Personality based Chatbot

Ayush Garg 111870086

Shubham Patel 112007216

Hitesh Sharma 112070337

Abstract

We are developing personality based chatbot. We are computing a chatbot model using person's dialogues, which can answer a question in a style/personality specific to the person.

1 Introduction

The objective of our project is to develop a chatbot model which can adapt a human personality for responding to any question. Chatbots like slack bot, fb messenger bot etc. are already there, but they respond in a very generic way, without remembering any context. We want to give chatbot a personality, so that it can act like a human while responding.

If a chatbot can adopt a personality, then user can relate better and open up more and give more information. It could change the way in which we interact with these artificial agents. This can help companies get more data about users. This type of chatbot can respond better specific to a person on which it is trained.

The problem is difficult to solve because :

1. The language we used for chats is not standard.
2. A large amount of user specific chat data is not easily available.
3. Even with data in hand, determining context is all together a different challenge

There are majorly three approaches:

1. **Speaker based model:**[1] In this model, each speaker is associated with an embedding vs learned during training. Whenever a response by speaker s is encountered during training, the corresponding embedding $\mathbf{v}_s$ is inserted into the first hidden layer of the decoder LSTM at each time step (i.e., conditioning each word in the utterance). The hidden states h_t of the decoder LSTM is thus calculated as follows (where y_t is the embedding of the response word at time t, and g stands for the LSTM cell operations):

   $h_t = g(h_{(t1)}, y_t, c_{(t1)}, \mathbf{v}_s)$

2. **Personality based model:**[2] Instead of leveraging speaker embeddings, the OCEAN scores are estimated for each speaker and a personality embedding $\mathbf{v}_o$ is inserted into the first layer of the LSTM decoder. OCEAN scores are 5-dimensional vectors o, where each dimension ranges from 1 to 7. Weights are learned during training. Whenever a response with personality traits o is encountered, we insert $\mathbf{v}_o$ into the first hidden layer of the decoder LSTM. Thus, the hidden states h_t are now calculated as:

   $h_t = g(h_{(t1)}, y_t, c_{(t-1)}, \mathbf{v}_o)$

3. **Speaker addressee model:**[3] A natural extension of the Speaker Model is a model that is sensitive to speaker-addressee interaction patterns within the conversation. The proposed Speaker-Addressee Model operates as follows: We wish to predict how speaker i would respond to a message produced by speaker j. Similarly to the Speaker model, we associate each speaker with a K dimensional speaker-level representation, namely v_i for user i and v_j for user j. We obtain an interactive representation $v_i{,}j \;\; R^(K1)$ by linearly combining user vectors v_i and v_j in an attempt to model the interactive style of user i towards user j,

   $V_{i,j} = tanh(W_1 v_i + W_2 v_2)$

One major issue in these approaches is that they require large amount of conversation data

or personal chats to train which is not easily available.
Evaluation is a problem and a general challenge in these cases. Since even if we can develop a model based on a specific person, we can never be sure if the model is indeed speaking like that person just by observing the chats.
Consider an example: Suppose we ask the same question to 2 different persons:

Question: *Do you love me?*
Ross(FRIENDS): *No I don't!*
Raj(Big Bang): *I dont love you.*
How do you decide who is who?

To address above problem of availability of large dataset, instead of completely relying on chats for personality, we tried to use our own features/knowledge along with the chat data.
For evaluation, One approach we tried is a turing test however actual evaluation metric is difficult to find in this case and that is something we need to explore further.
Another approach we tried is, we kept some test data containing question and answer pair, in style/personality of the user on which model is trained and then used this to evaluate our model, however, rarely our model could predict the exact answer, but captured personality of the user, in the answer provided by it. In this scenario, we need an evaluation system to determine how accurate our models output is corresponding to users personality.
We also calculated the BLEU scores for our model and various loss functions but these are not very reliable too in case of chatbots.

For the baseline model, we have developed a chatbot based on one of the characters of TV series FRIENDS. We found transcripts of all the 10 seasons of FRIENDS where there were dialogues of each of the characters. We then converted this data to a csv format which had speaker name and their dialogues as the fields. We trained our model to speak like a specific character Ross.

For Dataset, we used Friends TV series transcripts dataset. We modeled the personality of one of the character Ross in Friends TV series. We also applied it to our WhatsApp, Facebook, Hangouts and LinkedIn chat data. From Facebook, we downloaded complete profile data of the user, we used that data to model personality of the user. For personality of user, we explored other datasets like interview data or speakers speech data too.

The main outcomes of this project are:

1. We developed a personality based chatbot that can answer questions in the speaking style of a particular user.

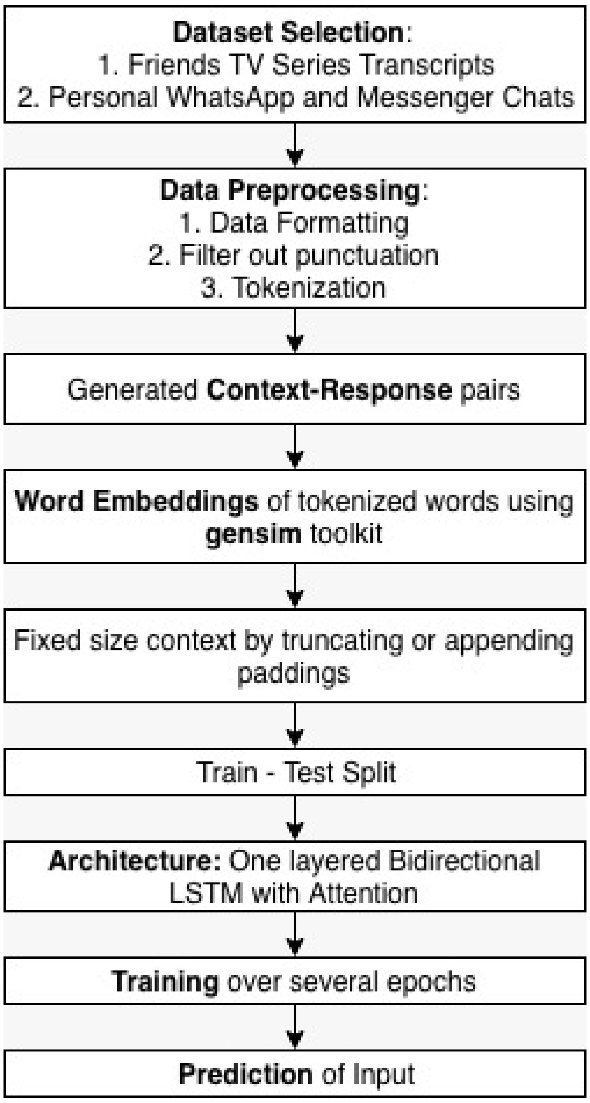
2. We tried to provide context of the conversation to the chatbot using a bidirectional LSTM.

3. Analysis shows that the bot works well when trained on large dataset with standard english language with pretrained glove embedding. However it struggles with Social media language (or a mixed non standard *Hinglish* language.

4. Based on our work, we conclude that investigating better evaluation metrics, better personality traits (other than OCEAN scores) and capturing more conversation context might be promising avenues for future work.

2 Task

In hand task here is to create a chatbot that is able to capture the speakers personality on which it was trained. To achieve this, training data must be provided consisting of context response pairs, where in response is provided by a speaker, whose personality we aim to capture in the bot. Due to this constraint over dataset, getting a reasonable size of dataset was the first hurdle that we need to pass through. We have explored multiple options as discussed in detail below. Apart from dataset, there were other challenges too, like how to pass data, what logic should be adopted for pair generations from dataset, what should be the fixed size of context and response that will enable our model to capture maximum information, what model architecture to use to make sure learning take place in an achievable timeframe.
Standard solutions for this task (the 3 models discussed above) usually have a personality vector that is trained, passed to the decoder LSTM corresponding to a particular speaker. For pair



Dataset Selection:
1. Friends TV Series Transcripts
2. Personal WhatsApp and Messenger Chats
Data Preprocessing:
1. Data Formatting
2. Filter out punctuation
3. Tokenization
Generated Context-Response pairs
Word Embeddings of tokenized words using gensim toolkit
Fixed size context by truncating or appending paddings
Train - Test Split
Architecture: One layered Bidirectional LSTM with Attention
Training over several epochs
Prediction of Input

Figure 1: Block Diagram of Baseline Model

generation, these models use sequential pairs.

2.1 Baseline Model(s)

For the baseline model, we have developed a chatbot based on one of the characters of TV series FRIENDS. We found transcripts of all the 10 seasons of FRIENDS where there were dialogues of each of the characters. We then converted this data to a csv format which had speaker name and their dialogues as the fields. We trained our model to speak like a specific character Ross.
For training data, we took the dialogue spoken by Ross as the Y label and concatenation of all dialogues before it as X label. Hence we are taking context-response pairs as all dialogues spoken before Ross replied Rosss reply.

We used Gensims Word2Vec for training word embeddings and a 4 layer LSTM for training the model. We trained our model on 9027 context response pairs and trained it for 100 epochs.

2.2 The Issues

As told above, no significant results were obtained from the baseline model. Model was sticking to probabilistic response which was mostly independent of what question was asked. On analysis, we concluded the root cause for this behavior is using too many LSTM layers, due to these 4 layers, count of parameters rose to 68K. Which ideally should have resulted in great output, provided ample amount of data and training. However, data in our case was nearly 9.5K, which was nowhere close to data on which state of the art chatbots are trained. Also, nor we have the computing power to train our model for days. Later, on further discussion over our project with TAs and Prof. we realized that there are certain combinations that we should try exploring, ahead of our baseline model. These combination includes, choosing less number of LSTM layers, employing attention to our system, using pretrained model. These discussion, helped us quantify our further approach, as mentioned in detail in approach section below.

3 Approach

We have tried a different set of modules in order to create a chatbot better than our baseline model, also we have tried tuning these different models under different hyper-parameters.
Below mentioned setups were done and tested in python with Keras framework. Also for word embedding generation we have used Gensim, which provided us an embedding size of 100 (we also tried with embedding size 300) -

1. 4 layer unidirectional LSTM, with sentence consisting of 30 and 50 words, hidden state size = 64,128, epochs ranging from 100-1000

2. 4 layer bidirectional LSTM, with sentence consisting of 30 and 50 words, hidden state size = 128, epochs ranging from 100-1000

3. Single-layer bidirectional LSTM, with sentence consisting of 30 and 50 words, hidden

```
[('phoebe', 0.7750962376594543),
(',', 0.9087491631507874),
('everything', 0.7403488159179688),
(',', 0.773642361164093),
('either', 0.7158159017562866),
('either', 0.6732187271118164),
('either', 0.652601957321167),
('either', 0.6304011940956116),
('either', 0.5696536302566528),
('either', 0.5397694110870361),
('either', 0.5200806856155396),
('either', 0.5205897092819214),
('either', 0.510658323764801),
('either', 0.4690850079059601),
('father-in-law', 0.43440473079681396)]
```

Figure 2: Probabilistic Output

state = 128, 512, epochs ranging from 100-1000

Below mentioned setups were done and tested in python with Pytorch framework. Also for word embedding generation we have a tried a one hot vector approach here:
A simple Seq2Seq model, with one encoder and decoder (with attention), modified to train as a chatbot by providing XY pairs from our dataset. Hidden state size 256, epochs -7500. Also data was normalized by converting it to lower case and removing all special symbols like [. ? !]
In outputs of all the first approach scenario mentioned above, we are still seeing that chatbot is responding, based on a probabilistic preference. Like for 4th model in Keras as mentioned above, after training, these words came out to be the most probabilistic responses to mostly every question we asked our chatbot. However, with Pytorch model we saw output is more organized based on the context, like response to "How are you", comes like "I am fine, u uu uu ", this is significant improvement compared to baseline.

We also tried different loss functions (cosine proximity, cross entropy). Next intuition was to involve a larger data set and train for more number of epochs. Thus we trained our seq2seq pytorch model, with Cornell movie dataset, at 4000 epoch, training time spanned around 5 hours, although we saw significant improvident but output is not with correct context for non trivial questions

We iteratively approached our work with 3 different ideas:

3.1 Idea 1:

Our first and the main idea was to train the above mentioned models on a conversation dataset with the context-response pairs as follows:
Context (X-label) : Concatenation of all the dialogues before him/her.
Response (Y-label): The dialogue of that particular person.
In case of personal chats, Suppose a person wants to train a model on his own personality, Then he would take his chats and context response pairs would be generated as:
Context (X-label): The response of the opposite person
Response (Y-label): My response.

This was a very basic idea with the idea that if we somehow train a model to respond to a question in a specific way, then ultimately it would learn to. However, due to time and computational constraints, we could not get any substantial results out of it.

3.2 Idea 2:

Then from paper [1], we got the idea of speaker and personality embeddings. The basic architecture for this idea was the same. That is, a SEQ2SEQ Bidirectional LSTM model developed, Single or multi-layer. This time we also made use of attention to capture more context and generate more relevant responses to our input question. Again, we tried this for both Keras and pytorch and trained for around 10000 epochs.
The basic intuition behind this idea was to have a speaker or personality embedding corresponding to each speaker which could then be fed into the decoder LSTM along with attention weights. These speaker embeddings would be randomly initialized in the beginning but would be trained with time. These embeddings could either be learned from the training data (dialogues of that speaker) or would be a combination of OCEAN traits (personality vectors) usually 5D in dimension.

3.3 Idea 3:

Finally, Mainly due to time and computational constraints, we zeroed in upon the idea to use a pre-trained model, trained on large Cornell movie dataset ,saving the weights for the same and then loading and retraining it for our own dataset to incorporate personality. That is the model would

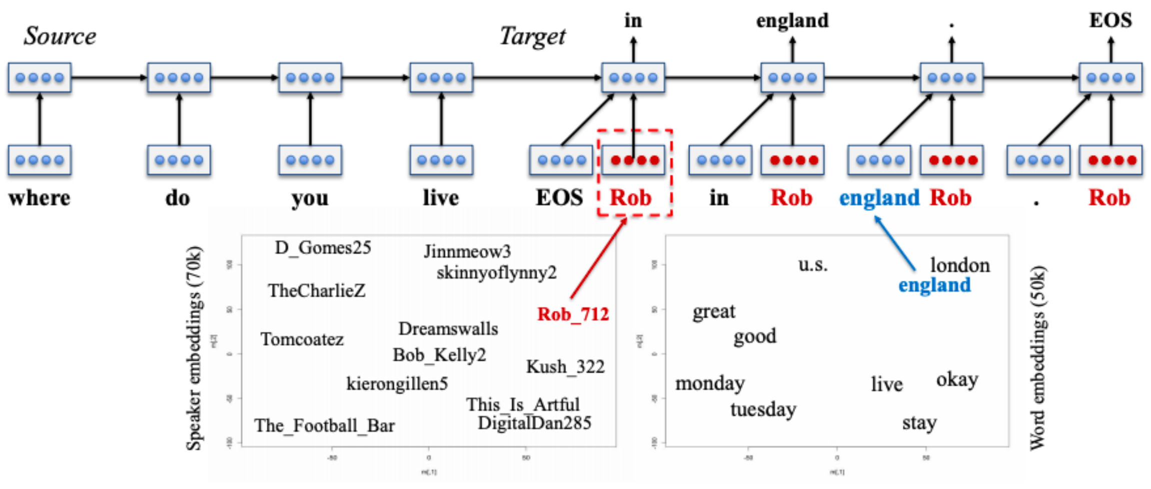


Figure 3: Implementation Details

first learn the language trend in general, that is how to speak. Because the personality data is usually small, we cannot expect it to learn to speak using that data. Hence we:

1. First pretrain the model using a large dataset like Cornell Movie Dataset which had around 150000 pairs. Save the model weights.
2. Load the weights. Retrain the model using FRIENDS dataset with context response pairs as mentioned in idea 1. Also, incorporate the speaker embeddings as mentioned in idea2.
3. We train the model for around 20000 epochs and our model was finally ready.

3.4 Implementation Details

As explained above, we used a bidirectional Attention Single layer LSTM model and word embeddings of size 100 trained using gensim. (We also tried Glove embeddings, However for personal chats and inconsistent data, gensim works better).

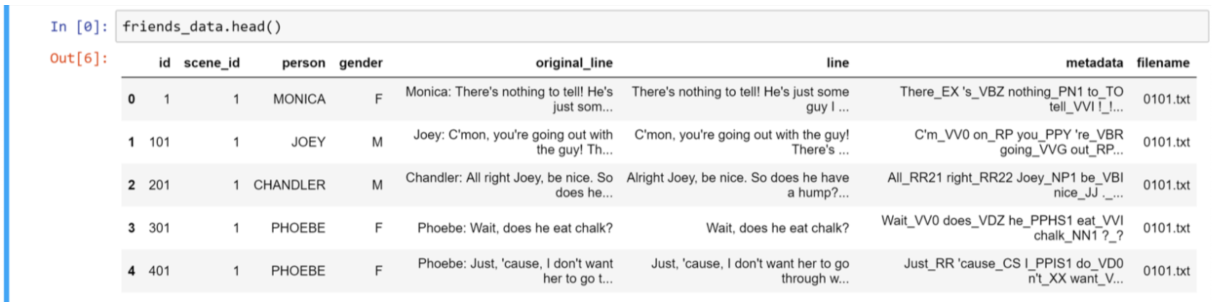
We finally inputted the speaker embeddings and personality vectors of character ROSS (determined empirically) in decoder LSTM and trained our bot for 20000 epochs on FRIENDS dialogue data: (Implementation is shown in above **Figure 3**)

4 Evaluation

Evaluating our model is a difficult task, as for evaluation, a matrix has to correctly judge if the speakers personality traits are captured in provided output or not. And for doing do, matrix in itself should have exact knowledge of speakers personality traits. Formulating such matrix is a challenge in itself. Here we will provide one extrinsic and one intrinsic model to evaluate our system. Extrinsic model evaluation can be done by turing test, which can be done by Human, who is familiar with the speaker on which model is trained. This can be done by, randomly mixing, few pairs X,Y and X,Y, where Y corresponds to actual response and Y corresponds to response from the bot. Next, person doing turing test, will be asked to classify, prior unknown to him, X Y pairs as those which are actually said by speaker, and those which are output from the bot. If at the end of the classification task, Z % of Y pairs lies in class of actually said by speaker, then Z can be seen proportional to bots accuracy. Intrinsic method of evaluation require using BLEU algorithm. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" this is the central idea behind BLEU. However, BLEU only gives us an estimate as to how much the bots output is alike to something said by an actual human being, it will not evaluate the output on the basis of speaker personality traits. Apart from that, one another problem with BLEU scores is that they tend to favor short translations, which can produce very high precision scores, even using modified precision.

4.1 Dataset Details

(Used Dataset Shown in **Figure 4** on next page.) For training our model like a specific personality, we explored several different types of Datasets:



```
In [0]: friends_data.head()
Out[6]:
     id  scene_id    person  gender                        original_line                                 line                              metadata  filename
0     1         1    MONICA       F  Monica: There's nothing to tell! He's  There's nothing to tell! He's just some  There_EX 's_VBZ nothing_PN1 to_TO  0101.txt
                                                              just som...                                guy I ...                      tell_VVI !_!...
1   101         1      JOEY       M     Joey: C'mon, you're going out with   C'mon, you're going out with the guy!   C'm_VV0 on_RP you_PPY 're_VBR  0101.txt
                                                              the guy! Th...                              There's ...                   going_VVG out_RP...
2   201         1  CHANDLER       M  Chandler: All right Joey, be nice. So  Alright Joey, be nice. So does he have  All_RR21 right_RR22 Joey_NP1 be_VBI  0101.txt
                                                                 does he...                              a hump?...                          nice_JJ ._....
3   301         1    PHOEBE       F        Phoebe: Wait, does he eat chalk?               Wait, does he eat chalk?  Wait_VV0 does_VDZ he_PPHS1 eat_VVI  0101.txt
                                                                                                                                        chalk_NN1 ?_?
4   401         1    PHOEBE       F       Phoebe: Just, 'cause, I don't want     Just, 'cause, I don't want her to go  Just_RR 'cause_CS I_PPIS1 do_VD0  0101.txt
                                                               her to go t...                             through w...                   n't_XX want_V...
```

Figure 4: Dataset Details

1. Friends TV Series conversation (Problem : Group conversation)
2. Interviews data (Problem : Small questions and Large answers)
3. Individual users speech data
4. WhatsApp and Facebook messenger conversations

Here are the details of the datasets (from the above) that we finally used:

1. Dialogue corpus from popular TV series FRIENDS. It consists of 60k dialogues, from which we generated context response pairs via different approach as mentioned in detail further below. Intention here was to train the model, on one specific character of the series, Ross Gelle r. Reason we choose Ross, was because he has more number dialogues than other 5 primary characters.
2. Whatsapp chat data. For this we extracted chat logs, of our own chats, in .txt format from whatsapp. Intuition here was to get chat logs of many of our contacts, and then make context response pairs out of it, and then train the model as us being the primary speaker whose personality we intend to capture. Another motivation here was to be able to informal talking style from these daily chat logs, which will be specific to individual whose chat data is used. We generated around 13727 context response pairs.
3. Under the same motivation as mentioned above, we downloaded our complete profile data from Facebook, which included ass messenger chats, status updates over the last 9 years.

4.2 Evaluation Measures

Evaluation measures, as we talked above, is in general, a very difficult problem to solve for our problem, primarily because there is no foolproof quantitative metric to do so. However, we used a combination of these metrics to evaluate our model:

1. **Accuracy (Perplexity)**: We tried with cosine similarity, categorical cross entropy and binary cross entropy. Categorical cross entropy gave the best results.

2. **Turing test**: As explained above, not a qualitative measure but effective nonetheless.

3. **BLEU scores** ”the closer a machine translation is to a professional human translation, the better it is” this is the central idea behind BLEU. However, Not very effective in our case if used separately.

4.3 Baselines

We followed 2 Ways to generate X and Y pairs:

1. Concatenate all sentences like we are doing with FRIENDS dataset (explained above).
2. Generate pairs sequentially. Example : A B C D are sequence of sentences in conversation. (A,B), (B,C), (C,D) will be (X,Y) paris.

We followed these steps to make our model:

1. We used 2 datasets. One of friends transcripts and other of our personal chats from whatsapp.
2. Data formatting and loading. For this, we performed stemmizing, lemmatizationa and stop words removal.
3. Creating X (Features) and Y(output) as training data for neural network. We followed both the above 2 ways and recorded results for both.
4. Tokenize words in X and Y.
5. Convert tokenized words into vectors (embeddings). We used gensim and glove word embeddings. Glove gave better results.

6. Now we have sentence vector in X and Y
7. For baseline 1 model, we have fixed sized input sentence, so to make sentence vector fixed size, we have trimmed longer sentences or added paddings to smaller sentences. Break X,Y pairs to train and test dataset using scikit learn train test split.
8. Defining architecture of our deep learning neural model. We tried with Single layer, Multi layer, bidirectional and attention based LSTM.
9. Train our model on X,Y train pairs. In keras we trained it for 500 epochs on friends Data and 1000 epochs on Personal chats.
10. Test our model on X,Y test pairs. We asked random questions like "How are you?". The results were decent but not very promising perhaps because of lack of training due to computational power constraints.

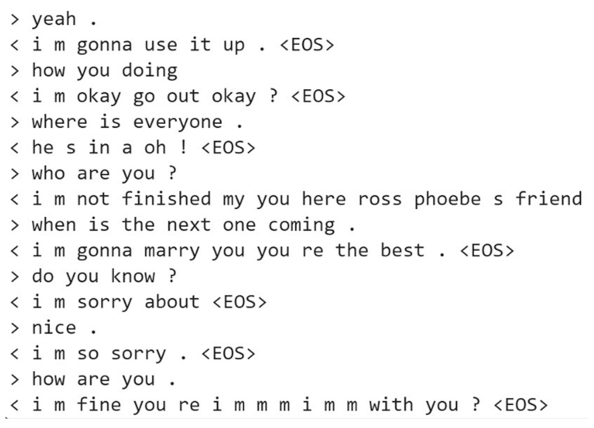
In the other baseline model, which was made in Pytorch, we used and existing seq2seq model with an encoder and an attention enabled decoder. For word embedding generation, we used one hot vector on all words in the training vocabulary. We trained this model over Friends data set initially, by processing data in both ways as before. In terms of tuning, we changes the hidden state size of encoder layers between 128,256,512 and tried on epochs ranging from 500 - 20000. However, our model post training was responding to usual greetings like Hi and Hello correctly, but for other inputs it still sticks to limited number of responses, consisting of nearly similar word. It is still providing probabilistic output, instead of one specific to context of input.

4.4 Results

Main results can be seen in the attached Table 1 in which we provided all the hyperparameter details and accuracy for all our models. Also Figure 5 shows sample output from our best model so far.

4.5 Analysis

1. We tried our model on two different datasets. We had used Cornell Movie Dataset for pretraining. We finally trained on Friends TV Dialogue datasets and Our personal Whatsapp Chats. Naturally, results were better on Friends TV dialogues than personal chats because its dataset is bigger and in more standard language. Results on personal chats were more interesting though.



```
> yeah .
< i m gonna use it up . <EOS>
> how you doing
< i m okay go out okay ? <EOS>
> where is everyone .
< he s in a oh ! <EOS>
> who are you ?
< i m not finished my you here ross phoebe s friend
> when is the next one coming .
< i m gonna marry you you re the best . <EOS>
> do you know ?
< i m sorry about <EOS>
> nice .
< i m so sorry . <EOS>
> how are you .
< i m fine you re i m m m i m m with you ? <EOS>
```

Figure 5: Output from Final Model

2. For making context response pairs, we tried with sequential pairs as well as dialogue concatenation pairs (explained above). Sequential pairs worked better but lacked any specific personality.
3. For word embeddings, we used Gensim as well as Glove pretrained word embeddings (50d, 100d). For obvious reasons, glove gave better results on friends data. However, we could not use glove on personal chats because most words were in language other than English. Hence we went ahead with gensim.
4. In our first baseline (default) model, we had trained our model on single LSTM with 100 hidden size and 100 sized vector word embeddings. We used cosine similarity as our loss function and trained on 25 epochs. Results were pretty decent with highest probabilistic word being 'absolutely'.
5. We then tried with 2,3 and 4 layered LSTMs. Results were decent but not very good. We then improved further by using Bidirectional that could use the whole context while answering. Finally, we implemented Attention layer on top to improve the accuracy further.
6. We did training on 25, 100, 500 and 1000 epochs on Google colaboratory GPU. After 1000 epochs, it starts overfitting and same word starts repeating.
7. For our data, a single layer attention based BLSTM trained on Gensim, trained on 1000 epochs (loss function Categorical cross entropy) worked best and gave around 25% accuracy (good for a personality based bot).

4.6 Code

Google drive link of code for all the implemented models along with dataset file and readme :

Model Name	Model Architecture	Dataset	Word Embeddings	Sentence Length	Accuracy	Loss Function	Loss	Epoch	Hidden Size
Baseline 1	4BLSTM	Friends	Gensim	30	7.2	Cosine Proximity	-0.63	100	100
	4BLSTM	Friends	Gensim	30	12	Cross Entropy	90	100	100
	2BLSTM	Friends	Gensim	30	8.3	Cosine Proximity	-0.34	100	100
	2BLSTM	Friends	Gensim	30	13.7	Cross Entropy	80	100	100
	1BLSTM	Friends	Gensim	50	24.5	Cross Entropy	26	500	300
	1BLSTM	Personal Chat	Gensim	50	25.4	Cross Entropy	292	500	300
	1BLSTM	Friends	Glove	50	35.4	Cross Entropy	42	100	300
	1BLSTM + Attn.	Friends	Gensim	50	37	Cross Entropy	60	100	300
	1BLSTM + Attn.	Personal Chat	Gensim	50	29.16	Cross Entropy	180	100	300
Baseline 2	Encoder Decoder + Attn.	Friends	Gensim	NA	NA	Cosine Proximity	1.42	20000	512
	Encoder Decoder + Attn.	Cornell	Gensim	NA	NA	Cosine Proximity	3.2	4000	256

Table 1: Result details

https://drive.google.com/open?id=1XallPdkIft1Z96B9fiPowxpWP9eh3N0j

5 Conclusions

1. It is a difficult problem, in general, because of lack of large amount of user specific data and evaluation metric.
2. The extension to our project is similar to the speaker-addressee model. For example, in our dataset of TV series Friends, the character Ross often talks differently to his sister Monica than to Rachel, with whom he is engaged in an off and on relationship throughout the series.
3. Having better and more exhaustive personality vector other than OCEAN is another challenge.
4. The main challenge of this extension is to collect conversation dataset of speaker with several other speakers.

References

[1] A Persona-Based Neural Conversation Model, Jiwei Li1, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, Bill Dolan http://www.aclweb.org/anthology/P16-1094

[2] Automatic Evaluation of Neural Personality-based Chatbots, Yujie Xing and Raquel Fernandez http://aclweb.org/anthology/W18-6524

[3] Neural Response Generation for Customer Service based on Personality Traits, Jonathan Herzig, Michal Shmueli-Scheuer, Tommy Sandbank and David Konopnicki http://www.aclweb.org/anthology/W17-3541