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

Amir Ali, Muhammad Zain Amin
Department of Computer Science and Engineering
University of Engineering and Technology,
Lahore, Pakistan

Abstract:

Conversational AI Chatbot development using Artificial Intelligence and Machine Learning technique is an interesting problem of Natural Language Processing. In many research and development projects scientists are using AI, Machine Learning algorithms and Natural Language Processing techniques for developing Conversational AI Chatbot. The research and development of automated help desk and customer services through these conversation agents are still under progress and experimentation. Conversational AI Chatbot is mostly deployed by financially organizations like the bank, credit card companies, businesses like online retail stores and startups. Virtual agents are adopted by businesses ranging from very small start-ups to large corporations. There are many AI Chatbot development frameworks available in the market both program-based and interface based. But they lack the accuracy and flexibility in developing real dialogues. Among these popular intelligent personal assistants are Amazon’s Alexa, Microsoft’s Cortana and Google’s Google Assistant. The functioning of these agents is limited, and retrieval based agent which are not aimed at holding conversations that 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 paper, we have developed a Conversational AI Chatbot using modern-day techniques. For developing Conversational AI Chatbot, We have implemented encoder-decoder attention mechanism architecture. This encoder-decoder is using Recurrent Neural Network with LSTM (LongShort-Term-Memory) cells.

Keywords: Chatbot, LSTM, encoder-decoder, attention mechanisms.

1. Introduction:

Conversational AI Chatbot is a language recognition system able to maintain a conversation with a user using a question/answer protocol. The communication can be done by audio or text media. As far as concerned with this project, we are going to focus on the textual models. In today’s world, the way we interact with our digital devices is largely restricted, based on what features and accessibility each device offers. However simple it may be, there is a learning curve associated with each new device we interact with. Chatbots solve this problem by interacting with a user using text autonomously. Chatbots are currently the easiest way we have for software to be native to humans because they provide an experience of talking to another person [1].

Since chatbots mimic an actual person, Artificial Intelligence (AI) techniques are used to build them. One such technique within AI is Deep Learning which mimics the human brain. It finds patterns from the training data and uses the same patterns to process new data. Deep Learning is promising to solve
long-standing AI problems like Computer Vision and Natural Language Processing (NLP), with Google investing 4.5 million in Montreal AI Lab in addition to a federal AI grant of $213 million[2].

Conversational AI Chatbot is a program that generates a 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 a 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 retrieve relevant answers based on the query string. They rely on keyword similarity and retrieved texts pulled from internal or external data sources including World Wide Web 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.

Recently, there has been a major increase in interest in the use and deployment of dialogue generation systems. Many major tech companies are using a virtual assistant or chat agent to ill the needs of customers. Some of them include Google’s Google Assistant, Microsoft’s Cortana and Amazon’s Alexa. Though they are primarily questioning answering systems, their adoption by major corporations has peaked interesting customers and seems promising for more advanced conversational agent system research and development.

2. Related Work

There have been much recent development and experimentation in the conversational agent system. Apart from traditional chatbot development techniques that use rule-based techniques, or simple machine learning algorithms, many advanced chatbots are using advanced Natural Language Processing (NLP) techniques and Deep Learning Techniques like Deep Neural Network (DNN) and Deep Reinforcement Learning (DRL).

2.1 Sequence to Sequence (Seq2Seq)

Some of the states of the art techniques involve using Deep Neural Network and it’s architectural variations. Sequence to Sequence (Seq2Seq) model based on encoder-decoder architecture is such an architecture that is very popular for dialogue generation, language modeling and machine translation. Seq2Seq uses Recurrent Neural Network (RNN) which is a popular Deep Neural Network architecture, especially for Natural Language Processing tasks. In Sequence to Sequence (Seq2Seq) model, many to many RNN architecture is used for the decoder. In this, encoder-decoder architecture, the input sequence is fed as a vector representation of text to the encoder. Then, the encoder produces some intermediate representation of information or thought vectors. Consequently, the thought vector generated by the encoder is fed into a decoder as input. Finally, the decoder processes the thought vector and converts the sequence one by one word and produces multiple outputs from the decoder in the form of the target sequence. Though vanilla RNN is default in Seq2Seq and works well for many NLP problems yet, due to higher complexity of language modeling problem, vanilla recurrent neural network cells often fail, especially, where long sequence of information needs to be remembered, as this information frequently becomes large for bigger datasets and turns to information bottleneck for
the RNN network. Therefore, researchers use variations of the recurrent neural networks to handle such a problem.

Long-Short-Term-Memory (LSTM) is a special variant of the cell type of Recurrent Neural Network which has empirically shown to work well for language modeling. LSTM has forgotten gates along with input gates and output gates. This helps remember more relevant and contextual information and discards the rest of the sequence which is desirable in language modeling where dependency within the sequence is sparse. Also, instead of using unidirectional cells, bidirectional LSTM cells can perform much better.

Another technique, Neural Attention Mechanism embedded in the Seq2Seq module has significantly improved performance in dialogue generation system and other NLP tasks and thus become industry standard practice. In the Neural attention mechanism, each hidden target compares with the source hidden state, generates attention vector by calculating score and preserves the attention vector in memory to choose over another candidate. Also, other techniques like, Beam Search can help improve decoding performance further by choosing top candidates. Seq2Seq has also been applied for other NLP tasks including machine translation, text summarization, and question-answering and image captioning.

2.2 Google’s Neural Machine Translation (GNMT)

Google Neural Machine Translation (GNMT) is a neural machine translation (NMT) system developed by Google and introduced in November 2016, that uses an artificial neural network to increase fluency and accuracy in Google Translate [3] [4] [5].

Google’s Neural Machine Translation (GNMT) model is a module for neural machine translation from and to other languages and English. GNMT has also been used for dialogue generation experimentally. It is based on the Seq2Seq model which is popular in dialogue generation. Also, GNMT has many techniques embedded in the module which are crucial for intelligent chatbot development. The GNMT model includes Sequence to Sequence modeling with an encoder-decoder architecture built using uni or bi-directional LSTM cells. They also have the option for Neural Attention Mechanism, Beam Search, and vocabulary generation using Google’s sub-word module. Also, they have the option of adjusting the hyperparameters for better model training.

2.3 Deep Reinforcement Learning

"Deep Reinforcement Learning for Dialogue Generation"[5] of Dan Jurafsky, Deep Reinforcement Learning (DRL) has been used for developing long conversation chatbots. Seq2Seq model can generate coherent dialogues but may produce repeated generic responses regardless of input and can get stuck in a loop in longer conversations. This occurs as Seq2Seq predicts utterances one at a time while ignoring their influence on future outcomes. Seq2Seq models tend to generate highly frequent repeated responses like “I don’t know”. This is due to the high frequency of generic responses in the training set, also these replies are more compatible with a wide range of input text.
In Dufarsky’s paper, they have generated an intermediate response using Seq2Seq model with attention where input was raw text. Then, the intermediate generated responses were fed into Reinforcement Model and were rewarded based on Ease of answering, Information Flow and Semantic Coherence. This is a forward centric model, where if the generated response is easy to answer, contribute to more information compared to previous dialogue history and grammatically and semantically correct, they are rewarded despite the success, in this paper Dufarsky has stated that the RL model is not optimized to predict the next utterance. This model increased the long-term reward for the long conversation to keep the conversation going by reducing generic responses. But, less relevant responses are produced in their experimentation as there is a trade-off between relevance and less repetitiveness.

2.4 Limitations

Although there are many chatbots currently available, the majority of them are limited in functionality, domain function, context, and coherence. They often fail in long conversations and have reduced relevancy in dialogue generation. Most of this chatbots are developed for the restricted domain. The majority of them are using simple rule-based techniques. They perform well in question answering sessions and in very structured conversational modes. But, fail to emulate real human conversation and lacks flexibility in functioning. Some of the chatbots using machine learning algorithms often adhere to simple algorithms. They lack in complexity and sophistication needed to
produce good results specifically in open domain conversations. Some chat engines are available in
the market which is often used by businesses for developing automated customer support. They are
also black box and business clients have limited knowledge of their internal architectures. Hence, they
produce results that can become unreliable and fail to fill the need of customers. Following is an
example of failed chatbot replies.

![Figure 2.1: Some undesirable chatbot replied](image)

3. System Architecture

Figure 3.1 shows a Proposed Architecture for Conversation AI chatbot of the entire process.

![Figure 3.1: System Architecture](image)
3.1 Data Collection

Data were processed to prepare for the input pipeline of the Sequence to Sequence model (Seq-to-Seq). In the original Sequence to Sequence model (Seq-to-Seq) model, there were two input data files and two vocabulary file generated from input files. The two input files were a translation from and translation to the language input data file. The vocabulary files contained the processed vocabulary for the two input data file of two different languages respectively. Also, there were separate test and development file for source and target.

For developing final chatbot, popular movie subtitle corpus “Cornell movie subtitle corpus” has been used. This corpus contains a metadata-rich large collection of conversations extracted from raw movie scripts from popular movies.[9]

The following are found in the corpus –
- 220,579 conversational exchanges between 10,292 pairs of movie characters.
- involves 9,035 characters from 617 movies
- in total 304,713 utterances

Other movie meta-data included genres, release year, IMDB rating, number of IMDB votes, IMDB rating

3.2 Data Preprocessing

Conversation data in the movie corpus contained Movie ID, Character ID, and Movie Line ID was separated by "+++++".

For preprocessing, conversation data were cleaned to remove this meta-data (eg. movie ID, character ID, Line ID). Also, data separators ("+++++") were eliminated. Additionally, some of the characters in the data contained an unsupported encoding format by UTF-8 standard and was hence removed.

Finally, data were separated into two different files to assimilate with the format of Sequence to Sequence model (Seq-to-Seq) model input pipeline format where first file is the dialogue 1 and the second one was the response to dialogue 1.

After separating the two files, data in both ile was cleaned simultaneously. Everything except alphabetical character and some punctuation (., ??) was removed as they hold little meaning in conversation. Also, all the text was converted to lowercase. Then, multiple consequent occurrences of this punctuation (., ??) was reduced to one in order to reduce punctuation overload. Next, all the punctuation except (‘) was separated with a single space before and after for better performance in the Sequence to Sequence model (Seq-to-Seq) module. Finally, all the consequent multiple spaces were reduced to single space and each text string was trimmed to remove before and after space.

Also, data was cleaned for removing extraneous dialogues. If multiple consequent utterances from a single person were present everything except the last utterance for the person was stored. Initially, an utterance with more than 100 lengths was discarded for both text dialogue and their reply as with the increase of length the text, context relevance starts to drop due to diversity and limited data. But later full-text length was embedded.
After cleaning, the source and target text were spilt for training, testing and development/validation set with source and target format and was saved in files for final input pipeline feed.

For vocabulary generation, Google’s Sub-word Sequence to Sequence model (Seqto-Seq) module was used as suggested by the Google Tensorflow and Sequence to Sequence model (Seq-to-Seq) module documentation. The sub-word application was only applied to training files source and target files.

3.3 Encoder-Decoder with Attention Mechanism Model

3.3.1 Encoder-Decoder

Sequence To Sequence model introduced in Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation has since then, become the Go-To model for Dialogue Systems and Machine Translation. It consists of two RNNs (Recurrent Neural Network): An Encoder and a Decoder. The encoder takes a sequence(sentence) as input and processes one symbol(word) at each timestep. Its objective is to convert a sequence of symbols into a fixed-size feature vector that encodes only the important information in the sequence while losing unnecessary information. You can visualize data flow in the encoder along the time axis, as the flow of local information from one end of the sequence to another.

![Encoder-Decoder Architecture](image)

Figure 3.2: Encoder-Decoder Architecture [7]

Each hidden state influences the next hidden state and the final hidden state can be seen as the summary of the sequence. This state is called the context or thought vector, as it represents the intention of the sequence. From the context, the decoder generates another sequence, one symbol(word) at a time. Here, at each time step, the decoder is influenced by the context and the previously generated symbols.
3.3.2 Attention Mechanism

The attention mechanism, introduced in this paper, Neural Machine Translation by Jointly Learning to Align and Translate [8], allows the decoder to selectively look at the input sequence while decoding. This takes the pressure off the encoder to encode every useful information from the input.

How does it work? During each timestep in the decoder, instead of using a fixed context (last hidden state of the encoder), a distinct context vector $c_i$ is used for generating word $y_i$. This context vector $c_i$ is basically the weighted sum of hidden states of the encoder.

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j$$

where $n$ is the length of the input sequence, $h_j$ is the hidden state at time step $j$. 
\[ \alpha_{ij} = \exp(e_{ij}) / \sum_{k=1}^{n} \exp(e_{ik}) \]

eij is the alignment model which is the function of the decoder’s previous hidden state si−1 and the jth hidden state of the encoder. This alignment model is parameterized as a feedforward neural network that is jointly trained with the rest of the model.

Each hidden state in the encoder encodes information about the local context in that part of the sentence. As data flows from word 0 to word n, this local context information gets diluted. This makes it necessary for the decoder to peak through the encoder, to know the local contexts. Different parts of the input sequence contain information necessary for generating different parts of the output sequence. In other words, each word in the output sequence is aligned to different parts of the input sequence. The alignment model gives us a measure of how well the output at position i match with inputs at around position j. Based on this, we take a weighted sum of the input contexts (hidden states) to generate each word in the output sequence.

3.4 Training the Model

3.4.1 Training Dataset

Training has been completed on 225000*2 utterance of “Cornell movie subtitle corpus” conversation and has been tested with 5000+5000 utterance and validated with 5000+5000 utterance.

Table 3.1: Training Dataset

<table>
<thead>
<tr>
<th>Dataset (Text)</th>
<th>Total Number of Utterance(person 1 and person 2)</th>
<th>Total Size in MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Data</td>
<td>225000+225000</td>
<td>17+17</td>
</tr>
<tr>
<td>Testing Dataset</td>
<td>5000+5000</td>
<td></td>
</tr>
<tr>
<td>Validation Dataset</td>
<td>5000+5000</td>
<td></td>
</tr>
</tbody>
</table>

3.4.2 Training Model and Parameters

For training, Attention Mechanism has been applied to improve performance.

Table 3.2: Parameters

<table>
<thead>
<tr>
<th>Epochs</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>32</td>
</tr>
<tr>
<td>rnn size</td>
<td>1024</td>
</tr>
<tr>
<td>num layers</td>
<td>3</td>
</tr>
<tr>
<td>encoding embedding size</td>
<td>1024</td>
</tr>
<tr>
<td>decoding embedding size</td>
<td>1024</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>learning rate decay</td>
<td>0.9</td>
</tr>
<tr>
<td>min learning rate</td>
<td>0.0001</td>
</tr>
<tr>
<td>keep probability</td>
<td>0.5</td>
</tr>
</tbody>
</table>
4. Results

The following are some responses derived after training on the full dataset with 34MB of training text with 225000+225000 utterance. The initial test result produced moderately coherent sentences. The following responses were generated after the inference from the trained model. Inference, the trained model produced a 5-8 candidate response for each input. Hence, the following examples are most suited to 5-8 candidate responses during inference.

<table>
<thead>
<tr>
<th>Input from Person</th>
<th>Attention Mechnishm Model Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>hi</td>
<td>hows the weekend</td>
</tr>
<tr>
<td>great thanks</td>
<td>You're welcome</td>
</tr>
<tr>
<td>Hey happy birthday have a nice day</td>
<td>Thank you</td>
</tr>
<tr>
<td>Who are you</td>
<td>I am chatbot</td>
</tr>
<tr>
<td>You are bragging</td>
<td>I'm not bragging, I'm only answering ur questions.</td>
</tr>
<tr>
<td>What is your mood</td>
<td>I do not have any emotions.</td>
</tr>
<tr>
<td>Do you get mad</td>
<td>I could get mad</td>
</tr>
</tbody>
</table>

Following are the Perplexity and Bleu for test and development dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Perplexity</th>
<th>Bleu</th>
</tr>
</thead>
<tbody>
<tr>
<td>eval dev</td>
<td>50.01</td>
<td>10.0</td>
</tr>
<tr>
<td>eval test</td>
<td>48.22</td>
<td>10.3</td>
</tr>
</tbody>
</table>

5. Conclusion

The training data on Cornell Movie Subtitle corpus produced a result that 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 datasets for further experimentation.

6. References


[6] Sequence-to-Sequence learning and Neural Conversation model 2017/08/02
https://isaacchanghau.github.io/2017/08/02/Seq2Seq-Learning-andNeuralConversationalModel


[8] Neural Machine Translation by Jointly Learning to Align and Translate Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio (Submitted on 1 Sep 2014 (v1), last revised 19 May 2016 (this version, v7))


