



National Institute of Technology Karnataka, Surathkal

Sentiment Analysis for Mental Health

Project Report

MA722 - Foundations of Machine Learning Algorithms

Master of Technology (Research)
Computational and Data Science

Rashmi R
243CD003

Table of Contents

Project Title: Sentiment Analysis for Mental Health.....	1
Table of Contents.....	2
Introduction.....	3
Problem Statement.....	3
Motivation.....	3
Architecture.....	3
Existing Methods.....	4
How Is This Project's Approach Better?.....	5
Implementation.....	6
Data Preprocessing and Visualization.....	6
Working.....	7
Handling Imbalanced Data - SMOTE Oversampling Technique.....	10
Model Development & Evaluation.....	11
Working.....	11
Model.....	12
ChatBot Development.....	14
Graph Modeling.....	14
Cypher.....	14
Graph: Sample Architecture.....	16
Conversation Chain.....	16
Knowledge Graph (of all conversations).....	17
Final Output.....	18
Chat - 1.....	18
Chat - 2.....	19
Conclusion.....	20
References.....	20

Introduction

Problem Statement

Mental health issues such as Depression, Anxiety, Stress, and other disorders have become increasingly prevalent, necessitating effective tools for early detection and intervention. Traditional methods of mental health assessment involve a psychiatrist but not everyone in this generation is open to other people due to the fear of judgement.

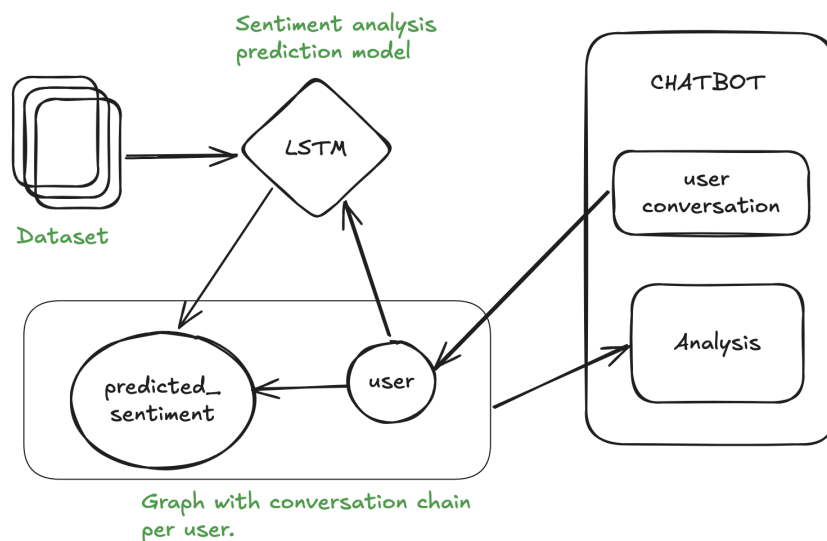
Motivation

This project aims to develop a chatbot that allows individuals to openly discuss their mental health without fear of judgment, providing early detection and intervention for mental health issues such as depression, anxiety, and stress. By leveraging a Bidirectional LSTM model for sentiment analysis, the system classifies users' mental health statuses based on their textual input. The chatbot processes and analyzes conversations, offering valuable insights into the user's emotional well-being. Additionally, a graph-based model predicts the overall sentiment of the user's conversation. The solution is deployed enabling seamless interaction and offering users a safe, anonymous platform to express themselves.

Source for dataset:

<https://www.kaggle.com/code/muhammadfaizan65/sentiment-analysis-for-mental-health-nlp/input>

Architecture



Existing Methods

1. Traditional Machine Learning Approaches

- a. Techniques: Naive Bayes, Adaptive Boosting (AdaBoost), Support Vector Machine (SVM), Decision Trees.
- b. Strengths:
 - i. Time-efficient when dealing with large, labeled datasets.
 - ii. Simpler to train and interpret.
- c. Limitations:
 - i. Poor performance on datasets with complex linguistic patterns or large-scale data.
 - ii. Algorithms like SVM struggle with dynamic datasets requiring constant updates.
 - iii. Overfitting issues in models like AdaBoost.

2. Deep Learning Approaches

- a. Techniques: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), including Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM).
- b. Strengths:
 - i. Capable of handling large-scale labeled or unlabeled datasets.
 - ii. High accuracy in sentiment analysis due to better feature extraction and context understanding.
 - iii. Bidirectional models (e.g., Bi-LSTM) consider both previous and future context in a sequence, improving classification precision.
- c. Limitations:
 - i. Computationally intensive and requires significant resources.
 - ii. Lack of interpretability makes it challenging for diagnostic purposes.

3. Existing Deep Learning Studies

- a. Example: A study using the Guardian newspaper dataset reported the highest accuracy of 0.939 with a Single Layer CNN.
- b. Drawbacks:
 - i. CNN models may lose temporal information critical in sequential text.
 - ii. Black-box nature hinders explainability for mental health diagnosis.

How Is This Project's Approach Better?

1. Bi-LSTM Model with Context Preservation
 - a. Advantage: Bi-LSTM captures the temporal relationships in text while considering both past and future context, making it particularly suitable for mental health-related text with subtle emotional cues.
2. Graph-based Sentiment Aggregation
 - a. Innovation: Incorporating graph modeling for conversation-level sentiment analysis is unique. It provides an additional layer of analysis by considering the flow and progression of user sentiment over time.
 - b. Advantage: This approach addresses the lack of interpretability by offering a structured visualization of conversation sentiments, by aiding mental health professionals.
3. Judgment-free User Interaction
 - a. User-centric: The chatbot fosters a safe space for expression, unlike traditional clinical settings where individuals face fear of judgment.
 - b. Human-like Interaction: The chatbot uses gpt-4o-mini for conversational interactions, ensuring responses feel empathetic, natural, and engaging.
4. Hybrid Approach: Individual + Conversation Sentiment Analysis
 - a. Advantage: Combining Bi-LSTM's sequential text analysis with graph-based aggregation ensures both granular (message-level) and holistic (conversation-level) sentiment detection.
5. Deployment Focus
 - a. Ease of Access: Deployed as an interactive chatbot, the system is accessible to users 24/7, bridging the gap in mental health resources and reducing the stigma of seeking help.

This combination of Deep Learning methodology and user-focused design makes my approach a significant improvement over existing methods.

Implementation

Data Preprocessing and Visualization

To train and evaluate the sentiment analysis model, we need a large dataset of text labeled with sentiment categories. For this project, we use publicly available datasets such as:

Mental Health Text Dataset: Includes text from online forums and social media where users discuss mental health, annotated with sentiment labels.

Sentiment Analysis Dataset: Contains a collection of tweets labeled as positive, negative, or neutral.

The dataset was obtained from Kaggle and KDNuggets focussing on the format of the data required for the sentiment analysis. The dataset contains 51074 unique values. The data will be preprocessed to remove noise, handle missing values, and prepare it for use in the model.

Data preprocessing involves several steps:

- **Tokenization:** Breaking down text into individual words or tokens.
- **Lowercasing:** Converting all text to lowercase to maintain consistency.
- **Stopword Removal:** Removing common words like "the," "is," "in," etc., that do not contribute much to sentiment classification.
- **Lemmatization:** Reducing words to their base form (e.g., "running" becomes "run").
- **Padding:** Ensuring all text inputs are of the same length by padding shorter inputs with zeros.

Working

```
import pandas as pd
import numpy as np
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Bidirectional, Dropout
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
df = pd.read_csv("Combined_data.csv")
df.head()
```

Unnamed: 0		statement	status
0	0	oh my gosh	Anxiety
1	1	trouble sleeping, confused mind, restless hear...	Anxiety
2	2	All wrong, back off dear, forward doubt. Stay ...	Anxiety
3	3	I've shifted my focus to something else but I'...	Anxiety
4	4	I'm restless and restless, it's been a month n...	Anxiety

```
df["status"].value_counts()
```

count	
status	
Normal	16351
Depression	15404
Suicidal	10653
Anxiety	3888
Bipolar	2877
Stress	2669
Personality disorder	1201

dtype: int64

```
df = df.dropna()
len(df)
```

52681

```
df.columns
```

```
Index(['statement', 'status'], dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
Index: 53043 entries, 0 to 53042  
Data columns (total 2 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   statement  52681 non-null  object  
1   status     53043 non-null  object  
dtypes: object(2)  
memory usage: 1.2+ MB
```

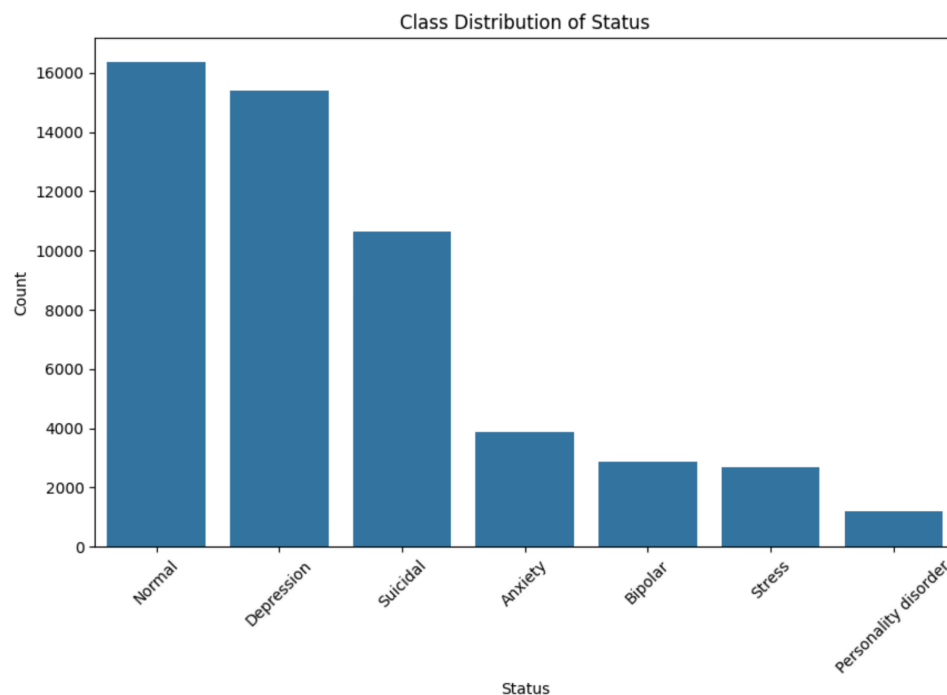
```
df.isna().sum()
```

```
statement    362  
status        0  
dtype: int64
```

```
df['status'].describe()
```

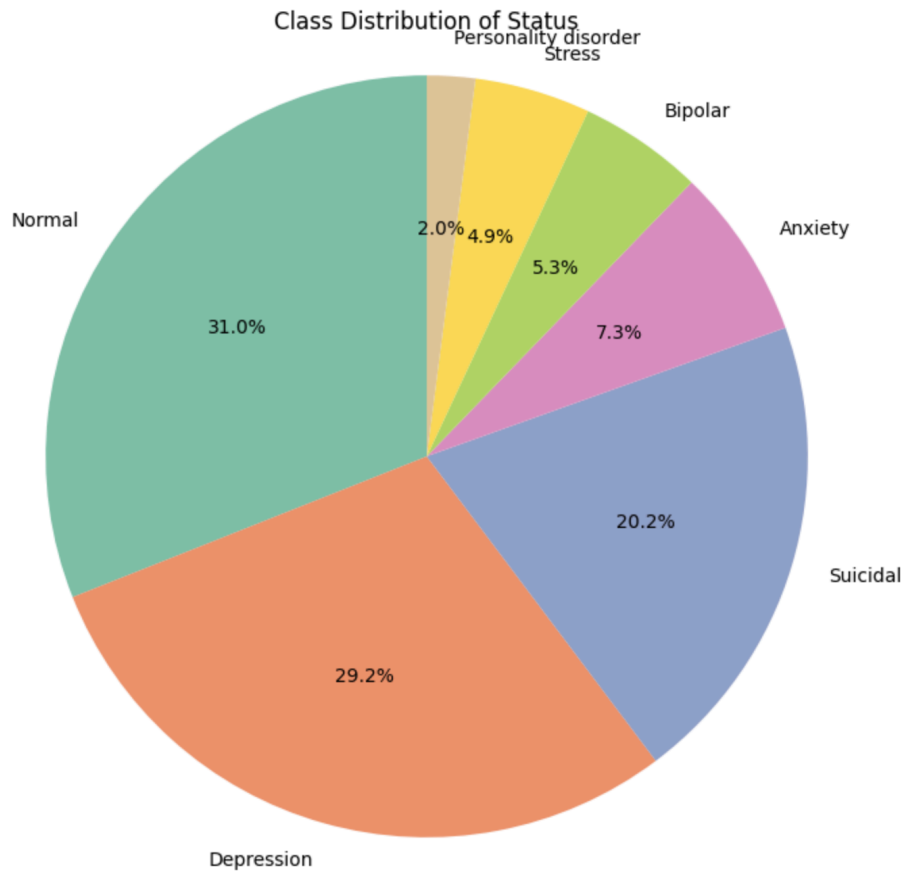
```
count      53043  
unique        7  
top      Normal  
freq     16351  
Name: status, dtype: object
```

```
# Plot the class distribution  
plt.figure(figsize=(10, 6))  
sns.barplot(x=class_counts.index, y=class_counts.values)  
plt.title('Class Distribution of Status')  
plt.xlabel('Status')  
plt.ylabel('Count')  
plt.xticks(rotation=45)  
plt.show()
```




```
# Count the occurrences of each class in the 'status' column
class_counts = df['status'].value_counts()

# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(class_counts, labels=class_counts.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Set2.colors)
plt.title('Class Distribution of Status')
plt.axis('equal') # Equal aspect ratio ensures that pie chart is circular.
plt.show()
```



```
texts = df['statement'].values
labels = df['status'].values
```

```
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)
```

```
label_encoder.classes_
```

```
array(['Anxiety', 'Bipolar', 'Depression', 'Normal',
      'Personality disorder', 'Stress', 'Suicidal'], dtype=object)
```

Handling Imbalanced Data - SMOTE Oversampling Technique

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

1. Identifying Class Imbalance

- a. The dataset's class distribution was analyzed.
- b. Certain mental health categories (e.g., Suicidal or Bipolar) had significantly fewer samples compared to others (e.g., Normal or Stress).
- c. This imbalance could lead to biased model predictions favoring majority classes.

2. Understanding SMOTE Logic

- a. SMOTE creates synthetic samples for minority classes by interpolating between existing samples.
- b. For each minority class sample, SMOTE selects k-nearest neighbors using Euclidean distance in the feature space.
- c. A synthetic sample is generated as:

$$x_{\text{new}} = x_i + \lambda \cdot (x_{\text{neighbor}} - x_i)$$

3. Implementation

- a. k-Nearest Neighbors (k=5 by default) were computed for each minority class sample.
- b. Synthetic samples were generated until the class sizes were balanced.
- c. Balancing the dataset ensures the model optimizes for all classes by minimizing the weighted loss function.

4. Outcome

- a. Balanced data improved the training process, reducing overfitting to the majority classes.
- b. Enhanced model's ability to generalize across all mental health statuses.

Here, I have used SMOTE library from imblearn python

```
from imblearn.over_sampling import SMOTE

from collections import Counter

counter = Counter(y_train)

print('Before', counter)

# oversampling the train dataset using SMOTE
```

```
smt = SMOTE()

X_train_sm, y_train_sm = smt.fit_resample(X_train, y_train)

counter = Counter(y_train_sm)

print('After', counter)
```

Model Development & Evaluation

The Bi-LSTM model is chosen for its ability to process sequences of text in both forward and backward directions. The Bi-LSTM is suitable for sentiment analysis because it can capture contextual information from both past and future words in a sentence. The architecture includes:

- **Embedding Layer:** Converts words into dense vectors of fixed size.
- **Bi-LSTM Layer:** Processes the word embeddings in both directions to capture the context.
- **Dense Layer:** Maps the output of the Bi-LSTM layer to sentiment classes.
- **Softmax Activation:** Outputs a probability distribution over sentiment classes (positive, negative, neutral).

The model is trained on the preprocessed text data using categorical cross-entropy loss and the Adam optimizer.

Working

Initialized a tokenizer that retains the top 20,000 most frequent words from the training data (X_train). It then fits the tokenizer to the training texts, preparing it to convert these texts into sequences of integers for further processing, with a maximum sequence length of 1,000 tokens.

```
# Tokenize and Pad the sequences

# parameters for tokenization
max_words = 20000 # max num of words to keep, based on word frequency
max_len = 1000 # max length of all sequences

# Tokenizer
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_train)
```

Convert the training and testing texts (X_train and X_test) into sequences of integers using the previously fitted tokenizer. This transformation allows the model to work with numerical representations of the text data.

```
# convert texts to sequences of integers
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
```

Pad the sequences of integers for both the training and testing data (X_train_seq and X_test_seq) to ensure a uniform input size of 1,000 tokens. This padding ensures that all sequences have the same length, which is essential for model training and evaluation.

```
# Pad sequences to ensure uniform input size
X_train_padded = pad_sequences(X_train_seq, maxlen=max_len)
X_test_padded = pad_sequences(X_test_seq, maxlen=max_len)
```

Model

```
# Build the BiLSTM model
# model parameters
embedding_dim = 128 # embedding layer output dimension

# Build the model
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=max_len))
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.5))
model.add(Dense(16, activation='relu'))
model.add(Dense(7, activation='softmax')) # 7 categories in total
```

```
# compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# train the model
history = model.fit(X_train_padded, y_train, epochs=20, batch_size=128, validation_split=0.2)
```

```
Epoch 1/20
264/264 — 56s 176ms/step - accuracy: 0.4626 - loss: 1.3713 - val_accuracy: 0.5543 - val_loss: 1.0229
Epoch 2/20
264/264 — 75s 168ms/step - accuracy: 0.6232 - loss: 0.8497 - val_accuracy: 0.6608 - val_loss: 0.7987
Epoch 3/20
264/264 — 82s 169ms/step - accuracy: 0.6998 - loss: 0.7068 - val_accuracy: 0.6943 - val_loss: 0.7777
Epoch 4/20
264/264 — 82s 169ms/step - accuracy: 0.7537 - loss: 0.6142 - val_accuracy: 0.7197 - val_loss: 0.7375
Epoch 5/20
264/264 — 82s 169ms/step - accuracy: 0.7947 - loss: 0.5508 - val_accuracy: 0.7173 - val_loss: 0.7826
Epoch 6/20
264/264 — 82s 171ms/step - accuracy: 0.8259 - loss: 0.4923 - val_accuracy: 0.7431 - val_loss: 0.7468
Epoch 7/20
264/264 — 82s 169ms/step - accuracy: 0.8565 - loss: 0.4082 - val_accuracy: 0.7166 - val_loss: 0.9637
Epoch 8/20
264/264 — 45s 170ms/step - accuracy: 0.8389 - loss: 0.4719 - val_accuracy: 0.7414 - val_loss: 0.8318
Epoch 9/20
264/264 — 82s 170ms/step - accuracy: 0.8775 - loss: 0.3679 - val_accuracy: 0.7316 - val_loss: 0.8252
Epoch 10/20
264/264 — 82s 169ms/step - accuracy: 0.8724 - loss: 0.3912 - val_accuracy: 0.7367 - val_loss: 0.8804
Epoch 11/20
264/264 — 82s 169ms/step - accuracy: 0.9018 - loss: 0.3163 - val_accuracy: 0.7337 - val_loss: 0.9266
Epoch 12/20
264/264 — 82s 170ms/step - accuracy: 0.9153 - loss: 0.2658 - val_accuracy: 0.7346 - val_loss: 0.9922
Epoch 13/20
264/264 — 83s 174ms/step - accuracy: 0.9281 - loss: 0.2332 - val_accuracy: 0.7351 - val_loss: 0.9868
Epoch 14/20
264/264 — 81s 170ms/step - accuracy: 0.9310 - loss: 0.2218 - val_accuracy: 0.7341 - val_loss: 0.9712
Epoch 15/20
264/264 — 83s 174ms/step - accuracy: 0.9414 - loss: 0.1983 - val_accuracy: 0.7312 - val_loss: 1.0812
Epoch 16/20
264/264 — 81s 169ms/step - accuracy: 0.9526 - loss: 0.1650 - val_accuracy: 0.7335 - val_loss: 1.1860
Epoch 17/20
264/264 — 82s 170ms/step - accuracy: 0.9528 - loss: 0.1624 - val_accuracy: 0.7363 - val_loss: 1.2560
Epoch 18/20
264/264 — 45s 170ms/step - accuracy: 0.9536 - loss: 0.1585 - val_accuracy: 0.7315 - val_loss: 1.1812
Epoch 19/20
264/264 — 82s 169ms/step - accuracy: 0.9630 - loss: 0.1295 - val_accuracy: 0.7378 - val_loss: 1.2583
Epoch 20/20
264/264 — 82s 170ms/step - accuracy: 0.9683 - loss: 0.1160 - val_accuracy: 0.7375 - val_loss: 1.3063
```

Training Accuracy: 96.83%

Testing Accuracy: 93.66%

330/330 — 16s 50ms/step - accuracy: 0.9366 -
loss: 0.2211 Test Accuracy: 0.9366

ChatBot Development

Graph Modeling

In addition to classifying individual inputs, the sentiment of an entire conversation can be predicted using graph modeling. A graph is built where each node represents a message, and edges represent the flow of conversation. Sentiments are assigned to each node, and the overall conversation sentiment is predicted by aggregating the sentiments from all nodes.

To implement this:

- **Graph Construction:** Each user's message is a node in the graph. Directed edges are created between consecutive messages.
- **Sentiment Prediction:** For each node (message), the Bi-LSTM model predicts the sentiment (positive, negative, neutral).
- **Aggregation:** The graph model aggregates sentiments to determine the overall conversation sentiment.

Cypher

```
MERGE (u:User {session_id:SESSION_ID}) -[:NEXT]->(c:Conversation {message:MESSAGE})
```

```
def get_memory(session_id):  
  
    return Neo4jChatMessageHistory(session_id=session_id, graph=graph)  
  
chat_chain = prompt | chat_llm | StrOutputParser()  
  
chat_with_message_history = RunnableWithMessageHistory(  
  
    chat_chain,  
  
    get_memory,  
  
    input_messages_key="question",  
  
    history_messages_key="chat_history",
```

```
)

list_of_emotions=[]

while True:

    question = input(">> ")

    list_of_emotions.append(question)

    response = chat_with_message_history.invoke(

        {

            "question":question

        },

        config={

            "configurable":{"session_id":SESSION_ID}

        }

    )

    if "bye" in question.lower():

        break

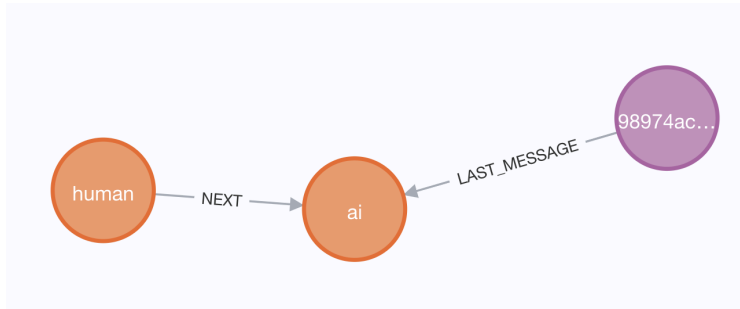
    print(response)

output = pipe(list_of_emotions)

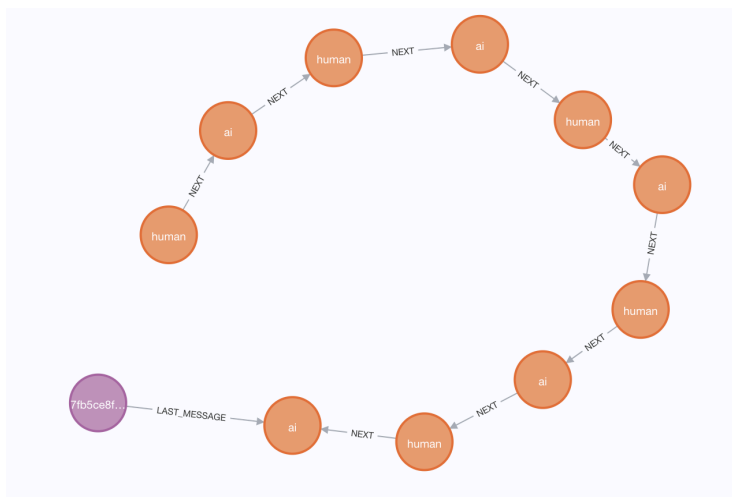
for option in output:

    print(option[0],option[1])
```

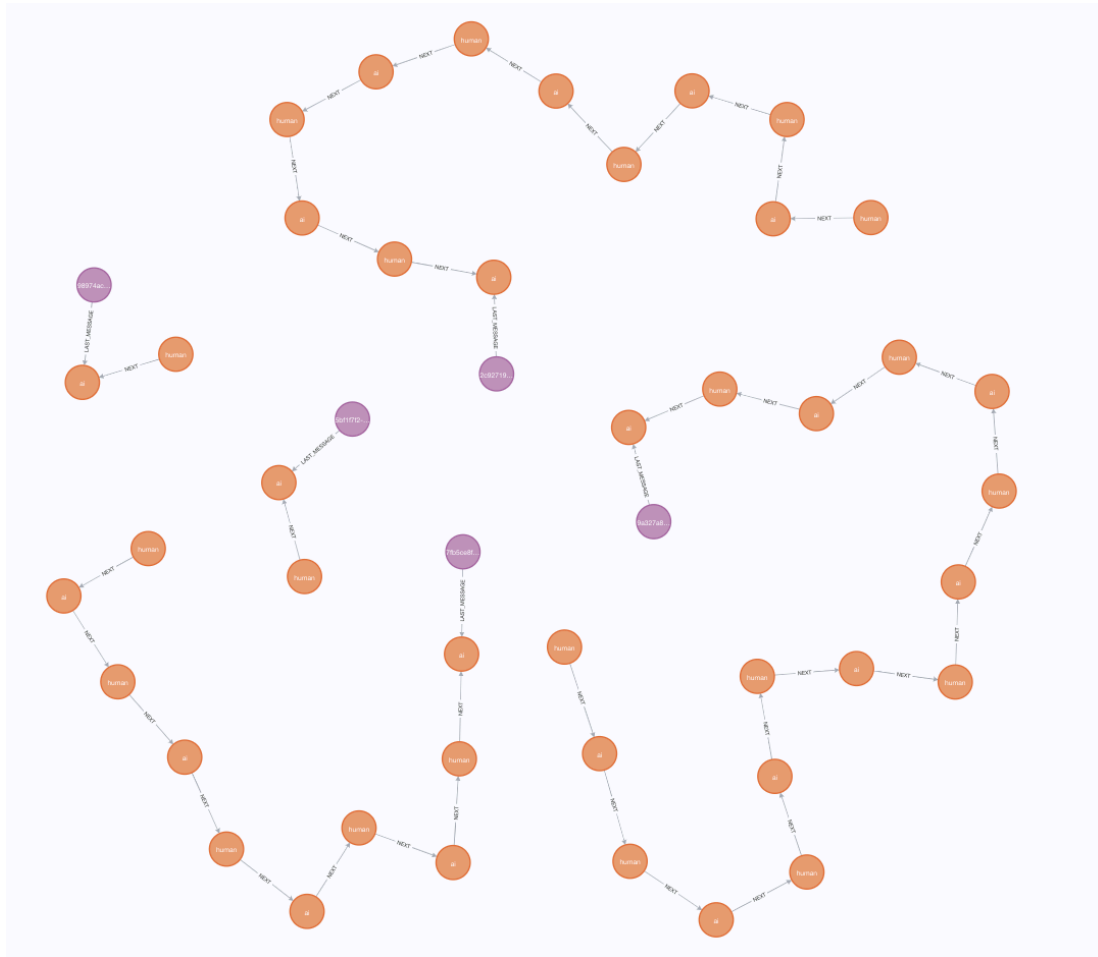
Graph: Sample Architecture



Conversation Chain



Knowledge Graph (of all conversations)



Overview

Node labels

* (47) Session (5) Message (42)

Relationship types

* (42) LAST_MESSAGE (5) NEXT (37)

Final Output

Chat - 1

Session ID: 2c927195-553f-4c01-b902-a2d4bba2c592

>> I feel stressed

I see, can you tell me more about what has been causing you stress lately? How have you been coping with it?

>> too many things on my plate i don't know what to do

It sounds like you have a lot going on. Have you been feeling overwhelmed by all the tasks and responsibilities? How have you been managing your time and priorities?

>> i am procrastinating a lot

I understand that procrastination can be a common response to feeling overwhelmed. Can you tell me more about what tasks you are procrastinating on and why you think you are avoiding them?

>> it's too confusing

I hear you, it can be difficult to navigate through confusing situations. Can you give me some specific examples of tasks or decisions that have been confusing for you? How do you typically approach these situations?

>> i feel like throwing away everything

It sounds like you are feeling a lot of frustration and possibly even a sense of hopelessness. Have you been experiencing any other emotions such as sadness, anger, or anxiety? Have you ever felt like this before, or is this a new experience for you?

>> ok i will talk, bye

```
WARNING:stdout
{'label': 'sadness', 'score': 0.32754749059677124} {'label': 'nervousness', 'score': 0.2918572723865509}
-----
{'label': 'confusion', 'score': 0.7369029521942139} {'label': 'disappointment', 'score': 0.10574560612440109}
-----
{'label': 'disappointment', 'score': 0.4744361340999603} {'label': 'neutral', 'score': 0.29723626375198364}
-----
{'label': 'confusion', 'score': 0.9376742243766785} {'label': 'neutral', 'score': 0.06897132098674774}
-----
{'label': 'sadness', 'score': 0.39953210949897766} {'label': 'annoyance', 'score': 0.1750536859035492}
-----
{'label': 'neutral', 'score': 0.7712337970733643} {'label': 'approval', 'score': 0.27930164337158203}
-----
```

Chat - 2

Session ID: 97b1c206-9615-4be6-b9bc-28a6428c999f

Hi, How are u feeling today?

User >> I dont know, things are not going well, its so saddening

Bot: >> I understand that you're feeling down. Can you tell me more about what's been going on? How long have you been feeling this way?

User >> Its the weather, I miss Bangalore and my home

Bot: >> It sounds like you're feeling homesick and the change in environment is affecting you. Can you share what specifically you miss about Bangalore and how it's impacting your mood?

User >> Bangalore weather is so perfect, here water and climate is affecting my health

Bot: >> It must be challenging to adjust to a different climate and environment. How is the change in weather impacting your health specifically? Have you noticed any physical symptoms or changes in your mood since moving?

User >> Yes I am trying to cope up. thanks, bye

Bot: >> I'm glad to hear that you're trying to cope. Remember, it's important to take care of your mental health as well. If you ever need to talk or need support, don't hesitate to reach out. Take care. Goodbye.

User >> bye

```
{'label': 'sadness', 'score': 0.8781848549842834} {'label': 'disappointment', 'score': 0.13053037226200104}
```

```
{'label': 'sadness', 'score': 0.875616729259491} {'label': 'disappointment', 'score': 0.17501196265220642}
```

```
{'label': 'admiration', 'score': 0.26267021894454956} {'label': 'approval', 'score': 0.17549003660678864}
```

```
{'label': 'gratitude', 'score': 0.9813748598098755} {'label': 'approval', 'score': 0.11549076437950134}
```

```
{'label': 'neutral', 'score': 0.8250218629837036} {'label': 'sadness', 'score': 0.2513568103313446}
```

Conclusion

This project highlights the effectiveness of combining deep learning and natural language processing for mental health assessment. By leveraging a Bidirectional LSTM model, GPT-4, and graph-based sentiment analysis, the system provides accurate sentiment classification and a supportive conversational platform. SMOTE oversampling addressed class imbalance, ensuring fair model training and reliable predictions. The chatbot creates a safe, judgment-free environment, aiding in early mental health interventions.

Future Work

1. **Enhancing Explainability:** Develop methods to make model predictions more interpretable.
2. **Multilingual Support:** Extend capabilities to analyze and interact in multiple languages.

References

Sl. No.	Title	Link
1.	An Overview of the Application of Sentiment Analysis in Mental Well-being	https://www.researchgate.net/publication/372822997_An_Overview_of_the_Application_of_Sentiment_Analysis_in_Mental_Well-being
2.	Mental Health Status Detection Model Based on LSTM Neural Network	https://www.sciencedirect.com/science/article/pii/S1877050924021082
3.	Artificial intelligence based social robots in the process of student mental health diagnosis	https://www.sciencedirect.com/science/article/pii/S1875952124001678
4.	WellXplain: Wellness concept extraction and classification in Reddit posts for mental health analysis	https://www.sciencedirect.com/science/article/pii/S0950705123009784
5.	A comprehensive review of predictive analytics models for mental illness using machine learning algorithms	https://www.sciencedirect.com/science/article/pii/S2772442524000522