Importing Packages

```
import numpy as np
import pandas as pd
import json
import matplotlib.pyplot as plt
import re
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from keras.layers import Input, LSTM, Embedding, Dense
from keras.models import Model
```

Loading Data

```
data = []
with open('gigaword/train.jsonl','r') as file:
    for line in file:
        data.append(json.loads(line))

data = pd.DataFrame(data)

data.head(10)
```

```
                 id                                                text
\
0  gigaword-train-0  australia 's current account deficit shrunk by...

1  gigaword-train-1  at least two people were killed in a suspected...

2  gigaword-train-2  australian shares closed down #.# percent mond...

3  gigaword-train-3  south korea 's nuclear envoy kim sook urged no...

4  gigaword-train-4  south korea on monday announced sweeping tax r...

5  gigaword-train-5  taiwan share prices closed down #.## percent m...

6  gigaword-train-6  australian shares closed down #.# percent mond...

7  gigaword-train-7  spanish property group colonial , struggling u...

8  gigaword-train-8  libyan leader moamer kadhafi monday promised w...

9  gigaword-train-9  the united nations ' humanitarian chief john h...


                                             summary
0  australian current account deficit narrows sha...
1    at least two dead in southern philippines blast
2           australian stocks close down #.# percent
3  envoy urges north korea to restart nuclear dis...
```

```
4     skorea announces tax cuts to stimulate economy
5               taiwan shares close down #.## percent
6            australian stocks close down #.# percent
7     spain 's colonial posts #.## billion euro loss
8   kadhafi promises wide political economic reforms
9  un 's top aid official arrives in drought-hit ...
```

Data Pre-Processing

```
##Lowercasing
data = data.applymap(lambda x: x.lower())

## Removing `#`
data = data.applymap(lambda x: str(x).replace('#', ''))

## Adding `start` nd `end` tokens to the summaries
data['summary'] = data['summary'].apply(

    lambda x: 'START_ ' + x + ' _END'

)

data['summary'].head(2)
```

```
0    START_ australian current account deficit narr...
1    START_ at least two dead in southern philippin...
Name: summary, dtype: object
```

```
data = data[:10000]
```

Vocabulary Creation

```
vocab_text = set()
for t in data['text']:
    for word in t.split():
        if word not in vocab_text:
            vocab_text.add(word)

len(vocab_text)
```

```
15845
```

```
vocab_summary = set()
for t in data['summary']:
    for word in t.split():
        if word not in vocab_summary:
            vocab_summary.add(word)

len(vocab_summary)
```

```
8757
```

Data Params Declaration

```
input_words = sorted(list(vocab_text))
target_words = sorted(list(vocab_summary))
num_encoder_tokens = len(vocab_text)
num_decoder_tokens = len(target_words)

num_encoder_tokens,num_decoder_tokens
```

```
(15845, 8757)
```

```
data['length_text_sentence']=data['text'].apply(lambda x:len(x.split("
")))
data['length_sum_sentence']=data['summary'].apply(lambda
x:len(x.split(" ")))

max_length_src = data['length_text_sentence'].max()
max_length_tar = data['length_sum_sentence'].max()

max_length_src,max_length_tar
```

```
(59, 25)
```

```
##For zero-pad
num_decoder_tokens+=1
```

Index Creation

```
input_token_index = dict([(word, i+1) for i, word in
enumerate(input_words)])
target_token_index = dict([(word, i+1) for i, word in
enumerate(target_words)])

reverse_input_char_index = dict((i, word) for word, i in
input_token_index.items())
reverse_target_char_index = dict((i, word) for word, i in
target_token_index.items())
```

Train and Test Split

```
x, y = data['text'], data['summary']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size =
0.2,random_state=9)

X_train.shape, X_test.shape
```

```
((8000,), (2000,))
```

```
##Save the data for future purpose
X_train.to_pickle('X_train.pkl')
X_test.to_pickle('X_test.pkl')
```

Batch Generation

```
def generate_batch(X = X_train, y = y_train, batch_size = 128):
    while True:
        for j in range(0, len(X), batch_size):
            encoder_input_data = np.zeros((batch_size,
max_length_src),dtype='float32')
            decoder_input_data = np.zeros((batch_size,
max_length_tar),dtype='float32')
            decoder_target_data = np.zeros((batch_size,
max_length_tar, num_decoder_tokens),dtype='float32')
            for i, (input_text, target_text) in
enumerate(zip(X[j:j+batch_size], y[j:j+batch_size])):
                for t, word in enumerate(input_text.split()):
                    encoder_input_data[i, t] = input_token_index[word]
##encoder input seq
                for t, word in enumerate(target_text.split()):
                    if t<len(target_text.split())-1:
                        decoder_input_data[i, t] =
target_token_index[word] ##decoder input seq
                    if t>0:
                        ##decoder target sequence (one hot encoded)
                        ##does not include the START_ token
                        ##Offset by one timestep
                        decoder_target_data[i, t - 1,
target_token_index[word]] = 1.
            yield([encoder_input_data, decoder_input_data],
decoder_target_data)
```

```
for j in range(0, len(X_train), 128):
            encoder_input_data = np.zeros((batch_size,
max_length_src),dtype='float32')
            decoder_input_data = np.zeros((batch_size,
max_length_tar),dtype='float32')
            decoder_target_data = np.zeros((batch_size,
max_length_tar, num_decoder_tokens),dtype='float32')
            for i, (input_text, target_text) in
enumerate(zip(X_train[j:j+batch_size], y_train[j:j+batch_size])):
                for t, word in enumerate(input_text.split()):
                    encoder_input_data[i, t] = input_token_index[word]
##encoder input seq
                for t, word in enumerate(target_text.split()):
                    if t<len(target_text.split())-1:
                        decoder_input_data[i, t] =
target_token_index[word] ##decoder input seq
                    if t>0:
                        ##decoder target sequence (one hot encoded)
                        ##does not include the START_ token
                        ##Offset by one timestep
```

```
                        decoder_target_data[i, t - 1,
target_token_index[word]] = 1.
```

Encoder Decoder Architecture

```
hidden_state = 300
```

Encoder

```
encoder_inputs = Input(shape=(None,))
enc_emb =  Embedding(num_encoder_tokens, hidden_state, mask_zero =
True)(encoder_inputs)
encoder_lstm = LSTM(hidden_state, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(enc_emb)
encoder_states = [state_h, state_c]
```

Decoder

```
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(num_decoder_tokens, hidden_state, mask_zero
= True)
dec_emb = dec_emb_layer(decoder_inputs)
decoder_lstm = LSTM(hidden_state, return_sequences=True,
return_state=True)
decoder_outputs, _, _ = decoder_lstm(dec_emb,
                                     initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

Model

```
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model.compile(optimizer='rmsprop', loss='categorical_crossentropy')

model.summary()
```

```
Model: "model_3"
__________________________________________________________________________
_____________________________
 Layer (type)                Output Shape              Param #
Connected to
==========================================================================
============================
 input_5 (InputLayer)        [(None, None)]            0         []



 input_6 (InputLayer)        [(None, None)]            0         []
```

```
 embedding_2 (Embedding)     (None, None, 300)            4753500
['input_5[0][0]']


 embedding_3 (Embedding)     (None, None, 300)            2627400
['input_6[0][0]']


 lstm_2 (LSTM)               [(None, 300),                721200
['embedding_2[0][0]']
                              (None, 300),

                              (None, 300)]



 lstm_3 (LSTM)               [(None, None, 300),          721200
['embedding_3[0][0]',
                              (None, 300),
'lstm_2[0][1]',
                              (None, 300)]
'lstm_2[0][2]']


 dense_1 (Dense)             (None, None, 8758)           2636158
['lstm_3[0][0]']


==================================================================
============================
Total params: 11459458 (43.71 MB)
Trainable params: 11459458 (43.71 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________
_____________________________
```

Training Config

```
train_samples = len(X_train)
val_samples = len(X_test)
batch_size = 64
epochs = 100
```

Training

```
model.fit([encoder_input_data, decoder_input_data],
decoder_target_data,
          batch_size = batch_size,
          epochs = epochs,
```

```
          validation_split=0.2,
         steps_per_epoch = train_samples//batch_size)

Epoch 1/100
100/125 [=====================>......] - ETA: 8s - loss:
2.6591WARNING:tensorflow:Your input ran out of data; interrupting
training. Make sure that your dataset or generator can generate at
least `steps_per_epoch * epochs` batches (in this case, 12500
batches). You may need to use the repeat() function when building your
dataset.
125/125 [==============================] - 34s 274ms/step - loss:
2.6591 - val_loss: 11.0432

<keras.src.callbacks.History at 0x2726ff10690>
```

Inference

```
model.save_weights('n_sq_comp_t_weights.h5')

latent_dim = 300

encoder_model = Model(encoder_inputs, encoder_states)

decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]

dec_emb2= dec_emb_layer(decoder_inputs) ##Get the embeddings of the
decoder sequence

decoder_outputs2, state_h2, state_c2 = decoder_lstm(dec_emb2,
initial_state=decoder_states_inputs)
decoder_states2 = [state_h2, state_c2]
decoder_outputs2 = decoder_dense(decoder_outputs2) ##A dense softmax
layer to generate prob dist. over the target vocabulary

##Decoder
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs2] + decoder_states2)

def decode_sequence(input_seq):
    ##Encode the input as state vectors.
    states_value = encoder_model.predict(input_seq)
    ##Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))
    target_seq[0, 0] = target_token_index['START_']
    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] +
```

```
states_value)
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = reverse_target_char_index[sampled_token_index]
        decoded_sentence += ' '+sampled_char
        if (sampled_char == '_END' or
           len(decoded_sentence) > 50):
            stop_condition = True

        ##Update the target sequence (of length 1).
        target_seq = np.zeros((1,1))
        target_seq[0, 0] = sampled_token_index

        ##Update states
        states_value = [h, c]

    return decoded_sentence

train_gen = generate_batch(X_train, y_train, batch_size = 1)
k=-1

k+=6
(input_seq, actual_output), _ = next(train_gen)
decoded_sentence = decode_sequence(input_seq)
print('Input Text sentence:', X_train[k:k+1].values[0])
print('Actual Summary:', y_train[k:k+1].values[0][6:-4])
print('Predicted Summary:', decoded_sentence[:-4])
```

```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
Input Text sentence: china 's only wholly chinese-owned passenger car
production line has launched an expansion project to more than triple
production in  , the overseas edition of the people 's daily said
saturday .
Actual Summary:  tianjin automobile launches massive expansion drive
Predicted Summary:  us condemns of markets in on bid
```