Machine Learning Engineer Nanodegree

Capstone Proposal

Sujeet Kumar
January 16, 2018

Domain Background:

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.

Chat bots can have a lot of applications in the modern world. They can become the first line of agents in the Call centres, they can act as SMEs for specific fields, act as the user interface for applications etc.

There are vast applications that haven't been identified yet. The prospects are huge and this excites me to dive into this field.

Problem Statement:

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 questions not in the database. Developments in Natural Language Processing and Machine Learning has given answers to the problems posed by the first generation of chat bots.

Present times people are busy, not having time for each other. This leaves us lonely and a chat bot can help alleviate some of that boredom and monotony. In addition to that, it can also perform tasks as a bonus.

Datasets and Inputs:

There are a lot of datasets that may be used for this purpose –

- Reddit comments and responses dataset.
- Dataset from Facebook chat.

- Dataset from movies dialogues.
- Dataset from call transcripts of call centers.

I shall be using the Reddit dataset from May 2015. (https://www.kaggle.com/reddit/reddit-comments-may-2015/data)

The database has one table, May2015, with the following fields:

- created_utc
- ups
- subreddit_id
- link_id
- name
- score_hidden
- author_flair_css_class
- author_flair_text
- subreddit
- id
- removal_reason
- gilded
- downs
- archived
- author
- score
- retrieved_on
- body
- distinguished
- edited
- controversiality
- parent_id

Solution Statement:

Replying to a conversation is a very difficult and complex task. Apart from the dictionary, the arrangement of words in a sentence plays a very vital role in determining the meaning of the sentence. The same words used in different sequence, may express a completely different idea.

Sequence to Sequence and Time series models are the way to go to solve this problem.

Recurrent Neural Network is best suited for Sequence to Sequence predictions. To include Time series memory to it, we can add a LSTM or GRU cell to the model.
To better predict the importance of each of the words at different locations of the sentence, Attention mechanism may be put to use (a small neural net that decides the importance of the word).

Benchmark Model:

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 propose to develop would be 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.

Evaluation Metrics:

1. Automatic - BLEU Score
   BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations. Translations can be considered as trained replies. For a chat bot we can consider the chat bot reply as the generated translation.
2. Manual – Suitability of reply
   Although BLEU score gives a good measure of the quality of translation/reply, it's also good to check suitability of replies for a conversation manually. This helps because in a conversation one question can have multiple correct answers but only a handful of answers of relevance.

Project Design:

Language – Python 3
Dataset - Reddit dataset from May 2015
Libraries – TensorFlow, Keras, Scikit Learn.

A Bidirectional Recurrent Neural Network with either GRU or LSTM cells will be used for the solution.
To impart positional significance of the words in the meaning of the sentence Attention will be used.
Attention is the weighted sum of importance of different words in the sentence to determine the response.
A RNN is a sequence of neural net, in which the output of one stage is fed as input to the next stage. This helps to predict an output based on a sequence of inputs.
In a RNN the predicted response/output is determined not only by the feature input but also by the output of the RNN earlier in the sequence, this makes it the ideal choice for Sequence to Sequence prediction problems.
A framework based on the – Embed, Encode, Attend, Predict framework by Matthew Honnibal will be used.
Stages to solution-

1. Embed – In this stage the words in the dictionary of the training data is tokenized. This is done so that each word can be represented by a number.
   If we use the simple 'One Hot Encoding', the resultant vectors can be very long. To avoid this, we represent words as a vectors of ~100 real numbers with decimals.
   This step can be performed in three ways –
   a. Training a small ANN to learn the numbers from scratch.
   b. Using an already trained Embedding
   c. By using a transfer learning.

   We are going to use a hybrid approach. The embedding we will make use of is Word2Vec or GloVe.
2. Encode – The output from the last stage in the RNN sequence is the unrolled encoded sentence. This is the encoded representation of the input sentence.
3. Attend – Attention is the process in which the importance of each word in the sentence is determined and is used to make the final prediction. This is done by training a set of small NNs to get the weights for each word. As a result, each stage in the RNN sequence

gives a scalar for a word. Using this a vector is formed. The vector is then passed through softmax(to normalize the value between 0 and 1) and is called as the Attention vector.

4. Decode – Once the attention vector is calculated the encoded output of Encoder is fed as input to the first cell of the Decoder. The first cell of the Decoder takes as input the encoder output as well as a trigger (appended with a 0 to maintain the shape of all the cells). The output of the first cell is then combined with the attention vector from the encoder using a single layer NN and fed to the activation function(BeamSearch) to predict the first word. This way the information from the earlier words in the encoder is not lost in the Decoder. Now the output from the first cell of the Decoder along with Attention vector (concatenated) is fed to the second cell in the RNN sequence along with the output state of the first cell. This predicts the second word with help from the activation function. And the same is continued until all the words are identified.

5. Predict – This is the step where the final predicted output is received.

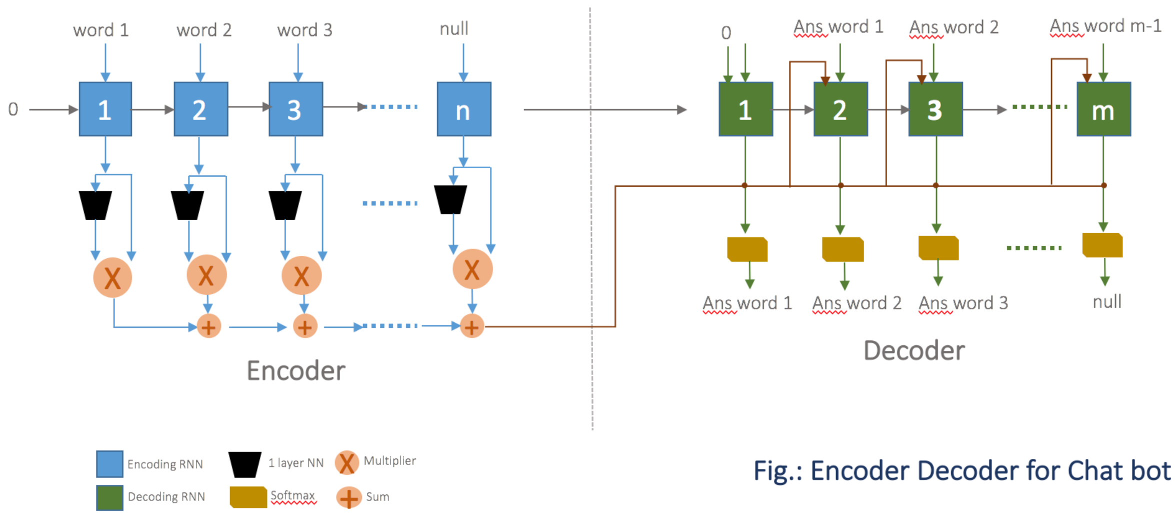

Fig.: Encoder Decoder for Chat bot

Implementation-

1. Using Bidirectional Dynamic RNN that is available in TensorFlow.
   Dynamic RNN is used because the length of the sentence (number of words) will not be constant. As a result '0' padding of the sentences will need to be done. This would be a waste of computation, when the difference in the number of words in the training data batch is large. TensorFlow's Dynamic RNN will automatically unroll the sentence sequences in the batch. Hence, no need for padding. However, a vector with the size of each sentence will need to be passed to TensorFlow for training.
   [*tf.nn.bidirectional_dynamic_rnn*].
2. Preprocess the data to remove unnecessary features (sklearn).
3. Get embedding for a batch of sentences with different sizes –
   [*tf.nn.embedding_lookup_sparse()*].

4. Using two sets of GRU/LSTM cells the forward and backward run vectors are obtained. [*tf.contrib.rnn.GRUCell(SIZE_OF_CELL)*]
5. The obtained vectors are concatenated.
6. Then the concatenated output is sent to the Attention network.
7. Which is then subjected to weighted – average process.
8. Weighted Average is the attention vector and is passed through to the decoder cells.
9. The decoder takes the encoded sentence from the last cell of the encoder and embeds it with the input trigger.
10. Then the embedded output is fed to the Decoder RNN.
11. The output from the RNN is combined with the Attention vector using a single layer NN.
12. Which is then passed on to an activation function(BeamSearchDecoder) to predict the probabilities of possible words.
13. The predicted word is fed to the next stage in the RNN sequence, along with its output state and the attention(concatenated). This process is repeated unless the complete answer is predicted.

References:

1. NLP models- Matthew Honnibal: https://explosion.ai/blog/deep-learning-formula-nlp
2. Vinyals etal: https://arxiv.org/pdf/1506.05869v1.pdf
3. Sutskever etal: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf
4. A Critical Review of Recurrent Neural Networks for Sequence Learning – Lipton etal: https://arxiv.org/pdf/1506.00019.pdf
5. RNNs in TensorFlow: https://www.tensorflow.org/tutorials/recurrent
6. Reddit Comments May 2015: https://www.kaggle.com/reddit/reddit-comments-may-2015
7. Chatbots: https://en.wikipedia.org/wiki/Chatbot
8. Word2Vec: https://en.wikipedia.org/wiki/Word2vec
9. Keras: https://keras.io
10. SkLearn: http://scikit-learn.org/stable/