```
import numpy as np
import pandas as pd
import json
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding,LSTM,Dense,SimpleRNN
```

```
from google.colab import drive

drive.mount('/content/drive')
file='/content/drive/MyDrive/train.jsonl'

with open(file,'r') as f:
  train_data=[json.loads(line) for line in f]
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df=pd.DataFrame(train_data)
```

```
df
```

	id	text	summary
0	gigaword-train-0	australia 's current account deficit shrunk by...	australian current account deficit narrows sha...
1	gigaword-train-1	at least two people were killed in a suspected...	at least two dead in southern philippines blast
2	gigaword-train-2	australian shares closed down #.# percent mond...	australian stocks close down #.# percent
3	gigaword-train-3	south korea 's nuclear envoy kim sook urged no...	envoy urges north korea to restart nuclear dis...
4	gigaword-train-4	south korea on monday announced sweeping tax r...	skorea announces tax cuts to stimulate economy
...	...	...	...
999995	gigaword-train-999995	after proclaiming a special relationship with ...	indian leader vajpayee to meet with bush to di...
999996	gigaword-train-999996	a group of people expelled by the british from...	former residents of indian ocean island demand...

```
token=Tokenizer()
token.fit_on_texts(d['text'] for d in train_data)
```

```
train_seq=token.texts_to_sequences(d['text'] for d in train_data)
train_sum=token.texts_to_sequences(d['summary'] for d in train_data)
```

```
def max_len_sequences(dataset):
    return max([len(seq) for seq in dataset])


print(f"Maximum length of sequences in train:{max_len_sequences(train_seq)}")
```

```
Maximum length of sequences in train:82
```

```
max_length=82

train_seq=pad_sequences(train_seq,maxlen=max_length,padding='post')
train_sum=pad_sequences(train_sum,maxlen=max_length,padding='post')
```

```
max_length = max([len(seq) for seq in train_seq])
```

```
max_length
```

```
82
```

```
vocab_size=len(token.word_index)+1
embedding_dim=100
hidden_units=128

rnn=Sequential()
rnn.add(Embedding(vocab_size,max_length))
rnn.add(SimpleRNN(hidden_units,return_sequences=True))
#rnn.add(hidden_units,return_sequences=True)
rnn.add(Dense(vocab_size,activation='softmax'))

rnn.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
train_seq_subset = train_seq[:1000]
train_sum_subset = train_sum[:1000]
```

```
rnn.fit(train_seq_subset,train_sum_subset,epochs=20,batch_size=32)
```

```
Epoch 1/20
32/32 [==============================] - 12s 333ms/step - loss: 8.5705 - accuracy: 0.8355
Epoch 2/20
32/32 [==============================] - 9s 292ms/step - loss: 1.9859 - accuracy: 0.9057
Epoch 3/20
32/32 [==============================] - 10s 292ms/step - loss: 0.9965 - accuracy: 0.9057
Epoch 4/20
32/32 [==============================] - 8s 265ms/step - loss: 0.9652 - accuracy: 0.9057
Epoch 5/20
32/32 [==============================] - 8s 250ms/step - loss: 0.9544 - accuracy: 0.9057
Epoch 6/20
32/32 [==============================] - 9s 266ms/step - loss: 1.0226 - accuracy: 0.9030
Epoch 7/20
32/32 [==============================] - 8s 253ms/step - loss: 1.0176 - accuracy: 0.9030
Epoch 8/20
32/32 [==============================] - 8s 245ms/step - loss: 0.9612 - accuracy: 0.9048
Epoch 9/20
32/32 [==============================] - 8s 262ms/step - loss: 0.9506 - accuracy: 0.9049
Epoch 10/20
32/32 [==============================] - 8s 241ms/step - loss: 0.8933 - accuracy: 0.9050
Epoch 11/20
32/32 [==============================] - 8s 238ms/step - loss: 1.1292 - accuracy: 0.8965
Epoch 12/20
32/32 [==============================] - 8s 257ms/step - loss: 0.8844 - accuracy: 0.9059
Epoch 13/20
32/32 [==============================] - 7s 225ms/step - loss: 0.8714 - accuracy: 0.9050
Epoch 14/20
32/32 [==============================] - 8s 248ms/step - loss: 0.7989 - accuracy: 0.9059
Epoch 15/20
32/32 [==============================] - 7s 230ms/step - loss: 0.7430 - accuracy: 0.9061
Epoch 16/20
32/32 [==============================] - 8s 251ms/step - loss: 0.7033 - accuracy: 0.9063
Epoch 17/20
32/32 [==============================] - 8s 244ms/step - loss: 0.6779 - accuracy: 0.9065
Epoch 18/20
32/32 [==============================] - 7s 226ms/step - loss: 0.6613 - accuracy: 0.9069
Epoch 19/20
32/32 [==============================] - 8s 240ms/step - loss: 0.6486 - accuracy: 0.9068
Epoch 20/20
32/32 [==============================] - 7s 234ms/step - loss: 0.6382 - accuracy: 0.9074
<keras.src.callbacks.History at 0x796d8bc86740>
```

```
vocab_size=len(token.word_index)+1
embedding_dim=100
hidden_units=128

lstm=Sequential()
lstm.add(Embedding(vocab_size,max_length))
lstm.add(LSTM(hidden_units,return_sequences=True))
#lstm.add(hidden_units,return_sequences=True)
lstm.add(Dense(vocab_size,activation='softmax'))
lstm.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
lstm.fit(train_seq_subset,train_sum_subset,epochs=20,batch_size=32)
```

```
Epoch 1/20
32/32 [==============================] - 10s 234ms/step - loss: 9.4223 - accuracy: 0.8700
Epoch 2/20
32/32 [==============================] - 7s 226ms/step - loss: 2.0950 - accuracy: 0.9057
Epoch 3/20
```

```
32/32 [==============================] - 7s 226ms/step - loss: 0.9904 - accuracy: 0.9057
Epoch 4/20
32/32 [==============================] - 7s 216ms/step - loss: 0.9613 - accuracy: 0.9057
Epoch 5/20
32/32 [==============================] - 7s 210ms/step - loss: 0.9421 - accuracy: 0.9057
Epoch 6/20
32/32 [==============================] - 6s 199ms/step - loss: 0.9084 - accuracy: 0.9057
Epoch 7/20
32/32 [==============================] - 6s 197ms/step - loss: 0.8637 - accuracy: 0.9057
Epoch 8/20
32/32 [==============================] - 6s 197ms/step - loss: 0.8120 - accuracy: 0.9057
Epoch 9/20
32/32 [==============================] - 7s 205ms/step - loss: 0.7682 - accuracy: 0.9057
Epoch 10/20
32/32 [==============================] - 6s 196ms/step - loss: 0.7387 - accuracy: 0.9057
Epoch 11/20
32/32 [==============================] - 7s 212ms/step - loss: 0.7148 - accuracy: 0.9057
Epoch 12/20
32/32 [==============================] - 6s 196ms/step - loss: 0.6969 - accuracy: 0.9057
Epoch 13/20
32/32 [==============================] - 6s 199ms/step - loss: 0.6832 - accuracy: 0.9062
Epoch 14/20
32/32 [==============================] - 6s 191ms/step - loss: 0.6726 - accuracy: 0.9062
Epoch 15/20
32/32 [==============================] - 6s 200ms/step - loss: 0.6619 - accuracy: 0.9067
Epoch 16/20
32/32 [==============================] - 6s 194ms/step - loss: 0.6528 - accuracy: 0.9070
Epoch 17/20
32/32 [==============================] - 6s 190ms/step - loss: 0.6466 - accuracy: 0.9070
Epoch 18/20
32/32 [==============================] - 6s 196ms/step - loss: 0.6384 - accuracy: 0.9074
Epoch 19/20
32/32 [==============================] - 6s 192ms/step - loss: 0.6317 - accuracy: 0.9073
Epoch 20/20
32/32 [==============================] - 6s 193ms/step - loss: 0.6264 - accuracy: 0.9076
<keras.src.callbacks.History at 0x796d8fa922f0>
```

```python
def generate_summary(text):
  seq=token.texts_to_sequences([text])
  seq=pad_sequences(seq,maxlen=max_length,padding='post')
  summary=lstm.predict(seq)[0]
  dec_summary=' '.join([token.index_word.get(idx,'?') for idx in summary[summary>0]])
  return dec_summary
```

```python
text = "New jobless numbers are a bit of a mixed bag for President Obama and his re-election bid."
summary = generate_summary(text)
print(f"Original text: {text}")
print(f"Summary: {summary}")
```

```
WARNING:tensorflow:5 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x796d8fa4c940> trigg
1/1 [==============================] - 0s 364ms/step
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```python
summary
```

```
'? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
```