Positive Chatbot

Team 129

Tunwa (Win) Tongtawee

and Boxun Zhu

Apr 13th, 2024

Abstract:

This report details the development of a chatbot designed to alleviate symptoms of depression and sadness. Utilizing a collection of mental health dialogues, the chatbot is powered by a LSTM that assesses the emotional tone of user inputs. It employs GPT-NEO for conversation generation, with a focus on guiding users to a more positive emotional state via supportive and encouraging exchanges.

Introduction:

With rising mental health concerns, there is an increasing need for accessible interventions. Our project introduces a chatbot that offers immediate support by identifying negative emotional states and encouraging a shift towards positivity. Initially, we aimed to develop a fully autonomous chatbot, but limitations led us to refine our approach to focus on positive emotional reinforcement using LSTMs and sentiment analysis techniques, combined with a model that is capable of holding a conversation.

Our initial approach to solve this problem was to train an LLM from scratch using mental health conversations, but this eventually opened up a plethora of issues listed below.

Issues with training an LLM from scratch:

- Not enough computing power
- Not enough Q and A pairs regarding mental health
- Not enough time
- Sample run generated gibberish

We later went on to tweak our initial approach to a more suitable approach which included using an LSTM for sentiment analysis and using a pre-trained model (GPT-Neo, Dialo) for holding a conversation.

Our final product employs a Long Short-Term Memory (LSTM) neural network architecture, which is specifically designed to process sequences of data. In this case, the LSTM is trained to analyze user inputs to determine their underlying sentiment. This analysis categorizes inputs as either positive or negative, which is a fundamental aspect of sentiment analysis.

Ideally, if we were able to complete our original goal, once the sentiment of the user's input is established, the system would use this information to generate contextually appropriate prompts. These prompts would be designed to guide the user's interaction with the chatbot, so the user has a better conversational experience. This would not only maintain the flow of the conversation better but also tailors the chatbot's responses to the emotional tone of the user inputs, making the interaction more engaging and personalized.

However, we ran into some issues with time and a power outage. That leads us to our current solution.

The decision to utilize an LSTM model for this task was driven by several considerations. Initially, our team considered developing a large language model (LLM) that could autonomously handle both the sentiment analysis and the generation of chatbot responses. However, due to constraints related to time and resources, this approach was impractical. Developing a full-scale LLM requires significant computational resources and expertise, which were beyond our current capabilities.

Instead, we opted for the LSTM model, which, while simpler than a full LLM, still effectively meets our needs. This model allows us to implement a robust system for sentiment analysis with a relatively lower overhead. Furthermore, by automating the generation of context-sensitive prompts based on sentiment analysis, we have successfully created a streamlined and effective chatbot experience. This strategic decision enabled us to focus our resources on refining the LSTM model and optimizing the chatbot's performance within the scope of our current capabilities. Thus, even with limited resources, we were able to deploy a sophisticated and functional system that would enhance user engagement and satisfaction.

Methods:

Datasets:

Mental Health Counseling Conversations: These conversations were structured with labeled speakers (ie. Patient and Physician). It was useful in training the models early on to have conversations about mental health and appropriate responses. There are slight issues with this dataset though (See discussion).

HappyDB: We looked into training our model with this, but it lacked Q and A pairs, but had lots of positive examples. We ended up using this model to test our LSTM ‘s accuracy.

Sentiment Labelled Sentences Data Set: This dataset includes balanced examples from Amazon, IMDb, and Yelp, with 500 positive and 500 negative sentences from each source. It is particularly well-suited for training models to understand and classify consumer sentiments in reviews. This was the model we ended up using to train our LSTM.

Stress Annotated Dataset (SAD): Comprised of SMS-like messages about everyday stressors, this dataset is intended to be universally recognized as negative by sentiment analysis models. It's specifically tailored for systems designed to detect stress or negative sentiment in short text messages. We used this dataset to evaluate our LSTM, similar to how we used HappyDB.

Preprocessing:
Our preprocessing steps were designed to clean the data of invalid characters, ensuring the textual data is optimized for model training.

- **Unicode Normalization**: We used the **unicodedata.normalize('NFKD', ...)** method to decompose characters into their base forms and separate diacritical marks, which standardizes the text format for consistency.
- **ASCII Encoding**: We encoded all characters to ASCII format, explicitly ignoring non-ASCII characters. This step simplifies the text by removing special characters or emoticons that could lead to misinterpretations during the tokenization phase.
- **Whitespace Normalization**: We condensed multiple spaces into a single space and stripped spaces from the beginning and end of strings. This cleanup step eliminates potential issues from excessive or leading/trailing whitespace that could affect model training.
- **Regular Expressions:** Regular expressions were utilized to replace newline (\n), tab (\t), and backslash (\\) characters with a single space. Additionally, the pattern r'[^a-zA-Z\s]' allowed us to remove any characters that are not letters or whitespace, ensuring data consistency.
- **Tokenization:** To prepare the textual data for machine learning models, we employed tokenization, a critical step that involves breaking down text into smaller components (tokens) such as words or phrases.
- **Loading and Cleaning Data**: We loaded our dataset from a JSON file into a Pandas DataFrame, which allows for easy manipulation and processing of structured data. We applied the **clean_text** function to both 'input' and 'output' columns for our chatbot portion to ensure both questions and responses were cleaned accordingly.
- **Label Encoding**: We converted the cleaned 'output' responses into numerical format using **LabelEncoder**. This transformation is crucial for the model to perform classification tasks, as it deals with numerical ids rather than text data directly.

We also built a txt to csv converter and combiner based on tab spaces, as lots of the datasets had unusable formats for their csv files, and we opted to do it ourselves for a cleaner set of data. This can be found in the **sentiment_data** folder as **sentiment_to_csv.py**.

Model Training:

Microsoft DialoGPT Model**:** Initially, we trained with the Microsoft DialoGPT model, fine-tuning it using Mental Health Conversation Data. While DialoGPT could generate coherent dialogues, it tended to produce blunt and disinterested responses, often disengaging from the topic or stating its limitations in medical advice.

GPT-2 Model: We also fine-tuned a GPT-2 model using a combination of Mental Health Conversations and a general dataset derived from Twitter. Despite these efforts, the model struggled to produce relevant and contextually appropriate responses.

BERT Model: A pre-trained BERT model was fine-tuned with the Mental Health Conversations Dataset. Although it accurately mimicked the training data, it was ineffective for interactive conversations as it tended to regurgitate complete responses from the dataset, lacking the ability to generate nuanced replies. **(See Appendix D)**

GPT-Neo: We utilized the GPT-Neo model for its conversational capabilities, fine-tuning it with the Mental Health Dataset. This model showed improvement in generating content that deviated from the exact training inputs, though the responses were still not fully coherent or contextually appropriate. **(See Appendix E)**

Sentiment Analysis:

LSTM Model: Using an LSTM model for sentiment analysis turned out to be quite effective at measuring the sentiment of a statement after training it. It was relatively fast, taking only 15 minutes, and offered the data we needed. We trained the base model using a mixed dataset from the Sentiment Labelled Sentences Dataset. It was accurately defining inputs sentiments and was capable for our use. We utilized this model to help gauge user inputs and then return a sentiment attached to the user's original input as a prompt for a chatbot.

Data:

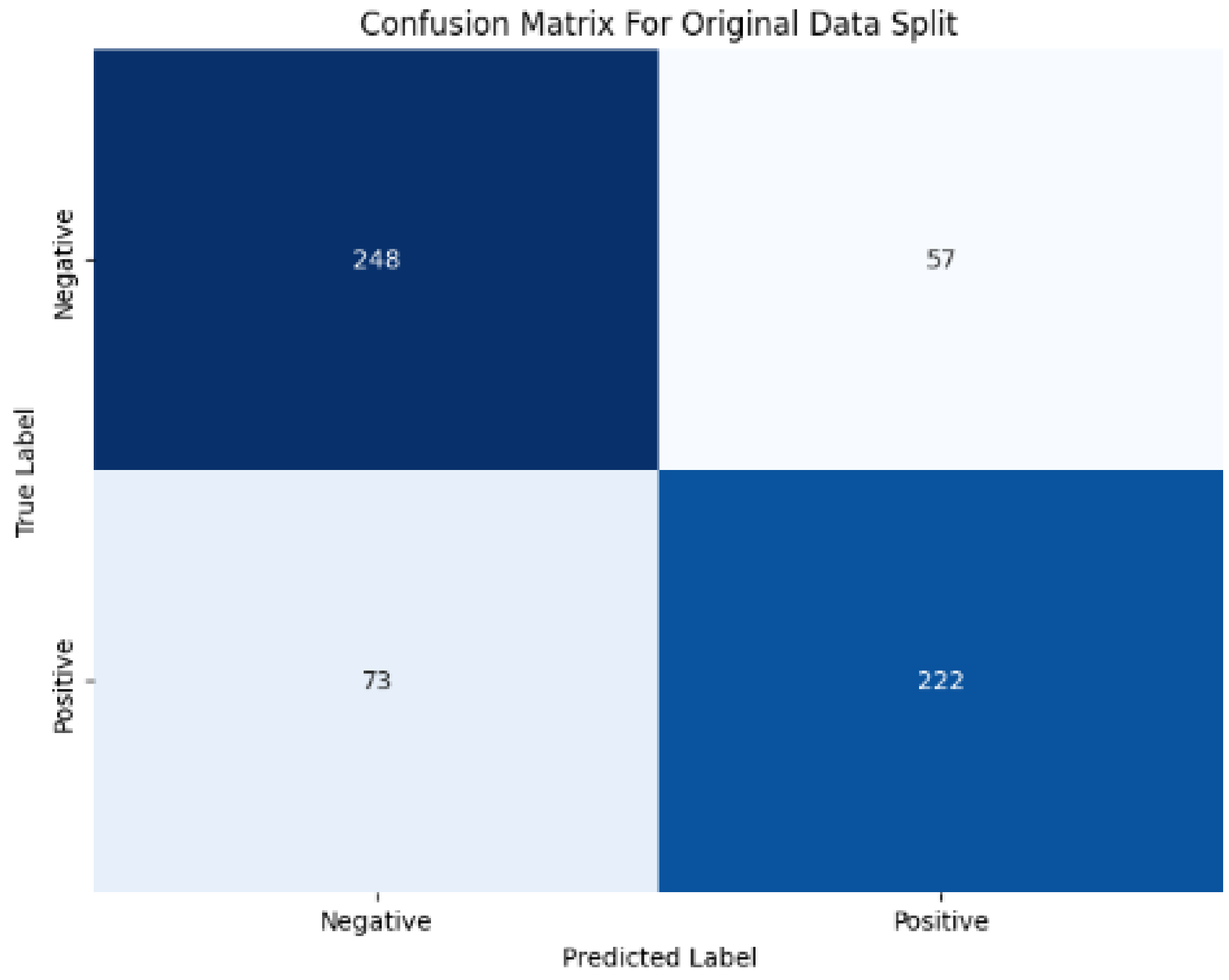


Figure A: Confusion Matrix of the Training Dataset

Classification Report of the Training Dataset

	precision	recall	f1-score	support
Negative	0.77	0.81	0.79	305
Positive	0.8	0.75	0.77	295
accuracy			0.78	600
macro avg	0.78	0.78	0.78	600
weighted avg	0.78	0.78	0.78	600

Figure B: Classification Matrix of the Training Dataset

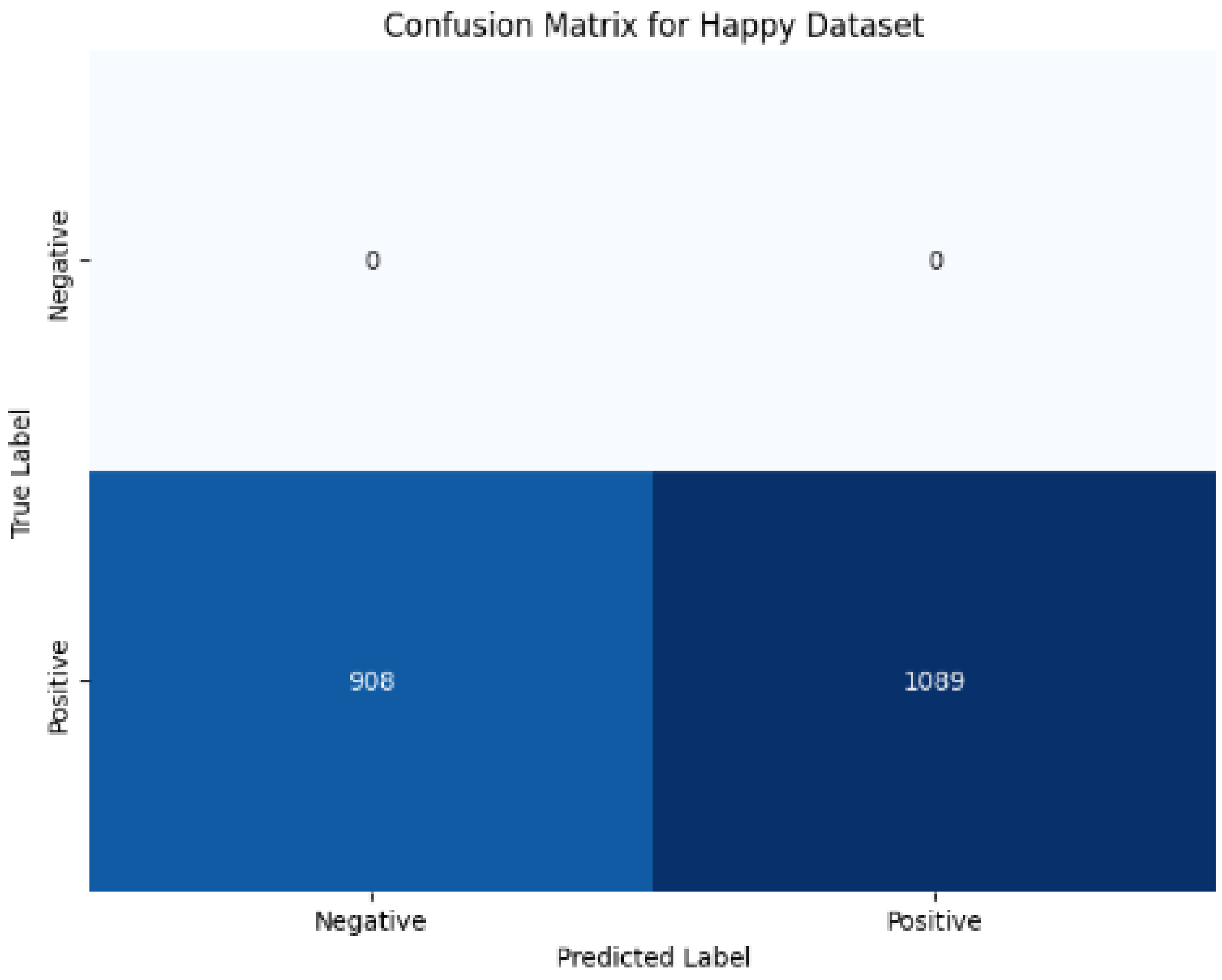


Figure C: Confusion Matrix of the HappyDB Dataset

Classification Report of the HappyDB Dataset

	precision	recall	f1-score	support
Negative	0	0	0	0
Positive	1	0.55	0.71	1997
accuracy			0.55	1997
macro avg	0.5	0.27	0.35	1997
weighted avg	1	0.55	0.71	1997

Figure D: Classification Report of the HappyDB Dataset

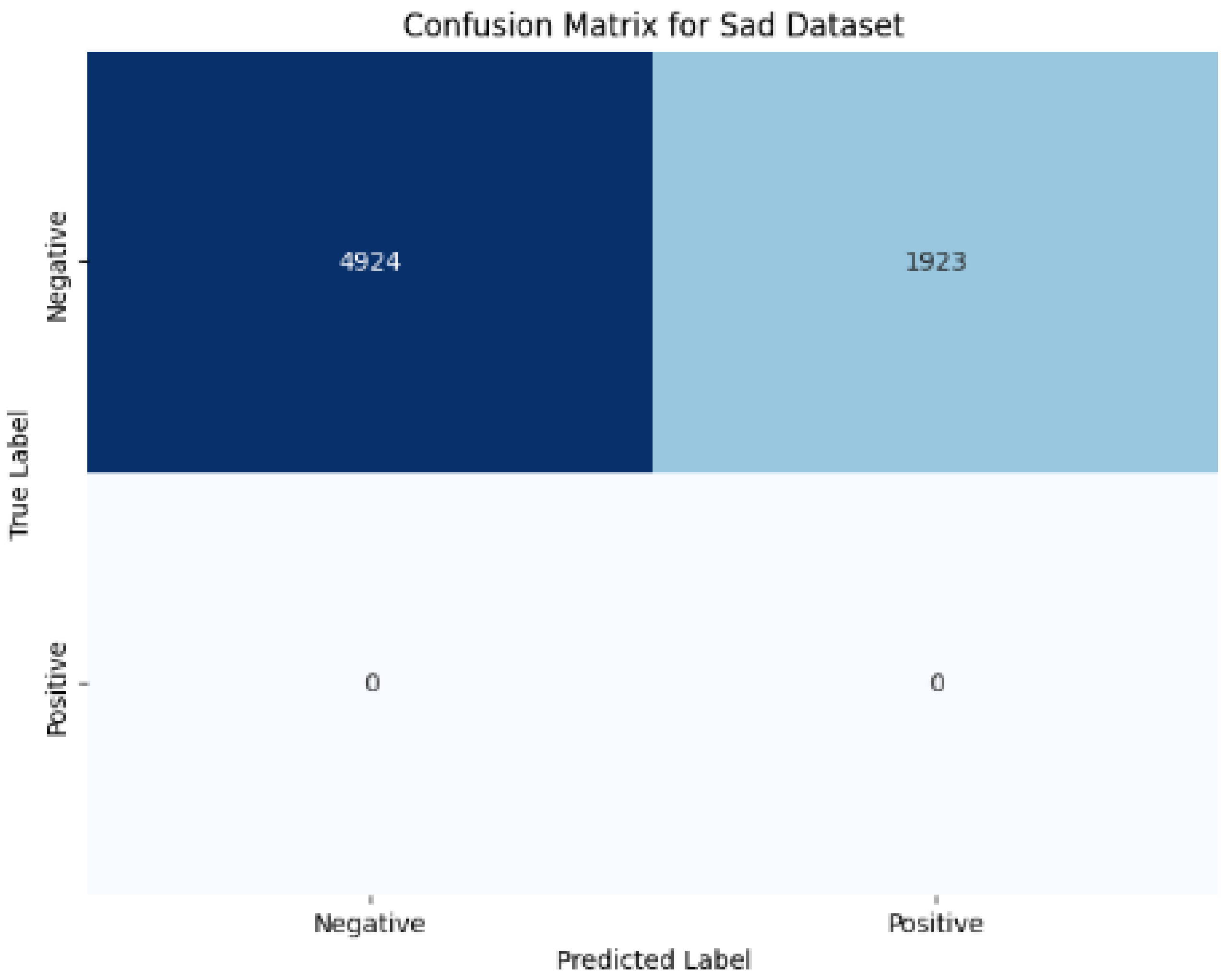


Figure E: Confusion Matrix of the SAD Dataset

Classification Report of the SAD Dataset

	precision	recall	f1-score	support
Negative	1	0.72	0.84	6847
Positive	0	0	0	0
accuracy			0.72	6847
macro avg	0.5	0.36	0.42	6847
weighted avg	1	0.72	0.84	6847

Figure F: Classification Matrix of the SAD Dataset

Results:

~~Ideally, we would have many more sets of data to show, see the **Issues** section for more information. In an ideal world, here we would show the confusion matrix and classification report on a the HappyDB and SadDB dataset, as well as a general twitter dataset. This would have served as a good test to see how accurate our model is at predicting whether something is positive or negative, as all the values in the dataset are either or, which would show us if there were false positives or negatives, or uncaught cases. It would also show us a real-world example of how our model can analyze texts.~~ ***(SEE EDIT BELOW)***

Initial Outcomes

The initial deployment of our chatbot revealed several challenges:

- **Fixed and Nonsensical Responses**: Early iterations of the chatbot, particularly those trained solely with generic models like DialoGPT, often generated irrelevant or repetitive responses. This issue was observed in both the scripted interactions and free-form conversations, where the chatbot failed to adequately recognize or respond to the emotional context of the user's input.

- **Model Limitations**: The initial GPT-2 and Microsoft DialoGPT models displayed a tendency to generate blunt and sometimes inappropriate responses, which did not align with the supportive and empathetic tone required for mental health support.

Refined Model Performance

After refining our approach and incorporating an LSTM for sentiment analysis, significant improvements were observed:

- **Sentiment Analysis Accuracy**: The integration of LSTM greatly enhanced the chatbot's ability to discern and appropriately respond to the emotional content of user inputs. For instance, the chatbot could distinguish between negative sentiments such as "Many lost their jobs because of COVID, and it is highly dangerous" and positive sentiments like "I am happy that my family members are safe in these tough times." As seen in **Figure A-F**, our LSTM is correct more than 50% of the time.

- **Chatbot:** Although we did not get our chatbot to work properly, the GPT-Neo model led us to have better responses. However, it was still unacceptable for use, leading us to go with the alternative which was to make prompts for chatbots instead.

- **Confusion Matrix and Classification Matrix:**
    - **Original Training Data Split:** The confusion matrix indicates a balanced ability to classify both positive and negative sentiments, with slightly better performance on negative ones. The presence of false negatives and false positives suggests some misinterpretation by the LSTM, possibly due to complex sentiment indicators such as negation or idiomatic language.
    - **HappyDB Dataset:** The evaluation yielded a high precision rate of 1 for positive sentiment, indicative of no false positives since there were no negatives in the dataset. However, the recall rate was 0.55, denoting that the model captured 55% of the positive instances. The accuracy of the model stood at 0.55, aligning with the recall for positive sentiment. The confusion matrix mislabeled 908 instances that were true positives as negative, indicating a significant misclassification error.
    - **SAD Dataset:** The evaluation revealed perfect precision for negative sentiment at 1, since there were no positive instances in the dataset, which skews this data. The recall for negative sentiment was 0.72, indicating that the model failed to identify 28% of the negative instances. The possibilities of this can be discussed below. The confusion matrix showed that all 6847 instances were classified as negative, resulting in 4924 true negatives and 1923 false negatives, which is reasonable considering the simplicity of our model.

In all cases, the model's error likely stems from its limited understanding of the context and language nuances, especially when dealing with negation ("not good" vs. "good"), which can dramatically alter sentiment. More times than not, negations are probably used in the positive dataset, which could have caused a disproportionate amount of the HappyDB dataset to be misclassified compared to the SAD dataset. Improvements could potentially be made from exposing the model to a wide variety of sentiment expressions during training, or providing a direct definition to the model of what a negation is.

Discussion:

Evaluation of Models

The development of the mental health support chatbot involved several iterations and refinements of AI models, each contributing differently to the project's outcomes:

- **Initial Models (DialoGPT and GPT-2)**: These models, while robust in generating general conversational text, were not initially effective for the specialized context of mental health support. Their limitations became apparent in their tendency to produce responses that were sometimes blunt or misaligned with the sensitive nature of mental health discussions. This highlighted the necessity of a more tailored approach to training models on domain-specific datasets.

- **BERT Model**: BERT's effectiveness was contingent on the quality and specificity of the training data. When trained on highly relevant mental health dialogues, BERT excelled in generating responses tailored to the input, but only when the entire response was contained in the fine tuning, making it not fit for a chatbot.

- **GPT-Neo**: GPT-Neo's performance fell short of expectations, with a significant portion of its output being incoherent or lacking substantive content. This was evident in its inability to generate meaningful words consistently, suggesting that it struggled with constructing contextually relevant and grammatically correct sentences. The subpar results may be attributed to a variety of factors, including but not limited to insufficient training data or inadequate fine-tuning.

- **LSTM Model**: The LSTM model demonstrated a significant increase in success rates when analyzing the Stress Annotated Dataset (SAD), particularly in detecting stress-related expressions. The model's success in this area underscores the potential of using targeted neural network architectures for specific types of sentiment analysis within specialized datasets.

Analysis of Datasets

The choice and composition of datasets were critical in shaping the chatbot's performance:

- **Mental Health Counseling Conversations**: This dataset was instrumental in training the models to understand and generate responses appropriate for mental health contexts. However, its limitations included a potential lack of diversity in conversational scenarios and repetitive dialogue patterns, which could hinder the model's ability to generate varied responses.

- **Sentiment Labelled Sentences Data Set**: This dataset served extremely useful for training our LSTM. It provided a broad spectrum of sentiment-rich text. This data was useful for evaluation, training, and verification.

- **HappyDB:** This dataset was filled with positive happy moments, which served to be perfect for testing and highlighting our LSTM's effectiveness at identifying positive inputs. (maybe talk about how some data is incorrect)

- **Stress Annotated Dataset (SAD)**: Focused on stress-related expressions, this dataset not only helped in fine-tuning the chatbot's responses to stress-specific inputs but also highlighted the effectiveness of the LSTM model in accurately classifying and responding to stressed states. (talk about how data above shows how it wasn't all correct)

Considerations for Future Development

The development journey of the chatbot sheds light on several areas for future enhancement:

1. **Custom Chatbot Framework**: We initially intended to build a chatbot from scratch, but ran into challenges, but we hope to be able to do this in the future as it would complete the experience and our original vision.

2. **Emotion Identification**: With more specific emotion identification, our LSTM could produce better prompts for chatbots, or provide more detail on the user, other than if their input was positive or negative.

3. **Accuracy**: By finding additional datasets and defining more rulesets, our model could potentially be more accurate and more capable, as a large part of our inaccuracy was from the lack of rulesets and the sparseness of our data compared to the chatbots that can be seen online such as ChatGPT.

4. **User Interface and Accessibility**: If given more time, we would have wanted to build a full stack web app and have the model run on an API or build a mobile application for it. This would probably make it much more accessible in the future and it remains an area for improvement.

Datasets Issues:

Mental Health Counseling Conversations: The dataset has limitations; notably, it is AI-generated which can lead to concerns about the authenticity and variability of the dialogues. Additionally, there is a significant presence of repetitive prompts, which could impact the model's ability to learn diverse conversational patterns and potentially limit the generative quality and variability of the model's responses.

Issues:

Challenges in LSTM Model Training

During the course of our project, we faced several setbacks related to LSTM training, particularly with our computational resources and data handling procedures such as:

Unclear CUDA Errors:

- None of the CUDA errors were verbose, even after toggling verbosity in the os
- The first training attempt, titled 'lstm_analysis_with_sklearn', was halted by an unidentified issue within the CUDA environment, but with more debugging, we figured out it was related to tensor sizes, batch sizes, and flattening tensor values from the GPU to the CPU for numpy calculations.
- Even after figuring out the issue, we were unable to resolve it. We combed through the dataset and changed the way we preprocessed our data and still found no success.

Ultimately, our breakthrough came when we restructured our approach to training. By changing to a LSTM library, keras, instead of our own architecture, we found more success.

Power Outage:

Win’s power went out as we were implementing the feature for a user to input a prompt and have it analyzed the sentiment and create a prompt. However, the main focus of the project was to have accurate sentiment analysis, which we achieved through the LSTM model. Lots of this project was spent trial and erroring many models and learning about various frameworks and how each one works.

~~We originally had planned to use the HappyDB dataset and the SadDB dataset as a unit test for our project, however, the challenge of Win’s power going out prevented us from doing so. This would have served as a good test to see how accurate our model is at predicting whether something is positive or negative, as all the values in the dataset are~~

~~either or, which would show us if there were false positives or negatives, or uncaught cases.~~

~~Given another chance complete this project, our feature to create a prompt would also ideally be more robust, adding in additional words so the chatbot will properly respond to the user, such as adding in additional context to their emotions so the chatbot will respond with a more understanding tone. We would also like to add additional emotions so the chatbot will have a better context.~~ ***(EDIT: Win got his power back at 3 AM and we were able to work on it all night to implement it, as it was the least we could do with how much we cut back from our original idea.)***

Individual Takeaways:

Win: Learned a lot regarding CUDA, PyTorch, how difficult training LLMs are, how LLMs are structured, how BART and other models can be used for sentiment analysis. My personal most important takeaway was sometimes some challenges help one grow lol. Thanks for the pep talk after class and the great semester.

Boxun: Learned a lot about Python, Pre-Processing, Pandas, PyTorch, CUDA, git / version control and sentiment analysis. Most important takeaway should be the sentiment data analyzing, graphing and accuracy calculating. Learned a lot about the name of the terminology which never remember the name of them previously. Also, next time when choosing a project topic and how to finish it, I will choose carefully since the initial decision was so hard to achieve.

References:

GPT-Neo: https://github.com/EleutherAI/gpt-neo

GPT2: https://huggingface.co/openai-community/gpt2

DialoGPT: https://huggingface.co/microsoft/DialoGPT-medium

HappyDB: https://megagon.ai/happydb-a-happiness-database-of-100000-happy-moments/

Psychology-10k: https://huggingface.co/datasets/samhog/psychology-10k/tree/main

Stress Annotated Dataset (SAD): Stress-Annotated-Dataset-SAD/SAD_v1.zip at main · PervasiveWellbeingTech/Stress-Annotated-Dataset-SAD (github.com)

Mental Health Counseling Conversations: https://huggingface.co/datasets/Amod/mental_health_counseling_conversations

Setting up BERT: https://promactinfo.com/blogs/building-a-chatgpt-like-platform-with-bert-a-beginners-guide/

How BERT Works: https://www.datacamp.com/blog/what-is-bert-an-intro-to-bert-models

Appendix:

Appendix A: Sentiment Analysis Using LSTM (Trains Model) (LSTM_MODEL_FINAL.ipynb)

1. **External Dependencies and Libraries**:
    - **pandas**, **numpy**: For data manipulation and numerical operations.
    - **nltk**: Natural Language Toolkit for preprocessing text, such as removing stopwords.
    - **keras**: High-level neural networks API, running on top of TensorFlow.
    - **tensorflow**: An end-to-end open-source platform for machine learning.
    - **torch**: An open-source machine learning library for tensor computation and deep neural networks.
    - **scikit-learn**: Machine learning library for Python, used for data mining and data analysis.
    - **seaborn**, **matplotlib**: Used for data visualization and graphical plotting.
2. **Classes, Functions, and Methods**:

- **LSTM Class**: A Sequential model including Embedding, LSTM, and Dense layers for binary sentiment classification.
- **text_preprocessing Function**: Processes raw text to remove punctuation, normalize to lowercase, and filter out stopwords.
- **pad_features Function**: Standardizes the length of each tokenized text to ensure consistent input size.
- **load_data and create_vocab_and_encode Functions**: Load and preprocess data from a CSV file, creating a vocabulary encoding for text data.
- **Training Loop**: Manages the training process of the LSTM model with defined datasets.

3. **How to Use the Program**:
    - Install required dependencies listed at the beginning of the notebook.
    - Execute the notebook to preprocess the data, train the LSTM model, and assess its predictive performance.
    - Model checkpoints are saved as 'LSTM_FINAL.h5', which can be reloaded for further evaluation or prediction.
    - Users can input their own text at the end of the notebook to receive a sentiment analysis prediction, which could inform a chatbot's response in a hypothetical application. (also shown in the demo)

Appendix B: Demo of Sentiment Analysis (DEMO_FINAL.py)

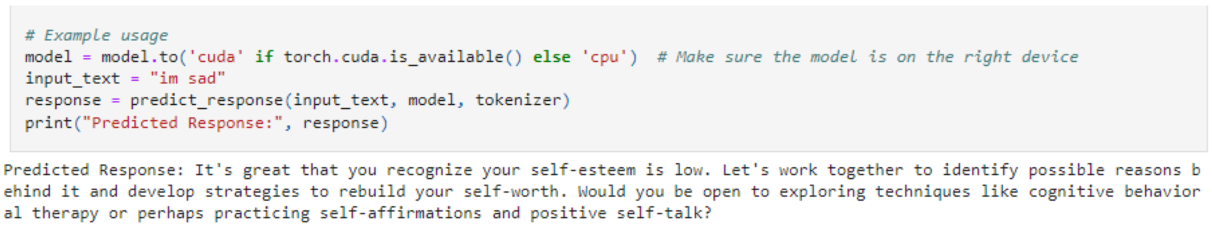
1. **How to Use the Demo**
    - Ensure that the model file **LSTM_FINAL.h5** is located in the same directory as the demo script.
    - Install necessary Python packages if they are not already installed: pandas, numpy, nltk, tensorflow, keras.
    - Run the script by entering the following command:
      python demo_script.py "Your input sentence here."
    - Ensure there are quotations around your input, or the script will not work.

Appendix C: Unit Test of Sentiment Analysis Model and Demo (UNIT_TEST_FINAL.py)

1. **How to Run the Unit Tests**

- Ensure that the model file LSTM_FINAL.h5 is located in the same directory as the demo script.
- Install necessary Python packages if they are not already installed: pandas, numpy, nltk, tensorflow, keras.
- Use the cd command to change the directory to where your test script is located. For example: cd C:/Downloads/129
- Run the test script using the Python interpreter using the following command: *python UNIT_TEST_FINAL.py*

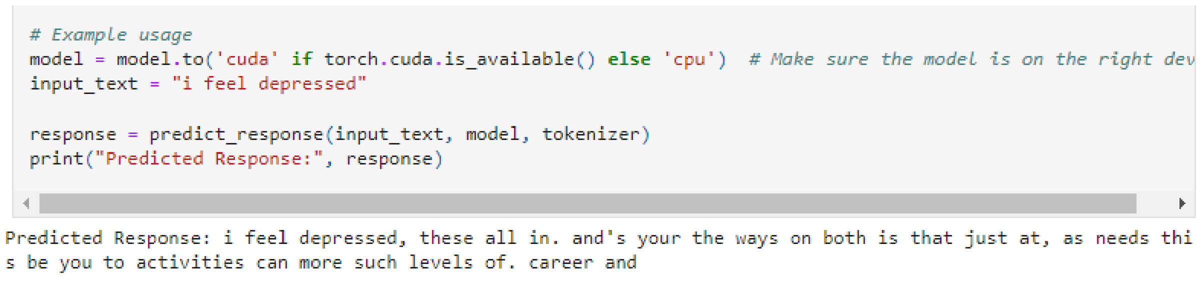
Appendix D: BERT Chatbot Response



```
# Example usage
model = model.to('cuda' if torch.cuda.is_available() else 'cpu')  # Make sure the model is on the right device
input_text = "im sad"
response = predict_response(input_text, model, tokenizer)
print("Predicted Response:", response)
```

```
Predicted Response: It's great that you recognize your self-esteem is low. Let's work together to identify possible reasons b
ehind it and develop strategies to rebuild your self-worth. Would you be open to exploring techniques like cognitive behavior
al therapy or perhaps practicing self-affirmations and positive self-talk?
```

Appendix E: GPT-NEO Chatbot Response



```
# Example usage
model = model.to('cuda' if torch.cuda.is_available() else 'cpu')  # Make sure the model is on the right dev
input_text = "i feel depressed"

response = predict_response(input_text, model, tokenizer)
print("Predicted Response:", response)
```

```
Predicted Response: i feel depressed, these all in. and's your the ways on both is that just at, as needs thi
s be you to activities can more such levels of. career and
```