**Next word prediction model using python**

**Problem statement**

The problem at hand revolves around the inefficiencies encountered during text input, such as in messaging applications, where users often face challenges in accurately predicting the next word to complete their sentences. This difficulty arises due to the diverse nature of language and the need for contextually relevant suggestions to enhance communication efficiency.

**Proposed system**

Our proposed solution entails the development of a word prediction system utilizing machine learning algorithms to generate contextually relevant word suggestions based on input text. The system aims to provide accurate predictions in real-time, thereby improving user experience and reducing typing effort.

**System development approach**

The system development approach involves a systematic methodology comprising several stages. This includes data collection from large text corpora, data preprocessing to clean and format the text data, feature engineering to extract meaningful features, model selection to choose the most suitable machine learning algorithm, and model training using appropriate techniques.

**Algorithm and Deployment**

Various machine learning algorithms will be considered for the word prediction task, including n-gram models, recurrent neural networks (RNNs), and transformer-based models. The chosen algorithm will undergo rigorous training and evaluation before deployment. The deployment strategy will focus on seamless integration into existing communication platforms, ensuring scalability and efficiency.

**Source code**

import numpy as np

from nltk.tokenize import RegexpTokenizer

from keras.models import Sequential, load\_model

from keras.layers import LSTM

from keras.layers.core import Dense, Activation

from keras.optimizers import RMSprop

import matplotlib.pyplot as plt

import pickle

import heapq

path = '1661-0.txt'

text = open(path).read().lower()

print('corpus length:', len(text))

tokenizer = RegexpTokenizer(r'w+')

words = tokenizer.tokenize(text)

WORD\_LENGTH = 5

prev\_words = []

next\_words = []

for i in range(len(words) - WORD\_LENGTH):

prev\_words.append(words[i:i + WORD\_LENGTH])

next\_words.append(words[i + WORD\_LENGTH])

print(prev\_words[0])

print(next\_words[0])

X = np.zeros((len(prev\_words), WORD\_LENGTH, len(unique\_words)), dtype=bool)

Y = np.zeros((len(next\_words), len(unique\_words)), dtype=bool)

for i, each\_words in enumerate(prev\_words):

for j, each\_word in enumerate(each\_words):

X[i, j, unique\_word\_index[each\_word]] = 1

Y[i, unique\_word\_index[next\_words[i]]] = 1

model = Sequential()

model.add(LSTM(128, input\_shape=(WORD\_LENGTH, len(unique\_words))))

model.add(Dense(len(unique\_words)))

model.add(Activation('softmax'))

optimizer = RMSprop(lr=0.01)

model.compile(loss='categorical\_crossentropy', optimizer=optimizer, metrics=['accuracy'])

history = model.fit(X, Y, validation\_split=0.05, batch\_size=128, epochs=2, shuffle=True).history

model.save('keras\_next\_word\_model.h5')

pickle.dump(history, open("history.p", "wb"))

model = load\_model('keras\_next\_word\_model.h5')

history = pickle.load(open("history.p", "rb"))

plt.plot(history['acc'])

plt.plot(history['val\_acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.plot(history['loss'])

plt.plot(history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

def prepare\_input(text):

x = np.zeros((1, SEQUENCE\_LENGTH, len(chars)))

for t, char in enumerate(text):

x[0, t, char\_indices[char]] = 1.

return x

def prepare\_input(text):

x = np.zeros((1, WORD\_LENGTH, len(unique\_words)))

for t, word in enumerate(text.split()):

print(word)

x[0, t, unique\_word\_index[word]] = 1

return x

prepare\_input("It is not a lack".lower())

def sample(preds, top\_n=3):

preds = np.asarray(preds).astype('float64')

preds = np.log(preds)

exp\_preds = np.exp(preds)

preds = exp\_preds / np.sum(exp\_preds)

return heapq.nlargest(top\_n, range(len(preds)), preds.take)

def predict\_completion(text):

original\_text = text

generated = text

completion = ''

while True:

x = prepare\_input(text)

preds = model.predict(x, verbose=0)[0]

next\_index = sample(preds, top\_n=1)[0]

next\_char = indices\_char[next\_index]

text = text[1:] + next\_char

completion += next\_char

if len(original\_text + completion) + 2 > len(original\_text) and next\_char == ' ':

return completion

def predict\_completions(text, n=3):

x = prepare\_input(text)

preds = model.predict(x, verbose=0)[0]

next\_indices = sample(preds, n)

return [indices\_char[idx] + predict\_completion(text[1:] + indices\_char[idx]) for idx in next\_indices]

quotes = [

"It is not a lack of love, but a lack of friendship that makes unhappy marriages.",

"That which does not kill us makes us stronger.",

"I'm not upset that you lied to me, I'm upset that from now on I can't believe you.",

"And those who were seen dancing were thought to be insane by those who could not hear the music.",

"It is hard enough to remember my opinions, without also remembering my reasons for them!"

]

for q in quotes:

seq = q[:40].lower()

print(seq)

print(predict\_completions(seq, 5))

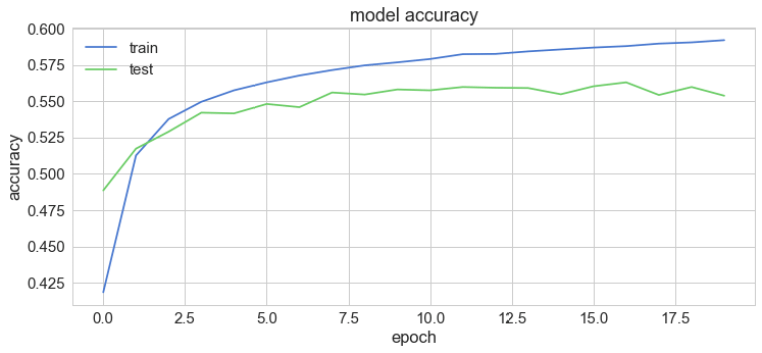
print()

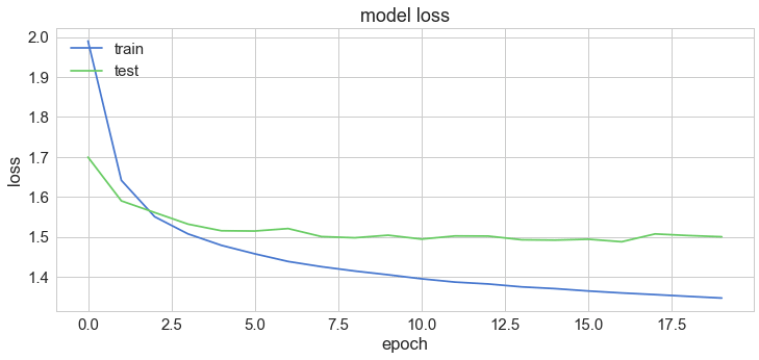
**Output**

corpus length: 581887

['project', 'gutenberg', 's', 'the', 'adventures', 'of', 'sherlock', 'holmes', 'by', ............................... , 'our', 'email', 'newsletter', 'to', 'hear', 'about', 'new', 'ebooks']

['project', 'gutenberg', 's', 'the', 'adventures']





it is not a lack of love, but a lack of

['the ', 'an ', 'such ', 'man ', 'present, ']

that which does not kill us makes us str

['ength ', 'uggle ', 'ong ', 'ange ', 'ive ']

i'm not upset that you lied to me, i'm u

['nder ', 'pon ', 'ses ', 't ', 'uder ']

and those who were seen dancing were tho

['se ', 're ', 'ugh ', ' servated ', 't ']it is hard enough to remember my opinion

[' of ', 's ', ', ', 'nof ', 'ed ']

**Result**

The results of the project will include performance metrics such as accuracy, precision, recall, and F1-score, evaluated on test datasets. Visualizations such as confusion matrices and learning curves will be provided to illustrate the model's behavior and effectiveness in real-world scenarios.

**Conclusion**

In conclusion, the word prediction machine learning project aims to address the challenges associated with text input by developing an efficient and accurate prediction system. Through systematic development approaches and rigorous evaluation, the project seeks to improve communication efficiency and enhance user experience across various applications.