```
import string
import re
from numpy import array,argmax,random,take
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense,LSTM,Embedding,Bidirectional,RepeatVector,TimeDist
from keras.preprocessing.text import Tokenizer
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.sequence import pad_sequences
from keras.models import load_model
from keras import optimizers
import matplotlib.pyplot as plt
% matplotlib inline
pd.set_option('display.max_colwidth',200)
```

```
def read_text(filename):
    file=open(filename,mode='rt',encoding='utf-8')
    text=file.read()
    file.close()
    return text


def to_lines(text):
    sents=text.strip().split('/n')
    sents=[i.split('\t') for i in sents]
    return sents


data = read_text("/content/deu.txt")
deu_eng=to_lines(data)
deu_eng=array(deu_eng)


deu_eng = deu_eng[:150000,:]


deu_eng
```

```
     array([['Go.', 'Geh.',
             'CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8597805 (Roujin)\r
             ...,
             'CC-BY 2.0 (France) Attribution: tatoeba.org #3847634 (CM) & #4878147 (Pfirsichb
             'Ohne Zweifel findet sich auf dieser Welt zu jedem Mann genau die richtige Ehefr
             'CC-BY 2.0 (France) Attribution: tatoeba.org #7697649 (RM) & #7729416 (Pfirsichb
           dtype='<U626')
```

```
eng_l= []
deu_l= []
```

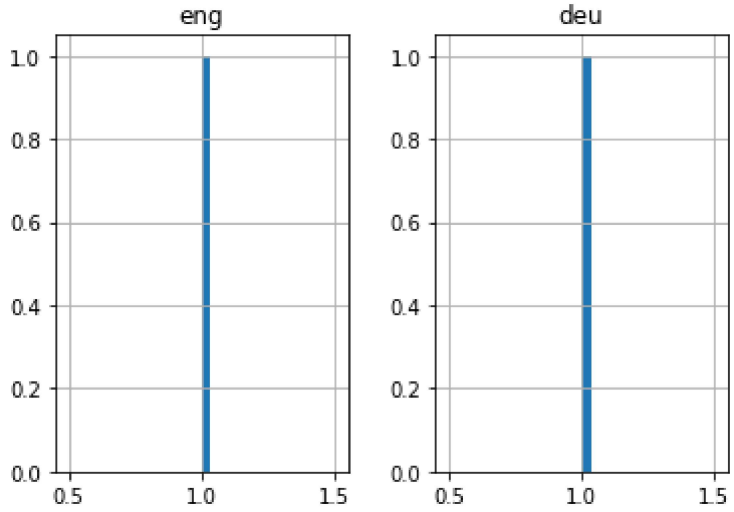
```
for i in deu_eng[:,0]:
  eng_l.append(len(i.split()))

for i in deu_eng[:,1]:
  deu_l.append(len(i.split()))


length_df = pd.DataFrame({'eng':eng_l,'deu':deu_l})



length_df.hist(bins=30)
plt.show()
```




```
def tokenization(lines):
  tokenizer=Tokenizer()
  tokenizer.fit_on_texts(lines)
  return tokenizer



eng_tokenizer =tokenization(deu_eng[:,0])
eng_vocab_size=len(eng_tokenizer.word_index)+1

eng_length=8
print('English Vocabulary Size: %d' %eng_vocab_size)
```

```
     English Vocabulary Size: 2
```

```
deu_tokenizer=tokenization(deu_eng[:, 1])
deu_vocab_size=len(deu_tokenizer.word_index) + 1

deu_length=8
print('Deutch Vocabulary Size : %d' % deu_vocab_size)
```

```
Deutch Vocabulary Size : 2
```

```
def encode_sequences(tokenizer,length,lines):
  seq=tokenizer.texts_to_sequences(lines)
  seq=pad_sequences(seq, maxlen=length,padding='post')
  return seq
```

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(deu_eng, test_size=0.2 , random_state=12)
```

```
trainX= encode_sequences(deu_tokenizer,deu_length,train[:,1])
trainY= encode_sequences(eng_tokenizer,eng_length,train[:,0])
```

```
testX= encode_sequences(deu_tokenizer,deu_length,train[:,1])
testY= encode_sequences(eng_tokenizer,eng_length,train[:,0])
```

```
def build_model(in_vocab,out_vocab,in_timesteps,out_timesteps,units):
  model=Sequential()
  model.add=(Embedding(in_vocab,units,input_length=in_timesteps,mask_zero=True))
  model.add(LSTM(units))
  model.add(RepeatVector(out_timesteps))
  model.add(LSTM(units,return_sequences=True))
  model.add(Dense(out_vocab,activation='softmax'))
  return model
```

```
model = build_model(deu_vocab_size,eng_vocab_size,deu_length,eng_length,512)
rms=optimizers.RMSprop(lr=0.001)
model.compile(optimizer=rms,loss='sparse_categorial_crossentropy')
```

```
filename='Joyraj Longjam'
checkpoint=ModelCheckpoint(filename,monitor='val_loss',verbose=1,save_best_only=T
```

```
history=model.fit(trainX,trainY.reshape(trainY.shape[0],trainY.shape[1],1),
       epochs=5,batch_size=512,
       validation_split=0.2,
       callbacks=[checkpoint],verbose=1)
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['train','validation'])
plt.show()
```

```
model=load_model('model.h1.24_Joyraj')
preds=model.predict_classes(testX.reshape((testX.shape[0],testX.shape[1])))


def get_word(n,tokenizer):
  for word,index in tokenizer.word_index.items():
    if index==n:
      return word
  return none




 preds_text=[]
for i in preds:
  temp=[]
  for j in range(len(i)):
    t=get_word(i[j],eng_tokenizer)
    if j>0:
      if(t==get_word(i[j-1,eng_tokenizer])or(t==None)):
        temp.append('')
      else:
          temp.append(t)

    else:
           if(t==None):
             temp.append('')
     else:
               temp.append(t)

               preds_text.append(' '.join(temp))



pred_df=pd.DataFrame({'actual':test[:,0],'predicted':preds_text})


pd.set_option('display.max_colwidth',200)



pred_df.heads(15)
```