VIDYA PAYGUDE

Data Science Intern at LetsGrowMore Virtual Internship Program (APRIL-2022)

ADVANCED LEVEL TASK 8 - Next Word Prediction

In [ ]:
```
import numpy as np
import pickle
import heapq
from nltk.tokenize import RegexpTokenizer
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import RMSprop
import matplotlib.pyplot as plt
```

In [ ]:
```
text = open('C:/Users/Lenovo/Desktop/textdata.txt',encoding='UTF-8').read().lower()
tr = RegexpTokenizer(r'\w+')
words = tr.tokenize(text)
unique_words = np.unique(words)
uw_index = dict((c, i) for i, c in enumerate(unique_words))
```

In [ ]:
```
len_w = 5
prev = []
nextw = []
for i in range(len(words) - len_w):
    prev.append(words[i:i + len_w])
    nextw.append(words[i + len_w])
print(prev[0],nextw[0])
```

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

In [ ]:
```
X = np.zeros((len(prev), len_w, len(unique_words)), dtype=bool)
Y = np.zeros((len(nextw), len(unique_words)), dtype=bool)
```

In [ ]:
```
for i, each_words in enumerate(prev):
    for j, each_word in enumerate(each_words):
        X[i, j, uw_index[each_word]] = 1
    Y[i, uw_index[nextw[i]]] = 1
```

In [ ]:
```
print(X[0][1])
```

```
[False False False ... False False False]
```

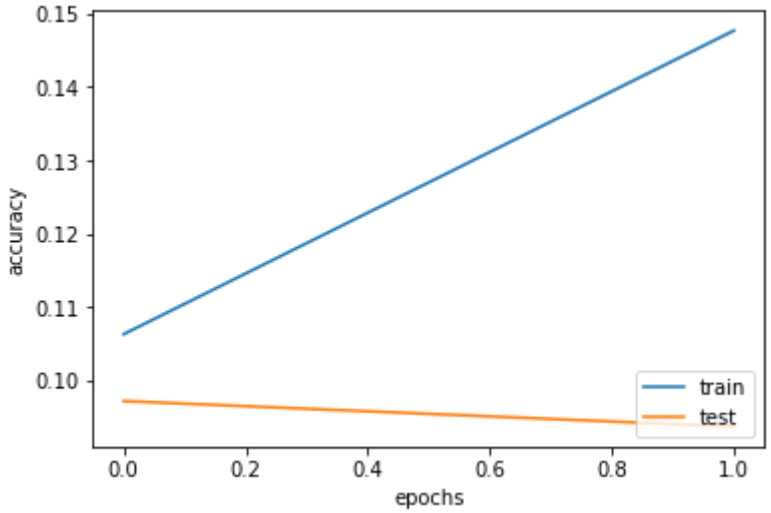
In [ ]:
```
model = Sequential()
model.add(LSTM(128, input_shape=(len_w, len(unique_words))))
model.add(Dense(len(unique_words)))
model.add(Activation('softmax'))
optimizer = RMSprop(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
history = model.fit(X, Y, validation_split=0.05, batch_size=128, epochs=2, shuffle=True).history
```

```
Train on 103759 samples, validate on 5462 samples
Epoch 1/2
103759/103759 [==============================] - 266s 3ms/sample - loss: 6.0194 - accuracy: 0.1063 - val_loss: 7.0757 - val_accura
cy: 0.0972
Epoch 2/2
103759/103759 [==============================] - 254s 2ms/sample - loss: 5.7708 - accuracy: 0.1477 - val_loss: 7.9357 - val_accura
cy: 0.0937
```

In [ ]:
```
model.save('word-pred-model.h5')
pickle.dump(history, open("history.p", "wb"))
model = load_model('word-pred-model.h5')
history = pickle.load(open("history.p", "rb"))

plt.plot(history['accuracy'])
plt.plot(history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc='lower right')
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2ae8f5de788>
```



In [ ]:

```
def prepare_input(text):
    x = np.zeros((1, len_w, len(unique_words)))
    for t, word in enumerate(text.split()):
        print(word)
        x[0, t, uw_index[word]] = 1
    return x

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_word(text, n=3):
    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]
```

In [ ]:

```
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:
    print("original sentence: ",q)
    seq = " ".join(tr.tokenize(q.lower())[0:5])
    print("sequence: ",seq)
    print("next possible words: ", predict_word(seq, 5))
```

```
original sentence:  It is not a lack of love, but a lack of friendship that makes unhappy marriages.
sequence:  it is not a lack
it
is
not
a
lack
next possible words:  ['of', 'which', 'man', 'a', 'that']
original sentence:  That which does not kill us makes us stronger.
sequence:  that which does not kill
that
which
does
not
kill
next possible words:  ['in', 'the', 'for', 'by', 'it']
original sentence:  I'm not upset that you lied to me, I'm upset that from now on I can't believe you.
sequence:  i m not upset that
i
m
```

```
not
upset
that
next possible words:  ['i', 'the', 'it', 'he', 'you']
original sentence:  And those who were seen dancing were thought to be insane by those who could not hear the music.
sequence:  and those who were seen
and
those
who
were
seen
next possible words:  ['in', 'which', 'and', 'with', 'upon']
original sentence:  It is hard enough to remember my opinions, without also remembering my reasons for them!
sequence:  it is hard enough to
it
is
hard
enough
to
next possible words:  ['be', 'think', 'have', 'me', 'see']
```