Natural Language Processing

Dialogue System

Abstract

In this Project, I have explored a dialogue system (Building a chatbot) with 3 different Datasets.

1. Cornell Movie Dialogues
2. QA Dataset
3. SQUAD (Stanford Question Answering Dataset) [**Incomplete**]

I have trained the chatbot both individually and combined to compare the performance of the chatbot. Due to memory restrictions and time constraints, I was only able to train the Cornell Movie dialogue dataset for **30000** iterations and QA dataset up to **2600** iterations, but as I have got a pretty significant improvement for loss of QA dataset, So I have stopped training the chatbot any further.
For the 3rd dataset, I was not able to train the chatbot due to some errors, Still, I have described the process and attached the code for performed for SQUAD Dataset.

For the second improvement, I have created a chatbot with the personality of **Barney Stinson** by adding his signature character traits and dialogues in the initial greeting and some extent of questions.

I have tested Initial and improved chatting experience 3 times (6 times in total) which can be checked in output_convo.txt.

Overall, we have noticed a significant improvement in chatting experience when trained with both the datasets when compared to individual ones even though the QA dataset individually provided a noticeable improvement in the loss.

Omkar Deshpande
(R00183613)

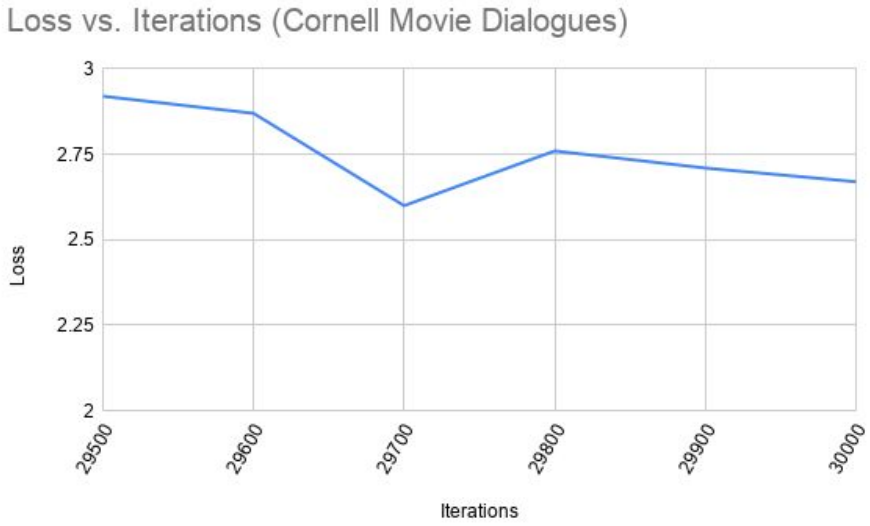
Improvement

Training for Multiple datasets

Cornell Movie Dialogue Dataset

For the Cornell movie dataset, I have trained the chatbot for **30000 iterations** and received a Loss rate of around **2.70.**
As the system was not able to handle the checkpoints after that (17 GB Folder) and killing the program repeatedly, I have stopped the training.

LOSS VS ITERATIONS :



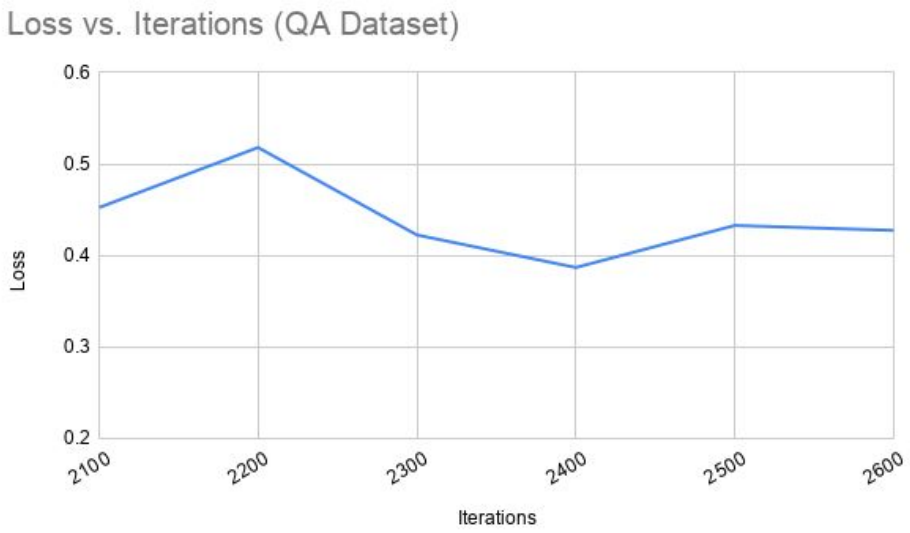
Parameter Tuning :
I have tried to tune 3 parameters, Learning rate and Iterations per save and Test size.
For the **Learning rate**, I have observed LR of **0.4** provided the best results for the 2000 iterations.
For **Iterations per save**, I have noticed an initial size of 50 and overall size of 100 to be best for the time complexity for my system.
For **Test size**, I have checked from 10000 to 20000 iterations with a step of 1000.

QA Dataset

The QA Dataset I have chosen is taken from **The WikiQA Corpus** and I have chosen 3 subsets of the Question answer-pairs provided in the dataset.

I have trained the dataset individually for **2600** iterations and noticed a loss rate of **0.4.**

LOSS VS ITERATIONS :



Parameter Tuning :

I have tried to tune 2 parameters, Learning rate, and Test size.

For the **Learning rate**, I have observed LR of 0.5 provided the best results for 500 iterations.

For **Test size**, I have given a fixed test size of 500.

Code Explanation :

For the second dataset, I have added a function **get_QA_Dataset2(),** in which I have taken the 3 subsets of the QA dataset and merged them into a single Dataset by using append function of the pandas data frame.

I have filled Null values with string NULL and considered Question and Answer columns which then I have taken into two different lists and passed those lists to prepare datasets where those lists are distributed into Training and test sets, as well as Encoding and decoding, is performed.

SQUAD Dataset

I have taken the SQUAD Dataset from Stanford's official website. From the 2 subsets, I have chosen Development set as the raw dataset's size of 40 MB.

For the second dataset, I have added a function **get_QA_Dataset3().**
As the Dev dataset was in JSON format, I have converted the dataset into CSV using an online tool (The python code was not able to load the file successfully).

In terms of preprocessing, I have removed the rows and columns with more than 10 NULL values.
I have filled the remaining NULL values with 0.
As there were many unnecessary fields, I have only taken Question and First Answer for the question (There were 3).
I have transposed the fields and tried to split the rows and columns to Question and Answer fields.
Due to time restriction, I was not able to train the semi-cleaned dataset and stopped there.

Create a chatbot with personality

For the second improvement, I have chosen to go with Rule-Based System in which I have given certain questions for which an answer will be given based on the set of responses present for the questions.
This System might not be best in terms of manual assignment of responses, But it makes the chatbot faster and as you can select the set of responses, You can personify the chatbot and add the elements such as **Critiquing** and **Sarcasm** which cant be done by an automated chatbot.
I have chosen to give my bot the personality of **Barney Stinson** by adding his signature dialogues such as "Legen..DAARRYYY" and "PLEASE" and I have also tried to implement an element of sarcasm in some of the questions.
I have tried to add several responses to the same questions in order to keep the random aspect and variety and also added the construct that if the person speaking to chatbot is sad or just types " . " to mention that he is not in the mood to talk, our chatbot will ask the question to keep the chat alive.

In the initial stage of talking, I have added a separate section of Greetings in which if any of the greeting words are typed, Our chatbot will greet the person or just nod depending on the random choice.
I have added this aspect separately to improve the customization of the bot.

Code Explanation :
I have created a function **basic_Response()** which takes the terminal input (chat) as a parameter.
In the function, I have defined 2 initial lists for Greeting inputs and responses and other than that 6 specific questions relating to our persona and a list of lists providing up to 3 sets of responses for each question.

```
responses = [
      [
          "I am the master of most, Barney Stinson!!",
          "Ask your wife *wink*, Just kidding! , I am Barney!!",
          "Whos asking?",
      ],
      ["Hello my dear chap!!", "Hello gentlemen!", "Hi, Have a drink!!"],
      ["what do you do?", "Why so glum?"],
      ["What happend? Did you a girl stood you up?"], ....
```

In the main function, I have checked whether the person speaking has greeted us or not, If no, Then if the person has asked any question and that question is present in our set or not,
If the person has asked any specific question which is present, we will randomly choose a response and reply back, Otherwise, we will generate tokens from the input and generate a response according to our trained data.

Conclusion

In Conclusion, We can say that the size and variation of the dataset is an essential part of training a chatbot as the increase in size increases the accuracy and variation increases the scope of the search space. We have also seen that using a rule-based system can increase the humane aspect of the bot and make the bot interesting to chat.