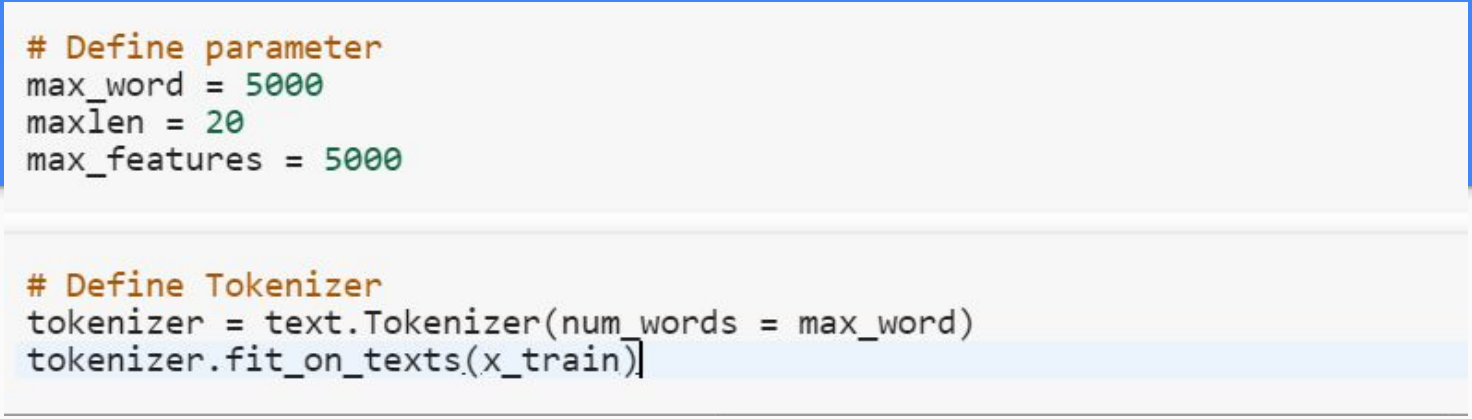
Train RNN_LSTM

ชื่อรองงานนำเสนอ

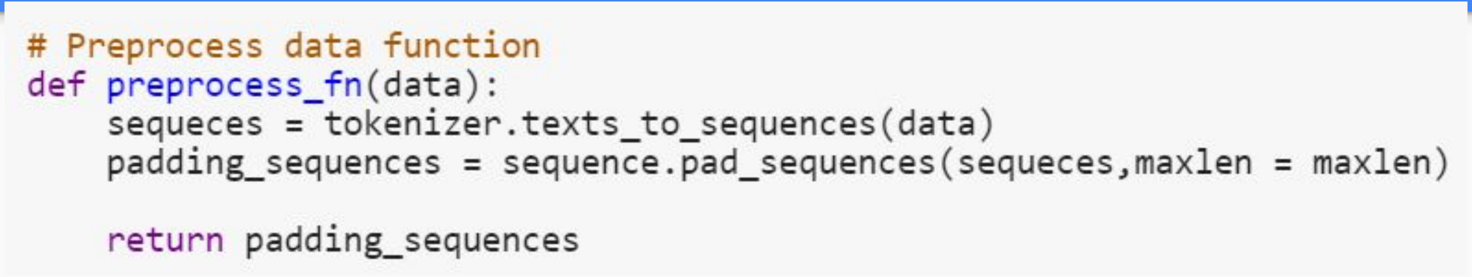


```
# Define parameter
max_word = 5000
maxlen = 20
max_features = 5000
```

```
# Define Tokenizer
tokenizer = text.Tokenizer(num_words = max_word)
tokenizer.fit_on_texts(x_train)
```

Ref::http://faroit.com/keras-docs/1.2.2/preprocessing/text/

text.Tokenizer is define corpus to keep maximum number of words (if set, tokenization will be restricted to the top nb_words most common words in the dataset)



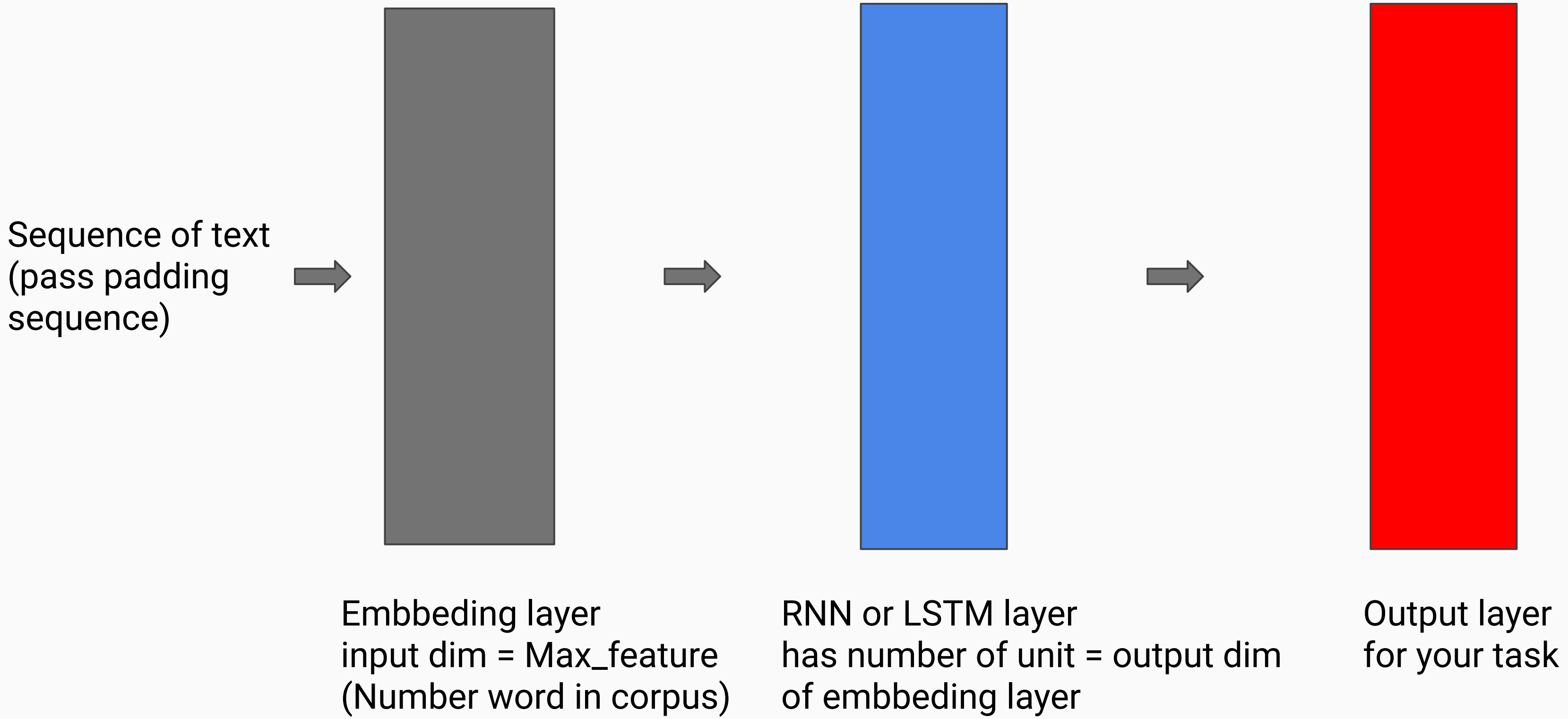
```python
# Preprocess data function
def preprocess_fn(data):
    sequeces = tokenizer.texts_to_sequences(data)
    padding_sequences = sequence.pad_sequences(sequeces,maxlen = maxlen)

    return padding_sequences
```

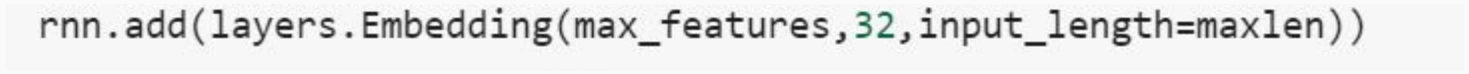
.texts_to_sequences list of texts to turn to sequences define by word_index

.pad_sequences Transform a list of sequences (lists of scalars) into a 2D Numpy array of shape max len

SImple Alchetecture

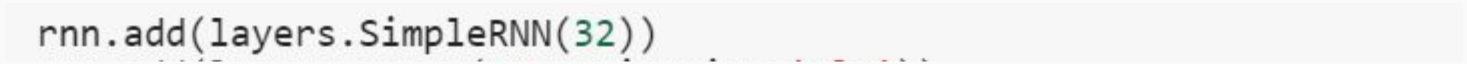

Sequence of text (pass padding sequence)
Embbeding layer input dim = Max_feature (Number word in corpus)
RNN or LSTM layer has number of unit = output dim of embbeding layer
Output layer for your task

keras.layers.Embedding(**input_dim,output_dim**,embeddings_initializer='uniform
', embeddings_regularizer=None, activity_regularizer=None,
embeddings_constraint=None, mask_zero=False, input_length=None)



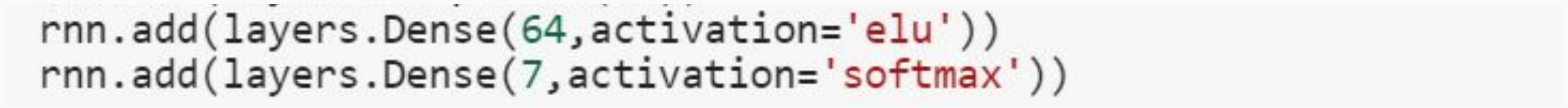
```
rnn.add(layers.Embedding(max_features,32,input_length=maxlen))
```

keras.layers.SimpleRNN(**units**, activation='tanh', use_bias=True, kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal', bias_initializer='zeros', kernel_regularizer=None, recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, recurrent_constraint=None, bias_constraint=None, dropout=0.0, recurrent_dropout=0.0, return_sequences=False, return_state=False, go_backwards=False, stateful=False, unroll=False)



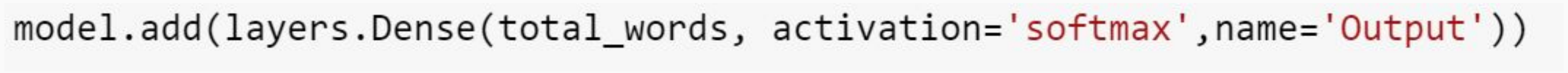
```
rnn.add(layers.SimpleRNN(32))
```

For Classification task :



```
rnn.add(layers.Dense(64,activation='elu'))
rnn.add(layers.Dense(7,activation='softmax'))
```

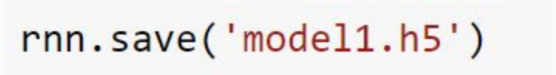
For Generate sentence task :
in the last layer in LSTM or RNN:



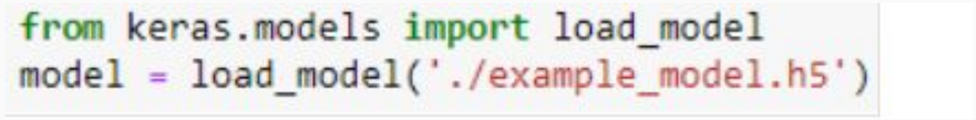
```
model.add(layers.Dense(total_words, activation='softmax',name='Output'))
```

Total word = len(Tokenizer.word_index) + 1 keras.layers.LSTM(**units**, activation='tanh', recurrent_activation='sigmoid', use_bias=True, kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal', bias_initializer='zeros', unit_forget_bias=True, kernel_regularizer=None, recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None, kernel_constraint=None, recurrent_constraint=None, bias_constraint=None, dropout=0.0, recurrent_dropout=0.0, implementation=2, return_sequences=False, return_state=False, go_backwards=False, stateful=False, unroll=False)

Check point



```python
rnn.save('model1.h5')
```



```python
from keras.models import load_model
model = load_model('./example_model.h5')
```

Google **Colab** tool is a Jupyter notebook-based system integrated with Google Drive. Note that when you connect to a GPU-based VM runtime, **you are given a maximum of 12 hours at a time on the VM.**