as np

import string

import re

import pickle

translate\_table = dict((ord(char), None) for char in string.punctuation)

global lrLangDetectModel

lrLangDetectFile = open('LRModel.pckl','rb')

lrLangDetectModel= pickle.load(lrLangDetectFile)

lrLangDetectFile.close()

text=" ".join(text.split())

text=text.lower()

text=re.sub(r"\d+","",text)

text=text.translate(translate\_table)

pred=lrLangDetectModel.predict([text])

prob=lrLangDetectModel.predict\_proba([text])

return pred[0]

print(lang\_detect("Hello i just built my own language detection model")) print(lang\_detect("hello ,ich habe gerade main eigines sprancherkennugsmodel erstellt")) print(lang\_detect(" Bonjourm je viens de crier mon propre"))

output:

THE SPECIAL CHARECTERS THAT ARE REMOVED ARE:

! " # $ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~ (2705, 2)

ALLOCATION OF TRAIN AND DATA SETS

(2164,)

(541,)

(2164,)

(541,)

ACCURACY SCORE

100.0 %

confusion matrix:

[[180 0 0]

[ 0 172 0]

[ 0 0 189]]

OUTPUT FOR THE GIVEN INPUT OF TEXTS:

english

german

french