```
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


from google.colab import files
uploaded = files.upload()
```

Choose Files 1661-0.txt

- **1661-0.txt**(text/plain) - 607791 bytes, last modified: 7/14/2022 - 100% done

```
Saving 1661-0.txt to 1661-0.txt
```

## Loading the data

```
text= open('1661-0.txt',encoding='UTF-8').read().lower()
print('corpus length:', len(text))
```

```
corpus length: 581888
```

## Splitting dataset into one word

```
tokenizer = RegexpTokenizer(r'\w+')
words = tokenizer.tokenize(text)
```

## Making list of sorted unique words

```
unique_words = np.unique(words)
unique_word_index = dict((c, i) for i, c in enumerate(unique_words))
```

## Feature Engineering and One Hot Encoding

```
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
```

Storing features and corresponding labels

```
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]
```

```
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).h
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/rmsprop.py:130: UserWarning
  super(RMSprop, self).__init__(name, **kwargs)
Epoch 1/2
811/811 [==============================] - 236s 288ms/step - loss: 6.0002 - accuracy
Epoch 2/2
811/811 [==============================] - 228s 281ms/step - loss: 5.7738 - accuracy
```

```
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"))
```

```
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("How are you ".lower())
```

```
how
are
you
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]
```

## Result

```
q =  "Do your work by your own instead of depending on someone"
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:  Do your work by your own instead of depending on someone
Sequence:  do your work by your
do
your
work
by
your
next possible words:  ['it', 'case', 'hands', 'house', 'way']
```

✓ 0s completed at 3:47 PM