Conversational Chatbot

Project review

For

Course : Artificial Intelligence
Course Code : CSE3013

By

Abhishek Thomas
17BCE0279

Under the guidance of

Prof. PREETHA EVANGELINE D

Associate Professor, SCOPE

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Abstract

Dialogue Generation or Intelligent Conversational Agent development using Artificial Intelligence or Machine Learning technique is an interesting problem in the field of Natural Language Processing. In many research and development projects, they are using Artificial Intelligence, Machine Learning algorithms and Natural Language Processing techniques for developing conversation/dialogue agent. Their research and development is still under progress and under experimentation. Dialogue/conversation agents are are predominately used by businesses, government organizations and non-profit organizations. They are frequently deployed by financial organizations like bank, credit card companies, businesses like online retail stores and start-ups. These virtual agents are adopted by businesses ranging from very small start-ups to large corporations. There are many chatbot development frameworks available in market both code based and interface based. But they lack the flexibility and usefulness in developing real dialogues. Among popular intelligent personal assistants includes Amazon's Alexa, Microsoft's Cortana and Google's Google Assistant. The functioning of these agents are limited, are retrieval based agent and also they are not aimed at holding conversations which emulate real human interaction. Among current chatbots, many are developed using rule based techniques, simple machine learning algorithms or retrieval based techniques which do not generate good results. In this project, I have developed intelligent conversational agent using state of the art techniques proposed in recently published research papers. For developing intelligent chatbot, I have used made use of the neural machine translation Model which is based on Sequence to Sequence(Seq2Seq) modeling with encoder-decoder architecture. This encoder-decoder is using Recurrent Neural Network with bi-directional LSTM (Long-Short-Term-Memory) cells. For performance optimization, I also applied Neural Attention Mechanism.

Introduction

Conversational agent or Chatbot is a program that generates response based on given input to emulate human conversations in text or voice mode. These applications are designed to simulate human-human interactions. Chatbots are predominantly used in business and corporate organizations including government, non-profit and private ones. Their functioning can range from customer service, product suggestion, product inquiry to personal assistant. Many of these chat agents are built using rule based techniques, retrieval techniques or simple machine learning algorithms. In retrieval based techniques, chat agents scan for keywords within the input phrase and retrieves relevant answers based on the query string. They rely on keyword similarity and retrieved text is pulled from internal or external data sources including world wide web or organizational database. Some other advanced chatbots are developed with natural language processing(NLP) techniques and machine learning algorithms. Also, there are many commercial chat engines available, which help build chatbots based on client data input. Figure 1.1: Ref 1 Recently, there have been major increase of interest in use and deployment of dialogue generation systems. Many major tech companies are using virtual assistant or chat agent to fill the needs of customers. Some of them include Google's Google Assistant, Microsoft's Cortana and Amazon's Alexa. Though they are primarily question answering systems, their adoption by major corporations has peaked interest in customers and seems promising for more advanced conversational agent system research and development.In this project we will be looking mainly at preprocessing the dataset and building a state-of-the-art neural machine translator and aim to get a moderately performing conversational chatbot.

Literature Survey

Intelligent Chatbot using Deep Learning [1]

-Anjana Tiha

This paper talks about how to use Deep Learning for a conversational chatbot. The model used in this paper is an encoder decoder model, ith LSTM and Attention mechanisms as well. It has compared it against basic rule based systems and has made a point about how deep learning can be used with better accuracy for conversational chatbots comparted to the former.It also has discussed in depth about LSTM and attention and how these will help increase the accuracy of any conversational chatbot.

Conversational AI Chatbot Based on Encoder-Decoder Architectures with Attention Mechanism [2]

-Amir Ali Muhammad Zain Amin

This paper has first stated the disadvantages of a rule-based chatbot and has gone on to implementing better techniques using deep learing methods to get better results.Among current chatbots, many are developed using rule-based techniques, simple machine learning algorithms or retrieval based techniques which do not generate good results. They have developed a Conversational AI Chatbot using modern-day techniques. For developing Conversational AI Chatbot, They have implemented encoder-decoder attention mechanism architecture. This encoder-decoder is using Recurrent Neural Network with LSTM (LongShort-Term-Memory) cells.

Towards Building A Neural Conversation Chatbot Through Seq2Seq Model [3]

-J.Prassanna, Khadar Nawas K, Christy Jackson J

In this work they use Neural Networks to train a chat model with a question and answer datasets that make models understand the patterns in it and behave intelligently. Here they build a domain-specific generative chatbot using Neural Networks to train a conversational Model which reads the pattern of data and reply answer when a new question is asked. Finally, they conclude by validating how relevant the response generated by the model to test data or test question and provide a further area of improvements to make the system more efficient and intelligent.

A Domain-Specific Generative Chatbot Trained from Little Data [4]

-Jurgita Kapociute-Dzikien ·

This research was performed with very small datasets having little variety in covered topics, which makes this research not only more difficult, but also more interesting. According, to their knowledge, it is the first attempt to train generative chatbots for a morphologically complex language.In this research, they seek effective solutions to create generative seq2seq-based chatbots from very small data. Since experiments are carried out in English and morphologically complex Lithuanian languages, they had an opportunity to compare results for languages with very different characteristics. They experimentally explore three encoder – decoder LSTM-based approaches (simple LSTM, stacked LSTM, and BiLSTM), three word embedding types (one-hot encoding, fastText, and BERT embeddings), and five encoder – decoder architectures based on different encoder and decoder vectorization units.

Chatbots with Personality Using Deep Learning [5]

-Susmit Gaikwad

This paper talks about the drawbacks of the current chatbots an how it can be solved.Natural Language Processing (NLP) requires the computational modelling of the complex relationships of the syntax and semantics of a language. While traditional machine learning methods are used to solve NLP problems, they cannot imitate the human ability for language comprehension. With the growth in deep learning, these complexities within NLP are easier to model, and be used to build many computer applications. A particular example of this is a chatbot, where a human user has a conversation with a computer program, that generates responses based on the user's input. In this project, they study the methods used in building chatbots, what they lack and what can be improved.They have also talked about using LSTM and also how convolution neural networks might be a good future for NLP based problems.

Deep reinforcement learning for dialogue generation

-Li J, Monroe W, Ritter A, Galley M, Gao J, Jurafsky D.

In this paper, they have generated intermediate response using Seq2Seq model with attention where input was raw text. Then, the intermediate generated responses were fed into Reinforcement Model and was rewarded based on Ease of answering, Information Flow and Semantic Coherence. This is forward centric model, where if generated response is easy to answer, contribute to more information compared to previous dialogue history and grammatically and semantically correct, they are rewarded. So they have basically combined the advantages of reinforcement learning with the basic seq2seq model

Overview

Problem statement

The main objective of this project Is to create a conversational chatbot , which gives moderately appropriate responses most of the time. We are going to stick with with the seq2seq encoder-decoder model as it has proven time and again to be an appropriate and accurate solution to many NLP problems. The entire model will be discussed later on in the upcoming sections. The dataset used in this project is the cornell movie dataset. Details about the dataset will also be mentioned in the upcoming sections.

Software Requirements

A) Tensorflow 2.1.0

B) matplotlib (Newest Version)

C) numpy

D) Jupyter notebook (Ipython)

E) sklearn

F) pandas

G) pillow

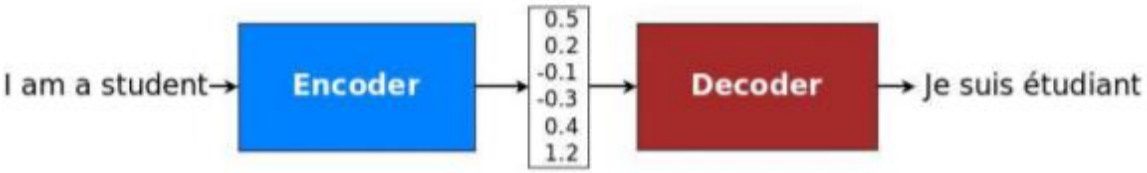
H) Pickle

There will be no specific hardware requirements for this project.

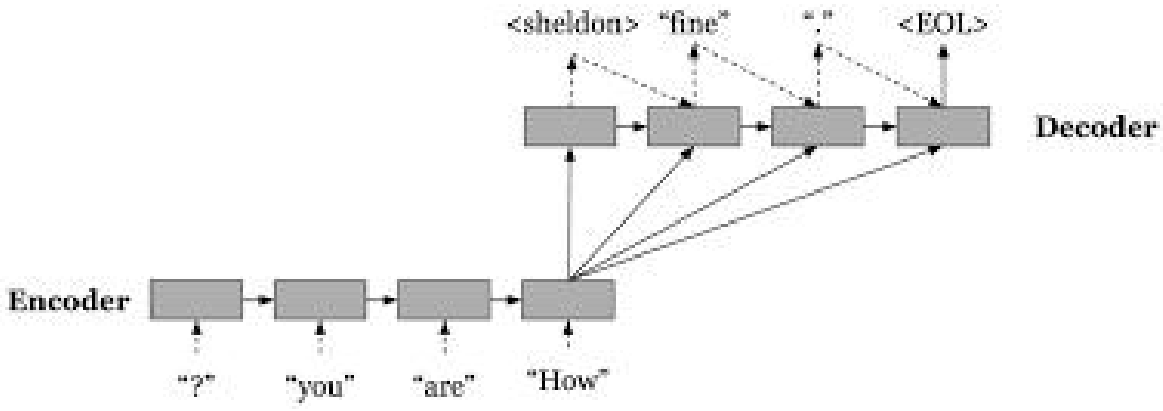
Algorithms and techniques

Google's Neural Machine Translator (GNMT)

Google's Neural Machine Translation (GNMT) is primarily used for machine translation. But, GNMT contains Sequence to sequence module with many enhancement techniques which can help build good dialogue generator. For translation, GNMT, does not apply traditional phrase-based translation systems where translation is performed by breaking up source sentences into multiple chunks and then translate phrase-by-phrase. It rather uses more human like translation approach.



The above is a general approach for NMT. An encoder converts a source sentence into a ” meaning” vector which is passed through a decoder to produce a translation



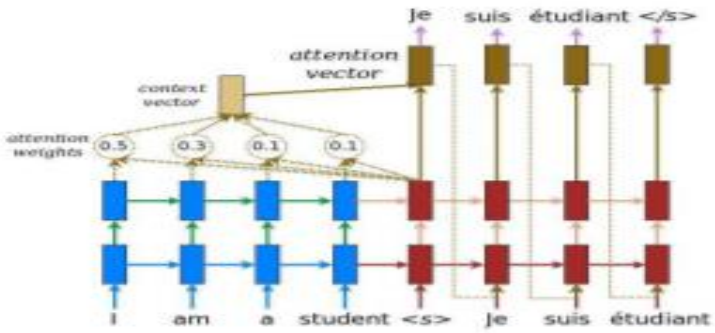
The Above is a more informative approach and description on how the encoder decoder model works.

GNMT is based on Seq2Seq architecture composed of encoder-decoder unit. GNMT can be 10 used with with different variation of architectures. The Seq2Seq module is composed of encoder and decoder. Encoder takes source text as input and processes the text to generate intermediate representation of input text called thought vector. The thought vector is then fed into the decoder input unit. The decoder now processes the thought vector and generates outputs. In case of dialogue generation problem the output is a response and for machine translation problem output is the target text. This architecture have shown to be capable of processing long-range dependencies and produce more fluent translations or responses.

Encoder-decoder units of GNMT's Seq2Seq module can have different architectures. Both encoder and decoder is composed of Recurrent Neural Network (RNN). But RNN of encoder and decoder unit can be composed of different cell types other than vanilla RNN cell, including Long Short-term Memory (LSTM), or a gated recurrent unit (GRU).
In this project we will be using a GRU along with the attention mechanism which we will see in the forthcoming sections.

Attention Mechanism

According to the schematic below, blue represents encoder and red represents decoder; and we could see that context vector takes all cell's outputs as input to compute the probability distribution of source language words for each single word decoder wants to generate. By utilizing this mechanism, it is possible for decoder to capture somewhat global information rather than solely to infer based on one hidden state.



We will now look into how to build the context vector. For a fixed target word, first, we loop over all encoder's states to compare target and source states to generate scores for each state in encoders. Then we could use softmax to normalize all scores, which generates the probability distribution conditioned on target states. At last, the weights are introduced to make context vector easy to train.

Given below are several formulae proposed by bahdanau explaining the process of calculating attention

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases} \quad (4)$$

The attention computation happens at every decoder time step. It consists of the following stages:

1. The current target hidden state is compared with all source states to derive attention weights

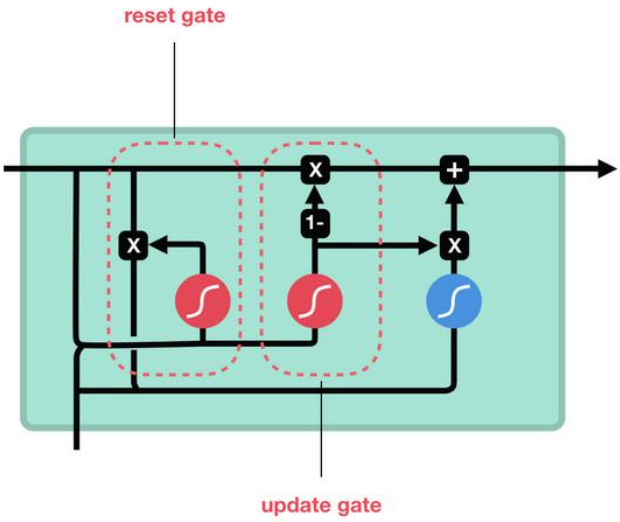
2. Based on the attention weights we compute a context vector as the weighted average of the source states.

3. Combine the context vector with the current target hidden state to yield the final attention vector.

4. The attention vector is fed as an input to the next time step.

GRU (Gated Recurrent Units)

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. So if you are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning. This is why LSTM and GRU were introduced.
LSTM's and GRU's were created as the solution to short-term memory. They have internal mechanisms called gates that can regulate the flow of information.These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions.



Given above is the a Basic GRU cell.

The update gate decides what information to throw away and what new information to add.
The reset gate is another gate is used to decide how much past information to forget.

The reason we Used GRU instead od LSTM in this project is GRU's are faster than LSTM computational wise as it has less operations to perform

Dataset

The dataset used in this project to train our model was the cornell movie Subtitle corupus.

About the Dataset

+)220,579 conversational exchanges between 10,292 pairs of movie characters.

+)involves 9,035 characters from 617 movies

+) in total 304,713 utterances

From This 300,00 sentences , only about 40,000 was included in this project as there was a time constraint and there was not enough computing power and space to train the model with the entire dataset. Around 32000 Training examples and 8000 Testing examples were taken

Preprocessing

Conversation data in the movie corpus contained Movie ID, Character ID, and Movie Line ID was separated by ”+++++”. For preprocessing, conversation data was cleaned to remove these meta-data(eg. movie ID, character ID, Line ID). Also, data separators(”+++++”) were eliminated. Additionally, some of the character in the data contained unsupported encoding format by UTF-8 standard and was hence removed. Finally, data was separated in two different files to assimilate with the format of Google’s Neural Machine Translation(GNMT) model input pipeline format where first file is the dialogue 1 and second one was response to dialogue 1.

For future work, more robust and real life conversation based corpus can be incorporated for dialogue generation model building if available. Other possible candidates for dialogue corpus includes, twitter data, reddit dataset and relay chat engine data corpus.

Implementation

Experimental setup

Language Used - Python

Primary Package used - Tensorflow

Editor used - Jupyter notebook

Data Storage - pickle and .data

GPU - 3GB NVidea GTX 1050

Ram - 8GB

Overview of Model Used

Deep Learning Module - Neural Machine Translator(GNMT)

Main algorithm - Seq2Seq Encoder Decoder with RNN

Enhancement techniques - GRU and Attention

Code

In [9]:
```
class Encoder(tf.keras.Model):
    def __init__(self,vocab_size,embedding_dim,enc_units,batch_size):
        super(Encoder,self).__init__()
        self.batch_size=batch_size
        self.enc_units=enc_units
        self.embedding=tf.keras.layers.Embedding(vocab_size,embedding_dim)
        self.gru=tf.keras.layers.GRU(self.enc_units,
                                     return_sequences=True,
                                     return_state=True,
                                     recurrent_initializer='glorot_uniform')
    def call(self,x,hidden):
        x=self.embedding(x)
        output,state=self.gru(x,initial_state=hidden)
        return output,state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_size,self.enc_units))
```

In [11]:
```
class BahdanauAttention(tf.keras.layers.Layer):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.W1 = tf.keras.layers.Dense(units)
        self.W2 = tf.keras.layers.Dense(units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, query, values):
        # query hidden state shape == (batch_size, hidden size)
        # query_with_time_axis shape == (batch_size, 1, hidden size)
        # values shape == (batch_size, max_len, hidden size)
        # we are doing this to broadcast addition along the time axis to calculate the score
        query_with_time_axis = tf.expand_dims(query, 1)

        # score shape == (batch_size, max_length, 1)
        # we get 1 at the last axis because we are applying score to self.V
        # the shape of the tensor before applying self.V is (batch_size, max_length, units)
        score = self.V(tf.nn.tanh(
            self.W1(query_with_time_axis) + self.W2(values)))

        # attention_weights shape == (batch_size, max_length, 1)
        attention_weights = tf.nn.softmax(score, axis=1)

        # context_vector shape after sum == (batch_size, hidden_size)
        context_vector = attention_weights * values
        context_vector = tf.reduce_sum(context_vector, axis=1)

        return context_vector, attention_weights
```

In [13]:
```
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.dec_units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')
        self.fc = tf.keras.layers.Dense(vocab_size)

        # used for attention
        self.attention = BahdanauAttention(self.dec_units)

    def call(self, x, hidden, enc_output):
        # enc_output shape == (batch_size, max_length, hidden_size)
        context_vector, attention_weights = self.attention(hidden, enc_output)

        # x shape after passing through embedding == (batch_size, 1, embedding_dim)
        x = self.embedding(x)

        # x shape after concatenation == (batch_size, 1, embedding_dim + hidden_size)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        # passing the concatenated vector to the GRU
        output, state = self.gru(x)

        # output shape == (batch_size * 1, hidden_size)
        output = tf.reshape(output, (-1, output.shape[2]))

        # output shape == (batch_size, vocab)
        x = self.fc(output)

        return x, state, attention_weights
```

Result

The model was Trained for 100 Epochs and the below loss was obtained. Considering the number of samples and the epochs the performance of the model is good.

```
Epoch 98 Loss 0.4342
Time taken for 1 epoch 87.63887429237366 sec

Epoch 99 Batch 0 Loss 0.4366
Epoch 99 Batch 100 Loss 0.4680
Epoch 99 Batch 200 Loss 0.3507
Epoch 99 Batch 300 Loss 0.4367
Epoch 99 Batch 400 Loss 0.4533
Epoch 99 Loss 0.4289
Time taken for 1 epoch 87.5113434791565 sec

Epoch 100 Batch 0 Loss 0.4673
Epoch 100 Batch 100 Loss 0.4748
Epoch 100 Batch 200 Loss 0.3857
Epoch 100 Batch 300 Loss 0.4323
Epoch 100 Batch 400 Loss 0.4544
Epoch 100 Loss 0.4252
Time taken for 1 epoch 87.93658494949341 sec
```

Some of the Translations to Basic sentences are also shown below

```
translate(u'<start> hi <end>')
```

```
Input: <start> hi <end>
Predicted translation: hi <end>
```

```
translate(u'<start> How are you <end>')
```

```
Input: <start> how are you <end>
Predicted translation: fine <end>
```

```
translate(u'<start> What are you doing <end>')
```

```
Input: <start> what are you doing <end>
Predicted translation: i don t know i m a real berserk <end>
```

```
translate(u'<start> where are you going <end>')
```

```
Input: <start> where are you going <end>
Predicted translation: i m going to go to the medicine <end>
```

```
translate(u'<start> can i come with you <end>')
```

```
Input: <start> can i come with you <end>
Predicted translation: of course <end>
```

```
translate(u'<start> would you like some tea <end>')
```

```
Input: <start> would you like some tea <end>
Predicted translation: no no <end>
```

```
translate(u'<start> am i disturbing you <end>')
```

```
Input: <start> am i disturbing you <end>
Predicted translation: maybe no <end>
```

Conclusion and future scope

The model produce good results taking into consideration the lack of computational power and training data.The training on Cornell Movie Subtitle corpus produced result which needs further improvement and more attention and speculation on training parameters. Adding more quality data will further improve performance. Also, the training model should be trained with other hyper-parameters and different dataset for further experimentation.To conclude I would say that rule-based system is not the only way to construct a good conversational chatbot, with a bit of parameter tuning and more training data i am sure deep learning will be a good solution for the conversational chatbot problem.

The chatbot developed using Google's Neural Machine Translation Model(GNMT) can be further improved with more robust, high quality real-life conversational datasets which could better emulate human interaction. Also, hyper-parameters of the GNMT model can be further fine-tuned and optimized for performance enhancement. Based on available opportunity to further advance the project, Deep Reinforcement Learning(RL) can be applied that could significantly improve performance as shown empirically in Dufarsky's paper. Reinforcement Learning algorithm can be applied after the initial training using Google's Neural Machine Translation.

REFERENCES

[1] https://www.researchgate.net/publication/328582617_Intelligent_Chatbot_using_Deep_Learning
[2] https://www.researchgate.net/publication/338100972_Conversational_AI_Chatbot_Based_on_Encoder-Decoder_Architectures_with_Attention_Mechanism
[3] http://www.ijstr.org/final-print/mar2020/Towards-Building-A-Neural-Conversation-Chatbot-Through-Seq2seq-Model.pdf
[4] https://www.researchgate.net/publication/340170837_A_Domain-Specific_Generative_Chatbot_Trained_from_Little_Data
[5] https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1678&context=etd_projects
[6] https://arxiv.org/abs/1606.01541
[7] https://towardsdatascience.com/seq2seq-model-in-tensorflow-ec0c557e560f
[8] https://blog.floydhub.com/attention-mechanism/
[9] https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21