Personality-Based Chat Bot

Sameer Kumar Behera (UIN: 526004296)
Srishti Agarwal (UIN: 525004684)
Shubham Bhargava(UIN: 127003902)

I. Abstract (*All*)

Conversation with machines has been a long-standing goal of human-computer interaction. We not only engage with such activities on our daily basis like using Google Voice, and Siri to set up alarms etc. but have grown reading fiction about such ideas. It makes sense for the natural language to become the primary way in which we interact with devices because that's how humans communicate with each other. Thus, the possibility of having conversations with machines would make our interaction much more smooth and human-like. In this project, we propose to build a theme or personality aware chatbot that will mimic the qualities of a person.

II. Introduction (*All*)

Previously, most of the conversational models were rule based and solved problems like booking airline tickets or providing recommendations. Hence, chatbots were commercially used for creating a dialog management system. ChatScript is one of such example which is a popular rule based conversational model and is one of the best conversational rule-based model. Although effective, but such systems are not flexible and are limited by the rules that derive them.

Recently, sequence to sequence (seq2seq) model broadened the scope of the chatbots by solving dialogues as a translation problem. Being a deep neural network, this model reduced the task of feature engineering and domain specific knowledge, hence solving those tasks for which rules are very hard to design manually. A basic sequence-to-sequence model, as introduced in Cho et al., 2014 [2], consists of two recurrent neural networks (RNNs): Encoder to process the input and a Decoder that generates the output. The encoder maps a variable-length source sequence to a fixed-length vector, and the decoder maps the vector representation back to a variable length target sequence. This model has been successfully used for many different natural language processing tasks, such as alignment, translation [3], and summarization [4].

Although the bots based on this model could converse fairly fluently but lacked a persona hence making them less convincing. We experiment with creating a personality chatbot based on popular TV series which could imitate popular characters like Sheldon from Big Bang Theory or Joey from Friends. In our model, the encoder processes a statement by human, and the decoder produces the response to that statement. We train the word embeddings as we train the model. We also experimented with using GoogleNews-vectors-negative300 pre-trained word vectors to initialize our word embeddings.

III. Related Work

Recent advances in dialogue systems can be attributed to deep learning. A lot of work has been done to develop chatbots, both task oriented and non-task oriented dialogue systems. For dialogue systems, deep learning can leverage a massive amount of data to learn meaningful feature representations and response generation strategies, while requiring a minimum amount of hand-crafted rules. This section discusses some of the recent and related work in this domain.

A. Rule-Based Models (*Sameer Kumar Behera*)

In rule-based chatbots, the bot answers the questions based on a set of rules which it is trained on. The rules may range from simple to complex depending upon the task, which is also cumbersome as it requires devising a new set of rules for every task. The bot may give more accurate answers but blunt answers. Also, writing rules for different scenarios is very time taking and it is impossible to write rules for every possible scenario. ELIZA[1] is one of the first ever chatbot programs written. It uses clever handwritten templates to generate replies that resemble the user's input utterances. Since then, countless hand-coded, rule-based chatbots have been developed.

B. Retrieval Models (*Srishti Agarwal*)

In retrieval based models, the answer is selected from the set of possible answers to the question. These bots are trained on a set of questions and their possible outcomes. For every question, the bot can find the most relevant answers from the sets of all possible answers and then outputs the answer. Since, the answers are pre-determined, they are syntactically and grammatically correct but the bot cannot generate any new responses and answers to the questions it has never heard of.

C. Generative Models(Sequence to Sequence) (*Shubham Bhargava*)

In Generative models, proper responses to conversations are generated such as in sequence-to-sequence models[6]. This type of methods usually uses an encoder-decoder framework which first encode input sentence as a vector representation, then feed this representation to decoder to generate output sequence. Generative models are better than rule-based models in a way better that they can generate the answers and not always replies with one of the answers from a set of answers. This makes them more intelligent as they take word by word from the query and generates the answers. It also makes them more prone to errors as they need to take the spelling and grammar into account. To make them better at handling these errors, these models need to be trained more precisely. Once trained, they can outperform the rule-based models as they can answer complex and unseen queries. Language translation models can be used in creating such a model.

D. Personality Chatbot with Speaker-Addressee model (*Shubham Bhargava*)

In 2016, Li et al.[5] at Stanford University published a paper that uses distributed embeddings to capture personas of chatbots. They proposed Speaker-Addressee model to predict how speaker A would respond to a message produced by speaker B achieving interesting results.

IV. Dataset (*Srishti Agarwal*)

For training our model we used scripts from the TV shows like Friends and Big Bang Theory. We used python BeautifulSoup for scraping the scripts from [6]. Some of the sample dialogues are:

```
Monica:  There's nothing to tell! He's just some guy I work with!
Joey:  C'mon, you're going out with the guy! There's gotta be something
wrong with him!
Chandler:   All right Joey, be nice.   So does he have a hump? A hump
and a hairpiece?
Phoebe:  Wait, does he eat chalk?

Phoebe:  Just, 'cause, I don't want her to go through what I went
through with Carl- oh!
Monica:  Okay, everybody relax. This is not even a date. It's just two
people going out to dinner and- not having sex.
Chandler:  Sounds like a date to me.
```

Table 1: This is an example script from famous sitcom Friends where all the characters are conversing with each other.

In order to preprocess this text we converted the script text to lowercase, removed rare alphanumeric characters. Also each scene has its own topic, context, and speakers and addressees, we decided to split the dialog scripts into conversations of contiguous statements. This way, statement from a completely new conversation won't be connected as responses to the previous scene. To separate the dialogues into these conversations, we introduced a separator ("===") each time the setting or scene changed. We arranged the dialogues into Question & Answer format with the required personality dialogues as answers and other character dialogue as question. Our final dataset looked like as shown below:

```
There's nothing to tell! He's just some guy I work with!
C'mon, you're going out with the guy! There's gotta be something
wrong with him!
=== (separator)
Then I look down, and I realize there's a phone... there.
Instead of...?
That's right.
Never had that dream.
===
```

Table 2: Sample preprocessed script

Our final dataset is stored in pkl format and loaded further to train encoder-decoder model. We choose to train our model for "Joey" (25%) from Friends and "Sheldon" (42%) from Big bang Theory as these characters have maximum number of dialogues in the dataset.

V. Implementation (*All*)

This section discusses how we formulated and tackled the problem statement, various evaluation metrics used to measure how well the model performs.

A. Our Approach

We used a sequence-to-sequence encoder decoder model as is used for machine translation by Sutskever et al.[7]. An example of the model used is shown in Fig.1. Seq2Seq is an encoder-decoder model where Encoder reads the input sequence, one time-step at a time, and learns the most effective fixed-dimensional vector representations of the input sentences(of variable lengths), whereas the Decoder extracts the output sequence from this vector representation. Both Encoder and Decoder are Recurrent Neural Networks except that decoder is conditioned on the input sequence. LSTMs have proved to be the best choice among RNNs in various Natural Language processing tasks owing to their ability to capture long term temporal dependencies in the input sequence.

Previous work in Sutskever et al.[7] English to French translation task empirically proved that LSTMs were not only highly efficient in mapping longer sequences but also learnt the meaningful phrases and sentences sensible to word order. Since then, seq2seq models have been widely used for other applications such as text summarization, code generation, conversational models, speech recognition,etc. We can have multiple variants of multi-layered seq2seq models, the most common being the one where output of one LSTM layer is passed as input to the layer above and the hidden state of the final encoder layer is then used to initialize the hidden state of the first decoding layer. Output of each decoding layer is then passed to the layer above until the final layer where a softmax activation function is used to predict the output sequence.

As described in the previous section, we have extracted out the sentences of the main speaker(target sentence) and those on which the main speaker responds to(source sentence). The model is trained on the scripts of all seasons of a particular TV series at a time to learn the behavior of the main character. The model is trained giving input sentences to the encoder generating a hidden thought vector that is sent to decoder as an input. The learning happens in a supervised learning fashion, hence, the target sentence is given to the decoder at each time step to help it learn to predict the next word. The beginning of sentence is marked with <GO> and <EOS> marks the end of sentence.

Each input sentence when given to the encoder and decoder models, is passed through an embedding layer. Embedding layer is a dense vector representation of a word, unlike one-hot

vector. Such a layer helps determine related words, we have used pretrained Google word2vec matrix.

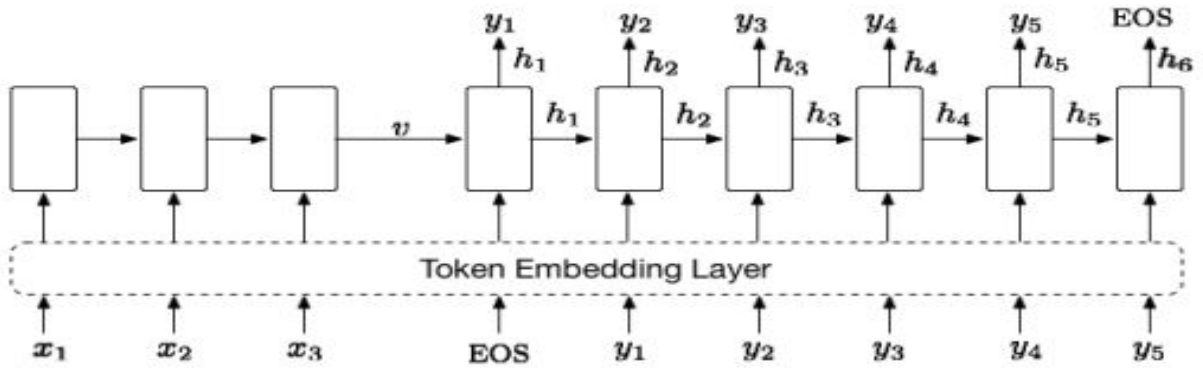


Fig. 1: An example of sequence-to-sequence encoder-decoder model. The model is fed input sentence X(words x1, x2 and x3) and outputs sentence Y(words y1, y2, y3, y4 and y5). EOS represents End of Sentence. V represents thought vector of x. The hidden state ht captures the sequential information in [x, y1, y2, . . . , yt−1]. [8]

B. Model Specifics:

We took help from the DeepQA repository which implements A Neural Conversational Model [9] (aka the Google chatbot). It uses a RNN (seq2seq model) for sentence predictions.
The implementation is done in python using Tensorflow.

The architectural details of our model is given below:
Embedding Layer: We use pre-trained Google word2vec dense vectors weights that consists of 3 million x 300 sized matrix.

RNN Cells: We use 2 layer LSTM cells in our model for encoder and decoder with embedding size as 256.

C. Losses:

Cross Entropy Loss:
Cross entropy indicates the distance between what the model believes the output distribution should be, and what the original distribution really is. It is defined as,

$$H(y,p) = -\Sigma y_i log(p_i)$$

It is a widely used alternative of mean squared error. It is used when node activations can be understood as representing the probability that each hypothesis might be true, i.e. when the output is a probability distribution.

Perplexity:

Perplexity metric in Natural Language Processing tasks is a way to capture the degree of 'uncertainty' a model has in predicting (assigning probabilities to) some text. It is calculated as a weighted geometric average of the inverses of the probabilities.

$$P = exp(\sum_{x} p(x) log \frac{1}{p(x)})$$

VI. Results and Experiments (*Shubham Bhargava)*

We ran and evaluated our model on two TV series datasets, namely Friends and Big Bang Theory on Seasons 1-10. All the experiments and results are stated in the subsequent paragraphs. The final results of our model on Friends dataset trained on character Joey are shown in Table 3.

To our surprise, our baseline model doesn't do very well with Google word2vec pre-trained word vectors. The responses generated are full of repetitive, common words. We suspect that this is due to the pre-trained word vectors we use are trained on general vocabulary (common crawl), which is very different from the vocabulary the characters use. Therefore, we only focus on the model without pre-trained word vectors.

We restricted the maximum length of sentences to 8 due to lack of enough computational resources. Also, the hidden states size is kept to 2048, embedding size to 256, and trained with number of epochs equal to 200 and learning rate to 0.001. Finally, the loss drops below 1 and Perplexity close to 2.

We tried evaluating our model with both attention and without attention. The results obtained are shown in Table 3. As we can see, the answers are good and related to what is being asked in the question. In some cases, the responses are perfect as in Table 3(a), when the chatbot replies "Joey Tribbiani! From the Wall!" in response to the question "Who are you?", while in some cases the response is not related to the query such as the last one in Table 3(a). Although our model does not give perfect answers all the times, it is fluent and mostly gives related answers.

Table 3: Result chat transcript of Joey from Friends and Sheldon from Big Bang Theory.

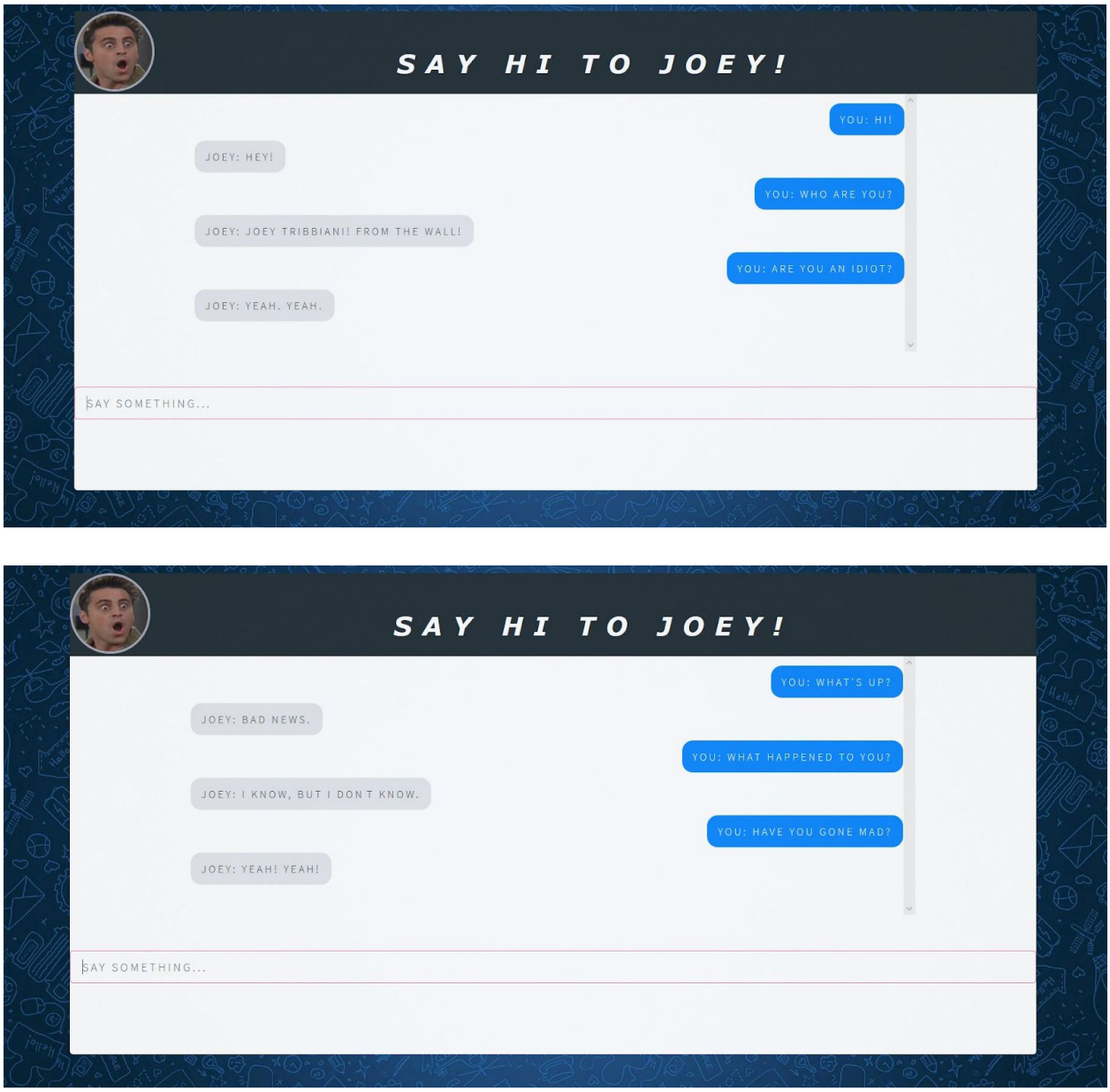
Question	Answer
Who are you?	Joey tribbiani! from the wall!
food?	That would be great!
You are stupid!	Yes! no! no.
Drake Ramore?	Yeah, what are you doing?

(a) Result chat transcript of character Joey

Question	Answer
Are you awesome?	I am. Oh
Why spot?	Because that changes everything.
Good afternoon.	Good night.
What is the time?	I'm not sure.

(b) Result chat transcript of character Sheldon

VII. User Interface Design Framework & Implementation (*Sameer Kumar Behera*)



Django, a Python-based Free and Open-Source Full Stack Web Development Framework is used to develop full fledged websites in Python. We used Django to develop our chatbot application. It encourages rapid development along with clean and pragmatic design which follows the model-view-template (MVT) architectural pattern. The framework emphasizes reusability and pluggability of components, less code and low coupling.

Redis, an open-source in-memory key-value structure project is used for implementing a distributed, in-memory key-value database with optional durability and also supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, bitmaps, streams

and spatial indexes. Lastly, but certainly not the least, is Redis's Pub/Sub Feature. This particular feature is used by us for the chat environment. It is also used for social network connections and for triggering scripts based on Pub/Sub events.

The ChatBot was run and tested on our local systems. Also we pushed the app on AWS Cloud Platform and ran the app on an EC2 instance.

The details of the implementation and installation requisites for a local machine can be seen on the following Github repository link: https://github.com/shbhmbhrgv/Personality-Chatbot

VIII. Conclusion (*All*)

This report discussed our approach to develop a personality based chatbot. A personality based chatbot can be used in various applications such as IT help desk or domain specific customer care applications. We demonstrate that it's possible to encode certain aspects of the personality and identity of a character in the chatbot. The model is trained entirely end-to-end with no handcrafted rules. While the bots speak fluently and most of the times related to the topic, sometimes it gives gibberish outputs. The bots also do not perform well when keeping into account the context from previous 2-3 sentences.

Also, the TV show dialogues scraped from Internet are of poor quality, and we had to use heuristics to separate conversations, which affected the performance of the bot. The lack of reliable evaluation metrics also made it hard for us to optimize. It was hard to differentiate whether the bot was getting better or worse, as with decrease in loss, the bot started outputting more generic or Yes, no responses. Future work includes developing a hybrid approach, by combining hand-crafted rules alongwith seq2seq model, and a better loss function. Also, we plan to enhance the use of context by taking into account all the sentences spoken whether they are related to the main character or not since it helps keeping the context better.

IX. References (*All*)

[1] "ELIZA", Weizenbaum, Joseph (1976). Computer Power and Human Reason: From Judgment to Calculation. New York: W.H. Freeman and Company. pp. 2,3,6,182,189. ISBN 0-7167-0464-1.

[2] Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

[3] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

[4] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXiv preprint arXiv:1509.00685 (2015).

[5] Li, Jiwei, et al. "A persona-based neural conversation model." arXiv preprint arXiv:1603.06155 (2016).

[6] Friends Corpus https://fangj.github.io/friends/, Big Bang Corpus https://bigbangtrans.wordpress.com/about/

[7] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (NIPS), pages 3104–3112.

[8] Li, Xiucheng & Zhao, Kaiqi & Cong, Gao & Jensen, Christian & Wei, Wei. (2018). Deep Representation Learning for Trajectory Similarity Computation. 617-628. 10.1109/ICDE.2018.00062.

[9] Vinyals, O., and Le, Q. 2015. A neural conversational model. Proceedings of the International Conference on Machine Learning, Deep Learning Workshop.