BUILDING A CONVERSATIONAL CHATBOT

A Capstone Project Report Submitted as part of AIML-PG Program To NITW E&ICT Academy

By MANIKONDA LAXMI SUDHA under the guidance of Edureka

April -2020

AIM OF THE PROJECT:

Aim of the project is to build an intelligent conversational Chabot, Riki that can understand complex queries from user and intelligently respond.

BACKGROUND OF THE PROJECT:

R-Intelligence Inc., an AI startup, has partnered with an online chat and discussion website bluedit.io. They have an average of over 5 million active customers across the globe and more than 100,000 active chat rooms. Due to the increased traffic, they are looking at improving their user experience with a chatbot moderator, which helps them engage in a meaningful conversation and keeps them updated on trending topics, while merely chatting with Riki, a chatbot. The Artificial Intelligence-powered chat experience provides an easy access to information and a host of options to the customers.

REQUIREMENT:

The chatbot should understand that users have different intents and make it extremely simple to work around these by presenting the users with options and recommendations that best suit their needs.

PYTHON LIBRARIES USED:

- keras.models
- keras.layers.recurrent
- keras.layers
- keras.preprocessing.sequence
- keras.callbacks
- collections
- numpy
- sklearn
- nltk
- pydot

ENVIRONMENT TO EXECUTE SCRIPT: **Google Colab**

SOLUTION APPROACH:

- Downloaded the glove model available at https://nlp.stanford.edu/projects/glove/glove.twitter.27B.zip and used glove.twitter.27B.100d for this project as source data.
- Loaded the glove word embedding into a dictionary where the key is a unique word token and the value is a d dimension vector.
- Data Prepared by filtering the conversations till max word length and converted the dialogues pairs into input text and target texts. Applied start and end token to recognize the beginning and end of the sentence token.
- Created two dictionaries - input_word2idx & target_word2idx and saved as NumPy file format in the drive.
- Prepared the input data with embedding. The input data is a list of lists:
    - First list is a list of sentences.
    - Each sentence is a list of words.
- Generated training Data per batch.
- Defined the model architecture and perform the following steps:

Step 1: Used a LSTM encoder to get input words encoded in the form of (encoder outputs, encoder hidden state, encoder context) from input words.

Step 2: Used a LSTM decoder to get target words encoded in the form of (decoder outputs, decoder hidden state, decoder context) from target words. Used encoder hidden states and encoder context (represents input memory) as initial state.

Step 3: Used a dense layer to predict the next token out of the vocabulary given decoder output generated by Step 2.

Step 4: Used loss ='categorical_crossentropy' and optimizer='rmsprop'

- Generated the model summary.
- Finally generated the predictions.

CHALLENGES FACED:

- It took many hours to execute as it needs to build deep learning model on a huge file for 10 epochs maximum.
- Preparing input data with embedding.
- Preparing encoders and decoders for inputs and outputs respectively

PROPOSED SOLUTION:

Some of the code snippet of submitted .ipynb file:

```
context = dict()
context['num_encoder_tokens'] = num_encoder_tokens
context['num_decoder_tokens'] = num_decoder_tokens
context['encoder_max_seq_length'] = encoder_max_seq_length
context['decoder_max_seq_length'] = decoder_max_seq_length

np.save( DATA_SET_NAME + '/word-context1.npy', context)

for input_text,input_text_embed in zip (input_texts,range(len(encoder_input_data))):
    assert (len(input_text)==len(encoder_input_data[input_text_embed]))

# custom function to generate batches

def generate_batch(input_data, output_text_data):
    num_batches = len(input_data) // BATCH_SIZE
    while True:
        for batchIdx in range(0, num_batches):
            start = batchIdx * BATCH_SIZE
            end = (batchIdx + 1) * BATCH_SIZE
            encoder_input_data_batch = pad_sequences(input_data[start:end], encoder_max_seq_length)
            decoder_target_data_batch = np.zeros(shape=(BATCH_SIZE, decoder_max_seq_length, num_decoder_tokens))
            decoder_input_data_batch = np.zeros(shape=(BATCH_SIZE, decoder_max_seq_length, num_decoder_tokens))
            for lineIdx, target_words in enumerate(output_text_data[start:end]):
                for idx, w in enumerate(target_words):
                    w2idx = 0
                    if w in target_word2idx:
                        w2idx = target_word2idx[w]
                    decoder_input_data_batch[lineIdx, idx, w2idx] = 1
                    if idx > 0:
                        decoder_target_data_batch[lineIdx, idx - 1, w2idx] = 1
            yield [encoder_input_data_batch, decoder_input_data_batch], decoder_target_data_batch
```

SCREENSHOTS OF RESULTS:

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
abcdefghijklmnopqrstuvwxyz1234567890?.,
1193514
-0.32053 99
They do not!
They do to!
I hope so.
She okay?
Let's go.
Wow
Okay -- you're gonna need to learn how t
They do not!
They do to!
I hope so.
She okay?
Let's go.
Wow
Okay -- you're gonna need to learn how t
```

```
Okay -- you're gonna need to learn how t
[['they', 'do', 'not', '!'], ['start', 'they', 'do', 'to', '!', 'end']]
[['they', 'do', 'to', '!'], ['start', 'i', 'hope', 'so', '.', 'end']]
[['i', 'hope', 'so', '.'], ['start', 'she', 'okay', '?', 'end']]
[['she', 'okay', '?'], ['start', 'let', "'s", 'go', '.', 'end']]
[['let', "'s", 'go', '.'], ['start', 'wow', 'end']]
[['wow'], ['start', 'okay', '--', 'you', "'re", 'gon', 'na', 'need', 'to', 'learn', 'how', 'to', 'lie', '.', 'end']]
[['okay', '--', 'you', "'re", 'gon', 'na', 'need', 'to', 'learn', 'how', 'to', 'lie', '.'], ['start', 'no', 'end']]
[['no'], ['start', 'i', "'m", 'kidding', '.', 'you', 'know', 'how', 'sometimes', 'you', 'just', 'become', 'this', '```', 'persona', "'`'", '?', 'ar
[['i', "'m", 'kidding', '.', 'you', 'know', 'how', 'sometimes', 'you', 'just', 'become', 'this', '```', 'persona', "'`'", '?', 'and', 'you', 'do',
[['like', 'my', 'fear', 'of', 'wearing', 'pastels', '?'], ['start', 'the', '```', 'real', 'you', "'`'", '.', 'end']]
[['the', '```', 'real', 'you', "'`'", '.'], ['start', 'what', 'good', 'stuff', '?', 'end']]
10002
Model: "model_2"
```

Layer (type)	Output Shape	Param #	Connected to
encoder_inputs (InputLayer)	(None, None)	0	
encoder_embedding (Embedding)	(None, 40, 256)	2560512	encoder_inputs[0][0]
decoder_inputs (InputLayer)	(None, None, 10001)	0	
encoder_lstm (LSTM)	[(None, 256), (None,	525312	encoder_embedding[0][0]
decoder_lstm (LSTM)	[(None, None, 256),	10504192	decoder_inputs[0][0] encoder_lstm[0][1] encoder_lstm[0][2]
decoder_dense (Dense)	(None, None, 10001)	2570257	decoder_lstm[0][0]

```
Total params: 16,160,273
Trainable params: 16,160,273
```

```
[['like', 'my', 'fear', 'of', 'wearing', 'pastels', '?'], ['start', 'the', '``', 'real', 'you', "''", '.', 'end']]
[['the', '``', 'real', 'you', "''", '.'], ['start', 'what', 'good', 'stuff', '?', 'end']]
10002
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
encoder_inputs (InputLayer)     (None, None)         0
__________________________________________________________________________________________________
encoder_embedding (Embedding)   (None, 40, 256)      2560512     encoder_inputs[0][0]
__________________________________________________________________________________________________
decoder_inputs (InputLayer)     (None, None, 10001)  0
__________________________________________________________________________________________________
encoder_lstm (LSTM)             [(None, 256), (None, 525312      encoder_embedding[0][0]
__________________________________________________________________________________________________
decoder_lstm (LSTM)             [(None, None, 256),  10504192    decoder_inputs[0][0]
                                                                 encoder_lstm[0][1]
                                                                 encoder_lstm[0][2]
__________________________________________________________________________________________________
decoder_dense (Dense)           (None, None, 10001)  2570257     decoder_lstm[0][0]
==================================================================================================
Total params: 16,160,273
Trainable params: 16,160,273
Non-trainable params: 0
__________________________________________________________________________________________________
/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/indexed_slices.py:434: UserWarning: Converting s
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
Epoch 1/1
7611/7611 [==============================] - 2403s 316ms/step - loss: 1.4538 - val_loss: 1.6591
```

CONCLUSION:

A chatbot is a computer program that can converse with humans using artificial intelligence in messaging platforms. Chatbots are one of the emerging Apps nowadays. This project brings the power to chatbots and enriches its usability. Chatbots can give a human like touch to some aspects and make it a useful conversation. And they are completely focused entirely on providing information and completing tasks for the humans they interact with. By implementing this, I was able to add basic chatbot functionality.