Intelligent Chatbot using Deep Learning

(A project submitted in fulfillment of the requirements for the degree of Masters of Science in Computer Science)

Submitted By

Student Name : Anjana Tiha

UID : U00619942

University of Memphis

Spring, 2018

Date: 04/16/2018

Project Supervisor : Professor Vasile Rus

Project Committee Members -

Prof. Vasile Rus, Prof. Nirman Kumar, Prof. Kan Yang

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 Google's Neural machine Translation(NMT) 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 applied Neural Attention Mechanism and Beam Search during training.

Contents

1. 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.

2. Related Works

There have been many recent development and experimentation in 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 state 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 which 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 specially for Natural Language Processing tasks. In Sequence to Sequence (Seq2Seq) model, many to many RNN architecture is used for decoder. In this, encoder-decoder architecture, input sequence is fed as a vector representation of text to encoder. Then, encoder produces some intermediate representation of information or thought vectors. Consequently, the thought vector generated by encoder is fed into decoder as input. Finally, decoder processes the thought vector and converts the sequence one by one word and produces multiple output from the decoder in form of 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 fails, specially, 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 uses variations of recurrent neural network to handle such problem.

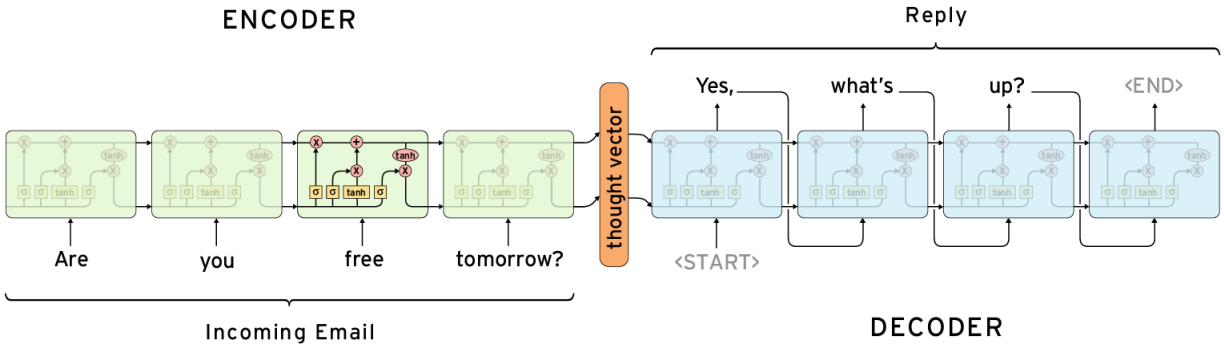


Figure 2.1: Sequence to Sequence Model

Long-Short-Term-Memory(LSTM) is a special variant of cell type of Recurrent Neural Network which has empirically shown to work well for language modeling. LSTM has forget 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 lan-

guage modeling where dependency within sequence is sparse. Also, instead of using unidirectional cells, bidirectional LSTM cells can perform much better.

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

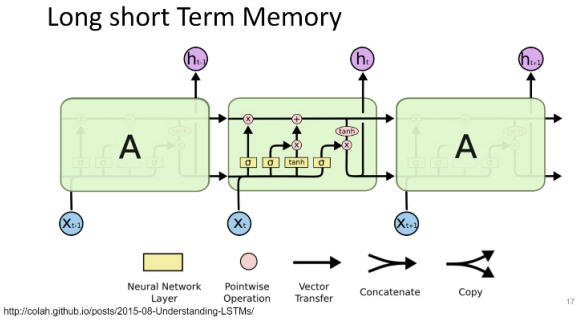


Figure 2.2: Ref 7

2.2 Google's Neural Machine Translation(GNMT)

Google's Neural Machine Translation(GNMT) model is a module for neural machine translation from and to other language and English. GNMT has also been used for dialogue generation experimentally. It is based on 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 encoder-decoder architecture built using uni or bi directional LSTM cells. They also have option for Neural Attention Mechanism, Beam Search, and vocabulary generation using Google's sub-word module. Also, they have option for adjusting the hyperparameters for better model training.

2.3 Deep Reinforcement Learning

"Deep Reinforcement Learning for Dialogue Generation"[1] 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 high frequency of generic responses in the training set, also this replies are more compatible with a wide range of input text.

Baseline mutual information model (Li et al. 2015)	**Proposed reinforcement learning model**
A: Where are you going? (1)	A: Where are you going? (1)
B: I'm going to the restroom. (2)	B: I'm going to the police station. (2)
A: See you later. (3)	A: I'll come with you. (3)
B: See you later. (4)	B: No, no, no, no, you're not going anywhere. (4)
A: See you later. (5)	A: Why? (5)
B: See you later. (6)	B: I need you to stay here. (6)
...	A: I don't know what you are talking about. (7)
...	...
A: how old are you? (1)	A: How old are you? (1)
B: I'm 16. (2)	B: I'm 16. Why are you asking? (2)
A: 16? (3)	A I thought you were 12. (3)
B: I don't know what you are talking about. (4)	B: What made you think so? (4)
A: You don't know what you are saying. (5)	A: I don't know what you are talking about. (5)
B: I don't know what you are talking about . (6)	B: You don't know what you are saying. (6)
A: You don't know what you are saying. (7)	...
...	...

Table 1: **Left Column**: Dialogue simulation between two agents using a 4-layer LSTM encoder-decoder trained on the OpenSubtitles dataset. The first turn (index 1) is input by the authors. Then the two agents take turns conversing, taking as input the other agent's prior generated turn. The output is generated using the mutual information model (Li et al., 2015) in which an N-best list is first obtained using beam search based on p(t|s) and reranked by linearly combining the backward probability p(s|t), where t and s respectively denote targets and sources. **Right Column**: Dialogue simulated using the proposed reinforcement learning model. The new model has more forward-looking utterances (questions like "Why are you asking?" and offers like "I'll come with you") and lasts longer before it falls into conversational black holes.

In Dufarsky's 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.

Input Message	Mutual Information Model	RL
How old are you?	I'm 16.	I'm 16, why are you asking?
What's your full name?	I have no idea.	What's yours?
I don't want to go home tonight.	Really?	Why?
Do you have any feelings for me?	I don't know what you are talking about.	I don't want to hurt your feelings.
How much time do you have here?	Not long enough. Sorry, sir.	Ten seconds.
Shall we get started?	Of course!	Yes. We've got a lot of work to do here.
Do you play football?	No, i don't	Yes. I love football!
We'd have to talk to him.	I mean, he's a good guy	About what ?
How come you never say it?	Because I don't want to hurt you.	I don't think it's a good idea to say it.

Table 3: Sampled responses generated from the mutual information models and the proposed RL model.

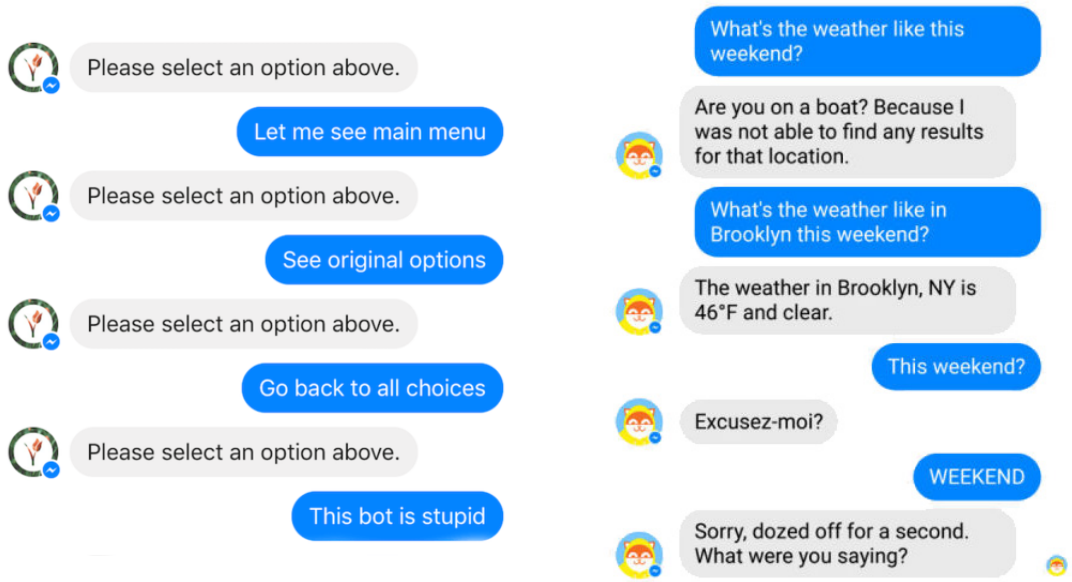
Despite success, in this paper Dufarsky has stated that the RL model is not optimized to predict the next utterance. This model increased long-term reward for longer conversation to keep conversation going by reducing generic response. But, less relevant responses are produced in their experimentation as there is trade-off between relevance and less repetitiveness.

3. Limitations

Although, there are many chatbots currently available, 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 these chatbots are developed for restricted domain. 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 result specifically in open domain conversation.

Some chat engines are available in market which are 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 3.1: Some undesirable chatbot replies (ref 5)



4. Deep Neural Network for Chatbot

4.1 Recurrent Neural Network

Recurrent Neural Network is a special Deep Neural Network Architecture used predominantly in Natural Language Processing (NLP) problems. In traditional Deep Neural Network, memory or sequence information is not taken into account. But, in Recurrent Neural Network, the sequential information is stored in memory and utilized for further processing which makes RNN suitable for sequential data or time series data where dependency exists in sequence.

4.1.1 Recurrent Neural Network Architecture

Recurrent Neural Network (RNN) is composed of input layer, multiple hidden layers and output layer. In input layer, input is feed as vector representation. Then, input vector is multiplied by some weight and some biases are added. Then, the output from input layer is passed to next hidden layer where each consecutive hidden layer is composed of numerous RNN cells. After getting output from input layer, the cells in hidden layer multiplies the generated output from input layer by their own cell weights and biases. Next, in each of the hidden layer cells, some global activation function (sigmoid, tangent) is applied to generate output from hidden layer. Then, output from each hidden layer cells is passed to successive hidden layer. Similar to previous hidden layer cells, some weight, biases and activation function is applied to the input of current hidden layer cell. This procedure propagates though all consequent hidden layers. Finally, output generated from the final hidden layer is passed to output layer and the output layer applies some function (e.g. Softmax) to generate final output. For RNN, the output vector from final output layer is then again fed into the input layer as an input vector. Hence, the sequence information is stored in the memory and utilized.

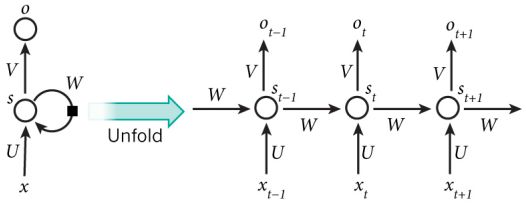


Figure 4.1: Recurrent Neural Network

But, vanilla RNN stores the complete sequence information. For large dataset with longer sequences, this can cause information bottleneck for the network. It may cause network to perform poorly due to information overload. As in many cases, complete sequential information may not be relevant in many NLP task including dialogue generation and can cause model to perform poorly. This problem is solved by special type of RNN cell Long-Short-Term-Memory(LSTM).

4.1.2 Long-Short-Term-Memory(LSTM)

Long-Short-Term-Memory(LSTM) is a special type of Recurrent Neural Network cell, which solves the data bottleneck for longer sequences. LSTM has forget gate along with the input and output gates. This helps remember longer sequence without overloading the network by discarding less relevant information.

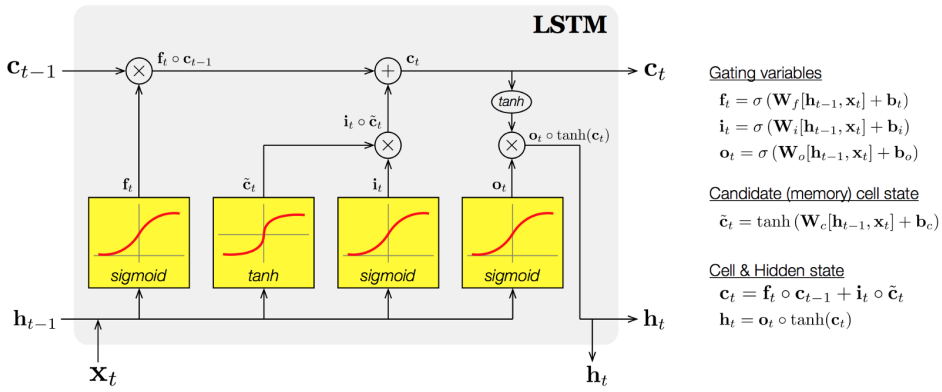


Figure 4.2: Long-Short-Term-Memory(LSTM)

There are couple of variation of LSTM. Following is the equation for LSTM with forget gate.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t.c_{t-1} + i_t.\sigma_g(\sigma_g(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t.\sigma_h(c_t)$$

Variables

$x_t \in R^d$: input vector to the LSTM unit
$f_t \in R^h$: forget gate's activation vector
$i_t \in R^h$: input gate's activation vector
$o_t \in R^h$: output gate's activation vector
$h_t \in R^h$: output vector of the LSTM unit
$c_t \in R^h$: cell state vector

$W \in R^{h \times d}, U \in R^{h \times h} \quad and \quad b \in R^h$: weight matrices and bias vector parameters which need to be learned during training where the superscripts d and h refer to the number of input features and number of hidden units, respectively.

Activation functions

σ_g : sigmoid function.
σ_c : hyperbolic tangent function.
σ_h : hyperbolic tangent function,$\sigma_h(x) = x$.

5. Architecture

In this project, I have developed Intelligent conversational agent following state of the art techniques proposed in recently published research papers. I have used Google's Neural Machine Translation(GNMT) module for building dialogue generator. Although, Google's Neural Machine Translation(GNMT) module is primarily used for Machine Translation tasks, they have empirically shown to be successful in other NLP tasks including dialogue generation and text summerization. GNMT has rich Seq2Seq module with many additional features for dialouge generation. Seq2Seq is a industry standard choice for dialogue generation and many NLP tasks. Althought, there exists a separate Seq2Seq module[4] which was the the early robust Seq2Seq module, but is not currently compatible with latest version of Google's machine learning framework Tensorflow and is only compatible with Tensorflow v1.0 whereas current Tensorflow is v1.6. Also, earlier version of Tensorflow had a minimalistic Seq2Seq module, it is not efficient for building robust model. Currently, Google does not offer any robust Seq2Seq module in it's official Machine Learning Framework Tensorflow and has moved many functionality of Seq2Seq to NMT model. Moreover, GNMT has many techniques embedded for dialogue generation agent development. GNMT model includes, Sequence to Sequence modeling (Seq2Seq) with encoder-decoder architecture based on uni or bi directional LSTM cells. GNMT also have option for Neural Attention Mechanism, Beam Search, and vocabulary generation using Google's sub-word module.

5.1 Google's Neural Machine Translation (GNMT)

5.1.1 Sequence to Sequence (Seq2Seq) Architecture

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.

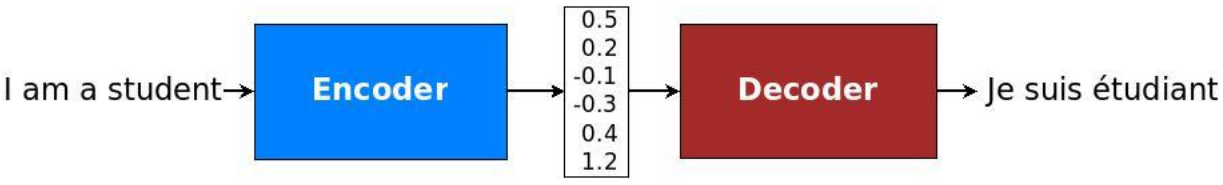


Figure 5.1: Encoder-decoder architecture – example of a general approach for NMT. An encoder converts a source sentence into a "meaning" vector which is passed through a decoder to produce a translation.

GNMT is based on Seq2Seq architecture composed of encoder-decoder unit. GNMT can be

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). Also, encoder and decoder can be composed of unidirectional or bidirectional RNN. But, it has been empirically found that, LSTM works well for dialogue generation and other language problems as full sequence text can become information bottleneck and in many cases complete sequence is not required for efficient dialogue generation. Also, bidirectional RNN can help improve performance further more. In my project I have used GNMT module with bidirectional LSTM cell. The architecture of this project consists of 2 layer encoder-decoder unit with 512 units.

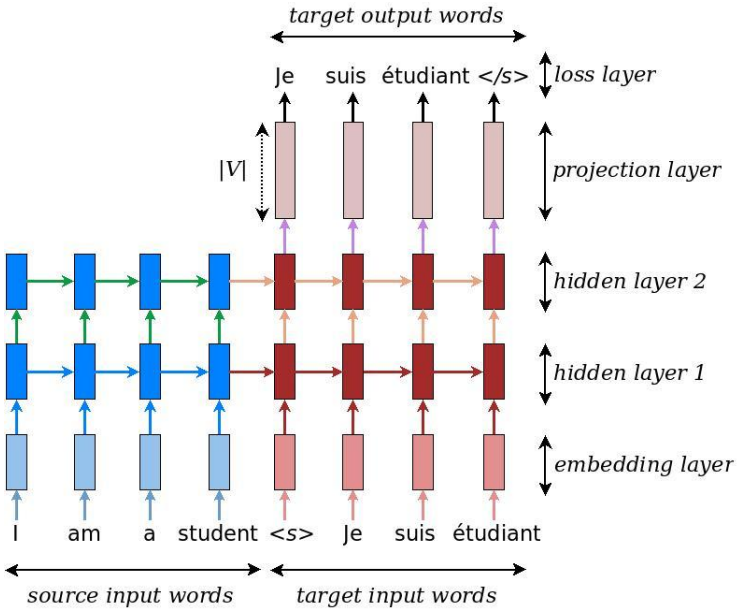


Figure 5.2: Neural machine translation – example of a deep recurrent architecture proposed by for translating a source sentence "I am a student" into a target sentence "Je suis étudiant". Here, "$< s >$" marks the start of the decoding process while "$< /s >$" tells the decoder to stop.

Neural Attention Mechanism[2]

To build state-of-the-art neural machine translation systems, the attention mechanism works specially well, which was first introduced in 2015. The idea of the attention mechanism is to form direct short-cut connections between the target and the source by paying ”attention” to relevant source content as we translate[2].

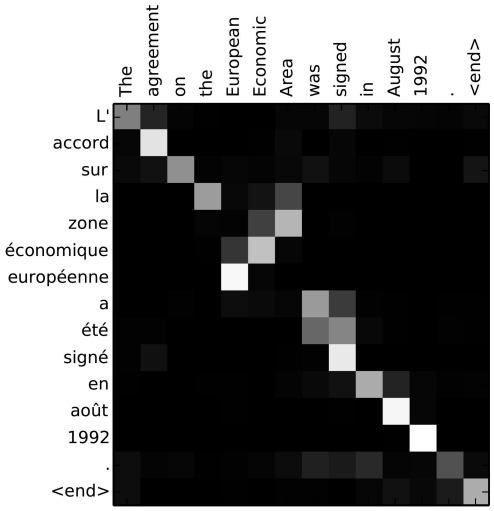


Figure 5.3: Attention visualization – example of the alignments between source and target sentences

In the vanilla seq2seq model, the last source state from the encoder to the decoder is passed when starting the decoding process. This works well for short and medium-length text string. However, for long sentences, the single fixed-size hidden state becomes an information bottleneck. Instead of discarding all of the hidden states computed in the source RNN, the attention mechanism provides an approach that allows the decoder to peek at them (treating them as a dynamic memory of the source information). This attention mechanism improves the translation of longer sentences. Attention mechanisms are the industry standard and applied to other NLP tasks including image caption generation, speech recognition, and text summarization.

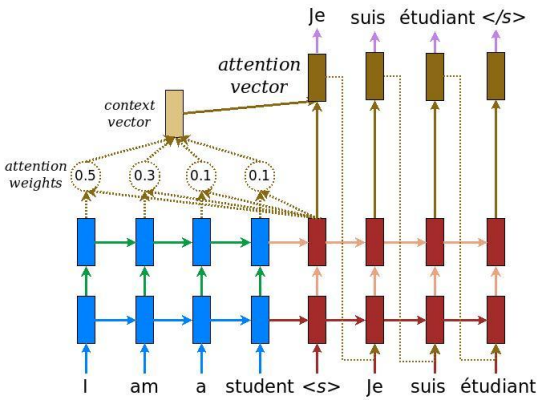


Figure 5.4: Attention mechanism – example of an attention-based NMT system.

As illustrated in Figure 5.4, 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. The first three steps can be summarized by the equations below:

$$a_{ts} = \frac{exp(score(h_t, \bar{h_s}))}{\sum_{s'=1}^{S} exp(score(h_t, \bar{h_{s'}}))} \quad \text{[Attention weights]} \quad (1)$$

$$c_t = \sum_s a_{ts}\bar{h_s} \quad \text{[Context vector]} \quad (2)$$

$$a_t = f(c_t, h_t) = tanh(W_c[c_t, h_t]) \quad \text{[Attention vector]} \quad (3)$$

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

Here, score function is used to compare the target hidden state "h_t" with each of the source hidden states "$\bar{h_s}$". The result is then normalized to produce attention weights which is a distribution over source positions. There are different choices for score function including multiplicative and additive forms. Once computed, the attention vector "a_t" is used to derive the softmax logit and loss.

6. Data

6.1 Data Collection

In this project, the dataset "Cornell Movie Subtitle Corpus" has been primarily used for final model training. Few other dialogue corpus based on movie subtitles have been preprocessed and cleaned for GNMT training including "Open Movie Subtitle Corpus" and "Movie Subtitle Corpus". But due to lack of data quality, they have been eliminated from final training.

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.

6.1.1 Dataset 1: Cornell Movie Subtitle Corpus

For developing final chatbot, popular movie subtitle corpus "Cornell movie subtitle corpus" has been used. This corpus contains metadata-rich large collection of conversations extracted from raw movie scripts from popular movies.
The following are found in 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

The data corpus can be found in http://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html.

6.2 Data Preprocessing

Data was processed to prepare for input pipe line of the Google's Neural Machine Translation(GNMT) model. In the original GNMT model, there were two input data files and two vocabulary file generated from input files. The two input files were translation from and translation to language input data file. The vocabulary files contained the processed vocabulary for the two input data file of two different language respectably. Also, there were separate test and development file for source and target.

6.2.1 Preprocessing of Cornell Movie Subtitle Corpus

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.

After separating the two files, data in both file 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 occurrence of these punctuation (. , ?!') was reduced to one in order to reduce punctuation overload. Next, all the punctuation except (') was separated with single space before and after for better performance in GNMT module. Finally, all the consequent multiple space was 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 utterance from single person was present everything except the last utterance for the person was stored. Initially, utterance with more than 100 length was discarded for both text dialogue and their reply as with increase of length the text, context relevance starts to drop due to diversity and limited data. But later full text length was embedded.

After through clenaing, the source and target text was splitted 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 Neural Machine Translation(NMT) module was used as suggested by the Google Tensorflow and Google's Neural Machine Translation module documentation. The sub-word application was only applied on training files source and target files.

7. Implementation Summary

Table 7.1: Algorithm Details

Deep Learning Module	:	Google's Neural Machine Translation (NMT).
Algorithm	:	Deep Neural Network (DNN), Recurrent Neural Network (RNN)
Main Technique	:	Sequence to Sequence (Seq2seq) modeling with encoder and decoder
Enhancement Techniques	:	Long Short Term Memory (LSTM) based RNN cell, Bidirectional LSTM, Neural Attention Model and Beam Search.

Table 7.2: Front-end and Back-end

Front-end	:	Python with PyQT.
Back-end	:	Python.

8. Hardware Specification

For training, personal laptop with 7th Gen Intel Core i5 Processor, dedicated graphics-NVIDIA GTX 1050 and 8GB Ram has been used. For further extended training, cloud computing platform Amazon AWS can be deployed.

Table 8.1: Hardware Platform- Local machines (personal laptop)

Processor	:	7th Gen Intel Core i5-7300HQ mobile processor.
Ram	:	8GB.
Graphics Card	:	NVIDIA GTX 1050 graphics.

9. Experiment

9.1 Training

9.1.1 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 9.1: Training Dataset

Dataset (Text)	Total Number of Utterance(person 1 and person 2)	Total Size in MB
Training Data	225000+225000	17+17
Testing Dataset	5000+5000	
Validation Dataset	5000+5000	

9.1.2 Training Model and Parameters

For training, GNMT module based on Bidirectional LSTM cells have been used. Neural Attention Mechanism have been applied to improve performance. Beam search was also applied on training but was later discarded. Total 3200 iteration with 512 hidden unit was trained with 2 layers.

Parameter Name	Value
Num of Train Steps	3200
Steps Per Stats	100
Num of Units	512
Num of Layers	2
Cell Type	LSTM
Encoder type	Bidirectional
Neural Attention	scaled luong

Parameter Name	Value
Optimizer	adam
Learning Rate	0.001
Decay Steps	1
Start Decay Step	1
Beam Width	10
Dropout	0.2
Metrics	bleu

10. Result

Following are some response derived after training on 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 inference from trained model. In inference, trained model produced 10-30 candidate response for each input. Hence, the following examples are most suited from 10-30 candidate responses during inference.

Input (Person 1)	GNMT Model Output (Person 2 Response)
how are you	how am i?
whats your plan for the weekend ?	i dont know
would you like some tea	would you like some help ?
You got something on your mind	yeah
would you like to go to watch movie ?	no .
where would you like to go for lunch today	i know where to go for lunch today .
i know where to go for lunch today .	you know where to go ?
Where is the library located ?	is that where you were ?
weather seem warm would you like to get some cold drinks	would you like to get some cold coffee ?
did you like the new startrek movie sequel	i did .
band played really well i am going for their next tour	i am going for their next interview .
am i disturbing you ?	no . not at all . where are you ?
i'm heading out . how about you ?	i got to wait for a call .

Following are the Perplexity and Bleu for test and development dataset

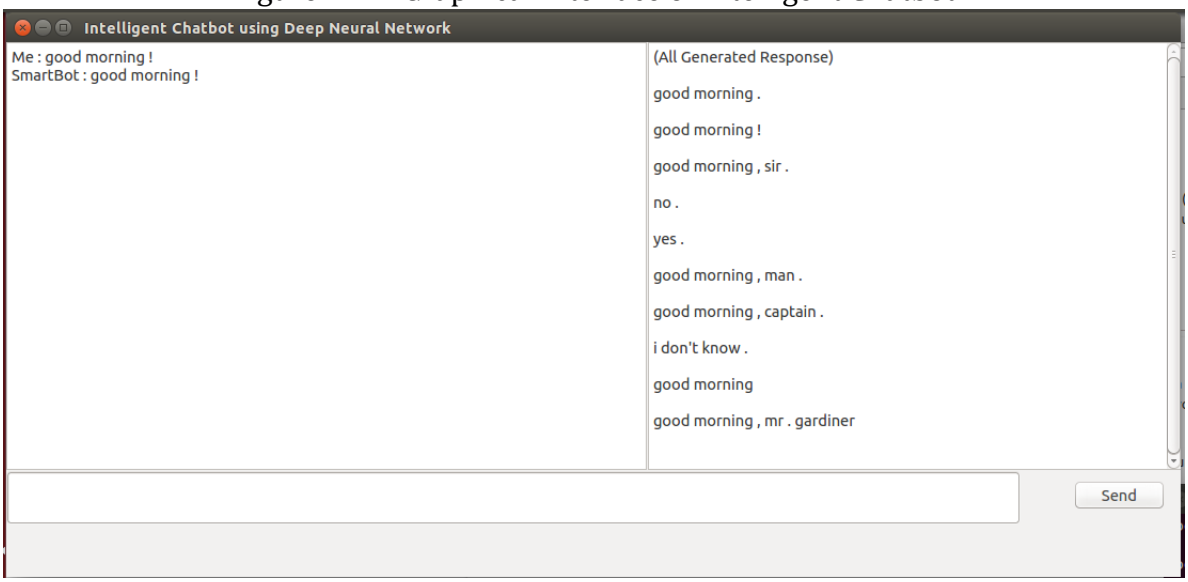
Table 10.1: Evaluation

Dataset	Perplexity	Bleu
eval dev	50.76	10.1
eval test	46.82	10.6

11. Graphical Interface (GUI)

Desktop Graphical Interface (GUI) was developed using Python and PyQT module. On left top side, the chat response and history is shown, and on the right textarea, all possible response of input text was shown. As during chat model training, number of output or translation was around 30, more than one possible response is shown. On the left, one response of input text was randomly chosen from resultant responses generated after interference.

Figure 11.1: Graphical Interface of Intelligent Chatbot



12. Challenges

The challenge in developing chatbot or dialogue generator lies in developing coherent dialogue generation system. As the model used in this experiment is for machine translation, the dialogue generation is treated as translation problem, where history of earlier conversations are not taken into account. Hence, the model can be limited in performance regarding long conversation.

Also, training is a long process which demands higher processing power and configured computing machine. Another problem is finding right hyper parameters to optimize the translation module for chat bot or dialogue generation system.

There are multiple chatbots developed using GNMT or Seq2Seq module. GNMT is a rather self-contained with Bi-directional LSTM cells, Neural Attention mechanism and Beam search techniques. Most of this features improve dialogue generation as well as machine translation. Bi-directional LSTM cells with attention mechanism seems to produce better output. Seq2Seq module also has some of the advanced features of GNMT. Nevertheless, developing chatbot algorithm from scratch by building RNN, bidirectional LSTM and neural attention techniques would be better suited option as GNMT is primarily for machine translation. But, this will require multiple trial and error before reaching optimal performance for the comprehensive chatbot module and hence is better suited as research problem.

After training, chatbot produced results with moderate relevancy. But many of the output were repeative and generic. Also, due to lack of real-life quality data the chatbot performed somehow below optimum for imitating human interaction. Also, many utterance was discarded due to longer length or discrepancy. And, number of training utterance was much less than required and test and development dataset was quite larger in comparison which might have caused the model to under-perform. Also, as data was limited, longer period of training may not have suited the dialogue generation problem.

13. Discussion

Development of dialogue agent using Neural Machine Translation(NMT) is widely practiced. Some other approaches use only Sequence to Sequence modeling. Many people also use their own Sequence to Sequence module. But, due to their lack complexity they fail to perform well. However, with more time and effort building a more comprehensive dialogue generation unit could solve dialogue generation problem is specifically better. Hence, Sequence to Sequence module with encoder-decoder architecture built on bi directional LSTM cells and Neural attention mechanism,Beam Search could be a more preferable task.

Apart from algorithm improvement, performance could be further optimized if better datasets were available. Before using Cornell Movie Dataset, I have experimented with other datasets and preprocessed and trained them on the GNMT module. But, due to lack of data quality, they were later discarded from training. Also, the datasets available used here are movie subtitles, which hardly mimics real human interaction. More realistic and high quality real life conversational data could further emulate the need and personality of the user. For future training, even personal chat history can be incorporated to give intelligent chatbot some personality.

The intelligent chatbot has produced moderate result. But, some of replies looks repetitive and lacks proper relevancy. This can be reduced with addition of more diverse and healthy data. Also, adding more length to original text could help improve responses and make them more relevant. A large proportion of data was lost due to choosing restrictive length where utterance more than 100 length was discarded but later full length text was adopted. This might have caused redundancy in training process. Also, repeated utterance by same character was eliminated and only last utterance by repeated speaker was kept,

which further reduced data size. Therefore more data with optimum sequence length can help build more intelligent chatbot.

14. Future Work

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.

15. Conclusion

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. This was an attempt to experiment with Deep Neural Network for dialogue generation inorder to develop intelligent chatbot.

Bibliography

[1] Li J, Monroe W, Ritter A, Galley M, Gao J, Jurafsky D. *Deep reinforcement learning for dialogue generation.* arXiv preprint arXiv:1606.01541. 2016 Jun 5.

[2] *Google's Neural machine translation(NMT)* https://github.com/tensorflow/nmt

[3] *Neural Machine Translation (seq2seq) Tutorial* https://www.tensorflow.org/tutorials/seq2seq

[4] *seq2seq Github Repository* https://github.com/google/seq2seq

[5] *Google's seq2seq* https://google.github.io/seq2seq/

[6] *Cornell Movie Dialogs Corpus* http://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

16. References

1. https://medium.com/swlh/chatbots-of-the-future-86b5bf762bb4
2. https://chatbotsmagazine.com/to-build-a-successful-chatbot-ask-these-5-questions-b7fe3776c74c
3. https://labs.bawi.io/creating-a-conversational-chatbot-using-wit-ai-6eba3c625f4f
4. https://www.ipsoft.com/2017/11/20/when-chatbots-fail-virtual-agents-step-in/
5. https://isaacchanghau.github.io/2017/08/02/Seq2Seq-Learning-and-Neural-Conversational-Model/
6. https://chatbot.fail/
7. https://www.altoros.com/blog/text-prediction-with-tensorflow-and-long-short-term-memory-in-six-steps/
8. https://medium.com/the-mission/11-best-uses-of-chatbots-right-now-1c27764b7e62
9. https://www.marutitech.com/7-reasons-why-business-needs-chatbot/
10. https://chatbotsmagazine.com/how-to-develop-a-chatbot-from-scratch-62bed1adab8c
11. https://www.marutitech.com/complete-guide-bot-frameworks/
12. https://en.wikipedia.org/wiki/Long_short-term_memory

17. Image Credits

1. https://regmedia.co.uk/2017/09/21/chatbot.jpg?x=442&y=293&crop=1
2. https://cognitiveseo.com/blog/wp-content/uploads/2017/06/Is-Chatbot-Marketing-the-future-1024x512.jpg
3. http://nyintl.net/site/wp-content/uploads/2018/01/robot-customer-service.png
4. https://cdn.zmescience.com/wp-content/uploads/2015/06/robot.jpg
5. http://www.netalogue.com/wp-content/uploads/2017/03/chatbot-880x541.png
6. https://betanews.com/wp-content/uploads/2017/08/chatbot-e1501792996887.jpg
7. https://www.adityathakker.com/wp-content/uploads/2017/05/Facebook-Messenger-ChatBot-tutorial.jpg
8. https://cdn-images-1.medium.com/max/1200/1*FYFI4jbAUMqbXxlo6V_lBA.png
9. http://sunnyapps.co/assets/images/temp/sunnyapps-chatbots-about.png
10. https://www.liveworld.com/wp-content/uploads/2017/07/Chatbots-in-customer-service_cropped.png
11. https://cdn-images-1.medium.com/max/1200/1*fNPqCg4vBiCQS1-nFs7gYA.jpeg
12. https://cdn-images-1.medium.com/max/1600/0*LyfY3Mow9eCYlj7o.
13. https://www.dailydot.com/wp-content/uploads/0da/df/dominos.jpg
14. https://bot-hub.com/media/print_screen/Yahoo-Weather/yahooweather__4.jpeg
15. https://chatfuel.com/images2/og_chatfuel.jpg
16. https://4.bp.blogspot.com/-UWs8Wger9LQ/WQ8Uh05oMzI/AAAAAAAABrc/oFhlMOasZZIxjDZOxmX s1600/bottr-for-blogger.jpg
17. http://static.hackingrevenue.com.s3.amazonaws.com/2016/08/motionai_0.png
18. https://botlist.co/system/BotList/Bot/logos/000/000/627/medium/recast_ai.png
19. https://techcrunch.com/wp-content/uploads/2016/09/api.jpg
20. http://www.dieproduktmacher.com/wp-content/uploads/2017/08/lex-vs-api.png
21. http://bodhiinfo.com/blog/wp-content/uploads/2016/09/google-apiai.jpg
22. https://cdn-images-1.medium.com/max/860/1*QoCscxdXl2mkWqL7mGfALA.png