Friendroid: A friendly chatter bot.

1st Ojasvi Aggarwal
Indraprastha Institute of Information Technology
ojasvi17033@iiitd.ac.in

2nd Prerna Kalla
Indraprastha Institute of Information Technology
prerna17042@iiitd.ac.in

I. Problem Statement

Chatter bot is an important man-machine interaction based on NLP. It plays vital role in fields like customer service, personal assistant etc.
Friendroid: A chatbot using RNN to predict responses for human interaction. This is based on Sequence-to-sequence model. Salient features: LSTM cells, bidirectional dynamic RNN, decoders with attention.
Project objectives are as follows:

- Personalizing the voice for the bot.
- Speech to text and text to speech functionality.
- Chatter bot learns from the database.
- GUI for the bot.

II. Motivation

Chatter bots are becoming a mainstream must have. In today's world, people are concerned about how frequently they are cared. There are phones, messages, social media etc. But still, everyone feels a need for someone special to be with them always. This helped to think of this friendly bot named "Friendroid".

III. Literature Review

RNNs have become major research topics for dialogue systems and is used in most modern chatting assistants. LSTM(a type of RNN) used by Apple for the Quicktype and Siri, by Amazon for Amazon Alexa, by Google Home for speech recognition, Allo, Google Translate etc.

We are referring other papers from A* conferences like AAAI, NIPS. etc

IV. Database Details

Database used: Cornell MovieDialogs Corpus
Link: https://drive.google.com/open?id=1xoxMiSq62Zd8fhG3PDVt7tXtZ8VTlPWf

V. Methodology

A. Approach

- We used the data set named "Cornell MovieDialogues Corpus".
- Trained the model using RNN with LSTM.
- Integrated GUI using Tkinter python.
- Incorporated features like speech to text and text to speech in python.

B. Algorithm

```
Algo_Chatbot(Cornell_Dataset):
        +Data Preprocessing
        +Input Preprocessing
        +Producing Intermediaries
        +Create 2 RNN : Encoder & Decoder
        +Sequence to sequence model
        +LSTM encoding to produce fixed length vector
        +LSTM decoder to decode fixed length vector
        +Producing model saver
        +Testing

Algo_Voice_TTS(Input_Text):
        +Provide text input
        +Use pyttsx3
        +Use pyttsx3.engine to speak voice ouput

Algo_Voice_STT(User_Voice):
        +Record voice using sounddevice.py
        +Use speech_recognition.recogniser
        +Show text output
```

```
|Algo_Voice_Produce(User_Voice):
|        +Record voice
|        +Extract MFCC features
|        +Produce MFCC graph
|        +Produce MFCC based voice(to be done)
|
|+Input - Cornell_Dataset
|+Data and Input Preprocessing:
|        Remove special symbols
|        Remove small word or lonng sentences
|        Everything to lower case
|Intermediaries:
|        jsons, csv's
|Final Output:
|        model.cpkt
|Testing:
|        Tested upon seq2seq model
|        Compare predicted output with ground truth
|        Higher similarity means higher accuracy
|
```
]

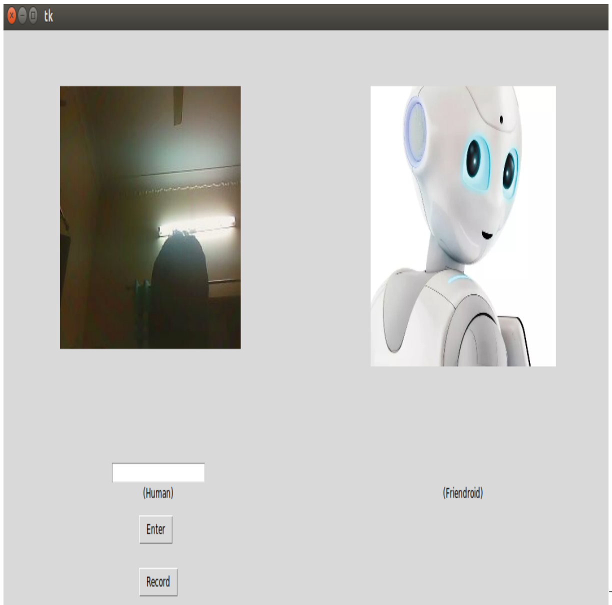
Fig. 1. Approach/ Algorithm

VI. Outputs and Results

The below given figure shows some requests and responses from the testing phase. Our model is able to respond well to basic conversational statements.

```
['good', 'morning', 'miss', '<UNK>']
['how', 'are', 'you', 'doing', '<EOS>']

['are', 'you', 'in', 'charge', 'here']
['yeah', 'that', 'is', 'it', '<EOS>']

['how', 'is', 'it', 'going']
['just', 'like', 'old', 'times', '<EOS>']

['would', 'you', 'mind']
['yes', 'i', 'am', 'afraid', '<EOS>']

['i', 'almost', 'forgot']
['yes', 'i', 'am', 'very', 'sorry', '<EOS>']

['you', 'want', 'my', 'advice']
['i', 'guess', 'so', '<EOS>']

['are', 'you', 'serious']
['no', 'not', 'really', '<EOS>']

['please', 'please', 'forgive', 'me']
['what', 'are', 'you', 'sorry', 'for', '<EOS>']
```
]

Fig. 2. Request and Response Examples

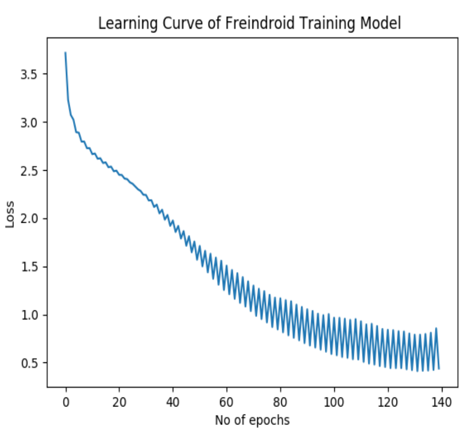


]

Fig. 3. Graphical User Interface

We are using the following matrices to evaluate the performance of our bot:

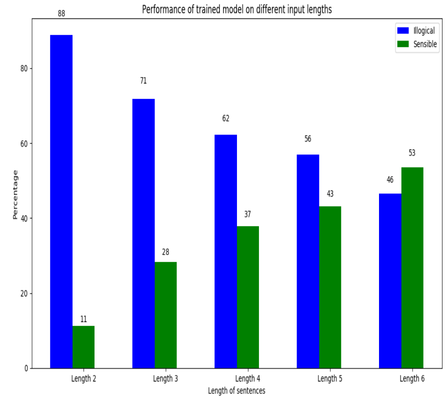
- Learning curve of the bot



]

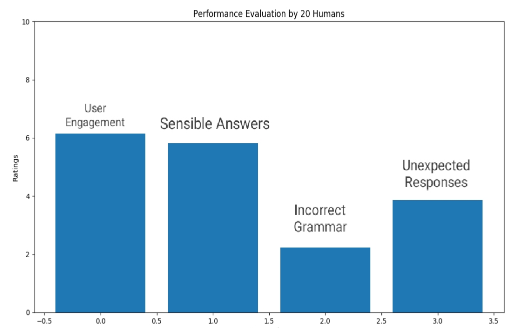
Fig. 4. Learning Curve

- Accuracy of model w.r.t. length of asked question
- Human surveys over various aspects of bot



]

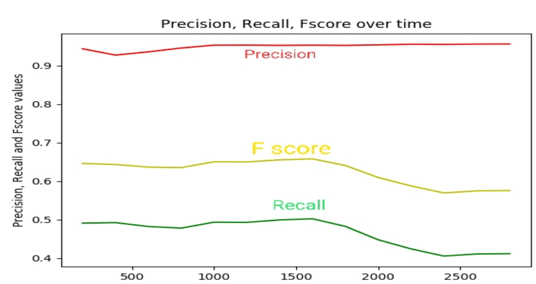
Fig. 5. Accuracy of model w.r.t. length of question



]

Fig. 6. Human surveys

- Precision, Recall and Fscore curve of the bot



]

Fig. 7. Precision, Recall and Fscore

VII. Milestones Achieved

Voice Conversion and Cloning
Speech to text using sound_recognition package ✓ Text to speech using Pyttsx3 package ✓ Create personalized voice- a) Extract MFCC features of any voice using librosa, numpy and python_speech_features ✓ b) Create digital voice of a person using lyrebird API ✓
Learning Chatbot
Dataset Preprocessing ✓ Train RNN model using tensorflow ✓ Initial testing of our RNN model ✓
Graphical User Interface
Create a user friendly GUI using Tkinter package ✓ Integrate GUI with backend ✓ GUI comprises of : a) Input button to provide text input to bot ✓ b) Record button for voice based user input ✓ c) Friendroid display of output message ✓ d) Background pronounce of bot's response ✓

]

Fig. 8. Milestones Achieved

VIII. Analysis

- Accuracy of our model on test data is around 43 %.
- The learning curve shows that training loss decreases as the number of epochs increase. It shows that the model is continuously learning.
  - The model learns faster initially and reaches a constant state eventually.
- Our model gives more meaningful responses on longer sentences than an shorter ones. Since our model does not remember context of the conservation, it may give illogical answers on certain questions of small length.
- According to 20 human evaluators, our model performs moderately in giving sensible responses. The model gives incorrect and illogical responses at times, when the question contains words not present in the model vocabulary.
- F1 score is 0.54. Precision value for our model is higher than recall.Precision is high when words of request and response and present in the model vocabulary.
- Our model has good results when all words used in request and response are present in the model vocabulary.If not, our model is prone to give illogical results.

IX. Summary

- Do not train your chat-bot on a small data-set. Our model performs moderately because it is trained on sentences of maximum length 6.It's performance can be improved by training on larger length sentences like 20-30.
- Do not use decoders in sequence-to-sequence model without attention mechanism as it will increase training time. It also solves the issue of decoding longer sentences.
- Do not set hyper-parameters of your model randomly. Experiment with different values and compare the performance of the generated models to find the best fit for your neural network.
- Do not use MFCC features to create personalized voice. It is extremely difficult and not efficient. Either create a neural network to solve this purpose or use and existing one to learn a new voice. API's like Lyrebird can also be used.

X. Conclusion

Conversational AI is extensively used in major product based companies and is the simplest way of communication between humans and AI. Introduction of recurrent neural networks with lstm's has improved the quality of chat-bots drastically.In this project we created a chat-bot using sequence-to-sequence model along with personalized voice based feature. We achieved 43% accuracy for our model. Shortcomings include training model on small sentences and lack of contextual understanding. Future work includes training our model on a larger data-set and adding a personality to the bot so that it can have themed conversations.

References

[1] Vinyals, Oriol, and Quoc Le. "A neural conversational model." arXiv preprint arXiv:1506.05869 (2015).

[2] Yan, Rui. "" Chitty-Chitty-Chat Bot": Deep Learning for Conversational AI." IJCAI. 2018.

[3] Radziwill, Nicole M., and Morgan C. Benton. "Evaluating quality of chatbots and intelligent conversational agents." arXiv preprint arXiv:1704.04579 (2017).

[4] Tensorflow documentation. Link : https://www.tensorflow.org/tutorials/