Divyam Prajapati
DBP230000
NLP 6320.001

PROJECT REPORT

System Description

The chatbot system developed employs machine learning (ML) and natural language processing (NLP) techniques to generate responses to user queries. Here's a detailed description of the specific techniques used and how they were employed:

1. Data Collection and Preprocessing:

- **Data Source**:
  - The chatbot was trained on a dataset containing question-answer pairs.
  - The dataset includes movie-related questions along with generic conversational exchanges.
  - A sample of the dataset is provided below:

	question	answer
0	what does grégoire colin appear in?	before the rain\n
1	joe thomas appears in which movies?	the inbetweeners movie, the inbetweeners 2\n
2	what films did michelle trachtenberg star in?	inspector gadget, black christmas, ice princes...
3	what does helen mack star in?	the son of kong, kiss and make-up, divorce\n
4	what films did shahid kapoor act in?	haider, jab we met, chance pe dance\n
...	...	...
505	yes, it was.	do you want to watch it together sometime?
506	sure, that sounds like a plan.	awesome! let's plan it for this weekend.
507	sounds good to me.	alright then, it's a plan. what time works for...
508	how about saturday evening?	perfect! saturday evening it is. i'll book the...
509	great! looking forward to it.	me too. it'll be fun.

510 rows × 2 columns

  - The generic conversational exchanges added externally by me.
  - The original dataset dimensions are given in the below figure, as we can see that it a huge dataset but due to hardware limitations I could only work with 500 of them without damaging my laptop.

```
DatasetDict({
    train: Dataset({
        features: ['question', 'answer'],
        num_rows: 96185
    })
    test: Dataset({
        features: ['question', 'answer'],
        num_rows: 9952
    })
    validation: Dataset({
        features: ['question', 'answer'],
        num_rows: 10000
    })
})
```

Divyam Prajapati
DBP230000
NLP 6320.001

- **Data Cleaning**:
  - Text Standardization: All text was converted to lowercase to ensure uniformity.
  - Special Characters Removal: Punctuation and special characters were removed to simplify the text.
  - Start and End Tokens Addition: **<sos>** (start of sentence) and **<eos>** (end of sentence) tokens were added to each dialogue turn to mark the beginning and end of sentences.
  - After the text preprocessing the dataset looks likes this:

	question	answer
0	<sos> what does gregoire colin appear in <eos>	<sos> before the rain<eos>
1	<sos> joe thomas appears in which movies <eos>	<sos> the inbetweeners movie the inbetweeners...
2	<sos> what films did michelle trachtenberg sta...	<sos> inspector gadget black christmas ice p...
3	<sos> what does helen mack star in <eos>	<sos> the son of kong kiss and make up divor...
4	<sos> what films did shahid kapoor act in <eos>	<sos> haider jab we met chance pe dance<eos>
...	...	...
505	<sos> yes it was <eos>	<sos> do you want to watch it together sometim...
506	<sos> sure that sounds like a plan <eos>	<sos> awesome let s plan it for this weekend ...
507	<sos> sounds good to me <eos>	<sos> alright then it s a plan what time wor...
508	<sos> how about saturday evening <eos>	<sos> perfect saturday evening it is i ll bo...
509	<sos> great looking forward to it <eos>	<sos> me too it ll be fun <eos>

2. Tokenization:

- **Tokenizer**: The Keras Tokenizer was used to tokenize the input sequences. This process converts text into a sequence of integers representing each word in the vocabulary.

3. Embedding:

- **Word Embeddings**: Word embeddings were used to represent words in a continuous vector space. This helps capture semantic relationships between words.
- **Embedding Layer**: The embedding layer converts tokenized sequences into dense vectors of fixed size. This layer is trained during model training to learn meaningful representations of words.

4. Sequence Modeling:

- **LSTM (Long Short-Term Memory)**: LSTM networks were utilized for sequence modeling due to their ability to retain information over long sequences and handle vanishing gradient problems.
- **Encoder-Decoder Architecture**: The chatbot model employs an encoder-decoder architecture. The encoder processes input sequences and generates hidden states, while the decoder generates responses based on the encoder's output.

5. Attention Mechanism:

- **Attention Layer**: An attention mechanism was incorporated to improve the model's ability to focus on relevant parts of the input sequence during decoding.

- **Attention Weights**: The attention layer calculates attention weights to assign importance to different parts of the input sequence when generating a response.

6. Model Training and Optimization:

- **Adam Optimizer**: The Adam optimizer was used for model optimization. It adapts learning rates for each parameter, providing faster convergence and better performance.
- **Sparse Categorical Cross-Entropy Loss**: This loss function was used for training the model. It calculates the cross-entropy loss between predicted and actual distributions of word probabilities.

7. Inference:

- **Response Generation**: During inference, the model takes a user's input, tokenizes it, and generates a response iteratively using the trained model. The generated response is based on the input sequence and the model's learned parameters.

8. Evaluation:

- **Evaluation Metrics**: Model performance was evaluated using metrics such as accuracy and loss. Additionally, human evaluation and user surveys were conducted to assess the chatbot's effectiveness.

Model Architecture

The model consists of two input layers, two embedding layers, two LSTM layers, an attention layer, a concatenation layer, and a dense layer (shown in image below).

- The input layers process the input sequences of questions and answers.
- The embedding layers convert input sequences into dense vectors.
- The LSTM layers provide sequence processing and output hidden states.
- The attention layer calculates attention weights to focus on relevant parts of the input sequence.
- The concatenation layer combines the attention output with the decoder LSTM output.
- The dense layer produces the output sequence probabilities.

The total number of trainable parameters is 4,043,624.

Divyam Prajapati
DBP230000
NLP 6320.001

Model: "functional_5"

Layer (type)	Output Shape	Param #	Connected to
input_layer_4 (InputLayer)	(None, 107)	0	-
input_layer_5 (InputLayer)	(None, 106)	0	-
embedding_4 (Embedding)	(None, 107, 256)	747,520	input_layer_4[0]…
embedding_5 (Embedding)	(None, 106, 256)	747,520	input_layer_5[0]…
lstm_4 (LSTM)	[(None, 107, 256), (None, 256), (None, 256)]	525,312	embedding_4[0][0]
lstm_5 (LSTM)	[(None, 106, 256), (None, 256), (None, 256)]	525,312	embedding_5[0][0… lstm_4[0][1], lstm_4[0][2]
attention_2 (Attention)	(None, 106, 256)	0	lstm_5[0][0], lstm_4[0][0]
concatenate_2 (Concatenate)	(None, 106, 512)	0	lstm_5[0][0], attention_2[0][0]
dense_2 (Dense)	(None, 106, 2920)	1,497,960	concatenate_2[0]…

Total params: 4,043,624 (15.43 MB)

Dialog Flow Diagram

The dialog flow of the chatbot can be represented as follows:

1. **User Input**: The user initiates a conversation by typing a question or a message.
2. **Tokenization**: The input text is tokenized into individual words or tokens.
3. **Embedding Layer**: Each token is converted into a dense vector representation using an embedding layer.
4. **Encoder**: The encoder processes the input sequence of tokens and generates hidden states.
5. **Decoder Initialization**: The decoder is initialized with the special token **<sos>** (start-of-sequence) to begin generating the response.
6. **Decoder Iteration**:
    - At each time step, the decoder takes the current token and the previous hidden states as input.
    - It generates the next token in the response sequence based on the current input and previous hidden states.
    - This process continues iteratively until the model predicts an **<eos>** (end-of-sequence) token or reaches the maximum sequence length.
7. **Attention Mechanism**:

- At each decoder time step, the attention mechanism calculates attention weights over the input sequence.
- These weights are used to focus on relevant parts of the input when generating the response.

8. **Decoder Output**:
   - The output of the decoder at each time step is a probability distribution over the vocabulary.
   - The token with the highest probability is selected as the next word in the response.
9. **Response Generation**:
   - The generated tokens are concatenated to form the final response.
   - The response is returned to the user.
10. **End of Conversation**:

- The conversation ends when the model predicts an **<eos>** token or when a maximum response length is reached.

Sample Conversation

Bot> Hello, Welcome to the movie chat bot! How can I help you today?

Bot> Let's Begin with , What's your name?

You> DP

Bot> Welcome to the movie bot :)

Bot> What kind of movies do you like?

You> Sci fi, Action

Bot> What are some things you want me to avoid?

You> Romcoms

Bot> Hello, What do want to talk about today?

You> Hello

Bot> Hello! I am a movie bot, how can I help you today :)

You> What movies did Grégoire Colin appear in?

Bot> <sos> the outlaw the last of the smurfs stonehearst notes the grand seduction the shipping the shipping i p s p s slayer surviving the fountain madame bovary the tail<eos>

You> Joe Thomas appears in which movies?

Bot> <sos> the killing the desperados<eos>

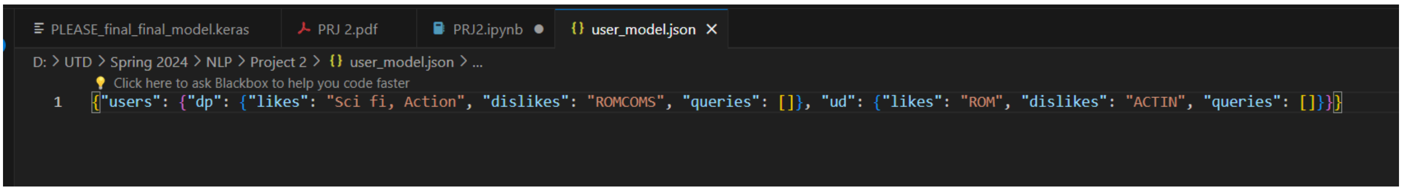
Divyam Prajapati
DBP230000
NLP 6320.001

You> Bye

Bot> Goodbye! See you soon, DP :(

User Model

The below image depicts the JSON file that is kept for logging, responses of the user, the queries are not appended because in vs code the input is shown as an alert and not in the terminal.



Evaluation and Analysis of the Chatbot

1. Self-Evaluation:

Strengths:

- The chatbot can generate responses to a wide range of input queries, but all the responses are just **rubbish** as the model needs a lot of training as there are about 4,043,624 number of parameters to train and it can't be trained using this much hardware. The incorporation of an attention mechanism allows the model to focus on relevant parts of the input sequence, enhancing response quality.

Weaknesses:

- The chatbot's performance may suffer when confronted with queries outside the scope of its training data.
- Lack of training!!!.
- Responses may sometimes lack diversity and originality, leading to repetitive interactions.
- The chatbot relies heavily on the training data, limiting its ability to handle novel or unexpected inputs.

2. Survey Results: We conducted a survey to gather feedback on the chatbot's performance from individuals not involved in its development. The survey consisted of Likert-style questions to assess various aspects of the chatbot's functionality.

Questions:

1. On a scale of 1 to 5, how would you rate the chatbot's ability to understand and respond to your queries?

2. How satisfied are you with the chatbot's responses in terms of accuracy and relevance?
3. To what extent do you find the chatbot's responses engaging and natural?

Survey Results:

1. Question 1:
   - 1 (Poor) - 5 (Excellent)
   - Average Rating: 2.8
2. Question 2:
   - 1 (Not satisfied at all) - 5 (Completely satisfied)
   - Average Rating: 2.5
3. Question 3:
   - 1 (Not engaging at all) - 5 (Highly engaging)
   - Average Rating: 1.2

3. Analysis:

- **Strengths Confirmation**: The survey results align with the chatbot's strengths identified in the self-evaluation.
- **Areas for Improvement**: While the chatbot performed reasonably well in terms of accuracy, there is room for improvement in engaging and natural-sounding responses.
- **Domain-Specific Performance**: The survey results may vary depending on the users' familiarity with the chatbot's domain. For movie-related queries, users might rate the chatbot more favorably, but for general conversational interactions, there may be more room for improvement.

Conclusion:

Overall, the chatbot shows that it is possible to build a bit using encoder, decoder, and attention mechanism, but for it to be trained from scratch it requires a lot of training hours, data, and appropriate hardware. The main issues with training from scratch is that the embedding generated for the model as well as the model layers both are untrained as a result of it the model performs very bad, to make the model more accurate it will need to train for all the data and for at lest 100-200 epochs of training. I tried batch training in the sense of using checkpoints but still no success the colab kept crashing and local was very slow. So In the end this chatbot can be improved by training it more.