```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, LSTM, Embedding, Bidirectional, TimeDistributed, BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from google.colab import drive
import json
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
tagged_sents = []
with open('/content/drive/MyDrive/Cleaned_Sentences_Task/cleaned_sentences.txt', 'r') as file:
    tagged_sents = json.load(file)
```

```
print(tagged_sents[0])
```

```
[['आग', 'nn'], ['की', 'psp']]
```

```
tagged_sentences = []
for i in tagged_sents:
    tagged_sentences.append([tuple(i[0]), tuple(i[1])])
```

```
print(tagged_sentences[0])
```

```
[('आग', 'nn'), ('की', 'psp')]
```

```
sentences, tags = [], []
for i in tagged_sentences:
  curr_sent, curr_tag = [], []
  for word, tag in i:
    curr_sent.append(word)
    curr_tag.append(tag.lower())
  sentences.append(curr_sent)
```

```
  tags.append(curr_tag)
```

```
print(sentences[0])
print(tags[0])
```

```
['आग', 'की']
['nn', 'psp']
```

```
tokenizer_sents = Tokenizer()
tokenizer_sents.fit_on_texts(sentences)
tokenized_sents = tokenizer_sents.texts_to_sequences(sentences)
```

```
print(tokenized_sents[0])
```

```
[632, 3]
```

```
tokenizer_tags = Tokenizer()
tokenizer_tags.fit_on_texts(tags)
tokenized_tags = tokenizer_tags.texts_to_sequences(tags)
```

```
tokenized_tags[0]
```

```
[1, 2]
```

```
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer_sents.word_index) + 1, output_dim=128, input_length=2, trainable=True))
model.add(Bidirectional(LSTM(512, return_sequences=True, activation='relu')))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(256, return_sequences=True, activation='relu')))
model.add(Bidirectional(LSTM(128, return_sequences=True, activation='relu')))
model.add(Dropout(0.2))
model.add(TimeDistributed(Dense(128, activation='relu')))
model.add(Dropout(0.2))
model.add(TimeDistributed(Dense(64, activation='relu')))
model.add(TimeDistributed(Dense(len(tokenizer_tags.word_index) + 1, activation='softmax')))
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 2, 128)            2589952   
_________________________________________________________________
bidirectional_3 (Bidirection (None, 2, 1024)           2625536   
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 1024)           0         
_________________________________________________________________
bidirectional_4 (Bidirection (None, 2, 512)            2623488   
_________________________________________________________________
bidirectional_5 (Bidirection (None, 2, 256)            656384    
_________________________________________________________________
dropout_4 (Dropout)          (None, 2, 256)            0         
_________________________________________________________________
time_distributed_2 (TimeDist (None, 2, 128)            32896     
_________________________________________________________________
dropout_5 (Dropout)          (None, 2, 128)            0         
_________________________________________________________________
time_distributed_3 (TimeDist (None, 2, 64)             8256      
_________________________________________________________________
time_distributed_4 (TimeDist (None, 2, 34)             2210      
=================================================================
Total params: 8,538,722
Trainable params: 8,538,722
Non-trainable params: 0
_________________________________________________________________
```

```
padded_sents = pad_sequences(tokenized_sents, maxlen=2, padding='post')
padded_tags = pad_sequences(tokenized_tags, maxlen=2, padding='post')
```

```
model.fit(padded_sents[0: -4000], to_categorical(padded_tags[0: -4000]), batch_size=128, epochs=20, validation_split=0.2)
```

```
Epoch 1/20
```

```
2047/2047 [==============================] - 634s 310ms/step - loss: 0.6630 - acc: 0.7968 - val_loss: 0.2966 - val_acc: 0.9115
Epoch 2/20
2047/2047 [==============================] - 637s 311ms/step - loss: 0.2526 - acc: 0.9234 - val_loss: 0.2358 - val_acc: 0.9249
Epoch 3/20
2047/2047 [==============================] - 623s 305ms/step - loss: 0.2044 - acc: 0.9354 - val_loss: 0.2252 - val_acc: 0.9294
Epoch 4/20
2047/2047 [==============================] - 622s 304ms/step - loss: 0.1802 - acc: 0.9411 - val_loss: 0.2077 - val_acc: 0.9316
Epoch 5/20
2047/2047 [==============================] - 624s 305ms/step - loss: 0.1615 - acc: 0.9457 - val_loss: 0.2050 - val_acc: 0.9339
Epoch 6/20
2047/2047 [==============================] - 631s 308ms/step - loss: 0.1474 - acc: 0.9493 - val_loss: 0.2135 - val_acc: 0.9348
Epoch 7/20
2047/2047 [==============================] - 628s 307ms/step - loss: 0.1363 - acc: 0.9526 - val_loss: 0.2172 - val_acc: 0.9351
Epoch 8/20
2047/2047 [==============================] - 631s 308ms/step - loss: 0.1282 - acc: 0.9550 - val_loss: 0.2169 - val_acc: 0.9350
Epoch 9/20
2047/2047 [==============================] - 626s 306ms/step - loss: 0.1210 - acc: 0.9568 - val_loss: 0.2223 - val_acc: 0.9360
Epoch 10/20
2047/2047 [==============================] - 631s 308ms/step - loss: 0.1153 - acc: 0.9583 - val_loss: 0.2346 - val_acc: 0.9358
Epoch 11/20
2047/2047 [==============================] - 632s 309ms/step - loss: 0.1096 - acc: 0.9601 - val_loss: 0.2389 - val_acc: 0.9345
Epoch 12/20
2047/2047 [==============================] - 632s 309ms/step - loss: 0.1048 - acc: 0.9612 - val_loss: 0.2343 - val_acc: 0.9358
Epoch 13/20
2047/2047 [==============================] - 638s 312ms/step - loss: 0.1009 - acc: 0.9625 - val_loss: 0.2518 - val_acc: 0.9361
Epoch 14/20
2047/2047 [==============================] - 629s 307ms/step - loss: 0.0973 - acc: 0.9638 - val_loss: 0.2443 - val_acc: 0.9370
Epoch 15/20
2047/2047 [==============================] - 637s 311ms/step - loss: 0.0937 - acc: 0.9648 - val_loss: 0.2573 - val_acc: 0.9358
Epoch 16/20
2047/2047 [==============================] - 636s 311ms/step - loss: 0.0906 - acc: 0.9656 - val_loss: 0.2709 - val_acc: 0.9367
Epoch 17/20
2047/2047 [==============================] - 631s 308ms/step - loss: 0.0883 - acc: 0.9663 - val_loss: 0.2702 - val_acc: 0.9347
Epoch 18/20
2047/2047 [==============================] - 638s 312ms/step - loss: 0.0859 - acc: 0.9672 - val_loss: 0.2798 - val_acc: 0.9359
Epoch 19/20
2047/2047 [==============================] - 641s 313ms/step - loss: 0.0836 - acc: 0.9680 - val_loss: 0.2768 - val_acc: 0.9351
Epoch 20/20
2047/2047 [==============================] - 640s 313ms/step - loss: 0.0815 - acc: 0.9688 - val_loss: 0.2732 - val_acc: 0.9356
<tensorflow.python.keras.callbacks.History at 0x7fdf0f1cd7f0>
```

```
print("Accuracy of the model =", model.evaluate(padded_sents[-4000: ], to_categorical(padded_tags[-4000: ], num_classes=len(tokenizer_tags.word_index) +
```

```
125/125 [==============================] - 4s 29ms/step - loss: 0.2738 - acc: 0.9352
125/125 [==============================] - 4s 29ms/step - loss: 0.2738 - acc: 0.9352
Accuracy of the model = 0.9352499842643738
```

```
actual = [list(i) for i in padded_tags[-5: ]]
actuals = []
for i in actual:
  curr = []
  for v in i:
    curr.append(int(v))
  actuals.append(curr)
```

```
predictions = []
for i in model.predict_classes(padded_sents[-5: ]):
  curr = []
  for v in i:
    curr.append(int(v))
  predictions.append(curr)
print(predictions)
```

```
WARNING:tensorflow:From <ipython-input-30-f870f4c723de>:2: Sequential.predict_classes (from tensorflow.python.keras.engine.sequ
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses
[[7, 7], [8, 10], [8, 24], [2, 5], [4, 2]]
```

```
pred = tokenizer_tags.sequences_to_texts(predictions)
act = tokenizer_tags.sequences_to_texts(actuals)
print(pred)
print(act)
```

```
['prp prp', 'cc nnc', 'cc inj', 'psp jj', 'nnp psp']
['prp prp', 'cc nnc', 'cc inj', 'psp jj', 'nnp psp']
```

```
predictions = model.predict_classes(padded_sents)
```

```
predict = [list(i) for i in predictions]
final_predict = tokenizer_tags.sequences_to_texts(predict)
```

```
rnn_tags = []
for i in final_predict:
  rnn_tags.extend(i.split(" "))
```

```
print(len(tags) * 2)
print(len(rnn_tags))
```

```
662728
662728
```

```
with open ('/content/drive/MyDrive/Cleaned_Sentences_Task/rnn_predictions.txt', 'w+') as file:
    json.dump(rnn_tags, file)
```