Machine Learning Engineer Nanodegree
Capstone Project

Domain Based ML Assistant

Sujeet Kumar

March 17, 2018

I Definition

Project Overview:

A Domain Based ML Assistant is a Chabot that can hold conversation on a particular domain. Such an assistant can be trained on the domain specific dataset to answer questions accordingly.

A Chatbot (also known as a talkbot, chatterbot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program which conducts a conversation via auditory or textual methods.
There has been an explosion of chat bots in the modern world, assistants like Alexa, Siri, Google Assistant have chat bots built into them.
There has been a lot of development in the field of Natural Language Processing. Sequence to Sequence (Vinyals etal, 2015) algorithms capable of handling Time series problems are a perfect fit for chat bots.

Domain based assistants can be used to answer questions to customers and can act as a first line of contact between a company and a customer. They can also act as a Q&A tool for employees looking for answers to particular questions related to a particular domain.

Problem Statement:

The objective of the project is to train a RNN such that it can answer questions based on the dataset that it was trained on.
Chat bots in the early days were explicitly programmed, with hundreds of hard coded responses
to questions. However, such systems were brittle and broke for inquiries not already coded.
Developments in Natural Language Processing and Machine Learning has given answers to the
problems posed by the first generation of chat bots.

This problem translates to a Sequence to Sequence translation problem. The only difference being, instead of training the model on dataset of two languages, we are training the model on valid questions and answers. The answers to questions act as the labels here which would otherwise be the translation to a different language.

One of the issues that would be faced in creating a time series model like this is the issue of disappearing gradient. This can be overcome by using memory cells (GRU/LSTM).
Another issue is the encapsulation of the meaning of the whole sentence in the output of the last encoder. This can be overcome by using Attention and passing the attention vector to hidden cells of the decoder.

Metrics:

The evaluation of the performance of a Conversational system is very tricky. Since for a given question more than one answer may be valid. There is no one to one relationship in such systems like other supervised learning problems.
To overcome this situation a two pronged approach was taken –

1. BLEU score evaluation.
2. Manual evaluation of the answers to see if the model produces appropriate responses to questions.

II Analysis

Data Exploration:

The Reddit comments dataset for the month of May-2015 was used for training the model.
The dataset can be found at - https://www.kaggle.com/reddit/reddit-comments-may-2015
The aforesaid dataset has about 54 million comments that adds to 30GB of data that was made on reddit.com for the month of May 2015.
The dataset comes in the form of a SQLite database with name database.sqlite with one table May2015.

The dataset has the following fields –

• created_utc – the time of creation of the comment in UTC epoch-second format.
• ups – the number of upvotes (included own).
• subreddit_id – the id of the Subreddit on which the comment is made.
• link_id – the id of the link the comment was in.
• name – unique id of the comment (t1_c3v7f8u)
• score_hidden – flag to say if the comment's score is currently hidden.
• author_flair_css_class – the CSS class of the author's flair. Subreddit specific.
• author_flair_text – the text of the author's flair. Subreddit specific.
• subreddit – Subreddit of the comment
• id – the item's name.
• removal_reason – reason for removal of the comment
• gilded – the number of times this comment received reddit gold.

• downs – the number of down votes.
• archived – flag to state if the comment is flagged.
• author – the account name of the poster.
• score – the net score of the comment.
• retrieved_on – the time at which the comment was retrieved in UTC.
• body – the raw text, this is unformatted. Which included markup characters like `**` for bold, <,> & etc.
• distinguished - to allow determining whether they have been distinguished by moderators/admins.
• edited – the edit date in UTC, false if not edited
• controversiality – the controversial level of the comment.
• parent_id – ID of the parent(link or comment) this comment was a reply to.

The fields considered relevant for the task at hand among the set mentioned above are –

- name- Since, this maps to the id of the comment. Also this is the field which is mentioned in the parent_id field of the comment that is the answer to this comment.
- parent_id- Since, this helps us to find out the parent (or question) comment of the present comment.
- body- Since, this is the raw comment. And acts as the questions and answers.
- score- This is a value between 0-10. This gives the relevance of the answer to the question. And hence can be used to filter good quality response.

The following SQL query is used to map questions to answers in the May2015 table –

```
SELECT a.name AS question_id, REPLACE(a.body, CHAR(10),'') AS question, b.name AS
ans_child_id, b.parent_id AS ans_parent_child,
REPLACE(b.body,CHAR(10),'') AS answer FROM May2015 a, May2015 b WHERE
b.parent_id=a.name AND b.score>6 AND a.body<>'[deleted]' AND b.body<>'[deleted]' LIMIT
100000;
```

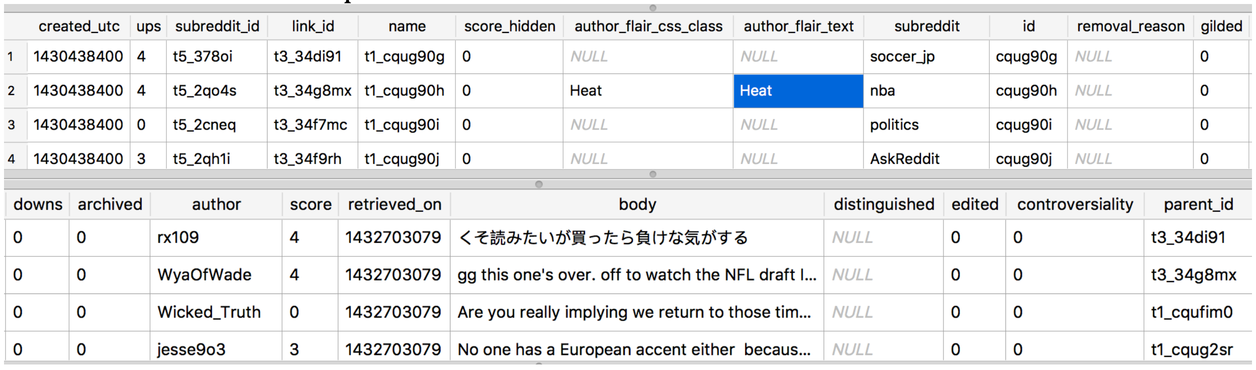
Also, for initial conditioning of the data the following SQL query is used –

```
SELECT a.name AS question_id,
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(a.bo
dy, CHAR(10),''),'[','')
,'[',''),')',''),'(',''),'?',''),'"',''),'=',''),'!',''),'#',''),'&',''),'<',''),'>','
'),'@',''),'^',''),'%',''),':',''),';',''),'*',''),'-',''),'_','')
AS question, b.name AS ans_child_id, b.parent_id AS ans_parent_child,
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
E(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(b.bo
dy, CHAR(10),''),'[','')
,'[',''),')',''),'(',''),'?',''),'"',''),'=',''),'!',''),'#',''),'&',''),'<',''),'>','
'),'@',''),'^',''),'%',''),':',''),';',''),'*',''),'-',''),'_','')
```

```
AS answer FROM May2015 a, May2015 b WHERE  b.parent_id=a.name AND b.score>6 AND
a.body<>'[deleted]' AND b.body<>'[deleted]' LIMIT 100000;
```

Exploratory Visualization:

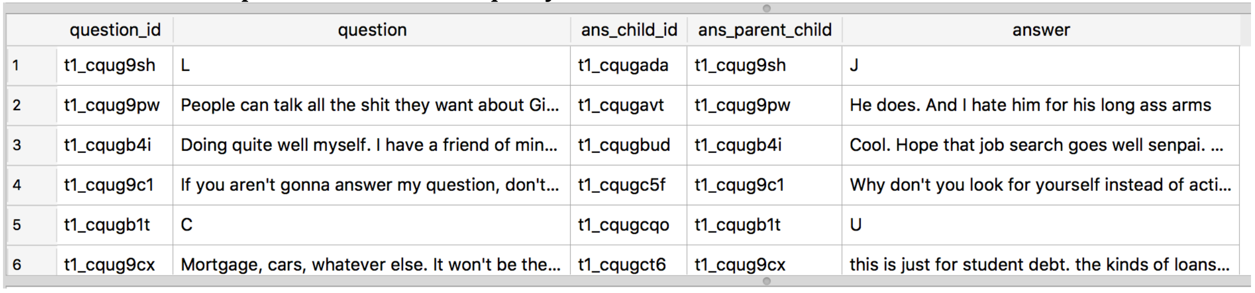
Here is a view of the sample data from the unaltered database with all the fields mentioned



	created_utc	ups	subreddit_id	link_id	name	score_hidden	author_flair_css_class	author_flair_text	subreddit	id	removal_reason	gilded
1	1430438400	4	t5_378oi	t3_34di91	t1_cqug90g	0	NULL	NULL	soccer_jp	cqug90g	NULL	0
2	1430438400	4	t5_2qo4s	t3_34g8mx	t1_cqug90h	0	Heat	Heat	nba	cqug90h	NULL	0
3	1430438400	0	t5_2cneq	t3_34f7mc	t1_cqug90i	0	NULL	NULL	politics	cqug90i	NULL	0
4	1430438400	3	t5_2qh1i	t3_34f9rh	t1_cqug90j	0	NULL	NULL	AskReddit	cqug90j	NULL	0

downs	archived	author	score	retrieved_on	body	distinguished	edited	controversiality	parent_id
0	0	rx109	4	1432703079	くそ読みたいが買ったら負けな気がする	NULL	0	0	t3_34di91
0	0	WyaOfWade	4	1432703079	gg this one's over. off to watch the NFL draft I...	NULL	0	0	t3_34g8mx
0	0	Wicked_Truth	0	1432703079	Are you really implying we return to those tim...	NULL	0	0	t1_cqufim0
0	0	jesse9o3	3	1432703079	No one has a European accent either becaus...	NULL	0	0	t1_cqug2sr

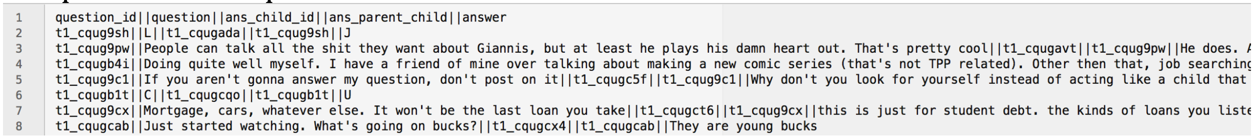
Final view of sample data from the query mentioned above -



	question_id	question	ans_child_id	ans_parent_child	answer
1	t1_cqug9sh	L	t1_cqugada	t1_cqug9sh	J
2	t1_cqug9pw	People can talk all the shit they want about Gi...	t1_cqugavt	t1_cqug9pw	He does. And I hate him for his long ass arms
3	t1_cqugb4i	Doing quite well myself. I have a friend of min...	t1_cqugbud	t1_cqugb4i	Cool. Hope that job search goes well senpai. ...
4	t1_cqug9c1	If you aren't gonna answer my question, don't...	t1_cqugc5f	t1_cqug9c1	Why don't you look for yourself instead of acti...
5	t1_cqugb1t	C	t1_cqugcqo	t1_cqugb1t	U
6	t1_cqug9cx	Mortgage, cars, whatever else. It won't be the...	t1_cqugct6	t1_cqug9cx	this is just for student debt. the kinds of loans...

The same is written to flat files with columns delimited by '||'

Sample of the raw question and answer data file-



```
question_id||question||ans_child_id||ans_parent_child||answer
t1_cqug9sh||L||t1_cqugada||t1_cqug9sh||J
t1_cqug9pw||People can talk all the shit they want about Giannis, but at least he plays his damn heart out. That's pretty cool||t1_cqugavt||t1_cqug9pw||He does. A
t1_cqugb4i||Doing quite well myself. I have a friend of mine over talking about making a new comic series (that's not TPP related). Other then that, job searching
t1_cqug9c1||If you aren't gonna answer my question, don't post on it||t1_cqugc5f||t1_cqug9c1||Why don't you look for yourself instead of acting like a child that
t1_cqugb1t||C||t1_cqugcqo||t1_cqugb1t||U
t1_cqug9cx||Mortgage, cars, whatever else. It won't be the last loan you take||t1_cqugct6||t1_cqug9cx||this is just for student debt. the kinds of loans you liste
t1_cqugcab||Just started watching. What's going on bucks?||t1_cqugcx4||t1_cqugcab||They are young bucks
```

This file is now divided into separate questions(input_text) and answers(output_text) files. With the questions and answers separated the files are now processed for data conditioning.

Algorithm and Techniques:

A good conversational system would need to consider the Time series aspect of the words used in the conversation to make meaningful conversation. This is due to the inherent nature of human to human conversations.
Recurrent Neural Networks are a perfect fit for Time Series problems such as Conversational Assistants.

RNNs are composed of a set of Neural Networks that are stacked horizontally, with the output of one NN feeding the input of the next NN placed after it.
Each NN act as the time step for the RNN.
For Conversational systems just using RNNs do not suffice, there are a number of additions needed to be done to make the system function properly.
These are –

1. Two RNNs one act as an Encoder while the other as the Decoder.
2. Number of units to be used in each hidden NN.
3. Number of NNs/time steps to be used in Encoder and Decoder RNNs (corresponding to the number of words in the sentence)
4. Activation function.
5. Dropout - Using dropout layers is a powerful way to prevent overfitting a model by randomly setting off a fixed percentages of NNs during training. This builds robustness into the model by forcing it to build redundant representations because it cannot count on the presence of any given neuron to form its predictions. I have used a drop out of 20%.
6. Optimizer - I have used the Adam optimizer, a modern stochastic gradient descent optimizer that has shown itself to work well under a variety of different conditions. An Adam optimizer is able to adapt its learning rate based on moment to more quickly converge on a solution.
7. Loss Function - This is the objective that the optimizer will seek to minimize. Each of the output classifiers will seek to minimize the softmax cross-entropy between the predicted and expected values.
8. Batch size. The choice of batch size must balance the efficiencies gained by training on multiple inputs at the same time with the requirements of holding all data within the GPU's memory. We will use a batch size of 64, which should give a reasonable compromise

This project was created using the high level Estimator APIs of TensorFlow.

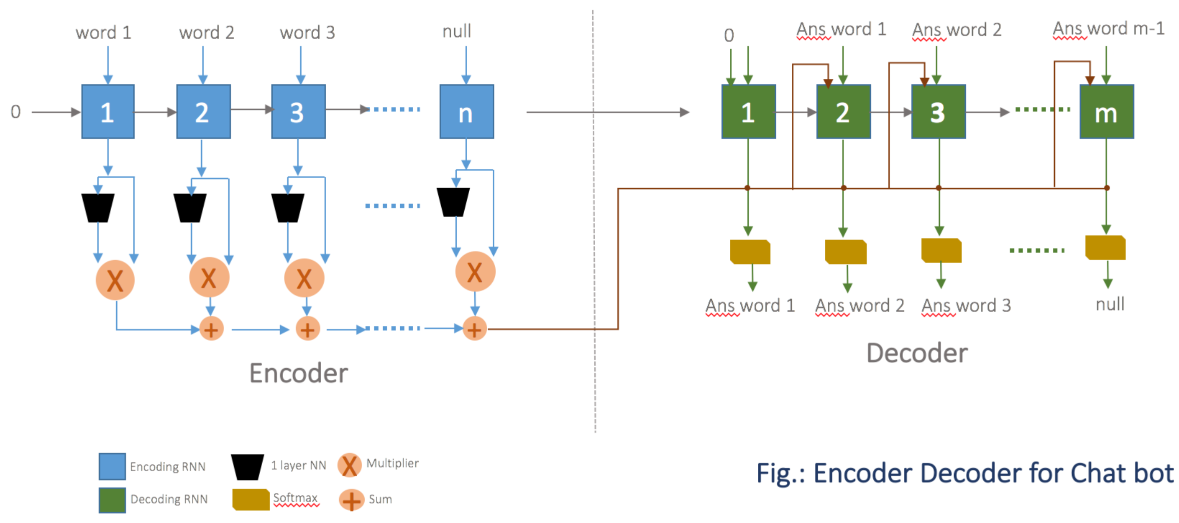


Fig.: Encoder Decoder for Chat bot

Benchmark:

Alice Bot (A.L.I.C.E. A.I Foundation) – This is a chatbot created in AIML (Artificial Intelligence Markup Language), an open, minimalist, stimulus-response language for creating
bot personalities. This is a template based chatbot.
The chatbot that I have developed is a generative chatbot that do not use templates.
I do not intend to compare the two, owing to the training infrastructure constraint that I have along with limited data that I would be using.

III Methodology:

Data Pre-processing:

The raw data file created from the Reddit comments corpus is divided into questions and answers file. With answers to a particular question residing in the same line.

After the dataset is divided into Questions and Answers, all the Questions and the corresponding answers that have less than 2 words and more than 30 words are removed from the dataset.

A vocabulary is created for the data set. Any word that is used at least 2 times in the sets of questions or answers is added to the vocabulary. The new questions and answers file is created by using only the words in the created vocabulary. Any word not found in the vocabulary is replaced by <UNK> in the questions/answers. This set act as the training set.
The vocabulary of words is also mapped with integer aliases for identification purposes.

This dataset is then sorted based on the number of words in the questions file. This is done to reduce the impact of padding required in the batch of questions put to training.

Implementation:

Project Structure:

The project has the following structure and files with their specific functionalities-

- Seq2Seq – This is the root of the project. All sub-directories and files reside under this directory.
- Api – This directory holds a helper sub-project that can be used to make API calls to the learned model to get answer out for the question asked. Flask is used for the same.
- Data – This directory holds all the data files for the project. They are as follows -

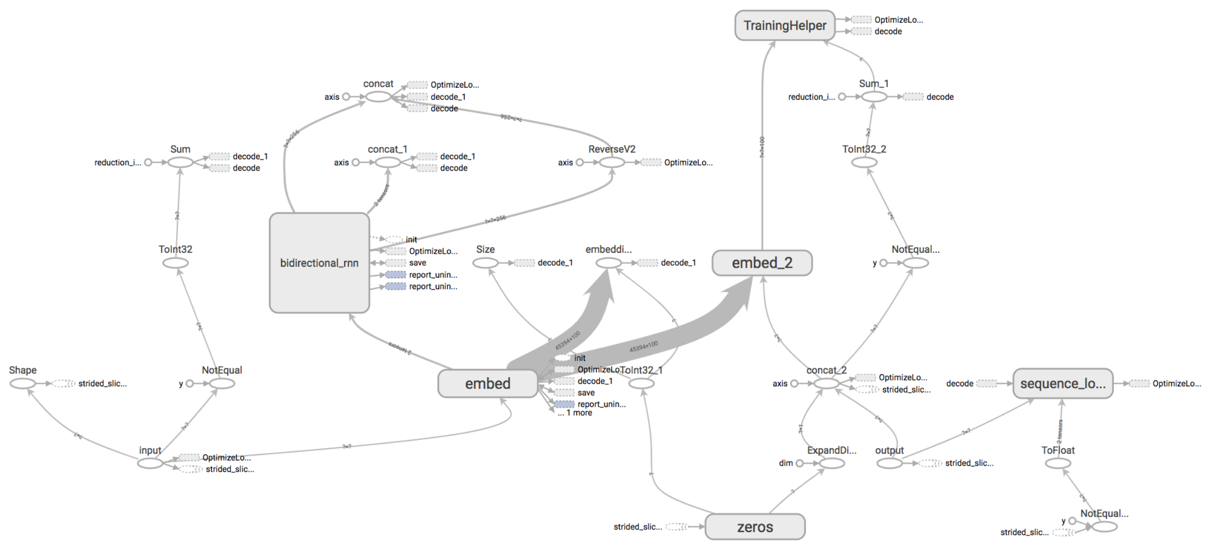
    - reddit_q_a.txt – The raw data file that is prepared using the query from the data exploration section. This has partially refined data from the SQLite database. Also, I have restricted the number of records to 100,000.
    - question_file – This is the separated question only file for all questions from the reddit_q_a.txt file.
    - answer_file – This is the separated answer only file for all questions from the reddit_q_a.txt file.
    - final_question_file – This is the final refined questions dataset file to be used in training.
    - final_answer_file – This is the final refined answers dataset file to be used in training.
    - vocab_map – This is the mapping of the words to be used as the vocabulary against the integer aliases. This is created using the answer_file and question_file.
    - testing_input_file – This is the question file used during testing for checking the BLEU score.
    - testing_ref_file – This is the answer file used during testing for checking the BLEU score.
    - prediction_input – This file carries some questions that may be used for inference check.
- model – This contains the trained model files. Files like the model checkpoint files, graph.pbtxt and the timeline files for the backup are stored here.
- Data_preparation.py – This is the python module that creates and refines the data files. The final output of this module is the final_question_file and final_answer_file.
- project_helper.py – This python file has some helper functions that are used by the Seq2Seq.py module to create input functions and load vocab.
- Seq2Seq.py – This python module has the logic for the model. Where the contents and structure of the model is defined.
- set_decoder.py – This is the python file that abstracts the decoder implementation from the Seq2Seq.py module.
- timeline_hook.py – A hook to take backups and print outputs during training.
- main.py – The main file of the project for training/inference.

Model Implementation:

The libraries used for the project include but not limited to-

- TensorFlow
- Numpy
- NLTK
- RE, OS, SYS and LOGGING

The TensorFlow graph for the implementation is added below.



I have used the TensorFlow Estimator API to implement the various components of the model.
TensorFlow Estimators is a high level TensorFlow API that greatly simplifies machine learning programming. Estimators encapsulate the following actions:

- Training
- Evaluation
- Prediction
- Export for servicing

Since the pre-made estimators are very generic and simple, the same cannot be used for our purpose. I have used Custom Estimators using a model function to define the inputs and the model.

The built model consists of the following components:

1. Memory Cell Units (both LSTM and GRU are added).
2. Dropout
3. Bidirectional Encoder.
4. Attention (Luang and Bahdanau are added)
5. Greedy Decoder for Training.
6. Beam Search Decoder for Inference.
7. Adam optimizer

Memory Cells:

I have added both the LSTM and GRU cells to the model. Based on the parameter passed to the trainer/predictor the cell can be changed.
However, I am using the GRU cells to showcase my final submission, since GRU gave me a better performance than LSTM.

Cell units are defined in TensorFlow using –
GRU cells - `tf.contrib.rnn.GRUCell(num_units=num_units)`
LSTM cells - `tf.contrib.rnn.BasicLSTMCell(num_units=num_units)`
Where, num_units is the number of units that make up a Cell. This is passed through parameters. I am using a num_units value of 256.

Dropout:

Dropout is the process in which during training some of the cells in Neural Network is removed temporarily (dropped) from the system. This is done to prevent the system from Overfitting.

I am using the Dropout Wrapper provided by the Estimator API to add dropout to my model.

```
tf.contrib.rnn.DropoutWrapper(cell=cell, input_keep_prob=(1.0 - dropout)
```

The arguments required –

a. cell – The memory cell used in the model.
b. input_keep_prob – The percentage of units that we want to include in a pass.

I am using a dropout of 20%, hence the usage of units would be 80% per pass.

Bidirectional Encoder:

I am using a Bidirectional Dynamic RNN encoder to encode the input sentence.

```
tf.nn.bidirectional_dynamic_rnn(cell_fw=fw_cell, cell_bw=bw_cell,
inputs=input_embed, dtype=tf.float32)
```

The Tensorflow API – `tf.nn.bidirectional_dynamic_rnn` can be used to create the same.

The API requires the following arguments –

a. cell_fw/cell_bw - The forward pass cell (GRU or LSTM) defined earlier and the other for the backward pass.
   I have used separate GRU cells for the same.
b. Input Embeddings – It needs the input embedding that the system has learnt based on the vocabulary. This is saved in input_embed variable.

The output returned is a tuple of 2 tuples –

a. outputs – The output for the forward and backward states of the RNN
b. output_States – The forward and backward final states of the RNN

The tuple elements of outputs and output_states are concatenated using `tf.concat(bd_encoder_op_values, -1)`. This combines the values from the forward and backward passes into a single tensor. And the same is used for further processing of the model.

Attention:

Attention is the mechanism that establishes direct relationship between the target and the source by giving importance to relevant words in the sentence as an answer to a question is getting generated.

Using TensorFlow Estimators attention can be applied to models using the AttentionWrapper.

```
tf.contrib.seq2seq.AttentionWrapper(
    cell, attention_mechanism, attention_layer_size=num_units)
```

where, cell – the Memory Unit Cell defined before
attention_mechanism – defined attention mechanism (Luong/Bahdanau)
attention_layer_size – number of units within each cell

The attention_mechanism is defined in TensorFlow using –
For Luong attention-

```
tf.contrib.seq2seq.LuongAttention(
    num_units=num_units, memory=encoder_outputs,
    memory_sequence_length=input_lengths)
```

OR for Bahdanau attention-

```
tf.contrib.seq2seq.BahdanauAttention(
    num_units=num_units, memory=encoder_outputs,
    memory_sequence_length=input_lengths)
```

where, num_units – number of units within each cell
memory – the concatenated output from the encoder
memory_sequence_length – the number of words in sentence

Greedy Decoder for Training:
I am using Greedy Decoder during training to check the loss of the model for training. This is chosen to make the training process faster.
In TensorFlow the same is achieved using the Basic Decoder that makes use of the Greedy Embedding Helper –

```
tf.contrib.seq2seq.GreedyEmbeddingHelper(
    embeddings, start_tokens=tf.to_int32(start_tokens), end_token=END_TOKEN)
```

where, embeddings – word embeddings created earlier,
start_tokens – a vector with the size equal to the batch size.
end_token – an alias depicting the end of the sentence. In the model used as 1.

```
tf.contrib.seq2seq.BasicDecoder(cell=out_cell, helper=helper,
            initial_state=out_cell.zero_state(dtype=tf.float32,
                         batch_size=batch_size).clone(cell_state=encoder_final_state))
```

where, cell – the created cell after adding attention,
helper – the GreedyEmbeddingHelper defined above,
initial_state – the encoder_final_state value calculated earlier

Beam Search Decoder for Inference:
In Greedy Decoding during the decoding process the first time step of decoder gives the word with the highest probability to the next time step which in turn gives the best word to the next time step. However, in Beam Search the first time step gives a list of ‘N’ best words(the

beam width) to the next time step. Using this list the next time step creates a set of output combination for itself (of two words). This time step then provides a list of 'N' possible combinations to the next time step which creates 'N' combinations (of three words) and passes them to the next time step. The process is propagated forward unless the EOS token is recieved.
In TensorFlow the BeamSearch Decoding can be performed by –

```
tf.contrib.seq2seq.BeamSearchDecoder(
    cell = out_cell, embedding = embeddings,
    start_tokens = tf.to_int32(start_tokens), end_token = END_TOKEN,
    initial_state = encoder_state,
    beam_width = beam_width,
    length_penalty_weight = 0.0)
```

where, cell – the memory cell with attention.
embedding – embeddings for the vocabulary
start_tokens – a vector with size equal to the batch size
end_token – the end token value for the sentences
initial_state – Concatenated encoder_state output
beam_width – the number of words that is desired to be passed to the next time step

Adam optimizer:

The Adam optimizer is the extension of Stochastic Gradient Descent. This is a process by which the weights of the models are updated during back propagation.
In TensorFlow Estimators the optimizer can be implemented using –

```
tf.contrib.layers.optimize_loss(
    loss, tf.train.get_global_step(),
    optimizer=params.get('optimizer', 'Adam'),
    learning_rate=params.get('learning_rate', 0.001),
    summaries=['loss', 'learning_rate'])
```

where, loss – the **sequence** loss, defined below,
global_step – Step counter to update on each step,
optimizer – The optimizer desired to be used,
learning_rate – the rate at which the correction is desired to be made,
Summaries – list of internal quantities to visualize in Tensorboard

```
loss = tf.contrib.seq2seq.sequence_loss(
    train_outputs.rnn_output, output, weights=weights)
```

where, logits – training outputs
target – the true class at each time step
weights – the weights of each prediction in the sequence

Hyperparameters used:

The hyperparameters used for training the model is as follows –

```
'batch_size': 64,
'input_max_length': 20,
'output_max_length': 20,
'embed_dim': 100,
'num_units': 256,
'dropout': 0.2,
'attention_mechanism_name': 'scaled_luong',
'cell_type': 'GRU',
'beam_width': 0
```

While the hyperparameters used for Inference is –

```
'batch_size': 3,
'embed_dim': 100,
'num_units': 256,
'input_max_length': 20,
'output_max_length': 20,
'dropout': 0.0,
'attention_mechanism_name': 'scaled_luong',
'cell_type': 'GRU',
'beam_width': 10
```

Refinement:

Since a time series Sequence to Sequence RNN model is very resource intensive, the amount of time that needs to be spent in training the model is huge.
To compensate for the long time required for training three attempts were made to check improvement of the model. The time for training was boxed at 2 days using CPU.

In the first run a unidirectional encoder was used with LSTM cell and Luong Attention, after 2 days of continuous training the loss was observed to fluctuate in the range of 3.8 to 4.1

To improve the performance, the model was enhanced with addition of a Bidirectional encoder, LSTM cell and Bahdanau attention. Dropout was added at 0.2 and the batch size was increased from 32 to 64. The new training loss was seen to be between 2.8 to 3.2.

For the last training check, the bidirectional model was enhanced with the addition of GRU cell and Scaled Luong attention. The batch size was kept at 64. Also, Dropout was maintained at 0.2. BeamSearch was added in place of Greedy decoding for Inference (Although this would not make any difference in training loss but Inference output would improve) With this combination, the loss was seen to be hovering around 2.2 – 2.6.

IV Results:

Model Evaluation and Validation:

The checkpoints from the final model was taken and an isolated set of questions were passed on to it to get the inference. The inference received from the model was run through the BLEU score calculator using a set of expected answers.
The BLEU score was found to be 0.48.

The BLEU score thus obtained is reasonable, considering the limitation of the training time available, usage of only CPU and relatively less number of data samples used.
On further manual evaluation of the responses generated (mentioned in the Conclusion section below), they were seen to be relatively satisfactory.

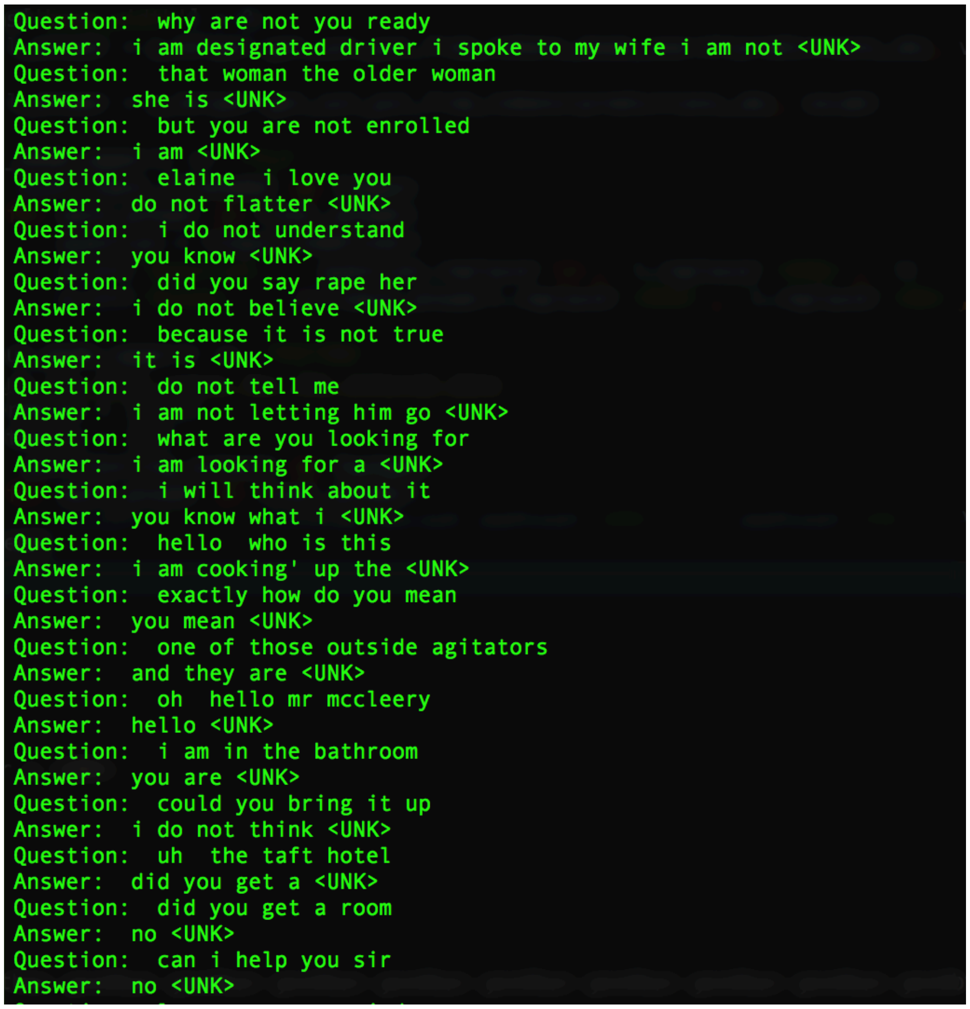
Justification:

The performance of the model could not match the performance set by state of the art machine learning models but it does give some meaning full answers to questions posed to it.
Since Time Sequence RNNs are very process intensive they take a lot of time to train. Also, the dataset required reaches millions of question-answer set. Training with such a huge amount of data would require a lot of resources. Training would need to be done for months using state of the art GPUs systems.

V. Conclusion:

Question and Answer Samples:

A sample set of the responses generated during the testing phase(used for BLEU score evaluation) is mentioned below –



```
Question:  why are not you ready
Answer:  i am designated driver i spoke to my wife i am not <UNK>
Question:  that woman the older woman
Answer:  she is <UNK>
Question:  but you are not enrolled
Answer:  i am <UNK>
Question:  elaine  i love you
Answer:  do not flatter <UNK>
Question:  i do not understand
Answer:  you know <UNK>
Question:  did you say rape her
Answer:  i do not believe <UNK>
Question:  because it is not true
Answer:  it is <UNK>
Question:  do not tell me
Answer:  i am not letting him go <UNK>
Question:  what are you looking for
Answer:  i am looking for a <UNK>
Question:  i will think about it
Answer:  you know what i <UNK>
Question:  hello  who is this
Answer:  i am cooking' up the <UNK>
Question:  exactly how do you mean
Answer:  you mean <UNK>
Question:  one of those outside agitators
Answer:  and they are <UNK>
Question:  oh  hello mr mccleery
Answer:  hello <UNK>
Question:  i am in the bathroom
Answer:  you are <UNK>
Question:  could you bring it up
Answer:  i do not think <UNK>
Question:  uh  the taft hotel
Answer:  did you get a <UNK>
Question:  did you get a room
Answer:  no <UNK>
Question:  can i help you sir
Answer:  no <UNK>
```

Reflection:

During the start of the project there were a lot of apprehensions, if the model would be able to provide reasonable answers based on the dataset that it was trained on.
Sequence to Sequence models are very process intensive. There are so many different combinations that can happen for a given question. To understand the question in itself is a very complex task. Providing answers and that to meaningful answers on top of that is even more difficult. Since answers may be correct but not relevant.
Eg. Q- How is the weather A- I don't know. Is a perfectly valid answer but not relevant may be.

Training such models takes a lot of time in GPUs. But with the limited resource of my computer the fact that the model could reach to a loss of 2.2 with a somewhat ok level of responses can be considered as a good start and a base to build upon.

Improvement:

Training for greater number of epochs would help a lot in improving the performance of the model.
At the current state the model was trained only for 1 epoch on a dataset of 100,000 sentences.
Having a greater dataset (~ 10-20 million sentences) would help improve the performance of the model. Having a bigger vocabulary will also diminish the instance of <UNK> in the answers predicted.

Most of the state of the art Conversational systems have a level of explicit programming included. Adding some explicit programming to catch exceptions would be helpful in improving the model.
The model can also be improved by incorporating Deep Reinforcement Learning model to learn which among the correct answer is relevant and better suited for the context.

References:

1. TensorFlow Estimator API: https://www.tensorflow.org/get_started/custom_estimators
2. Neural Machine Translation: https://www.tensorflow.org/tutorials/seq2seq
3. NLTK BLEU: https://www.nltk.org/_modules/nltk/translate/bleu_score.html
4. NLP models- Matthew Honnibal: https://explosion.ai/blog/deep-learning-formula-nlp
5. Vinyals etal: https://arxiv.org/pdf/1506.05869v1.pdf
6. Sutskever etal: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-withneural-networks.pdf
7. A Critical Review of Recurrent Neural Networks for Sequence Learning – Lipton etal: https://arxiv.org/pdf/1506.00019.pdf
8. RNNs in TensorFlow: https://www.tensorflow.org/tutorials/recurrent
9. Reddit Comments May 2015: https://www.kaggle.com/reddit/reddit-comments-may-2015
10. Chatbots: https://en.wikipedia.org/wiki/Chatbot
11. SkLearn: http://scikit-learn.org/stable/