**LGMVIP DATA SCIENCE**

**Komal Kaushik**

**TASK - Next Word Prediction**

## 1. Installing the libraries

!pip install numpy

!pip install tensorflow

!pip install keras

!pip install nltk

Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (1.19.5)

Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.6.0)

Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.12.1)

Requirement already satisfied: gast==0.4.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.4.0)

Requirement already satisfied: tensorflow-estimator~=2.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.6.0)

Requirement already satisfied: h5py~=3.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0)

Requirement already satisfied: grpcio<2.0,>=1.37.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.40.0)

Requirement already satisfied: six~=1.15.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0)

Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.2.0)

Requirement already satisfied: absl-py~=0.10 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.12.0)

Requirement already satisfied: tensorboard~=2.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.6.0)

Requirement already satisfied: numpy~=1.19.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.19.5)

Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.17.3)

Requirement already satisfied: astunparse~=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.6.3)

Requirement already satisfied: flatbuffers~=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.12)

Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.37.0)

Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0)

Requirement already satisfied: clang~=5.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (5.0)

Requirement already satisfied: termcolor~=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.0)

Requirement already satisfied: typing-extensions~=3.7.4 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.7.4.3)

Requirement already satisfied: keras~=2.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.6.0)

Requirement already satisfied: keras-preprocessing~=1.1.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)

Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py~=3.1.0->tensorflow) (1.5.2)

Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (1.8.0)

Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (0.6.1)

Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (2.23.0)

Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (1.35.0)

Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (3.3.4)

Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (57.4.0)

Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (0.4.6)

Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6->tensorflow) (1.0.1)

Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (0.2.8)

Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (4.2.2)

Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (4.7.2)

Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (1.3.0)

Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard~=2.6->tensorflow) (4.8.1)

Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (0.4.8)

Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2.10)

Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2021.5.30)

Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (1.24.3)

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (3.0.4)

Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (3.1.1)

Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->markdown>=2.6.8->tensorboard~=2.6->tensorflow) (3.5.0)

Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages (2.6.0)

Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.2.5)

Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from nltk) (1.15.0)

## 2. Importing the libraries

**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** **tensorflow.keras.optimizers** **import** RMSprop

**import** **matplotlib.pyplot** **as** **plt**

**import** **pickle**

**import** **heapq**

## 3. Loading the dataset

path = '1661-0.txt'

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

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

corpus length: 581888

## 4. Splitting the entire dataset into each word in order without the presence of special characters

tokenizer = RegexpTokenizer(r'\w+')

words = tokenizer.tokenize(text)

## 5. Dictionary(<key: value>) with each word form the unique\_words list as key and its corresponding position as value

unique\_words = np.unique(words)

unique\_word\_index = dict((c, i) **for** i, c **in** enumerate(unique\_words))

## 6. Feature engineering

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])

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

of

## 7. One-Hot encoding

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

print(X[0][0])

[False False False ... False False False]

## 8. Building the model

model = Sequential()

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

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

model.add(Activation('softmax'))

## 9. Training

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=10, shuffle=**True**).history

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/optimizer\_v2.py:356: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

"The `lr` argument is deprecated, use `learning\_rate` instead.")

Epoch 1/10

811/811 [==============================] - 185s 226ms/step - loss: 4.9602 - accuracy: 0.2857 - val\_loss: 8.4565 - val\_accuracy: 0.0970

Epoch 2/10

811/811 [==============================] - 178s 219ms/step - loss: 4.6376 - accuracy: 0.3281 - val\_loss: 8.6905 - val\_accuracy: 0.0842

Epoch 3/10

811/811 [==============================] - 177s 218ms/step - loss: 4.4276 - accuracy: 0.3676 - val\_loss: 9.0583 - val\_accuracy: 0.0817

Epoch 4/10

811/811 [==============================] - 177s 219ms/step - loss: 4.2731 - accuracy: 0.4005 - val\_loss: 8.9268 - val\_accuracy: 0.0857

Epoch 5/10

811/811 [==============================] - 178s 219ms/step - loss: 4.1585 - accuracy: 0.4302 - val\_loss: 9.1003 - val\_accuracy: 0.0815

Epoch 6/10

811/811 [==============================] - 179s 221ms/step - loss: 4.0426 - accuracy: 0.4580 - val\_loss: 9.0984 - val\_accuracy: 0.0731

Epoch 7/10

811/811 [==============================] - 178s 220ms/step - loss: 3.9590 - accuracy: 0.4804 - val\_loss: 9.1551 - val\_accuracy: 0.0732

Epoch 8/10

811/811 [==============================] - 178s 220ms/step - loss: 3.8924 - accuracy: 0.5009 - val\_loss: 9.2554 - val\_accuracy: 0.0714

Epoch 9/10

811/811 [==============================] - 182s 224ms/step - loss: 3.8429 - accuracy: 0.5183 - val\_loss: 9.2172 - val\_accuracy: 0.0663

Epoch 10/10

811/811 [==============================] - 181s 223ms/step - loss: 3.7827 - accuracy: 0.5337 - val\_loss: 9.3248 - val\_accuracy: 0.0688

## 10. Saving the model and loading it back

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"))

history

{'accuracy': [0.28571978211402893,

0.32805830240249634,

0.3675922155380249,

0.4004664719104767,

0.43023738265037537,

0.4579940140247345,

0.4804113507270813,

0.5009204149246216,

0.51833575963974,

0.5337368249893188],

'loss': [4.960208415985107,

4.637572288513184,

4.427563190460205,

4.273090839385986,

4.158470153808594,

4.042626857757568,

3.958982229232788,

3.892402172088623,

3.842939853668213,

3.782714366912842],

'val\_accuracy': [0.09703405201435089,

0.08421823382377625,

0.08165507018566132,

0.08568289875984192,

0.0814719870686531,

0.07305016368627548,

0.07323324680328369,

0.0714024156332016,

0.06627608835697174,

0.06883925199508667],

'val\_loss': [8.456538200378418,

8.690515518188477,

9.058286666870117,

8.926826477050781,

9.100311279296875,

9.09841251373291,

9.155068397521973,

9.255380630493164,

9.21721076965332,

9.324771881103516]}

## 11. Evaluation

plt.plot(history['accuracy'])

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

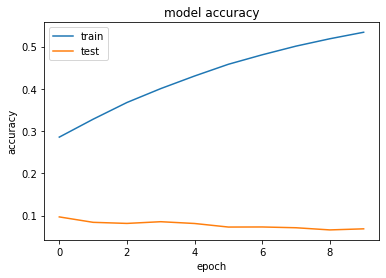
plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

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

<matplotlib.legend.Legend at 0x7f5e4dcb7cd0>



plt.plot(history['loss'])

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

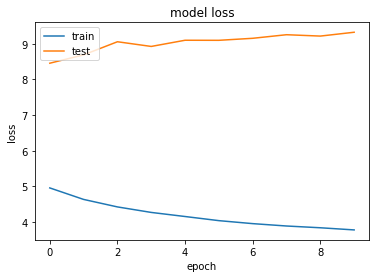
plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

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

<matplotlib.legend.Legend at 0x7f5e4de59e50>



## 12. Prediction

**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())

it

is

not

a

lack

array([[[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.]]])

**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\_completions(text, n=3):

**if** text == "":

**return**("0")

x = prepare\_input(text)

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

next\_indices = sample(preds, n)

**return** [unique\_words[idx] **for** idx **in** next\_indices]

q = "There is nothing more deceptive than an obvious fact"

print("correct sentence: ",q)

seq = " ".join(tokenizer.tokenize(q.lower())[0:5])

print("Sequence: ",seq)

print("next possible words: ", predict\_completions(seq, 5))

correct sentence: There is nothing more deceptive than an obvious fact

Sequence: there is nothing more deceptive

there

is

nothing

more

deceptive

next possible words: ['than', 'be', 'but', 'once', 'no']