

Conversation Agent – Chatbot

Team Members: Abhay Marathe, Kuldeepsinh Chudasama, Prashant Desai

1. Introduction

Conversational modelling (Chatbot) is an important activity in Machine Intelligence and Natural Language Processing. Initially chatbots were built on rule-based architecture in which pattern recognition was performed on input and response was chosen from canned output. e.g. Pattern-action rules (Eliza) and a mental model (Parry). Now-a-days, machine learning algorithms have enabled tuning of various inputs to enhance understanding of the question and chatbot is able to generate the output instead of selecting from canned responses. Earlier chatbots like Eliza were implemented using regex and now various algorithms are used to get the content-based response. Chatbot has wide range of applications in the field of business, education, banking, manufacturing etc. Moreover, it is used for customer support, product suggestion, e-commerce, finance, healthcare, insurance, media and travel industry. In our project, we have presented an approach for building chatbot using Seq2Seq model. This model is based on a recurrent neural network which reads the input sequence one token at a time. Our chatbot is first trained using cornel movie dialogue corpus and then chatbot is able to answer the questions using the learned data.

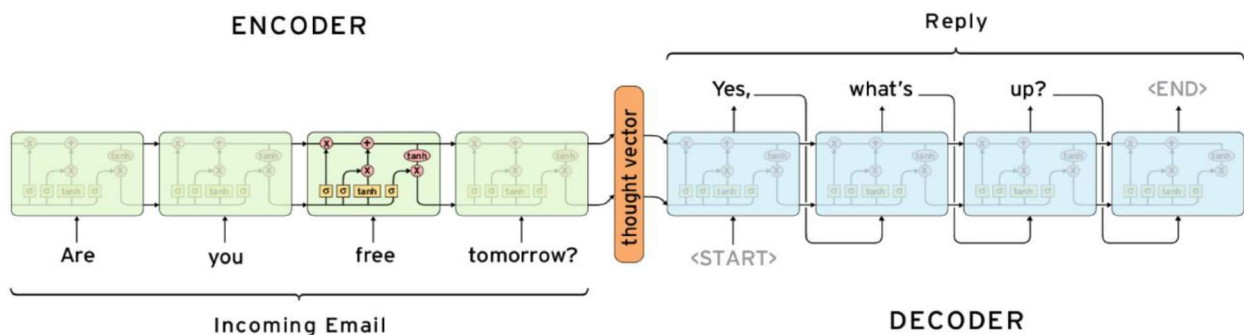
2. Background

We have referred several research papers related to chatbot development. A research paper authored by Q. Vera Liao, Muhammed Mas-ud Hussain and others - ‘*Conversations with a Question-and-Answer Chatbot in the Wild*’^[1] gave us an idea regarding deployment of a chatbot, where authors have reported on users’ interest areas in conversational interactions to inform the development of chatbot. By studying log data from a field deployment of a question and answer conversational agent, we characterize the rich forms of conversational interactions users had with the agent. **After collecting the data author performed various preprocessing functions using NLTK library. Also, we referred ‘*Neural Machine Translation by Jointly Learning to Align and Translate*’^[4] authored by Dzmitry Bahdanau, KyungHyun Cho and Yoshua Bengio and ‘*Sequence to Sequence Learning with Neural Networks*’^[3] authored by Ilya Sutskever, Oriol Vinyals and Quoc V. Le. In this referenced research paper, they used Tensorflow framework to perform sequence to sequence modelling. Authors proposed the Seq2seq modelling using LSTM and RNN. Moreover, they have used an LSTM-like RNN architecture to map sentences into vectors and back, although their primary focus was on integrating their neural network into an SMT (Statistical Machine Translation) system. In the RNN architecture they have proposed the bidirectional methods for encoders and decoders.** We observed that a fixed-length context vector is problematic for translating long sentences in their work. While in our project we implemented Multi-layered RNN cells for encoders and decoders.

3. Project

In this project, we used Cornell University's [Movie Dialogue Corpus](https://www.cornell.edu/movie-dialogue-corpus/) to design the chatbot. It includes 220,579 conversational exchanges between 10,292 pairs of movie characters, involving 9,035 characters from 617 movies with 304,713 total utterances. This corpus contains a large metadata collection of fictional conversations extracted from raw movie scripts. We used TensorFlow to develop our chatbot. It relies on the construction of dataflow graphs with nodes that represent mathematical operations and edges represent tensors. Along with that we implemented Seq2Seq model using LSTM (Long Short-Term Memory) cells, a Multi-layered cell RNN (Recurrent Neural Networks) and decoders with attention. Encoders job is to encapsulate the information of the input text into a fixed representation. The decoder takes this representation and generates a variable length text that best responds to it. We referred '*Sequence to Sequence Learning with Neural Networks*'^[3] to understand and implement the concept of Seq2Seq in chatbot.

RNN contains several hidden state vectors, which each represents information from previous time steps. For example, hidden state vector at 3rd time step will be a function of first 3 words. By using this logic, final hidden state vector of the encoder RNN can be thought of as a pretty accurate representation of the whole input text. The decoder is another RNN, which takes in the final hidden state vector of the encoder and uses it to predict the words of the output reply.



Source <https://research.googleblog.com/2015/11/computer-respond-to-this-email.html>

After gathering the data, we preprocessed the data as following:

- We have formatted Cornell Movie Data Corpus into a question-answer format by extracting the conversions from movie_conversations.txt. We have used normal string splicing and split operations to clean the text data.
- Further, we split the entire question-answer data into encoder-decoder files and have created train and test encoder-decoder files from them.
- We have also tried and tested different word embeddings like Word2Vec and Google News Pretrained Embeddings to get different results.
- We have used padding and bucketing to simplify the task for seq2seq model. Padding and Bucketing is used to handle variable length sequences to feed it to LSTM cells. We have used EOS, PAD, UNK, GO as four special vocabulary tokens. Also, we used UNK to remove the words from the vocabulary with a lesser frequency of appearances, EOS was to signify the end of sentence, PAD was used to make each question and answer of equal length. GO was used as an extra symbol for the decoder input before padding with PAD.

ITCS 4111/5111 Intro to NLP

Final Project Report

We used following dependencies in our project:

- **TensorFlow:** TensorFlow is the high-level neural networks APIs. We have used the **seq2seq** extensively for majority of the training and testing, seq2seq has been the front face for the most of the neural machine translation operations today. It provides us the crux of the RNN encoder and decoder functioning.
- **Numpy:** The numpy library was used to perform mathematical tasks on data wherever required, matrix manipulation, etc.

Seq2seq training and testing:

We created wrapper for seq2seq, constructor takes following parameters as input:

- `enc_vocab_size`
- `dec_vocab_size`
- `num_layers`
- `max_gradient_norm`
- `batch_size`
- `learning_rate`
- `learning_rate_decay_factor`
- `epochs`
- `forward_only`

While in our project we implemented Multi-layered RNN cells for encoders and decoders. We created a list of placeholders of datatype *tf.float32* (indices), of known length. Next, we created the LSTM cell, the most essential part. LSTM cells, are stacked together to form a stacked LSTM, using *MultiRNNCell*. Now we use a high-level function - *embedding_seq2seq_attention* provided by tensorflow's seq2seq module, to create an attention based seq2seq model, which does word embedding internally. We used another high-level function, to get the expression for loss. Then, we build a *train* operation that minimizes the loss. Training is fairly simple. We created a method *train*, that runs the train op, for a given number of epochs. In our encoder we have used tensorflow's *embedding_attention_seq2seq* function to encode our vocabulary. It handles word embeddings with attention. In decoder we have implemented the greedy decoder as well as beam search decoder.

• What Did/Didn't work: -

- In our project, we initially used Greedy decoder. It uses tensorflow embeddings with attention mechanism. We observed that sometimes it gave irrelevant responses. It gives fast responses compared to beam search decoder.
- Also, we used Multilayered RNN cell encoder and Beam search decoder. Beam search decoder has beam size of 5, thus it selects best response out of the five possible answers. Beam size decoder gives slower responses as the beam size increases. We have also used anti-LM parameter which penalizes all those words who have high probability for any input. We can disable this functionality at this parameter by setting it zero. This approach worked well in our project.
- While, the other approach that we tried was implementing Word2Vec embedding for generating our vocabulary. The Word2Vec model has window size of 7 and minimum count

ITCS 4111/5111 Intro to NLP

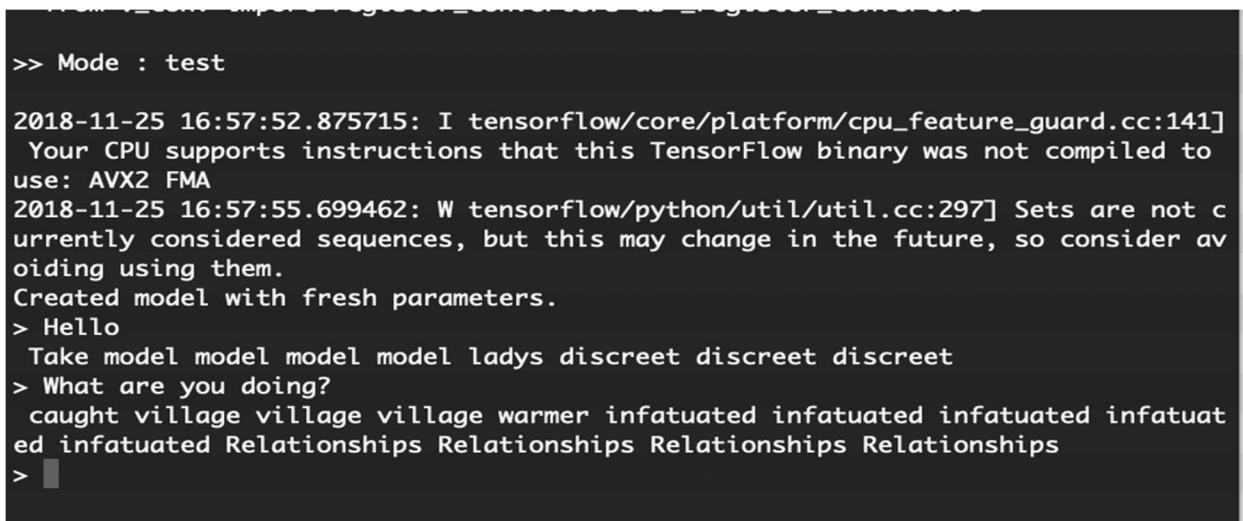
Final Project Report

of 1 for generating the word embedding. Then we trained these embeddings on our Seq2Seq model. It gave relatively poor responses as compared to Beam Search. Moreover, we also tried pretrained embeddings for generating vocabulary. It gave irrelevant and out of context responses. We found that this model had worst performance between all the models.

- **Results: -**

One of the interesting aspects of this project was getting a chance to look at how the responses changed as the model trained. We trained our model at different checkpoints. For different checkpoints we got different responses. At different points in the training loop, we tested the network on an input string, and outputted all of the non-pad and non-EOS tokens in the output.

At first, we could see that the responses were mainly blank, as the network repeatedly outputted padding and EOS tokens. This is normal since padding tokens are by far the most frequent token in the whole dataset. So here we have used the Word2Vec on our movie corpus data for word embeddings. Here, you can see that in following screenshot, the network is giving repetitive responses.



```
>> Mode : test

2018-11-25 16:57:52.875715: I tensorflow/core/platform/cpu_feature_guard.cc:141]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
2018-11-25 16:57:55.699462: W tensorflow/python/util/util.cc:297] Sets are not c
urrently considered sequences, but this may change in the future, so consider av
oiding using them.
Created model with fresh parameters.
> Hello
Take model model model model ladys discreet discreet discreet
> What are you doing?
caught village village village warmer infatuated infatuated infatuated infatuat
ed infatuated Relationships Relationships Relationships Relationships
>
```

Fig. Repetitive response

Then, you can see that the model starts to output 'okay', 'well' or other such high frequency for every single input string it is given. Here we have used the Word2Vec pretrained model of Google News data for word embeddings. This makes sense intuitively since these words gets used so often in the dataset and that makes it an acceptable response to anything. Slowly, you start to see more complete thoughts and grammatical structure come up in the responses. Could be due to a bit of overfitting as well.

ITCS 4111/5111 Intro to NLP

Final Project Report

```
>> Mode : test

2018-11-25 17:05:32.007142: I tensorflow/core/platform/cpu_feature_guard.cc:141]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
2018-11-25 17:05:35.053213: W tensorflow/python/util/util.cc:297] Sets are not c
urrently considered sequences, but this may change in the future, so consider av
oiding using them.
Reading model parameters from working_dir/seq2seq.ckpt-48000
> Hi
wouldn such I He _UNK _UNK put , the
> Where ya goin?
' . it be driven says the
> What's this?
trying _UNK
>
```

Fig. Google News data for word embeddings response

Next, we have implemented the greedy decoder, which performs well than previous models. In the following screenshot you can see the responses from greedy decoder,

```
C:\Windows\System32\cmd.exe - python main.py
2018-11-25 16:21:20.428698: W tensorflow/python/util/util.cc:159] Sets are not currently considered sequences, but this
may change in the future, so consider avoiding using them.
Reading model parameters from working_dir/seq2seq.ckpt-97200
> Good Morning
Good morning .
> Looks like things worked tonight, huh?
Yeah , but . . .
> You wanted to go out with 'me, did you?
Excuse me ?
> Sorry

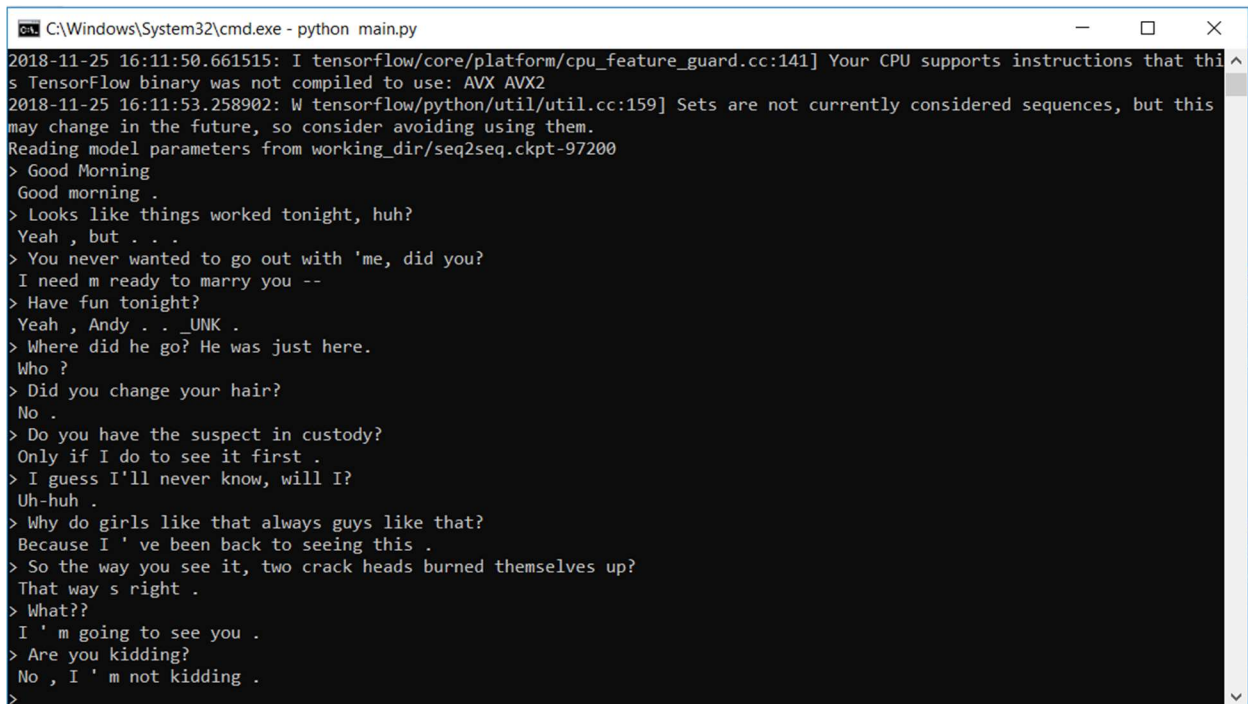
> Have fun tonight?
Yeah , _UNK .
> Where did he go? He was just here
He grow up !
> Did you change your hair?
No .
> Do you have the suspect in custody?
Only if I do to see it first .
> Why do girls like that always guys like that?
Because I ' ve been back to seeing this .
> Are you kidding?
No , I ' m not kidding .
> I'm tired
I ' ll do it .
> How do you get your hair to look like that?
He ' ll do it .
> I looked for you back at the party, but you always seemed to be "occupied".
I was ?
>
```

Fig. Greedy Decoder

ITCS 4111/5111 Intro to NLP

Final Project Report

Also, we implemented the beam search decoder to get better response,

A screenshot of a Windows command prompt window titled "C:\Windows\System32\cmd.exe - python main.py". The window shows a chatbot interface with a series of user prompts and bot responses. The bot's responses are generated using a beam search decoder. The conversation includes greetings, questions about going out, and a discussion about a suspect in custody. The bot's responses are often short and sometimes contain placeholders like "_UNK".

```
C:\Windows\System32\cmd.exe - python main.py
2018-11-25 16:11:50.661515: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2018-11-25 16:11:53.258902: W tensorflow/python/util/util.cc:159] Sets are not currently considered sequences, but this may change in the future, so consider avoiding using them.
Reading model parameters from working_dir/seq2seq.ckpt-97200
> Good Morning
Good morning .
> Looks like things worked tonight, huh?
Yeah , but . . .
> You never wanted to go out with 'me, did you?
I need m ready to marry you --
> Have fun tonight?
Yeah , Andy . . _UNK .
> Where did he go? He was just here.
Who ?
> Did you change your hair?
No .
> Do you have the suspect in custody?
Only if I do to see it first .
> I guess I'll never know, will I?
Uh-huh .
> Why do girls like that always guys like that?
Because I ' ve been back to seeing this .
> So the way you see it, two crack heads burned themselves up?
That way s right .
> What??
I ' m going to see you .
> Are you kidding?
No , I ' m not kidding .
>
```

Fig. Beam search Decoder

It's probably difficult to judge whether or not the bot actually gives appropriate answers, but we would say it does alright. The grammar is well structured, we can pick a good set of responses amongst all of the responses generated. We are measuring the performance of chatbot on basis of perplexity. The average perplexity of greedy decoder model for 48000 iteration is 5.70 on the other hand average perplexity of beam search decoder model for 48000 iteration is 4.32.

ITCS 4111/5111 Intro to NLP

Final Project Report

- **Contributions: -**

	Team Member Name (800 id)	Responsible For
1	Abhay Marathe (801053365)	-Data gathering and preprocessing (Gathered metadata and removed stop words) -Data Segregation in train, test files. -Greedy Decoder Implementation -Report
2	Kuldeepsinh Chudasama (801045027)	-Formatted the raw movie_lines corpus into lineId and text of utterance format -Converting variable length sentences to a fixed length sentences -Word embedding using Word2Vec model -Performance Optimization -Report
3	Prashant Desai (801044356)	-Formatted the movie_conversations corpus into list of utterances -Encoder Implementation (in progress) -Seq2seq training and testing -Beam search decoder implementation -Model Evaluation -Report

- **Challenges and In between modifications: -**

- While going through the preprocessing step of our dataset, we initially removed stop words and punctuation as mentioned in our project proposal. But, later we found that due to removing all stop words along with pronouns and punctuations, our model wouldn't be able to recognize questions, exclamations in the conversation and answer the queries properly. So, stepping down to our raw dataset, we performed a split using the '+++\$++\$' as the separator and then sliced the achieved list and we created a dictionary by appending the sliced conversations.

```
L1045 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ They do not!  
L1044 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ They do to!  
L985 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ I hope so.  
L984 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ She okay?  
L925 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Let's go.  
L924 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ Wow  
L872 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Okay -- you're
```


ITCS 4111/5111 Intro to NLP

Final Project Report

```
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L276', 'L277']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L280', 'L281']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L363', 'L364']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L365', 'L366']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L367', 'L368']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L401', 'L402', 'L403']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L404', 'L405', 'L406',
'L407']
```

- Also, we tried to add our own word embeddings using Word2Vec model for Cornell Movie Corpus as well as for pretrained GoogleNews embeddings. But while testing the model it was giving strange responses as shown in above screenshot. (i.e. Fig. Repetitive response and Fig. Google News data for word embeddings response)

4. Summary

- Chatbot has wide range of applications in the field of business, education, banking, manufacturing etc. Moreover, it is used for customer support, product suggestion, e-commerce, finance, healthcare, insurance, media and travel industry. In our project, we have presented an approach for building chatbot using Seq2Seq model. We trained chatbot using Cornell Movie Dialogue Corpus and then chatbot is able to answer the questions using the learned data.
- We have used TensorFlow to develop our chatbot. Along with that we implemented Seq2Seq model using as LSTM (Long Short-Term Memory) cells, a Multi-layered cell RNN (Recurrent Neural Networks) and decoders with attention. We have used padding and bucketing to handle variable sequences to feed it to the seq2seq model.
- In our project, we initially used Greedy decoder. It uses tensorflow embeddings with attention mechanism. We observed that sometimes it gives irrelevant responses.
- Also, we used Multilayered RNN cell encoder and Beam search decoder. Beam size decoder gives slower responses as the beam size increases. We have also used anti-LM parameter which penalizes all those words who have high probability for any input. This approach worked well in our project.
- While, the other approach that we tried was implementing Word2Vec embedding for generating our vocabulary. It gave relatively poor responses as compared to Beam Search. Moreover, we also tried pretrained embeddings for generating vocabulary. We found that this model had worst performance between all the models.

5. Conclusion

After interacting with the chatbot, we came to the conclusion that there is quite a lot of room for further enhancing for the chatbot. The bot isn't able to connect thoughts together and some responses are simply random and coherent making no sense given the context of the conversation. One of the possible reasons for that is lack of enough training data. Incorporating the other larger datasets may improve the performance of the chatbot. Furthermore, we think that the use of a fixed-length context vector is problematic for translating long sentences.

For the future scope of this project, we believe that the persona based chatbot can be built based on the characters who are speaking the dialogue. A well refined and tuned chatbot would be implemented as a virtual assistant for customer support, product suggestion, e-commerce, finance, healthcare, insurance etc. sector.

ITCS 4111/5111 Intro to NLP

Final Project Report

6. References

- [1] [Conversations with a Question-and-Answer Chatbot in the Wild](#)
- [2] [Deep Learning Based Chatbot Models](#)
- [3] [Sequence to Sequence Learning with Neural Networks](#)
- [4] [Neural Machine Translation by Jointly Learning to Align and Translate](#)
- [5] [Generative-model-encoder-decoder-chatbots](#)
- [6] [Practical-seq2seq](#)
- [7] [Seq2Seq model in TensorFlow](#)
- [8] [How to build your first chatbot](#)