CS584 Extra Credit

David Fu

In [30]:
```
import numpy as np
import re

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input, LSTM, Bidirectional, Concaten
ate, Dense, GRU, Dot, Activation, Concatenate, Dropout, SimpleRNN
from tensorflow.keras.models import Model

from tensorflow.keras.utils import plot_model

from tensorflow.keras import optimizers

from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import sentence_bleu

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
%matplotlib inline
```

Text Clean Up and Tokenizer

In [2]:
```
def read_input(input_path:str) -> str:
    file_data = open(input_path , 'rt')
    return file_data.read()

def clean_text(text:str):
    clean = re.sub('[\W_]+', ' ', text.lower())
    clean = re.sub('[\d]+', ' ', clean)
    return re.sub(' +', ' ', clean)

def line_break_tokenizer(input_text:str):
    return input_text.split('\n')
```

In [3]:
```
# Load the data from the txt file

english = './es-en/europarl-v7.es-en.en'
spanish = './es-en/europarl-v7.es-en.es'

en_data = line_break_tokenizer(read_input(english))
es_data = line_break_tokenizer(read_input(spanish))
print(len(en_data))
print(len(es_data))
```

```
1965735
1965735
```

In [64]:
```
target_token_index
```

Out[64]:
```
{' ': 1,
 'e': 2,
 'a': 3,
 'o': 4,
 's': 5,
 'n': 6,
 'r': 7,
 'i': 8,
 'l': 9,
 'd': 10,
 't': 11,
 'c': 12,
 'u': 13,
 'p': 14,
 'm': 15,
 'b': 16,
 'g': 17,
 'q': 18,
 'y': 19,
 'ó': 20,
 'v': 21,
 'f': 22,
 'h': 23,
 'í': 24,
 'á': 25,
 'j': 26,
 'z': 27,
 'é': 28,
 'ñ': 29,
 'x': 30,
 'ú': 31,
 'k': 32,
 'w': 33,
 'º': 34,
 'ü': 35,
 'ö': 36,
 'è': 37,
 'ä': 38,
 'ç': 39,
 'ò': 40,
 'č': 41,
 'à': 42,
 'ń': 43,
 'å': 44,
 'ł': 45,
 'ā': 46,
 'ṇ': 47,
 'š́': 48,
 'ė': 49}
```

In [65]:
```
def decode_sequence(input_seq):
    states_value = encoder_model.predict(input_seq)

    target_seq = np.zeros((1, 1, num_decoder_tokens))
    target_seq[0, 0, target_token_index[' ']] = 1.

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + state
s_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 == '\n' or
           len(decoded_sentence) > max_decoder_seq_length):
            stop_condition = True

        target_seq = np.zeros((1, 1, num_decoder_tokens))
        target_seq[0, 0, sampled_token_index] = 1.

        states_value = [h, c]

    return decoded_sentence
```

In [33]:
```
def blue_score(input_test_texts, target_test_texts):
    score = 0
    for i, input_sentence in enumerate(input_test_texts):
        input_sequence = text2sequences(len(input_sentence), input_sente
nce)[0]
        input_x = data_encoder(input_sequence, len(input_sentence), num_
encoder_tokens)
        translated_sentence = decode_sequence(input_x)
        reference = translated_sentence.split()
        candidate = target_test_texts[i].split()
        s = sentence_bleu(reference, candidate)
        score += s
    print("Average BLEU score {0:.4f}".format(score / len(input_test_tex
ts)))

def blue_score_attention(input_test_texts, target_test_texts):
    score = 0
    for i, input_sentence in enumerate(input_test_texts):
        input_sequence = text2sequences(len(input_sentence), input_sente
nce)[0]
        input_x = data_encoder(input_sequence, len(input_sentence), num_
encoder_tokens)
        translated_sentence = decode_sequence_attention(input_x)
        reference = translated_sentence.split()
        candidate = target_test_texts[i].split()
        s = sentence_bleu(reference, candidate)
        score += s
    print("Average BLEU score {0:.4f}".format(score / len(input_test_tex
ts)))
```

Check Text and Translation

In [4]:
```
for i in range(0, len(en_data)):
    en_data[i] = clean_text(en_data[i])
    es_data[i] = clean_text(es_data[i])
for i in range(120, 130):
    print('{' + en_data[i] + '} >>> {' + es_data[i] + '}\n')
```

```
{it would be useful for the record of the house to state how people per
ceive what we have just done in the light of their own political analys
is } >>> {sería útil que el historial de la cámara registrara cómo perc
ibe la gente lo que hemos hecho a la luz de sus propios análisis políti
cos }

{madam president i do not wish to reopen the debate but i had also aske
d for the floor to comment on mr barón crespo s motion } >>> {señora pr
esidenta no deseo reanudar el debate pero había pedido también la palab
ra para dar mi opinión acerca de la enmienda del sr barón crespo }

{you did not call me either } >>> {tampoco me ha nombrado usted }

{i regret this but the vote has already been taken and the decision is
made so let us leave the matter there } >>> {lo lamento pero la votació
n se ha realizado se ha adoptado la decisión y por consiguiente dejemos
así las cosas }

{i am terribly sorry mr hänsch and mr cox i did not see you asking to s
peak } >>> {lo siento mucho señor hänsch señor cox no he advertido que
ustedes pedían la palabra }

{even so i think the positions are quite clear and they shall be entere
d in the minutes } >>> {en estas circunstancias creo que las posiciones
están claras y que se reflejarán en el acta }

{when we adopt the minutes for today s sitting tomorrow then any member
s who think the positions have not been explained clearly enough may as
k for amendments } >>> {cuando mañana se trate la aprobación del acta d
e la sesión de hoy si sus señorías estiman que las posiciones no se han
explicado lo bastante bien podrán pedir modificaciones }

{this seems to me to be a workable solution } >>> {creo que es una buen
a fórmula }

{of course the minutes for tomorrow s sitting will take into account an
y additional explanations } >>> {por supuesto que el acta de la sesión
de mañana consignará todas las explicaciones complementarias }

{i think this is a better solution than proceeding now to extremely tim
e consuming explanations of votes } >>> {creo que esta fórmula es mejor
que la de proceder ahora a unas explicaciones de voto que nos llevarían
mucho tiempo }
```

In [5]:
```
x_data_1, x_data_2, y_data_1, y_data_2 = train_test_split(en_data, es_da
ta, test_size=0.997, random_state=1219)
x_train, x_2, y_train, y_2 = train_test_split(x_data_1, y_data_1, test_s
ize=0.40, random_state=1219)
x_valid, x_test, y_valid, y_test = train_test_split(x_2, y_2, test_size=
0.50, random_state=1219)
```

In [6]:
```
print(x_train[1])
print(y_train[1])

print(len(x_train))
```

```
mr president we trust europol will be strengthened and that it will ful
fil its duties efficiently so that the citizens feel protected this is
all perfectly compatible however with the democratic control the house
always insists on when the legal status of this instrument for police c
ooperation is debated
señor presidente queremos que europol se fortalezca que cumpla sus misi
ones de manera eficaz para que los ciudadanos se sientan protegidos y e
sto es perfectamente compatible con el control democrático que viene ex
igiendo este parlamento en cada ocasión en que se debate sobre el estat
us jurídico de este instrumento de cooperación policial
3538
```

In [7]:
```
max_encoder_seq_length = max(len(line) for line in x_train)
max_decoder_seq_length = max(len(line) for line in y_train)
print('max length of input  sentences: %d' % (max_encoder_seq_length))
print('max length of target sentences: %d' % (max_decoder_seq_length))
```

```
max length of input  sentences: 1154
max length of target sentences: 1257
```

set up seq2seq encoder and decoder

In [8]:
```
def text2sequences(max_len, lines):
    tokenizer = Tokenizer(char_level=True, filters='')
    tokenizer.fit_on_texts(lines)
    seqs = tokenizer.texts_to_sequences(lines)
    seqs_pad = pad_sequences(seqs, maxlen=max_len, padding='post')
    return seqs_pad, tokenizer.word_index


encoder_input_seq, input_token_index = text2sequences(max_encoder_seq_le
ngth,
                                                      x_train)
decoder_input_seq, target_token_index = text2sequences(max_decoder_seq_l
ength,
                                                       y_train)

print('shape of encoder_input_seq: ' + str(encoder_input_seq.shape))
print('shape of input_token_index: ' + str(len(input_token_index)))
print('shape of decoder_input_seq: ' + str(decoder_input_seq.shape))
print('shape of target_token_index: ' + str(len(target_token_index)))
```

```
shape of encoder_input_seq: (3538, 1154)
shape of input_token_index: 50
shape of decoder_input_seq: (3538, 1257)
shape of target_token_index: 49
```

In [9]:
```
num_encoder_tokens = len(input_token_index) + 1
num_decoder_tokens = len(target_token_index) + 1
print('num_encoder_tokens: ' + str(num_encoder_tokens))
print('num_decoder_tokens: ' + str(num_decoder_tokens))
```

```
num_encoder_tokens: 51
num_decoder_tokens: 50
```

In [10]:
```
def data_encoder(sequences, max_len, vocab_size):
    n = len(sequences)
    data = np.zeros((n, max_len, vocab_size))
    for i in range(n):
        data[i, :, :] = to_categorical(sequences[i], num_classes=vocab_s
ize)
    return data

encoder_input_data = data_encoder(encoder_input_seq, max_encoder_seq_len
gth, num_encoder_tokens)
decoder_input_data = data_encoder(decoder_input_seq, max_decoder_seq_len
gth, num_decoder_tokens)

decoder_target_seq = np.zeros(decoder_input_seq.shape)
decoder_target_seq[:, 0:-1] = decoder_input_seq[:, 1:]
decoder_target_data = data_encoder(decoder_target_seq,
                                   max_decoder_seq_length,
                                   num_decoder_tokens)

print(encoder_input_data.shape)
print(decoder_input_data.shape)
print(decoder_target_data.shape)
```

```
(3538, 1154, 51)
(3538, 1257, 50)
(3538, 1257, 50)
```

Set Up data traning and Model

In [11]:
```
latent_dim = 256

encoder_inputs = Input(shape=(None, num_encoder_tokens),
                       name='encoder_inputs')


encoder_bilstm = Bidirectional(LSTM(latent_dim, return_state=True,
                                    dropout=0.5, name='encoder_lstm'))
_, forward_h, forward_c, backward_h, backward_c = encoder_bilstm(encoder
_inputs)

state_h = Concatenate()([forward_h, backward_h])
state_c = Concatenate()([forward_c, backward_c])

encoder_model = Model(inputs=encoder_inputs,
                      outputs=[state_h, state_c],
                      name='encoder')
```

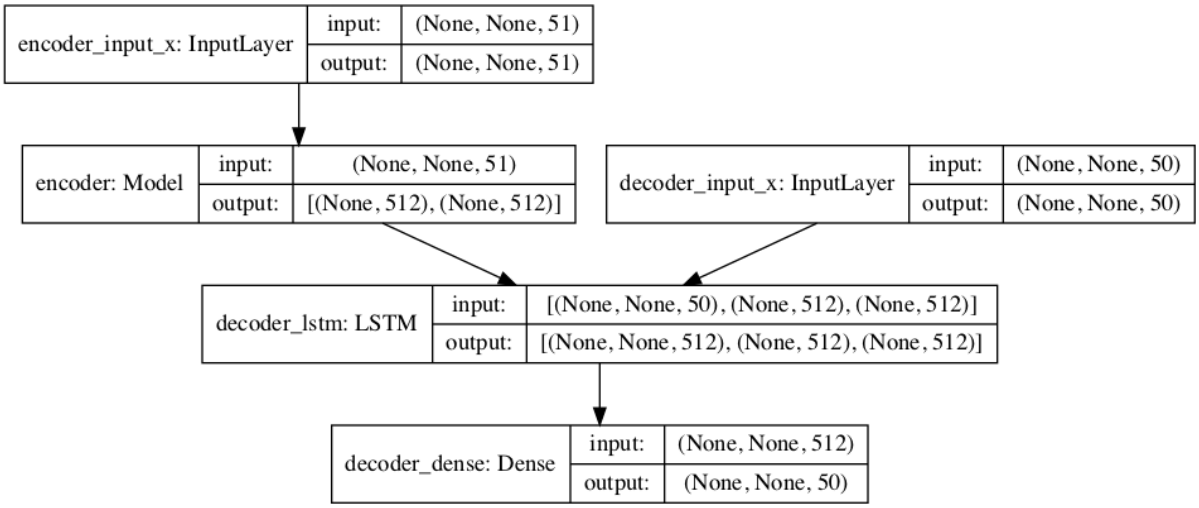
In [12]:
```
decoder_input_h = Input(shape=(latent_dim*2,), name='decoder_input_h')
decoder_input_c = Input(shape=(latent_dim*2,), name='decoder_input_c')
decoder_input_x = Input(shape=(None, num_decoder_tokens), name='decoder_
input_x')


decoder_lstm = LSTM(latent_dim*2, return_sequences=True,
                    return_state=True, dropout=0.5, name='decoder_lstm')
decoder_lstm_outputs, state_h, state_c = decoder_lstm(decoder_input_x,
                                                      initial_state=[dec
oder_input_h, decoder_input_c])


decoder_dense = Dense(num_decoder_tokens, activation='softmax', name='de
coder_dense')
decoder_outputs = decoder_dense(decoder_lstm_outputs)


decoder_model = Model(inputs=[decoder_input_x, decoder_input_h, decoder_
input_c],
                      outputs=[decoder_outputs, state_h, state_c],
                      name='decoder')
```

In [13]:
```
encoder_input_x = Input(shape=(None, num_encoder_tokens), name='encoder_
input_x')
decoder_input_x = Input(shape=(None, num_decoder_tokens), name='decoder_
input_x')


encoder_final_states = encoder_model([encoder_input_x])
decoder_lstm_output, _, _ = decoder_lstm(decoder_input_x, initial_state=
encoder_final_states)
decoder_pred = decoder_dense(decoder_lstm_output)

model = Model(inputs=[encoder_input_x, decoder_input_x],
              outputs=decoder_pred,
              name='model_training')
model.summary()
```

```
Model: "model_training"
__________________________________________________________________________
_______________________
Layer (type)                    Output Shape         Param #     Connec
ted to
==========================================================================
==========================
encoder_input_x (InputLayer)    (None, None, 51)     0
__________________________________________________________________________
_______________________
decoder_input_x (InputLayer)    (None, None, 50)     0
__________________________________________________________________________
_______________________
encoder (Model)                 [(None, 512), (None, 630784      encode
r_input_x[0][0]
__________________________________________________________________________
_______________________
decoder_lstm (LSTM)             [(None, None, 512),  1153024     decode
r_input_x[0][0]
                                                                 encode
r[1][0]
                                                                 encode
r[1][1]
__________________________________________________________________________
_______________________
decoder_dense (Dense)           (None, None, 50)     25650       decode
r_lstm[1][0]
==========================================================================
==========================
Total params: 1,809,458
Trainable params: 1,809,458
Non-trainable params: 0
__________________________________________________________________________
_______________________
```

In [14]:
```
plot_model(model, show_shapes=True, to_file='model_LSTM.png')
```

Out[14]:



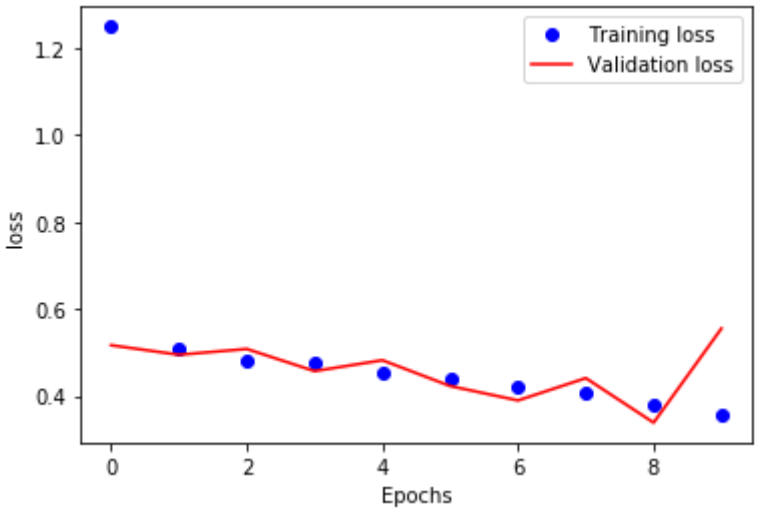
In [15]:
```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')

history = model.fit([encoder_input_data, decoder_input_data],  # trainin
g data
          decoder_target_data,                      # labels (left shif
t of the target sequences)
          batch_size=128, epochs=10, validation_split=0.2)

model.save('seq2seq.h5')
```

```
WARNING:tensorflow:From /Users/jig728/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.
<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Users/jig728/opt/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:422: The name tf.global_va
riables is deprecated. Please use tf.compat.v1.global_variables instea
d.

Train on 2830 samples, validate on 708 samples
Epoch 1/10
2830/2830 [==============================] - 631s 223ms/step - loss: 1.
2498 - val_loss: 0.5164
Epoch 2/10
2830/2830 [==============================] - 600s 212ms/step - loss: 0.
5075 - val_loss: 0.4945
Epoch 3/10
2830/2830 [==============================] - 586s 207ms/step - loss: 0.
4794 - val_loss: 0.5078
Epoch 4/10
2830/2830 [==============================] - 585s 207ms/step - loss: 0.
4761 - val_loss: 0.4568
Epoch 5/10
2830/2830 [==============================] - 587s 207ms/step - loss: 0.
4509 - val_loss: 0.4819
Epoch 6/10
2830/2830 [==============================] - 597s 211ms/step - loss: 0.
4396 - val_loss: 0.4225
Epoch 7/10
2830/2830 [==============================] - 595s 210ms/step - loss: 0.
4218 - val_loss: 0.3894
Epoch 8/10
2830/2830 [==============================] - 618s 219ms/step - loss: 0.
4093 - val_loss: 0.4407
Epoch 9/10
2830/2830 [==============================] - 603s 213ms/step - loss: 0.
3811 - val_loss: 0.3383
Epoch 10/10
2830/2830 [==============================] - 604s 213ms/step - loss: 0.
3568 - val_loss: 0.5549
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-15-fa2446370bfd> in <module>
      6           batch_size=128, epochs=10, validation_split=0.2)
      7
----> 8 model.save('seq2seq.h5')

~/opt/anaconda3/lib/python3.7/site-packages/keras/engine/network.py in save(self, filepath, overwrite, include_optimizer)
   1150             raise NotImplementedError
   1151         from ..models import save_model
-> 1152         save_model(self, filepath, overwrite, include_optimizer)
   1153
   1154     @saving.allow_write_to_gcs

~/opt/anaconda3/lib/python3.7/site-packages/keras/engine/saving.py in save_wrapper(obj, filepath, overwrite, *args, **kwargs)
    447                 os.remove(tmp_filepath)
    448         else:
--> 449             save_function(obj, filepath, overwrite, *args, **kwargs)
    450
    451     return save_wrapper

~/opt/anaconda3/lib/python3.7/site-packages/keras/engine/saving.py in save_model(model, filepath, overwrite, include_optimizer)
    539                 return
    540         with H5Dict(filepath, mode='w') as h5dict:
--> 541             _serialize_model(model, h5dict, include_optimizer)
    542     elif hasattr(filepath, 'write') and callable(filepath.write):
    543         # write as binary stream

~/opt/anaconda3/lib/python3.7/site-packages/keras/engine/saving.py in _serialize_model(model, h5dict, include_optimizer)
    161             layer_group[name] = val
    162     if include_optimizer and model.optimizer:
--> 163         if isinstance(model.optimizer, optimizers.TFOptimizer):
    164             warnings.warn(
    165                 'TensorFlow optimizers do not '

AttributeError: module 'tensorflow.keras.optimizers' has no attribute 'TFOptimizer'
```

In [ ]:
```
model.save('seq2seq.h5')
```

In [25]:
```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



In [ ]:
```
```

In [66]:
```
blue_score(x_valid, y_valid)
```

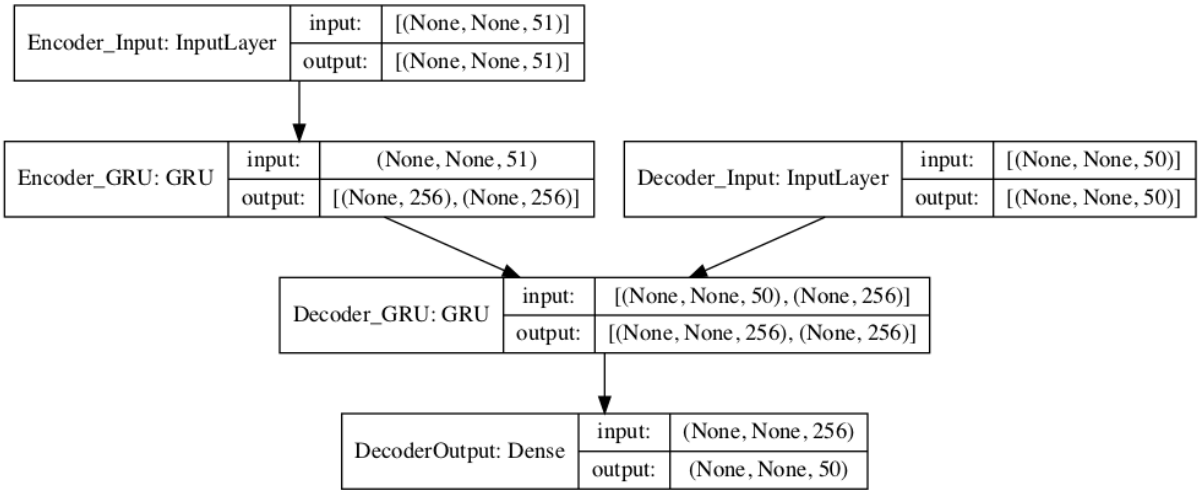
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-66-97baa390ba55> in <module>
----> 1 blue_score(x_valid, y_valid)

<ipython-input-33-0f46995ee91d> in blue_score(input_test_texts, target_test_texts)
      4         input_sequence = text2sequences(len(input_sentence), input_sentence)[0]
      5         input_x = data_encoder(input_sequence, len(input_sentence), num_encoder_tokens)
----> 6         translated_sentence = decode_sequence(input_x)
      7         reference = translated_sentence.split()
      8         candidate = target_test_texts[i].split()

<ipython-input-65-c9dd07d5bb13> in decode_sequence(input_seq)
     12         sampled_token_index = np.argmax(output_tokens[0, -1, :])
     13
---> 14         sampled_char = reverse_target_char_index[sampled_token_index]
     15         decoded_sentence += sampled_char
     16

KeyError: 0
```

In [40]:
```
encoder_inputs = Input(shape=(None, num_encoder_tokens), name="Encoder_I
nput")
encoder = GRU(latent_dim, return_state=True, name="Encoder_GRU")
_, encoder_state = encoder(encoder_inputs)
decoder_inputs = Input(shape=(None, num_decoder_tokens), name="Decoder_I
nput")
decoder_gru = GRU(latent_dim, return_sequences=True, return_state=True,
name="Decoder_GRU")

decoder_outputs, _ = decoder_gru(decoder_inputs, initial_state=encoder_s
tate)

decoder_dense = Dense(num_decoder_tokens, activation='softmax', name="De
coderOutput")
decoder_outputs = decoder_dense(decoder_outputs)

model_GRU = Model([encoder_inputs, decoder_inputs], decoder_outputs)

model_GRU.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
model_GRU.summary()
```

```
WARNING:tensorflow:From /Users/jig728/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/init_ops.py:1251: calling VarianceScalin
g.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecat
ed and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it
to the constructor
Model: "model"
_________________________________________________________________________
_________________________________
Layer (type)                 Output Shape              Param #     Connec
ted to
=========================================================================
==========================
Encoder_Input (InputLayer)   [(None, None, 51)]        0
_________________________________________________________________________
_________________________________
Decoder_Input (InputLayer)   [(None, None, 50)]        0
_________________________________________________________________________
_________________________________
Encoder_GRU (GRU)            [(None, 256), (None,      236544      Encode
r_Input[0][0]
_________________________________________________________________________
_________________________________
Decoder_GRU (GRU)            [(None, None, 256),       235776      Decode
r_Input[0][0]
                                                                   Encode
r_GRU[0][1]
_________________________________________________________________________
_________________________________
DecoderOutput (Dense)        (None, None, 50)          12850       Decode
r_GRU[0][0]
=========================================================================
==========================
Total params: 485,170
Trainable params: 485,170
Non-trainable params: 0
_________________________________________________________________________
_________________________________
```

In [41]:
```
plot_model(model_GRU, show_shapes=True, to_file='model_GRU.png')
```

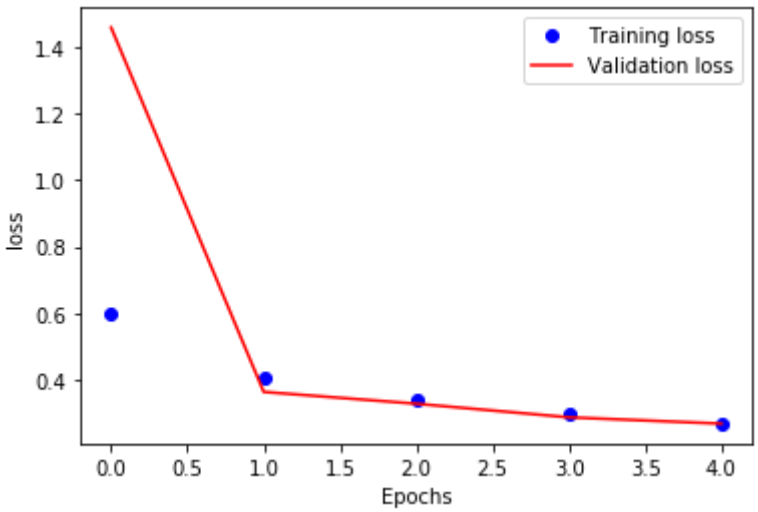
Out[41]:



In [43]:
```
history2 = model_GRU.fit([encoder_input_data, decoder_input_data], decod
er_target_data,
          batch_size=64,
          epochs=5,
          validation_split=0.2)
```

```
Train on 2830 samples, validate on 708 samples
Epoch 1/5
2830/2830 [==============================] - 636s 225ms/sample - loss:
0.5996 - acc: 0.8915 - val_loss: 1.4576 - val_acc: 0.8926
Epoch 2/5
2830/2830 [==============================] - 584s 206ms/sample - loss:
0.4074 - acc: 0.8948 - val_loss: 0.3637 - val_acc: 0.8945
Epoch 3/5
2830/2830 [==============================] - 576s 203ms/sample - loss:
0.3420 - acc: 0.9036 - val_loss: 0.3283 - val_acc: 0.9067
Epoch 4/5
2830/2830 [==============================] - 581s 205ms/sample - loss:
0.2992 - acc: 0.9133 - val_loss: 0.2874 - val_acc: 0.9144
Epoch 5/5
2830/2830 [==============================] - 573s 202ms/sample - loss:
0.2708 - acc: 0.9187 - val_loss: 0.2685 - val_acc: 0.9185
```

In [44]:
```
loss = history2.history['loss']
val_loss = history2.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



Check BLUE Score

In [45]:
```
blue_score(input_val_texts, target_val_texts)
```

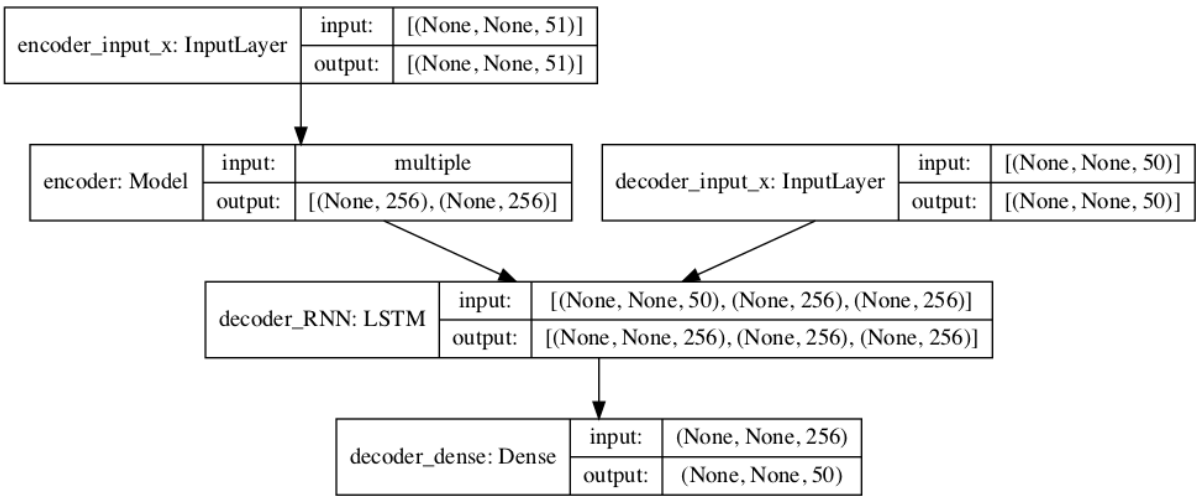
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-45-7791e7c383e7> in <module>
----> 1 blue_score(input_val_texts, target_val_texts)

NameError: name 'input_val_texts' is not defined
```

In [46]:
```
latent_dim = 256


encoder_inputs = Input(shape=(None, num_encoder_tokens),
                       name='encoder_inputs')

encoder_RNN = SimpleRNN(latent_dim, return_state=True,
                    dropout=0.5, name='encoder_RNN')
state_h, state_c = encoder_RNN(encoder_inputs)

encoder_model_RNN = Model(inputs=encoder_inputs,
                      outputs=[state_h, state_c],
                      name='encoder')

decoder_input_h = Input(shape=(latent_dim,), name='decoder_input_h')
decoder_input_c = Input(shape=(latent_dim,), name='decoder_input_c')
decoder_input_x = Input(shape=(None, num_decoder_tokens), name='decoder_
input_x')

decoder_lstm = LSTM(latent_dim, return_sequences=True,
                    return_state=True, dropout=0.5, name='decoder_RNN')
decoder_lstm_outputs, state_h, state_c = decoder_lstm(decoder_input_x,
                                                      initial_state=[dec
oder_input_h, decoder_input_c])

decoder_dense = Dense(num_decoder_tokens, activation='softmax', name='de
coder_dense')
decoder_outputs = decoder_dense(decoder_lstm_outputs)


decoder_model_RNN = Model(inputs=[decoder_input_x, decoder_input_h, deco
der_input_c],
                      outputs=[decoder_outputs, state_h, state_c],
                      name='decoder')

encoder_input_x = Input(shape=(None, num_encoder_tokens), name='encoder_
input_x')
decoder_input_x = Input(shape=(None, num_decoder_tokens), name='decoder_
input_x')

encoder_final_states = encoder_model_RNN([encoder_input_x])
decoder_lstm_output, _, _ = decoder_lstm(decoder_input_x, initial_state=
encoder_final_states)
decoder_pred = decoder_dense(decoder_lstm_output)

model_SimpleRNN = Model(inputs=[encoder_input_x, decoder_input_x],
              outputs=decoder_pred,
              name='model_training')
model_SimpleRNN.summary()
```

```
Model: "model_training"
__________________________________________________________________________
_____________________
Layer (type)                    Output Shape         Param #     Connec
ted to
==========================================================================
=========================
encoder_input_x (InputLayer)    [(None, None, 51)]   0
__________________________________________________________________________
_____________________
decoder_input_x (InputLayer)    [(None, None, 50)]   0
__________________________________________________________________________
_____________________
encoder (Model)                 [(None, 256), (None, 78848       encode
r_input_x[0][0]
__________________________________________________________________________
_____________________
decoder_RNN (LSTM)              [(None, None, 256),  314368      decode
r_input_x[0][0]
                                                                 encode
r[1][0]
                                                                 encode
r[1][1]
__________________________________________________________________________
_____________________
decoder_dense (Dense)           (None, None, 50)     12850       decode
r_RNN[1][0]
==========================================================================
=========================
Total params: 406,066
Trainable params: 406,066
Non-trainable params: 0
__________________________________________________________________________
_____________________
```

In [47]:
```
plot_model(model_SimpleRNN, show_shapes=True, to_file='model_SimpleRNN.p
ng')
```

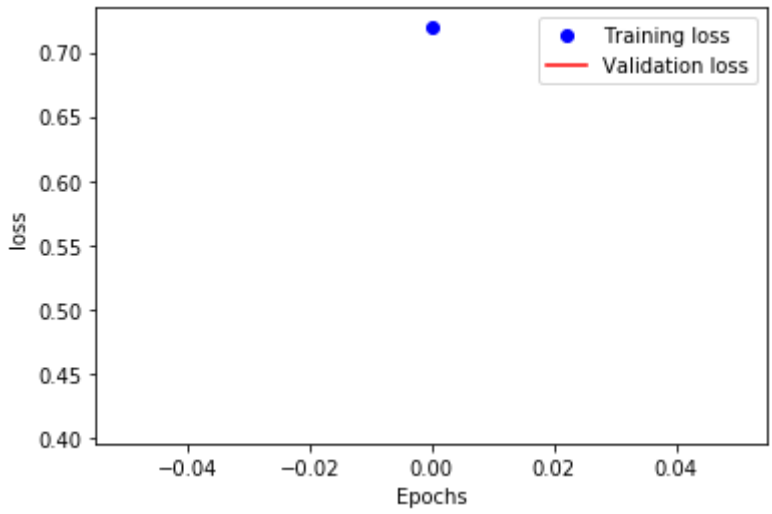
Out[47]:



In [49]:
```
model_SimpleRNN.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
history3 = model_SimpleRNN.fit([encoder_input_data, decoder_input_data], decoder_target_data,
          batch_size=64,
          epochs=1,
          validation_split=0.2)
```

```
Train on 2830 samples, validate on 708 samples
2830/2830 [==============================] - 463s 164ms/sample - loss: 0.7192 - acc: 0.8715 - val_loss: 0.4112 - val_acc: 0.8878
```

In [50]:
```
loss = history3.history['loss']
val_loss = history3.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



Validation

In [54]:
```
blue_score(x_valid, y_valid)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-54-97baa390ba55> in <module>
----> 1 blue_score(x_valid, y_valid)

<ipython-input-33-0f46995ee91d> in blue_score(input_test_texts, target_test_texts)
      4         input_sequence = text2sequences(len(input_sentence), input_sentence)[0]
      5         input_x = data_encoder(input_sequence, len(input_sentence), num_encoder_tokens)
----> 6         translated_sentence = decode_sequence(input_x)
      7         reference = translated_sentence.split()
      8         candidate = target_test_texts[i].split()

<ipython-input-38-ed288f190572> in decode_sequence(input_seq)
      3
      4     target_seq = np.zeros((1, 1, num_decoder_tokens))
----> 5     target_seq[0, 0, target_token_index['\t']] = 1.
      6
      7     stop_condition = False

KeyError: '\t'
```

In [52]:
```
latent_dim = 256
encoder_inputs = Input(shape=(None, num_encoder_tokens))
x_encoder, _, _, _, _ = Bidirectional(LSTM(latent_dim,return_sequences=T
rue, return_state=True))(encoder_inputs)
x_encoder = Dropout(0.5)(x_encoder)
x_encoder, forward_h, forward_c, backward_h, backward_c = Bidirectional(
LSTM(latent_dim,return_sequences=True, return_state=True))(encoder_input
s)
state_h = Concatenate()([forward_h, backward_h])
state_c = Concatenate()([forward_c, backward_c])
x_encoder = Dropout(0.5)(x_encoder)



decoder_inputs = Input(shape=(None, num_decoder_tokens))

x_decoder = LSTM(latent_dim*2,return_sequences=True)(decoder_inputs)
x_decoder = Dropout(0.5)(x_decoder)

# Attention
attention = Dot(axes=[2, 2])([x_decoder, x_encoder])
attention = Activation('softmax')(attention)

context = Dot(axes=[2, 1])([attention, x_encoder])
decoder_combined_context = Concatenate(axis=-1)([context, x_decoder])
x_decoder = LSTM(int(latent_dim/2),return_sequences=True)(x_decoder)
x_decoder = Dropout(0.5)(x_decoder)
# Output
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_combined_context)

model_attention = Model([encoder_inputs, decoder_inputs], decoder_output
s)
model_attention .summary()
```

```
WARNING:tensorflow:From /Users/jig728/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/init_ops.py:97: calling GlorotUniform.__
init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated a
nd will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it
to the constructor
WARNING:tensorflow:From /Users/jig728/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/init_ops.py:97: calling Orthogonal.__ini
t__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and
will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it
to the constructor
WARNING:tensorflow:From /Users/jig728/opt/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/init_ops.py:97: calling Zeros.__init__
(from tensorflow.python.ops.init_ops) with dtype is deprecated and will
be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it
to the constructor
Model: "model_1"
__________________________________________________________________________
_______________________________
Layer (type)                    Output Shape         Param #     Connec
ted to
==========================================================================
===========================
input_2 (InputLayer)            [(None, None, 50)]   0
__________________________________________________________________________
_______________________________
input_1 (InputLayer)            [(None, None, 51)]   0
__________________________________________________________________________
_______________________________
lstm_2 (LSTM)                   (None, None, 512)    1153024     input_
2[0][0]
__________________________________________________________________________
_______________________________
bidirectional_1 (Bidirectional) [(None, None, 512),  630784      input_
1[0][0]
__________________________________________________________________________
_______________________________
dropout_2 (Dropout)             (None, None, 512)    0           lstm_2
[0][0]
__________________________________________________________________________
_______________________________
dropout_1 (Dropout)             (None, None, 512)    0           bidire
ctional_1[0][0]
__________________________________________________________________________
_______________________________
dot (Dot)                       (None, None, None)   0           dropou
t_2[0][0]
                                                                 dropou
t_1[0][0]
__________________________________________________________________________
_______________________________
activation (Activation)         (None, None, None)   0           dot[0]
```

```
[0]
__________________________________________________________________________
_____________________________
dot_1 (Dot)                     (None, None, 512)    0           activa
tion[0][0]
                                                                 dropou
t_1[0][0]
__________________________________________________________________________
_____________________________
concatenate_2 (Concatenate)     (None, None, 1024)   0           dot_1
[0][0]
                                                                 dropou
t_2[0][0]
__________________________________________________________________________
_____________________________
dense (Dense)                   (None, None, 50)     51250       concat
enate_2[0][0]
==========================================================================
==========================
Total params: 1,835,058
Trainable params: 1,835,058
Non-trainable params: 0
__________________________________________________________________________
_____________________________
```

In [53]:
```
plot_model(model_attention, show_shapes=True, to_file='model_attention.png')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-816e71da4e36> in <module>
      1 # Show the structure of the nerual network
      2 from keras.utils.vis_utils import plot_model
----> 3 plot_model(model_attention, show_shapes=True, to_file='model_attention.png')

~/opt/anaconda3/lib/python3.7/site-packages/keras/utils/vis_utils.py in plot_model(model, to_file, show_shapes, show_layer_names, rankdir, expand_nested, dpi)
    238     """
    239     dot = model_to_dot(model, show_shapes, show_layer_names, rankdir,
--> 240                        expand_nested, dpi)
    241     _, extension = os.path.splitext(to_file)
    242     if not extension:

~/opt/anaconda3/lib/python3.7/site-packages/keras/utils/vis_utils.py in model_to_dot(model, show_shapes, show_layer_names, rankdir, expand_nested, dpi, subgraph)
    167             node_key = layer.name + '_ib-' + str(i)
    168             if node_key in model._network_nodes:
--> 169                 for inbound_layer in node.inbound_layers:
    170                     inbound_layer_id = str(id(inbound_layer))
    171                     if not expand_nested:

TypeError: 'InputLayer' object is not iterable
```

In [ ]:
```
model_attention.compile(optimizer='rmsprop', loss='categorical_crossentropy')

history4 = model_attention.fit([encoder_input_data, decoder_input_data], # training data
          decoder_target_data,                         # labels (left shift of the target sequences)
          batch_size=128, epochs=1, validation_split=0.2)
```

In [ ]:
```
loss = history4.history['loss']
val_loss = history4.history['val_loss']

epochs = range(len(loss))

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

In [ ]:

In [57]:
```
def decode_sequence_attention(input_seq):
    states_value = encoder_model_RNN.predict(input_seq)

    target_seq = np.zeros((1, 1, num_decoder_tokens))
    target_seq[0, 0, target_token_index['\t']] = 1.

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model_RNN.predict([target_seq] + s
tates_value)

        # this line of code is greedy selection
        # try to use multinomial sampling instead (with temperature)
        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 == '\n' or
           len(decoded_sentence) > max_decoder_seq_length):
            stop_condition = True

        target_seq = np.zeros((1, 1, num_decoder_tokens))
        target_seq[0, 0, sampled_token_index] = 1.

        states_value = [h, c]

    return decoded_sentence
```

In [ ]:

In [ ]:
```
blue_score(input_test_texts, target_test_texts)
```

In [ ]:
```
blue_score_attention(input_test_texts, target_test_texts)
```

In [67]:
```
for seq_index in range(1000,1020):
    input_seq = encoder_input_data[seq_index: seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)
    decode_sequence_R = decode_sequence_attention(input_seq)
    print('-'*40)
    print('English:       ', input_texts[seq_index])
    print('Spanish (true): ', target_texts[seq_index][1:-1])
    print('Spanish (pred): ', decoded_sentence[0:-1])
    print('Spanish (pred_attention):', decode_sequence_R[0:-1])
```

```
----------------------------------------------------------------------
----
KeyError                                  Traceback (most recent call l
ast)
<ipython-input-67-2fb7477ab363> in <module>
      1 for seq_index in range(1000,1020):
      2     input_seq = encoder_input_data[seq_index: seq_index + 1]
----> 3     decoded_sentence = decode_sequence(input_seq)
      4     decode_sequence_R = decode_sequence_attention(input_seq)
      5     print('-'*40)

<ipython-input-65-c9dd07d5bb13> in decode_sequence(input_seq)
     12         sampled_token_index = np.argmax(output_tokens[0, -1, :]
)
     13
---> 14         sampled_char = reverse_target_char_index[sampled_token_
index]
     15         decoded_sentence += sampled_char
     16

KeyError: 0
```

In [ ]:
```

```