Report

1. INTRODUCTION

Mental health has emerged as a critical concern in today's society, with increasing cases of anxiety, depression, and emotional instability due to stressful lifestyles, social isolation, and digital fatigue. Many individuals face emotional challenges daily but hesitate to consult therapists due to the stigma attached to mental illness, high costs, or limited accessibility to qualified professionals. In this context, there is an urgent need for an intelligent, accessible, and empathetic digital platform that can provide emotional support and therapeutic suggestions to users—instantly, affordably, and anonymously.

Therapy Connect is an AI-powered virtual mental health assistant designed to understand users' emotions by analyzing their spoken thoughts and daily experiences. The system listens to a user's voice through a microphone input, transcribes the speech into text, identifies the underlying emotion, and provides therapy-aligned recommendations drawn from trusted psychological resources. This creates an experience that mimics a conversation with a therapist while maintaining user privacy and emotional safety.

The platform detects six core emotional states—fear, sadness, love, joy, surprise, and anger—based on the user's spoken input. The process begins with Whisper, a robust automatic speech recognition (ASR) model developed by OpenAI. Whisper transcribes audio into high-quality text, even in the presence of background noise or accents. The transcribed text is then passed into a BiLSTM (Bidirectional Long Short-Term Memory) model trained to classify the input into one of the six emotion categories. BiLSTM, a type of recurrent neural network (RNN), is especially suitable for capturing the context and emotional tone of a sequence of words, both forward and backward.

Once the emotion is classified, the system invokes a Retrieval-Augmented Generation (RAG) pipeline to generate personalized therapy suggestions. This is accomplished using:

- MiniLM-L6-v2, a lightweight transformer-based model from Hugging Face, which creates sentence embeddings for semantic search.

- A ChromaDB (SQLite) vector store that holds embedded chunks of text (chunk size: 1000, overlap: 150) from a curated knowledge base of 31 psychology and therapy-related books.

- Gemini 2.5 Pro, an advanced generative model used to retrieve relevant passages and generate emotionally-aware and context-sensitive feedback for the user, including therapeutic suggestions and follow-up questions.

The user interface is built using Flask (Python) with HTML and CSS for front-end design, and a FastAPI backend powers the ASR service integration. The result is a lightweight, fully integrated web-based application that empowers users to share their thoughts and receive immediate, AI-driven emotional insights.

By combining state-of-the-art models in NLP and emotion detection with a carefully curated psychological knowledge base, Therapy Connect stands as a unique, scalable, and impactful solution for providing first-line emotional support.

2. PROBLEM STATEMENT

Mental health resources are either scarce, expensive, or not easily accessible to a large segment of the population. While therapy is essential, not everyone can access a human therapist at all times. There's a need for an intelligent, interactive system that can listen to users, understand their emotional state, and suggest helpful strategies derived from established psychological literature.

Key goals:

- Accurately detect emotions such as fear, sadness, love, joy, surprise, and anger from user input.
- Generate therapy-oriented feedback and recommendations.
- Utilize real psychological knowledge as a foundation for all suggestions.

3. LITERATURE SURVEY

3.1. Introduction

Emotion detection and intelligent therapy response generation are active research areas in the field of artificial intelligence and mental health informatics. Natural Language Processing (NLP) has significantly evolved in the past decade, allowing for context-aware understanding of human emotions and psychological states.

Recent advances in speech-to-text systems (like OpenAI's Whisper), sequence models (like BiLSTM), and language models (like Gemini, MiniLM) offer unprecedented capabilities for emotional inference and contextual recommendation systems. By combining these techniques with Retrieval-Augmented Generation (RAG), it becomes possible to create a virtual therapist that is both intelligent and grounded in authentic psychological knowledge.

3.2. Existing Methods

Several applications have been developed in the field of AI-powered emotional wellness and mental health support. These systems aim to simulate the therapist experience or provide mental well-being tools, but most come with certain limitations. Below is a review of key existing applications:

- **Woebot -** Woebot is an AI-driven mental health chatbot designed to deliver Cognitive Behavioral Therapy (CBT) techniques through casual conversation. It checks in with users daily and helps them reflect on their thoughts using structured CBT methods. While useful for offering guided self-help, Woebot is primarily rule-based and scripted. It lacks the ability to deeply personalize responses or generate novel suggestions based on individual emotional nuances, which limits its adaptability in real-time emotional scenarios.

- **Wysa -** Wysa is another popular AI mental health app that incorporates techniques from CBT, Dialectical Behavior Therapy (DBT), and meditation. It uses a text-based chatbot to help users manage stress, anxiety, and depression. While Wysa is praised for its therapeutic exercises and mental health journaling features, it does not accept voice input or detect emotions from speech. The interaction remains limited to user-typed messages, reducing natural communication and emotional context capture.

- **Replika -** Replika is a personal AI companion that builds emotional connections with users through continuous conversations. It adapts to the user’s communication style and learns over time to mimic human-like interactions. However, Replika focuses more on companionship and self-expression rather than structured therapy. It does not provide evidence-based therapeutic guidance and is not grounded in professional psychological frameworks or literature.

- **Tess -** Tess is an emotional wellness chatbot built for enterprises and healthcare institutions. It delivers supportive conversations via text messaging and uses NLP to respond empathetically. Tess is known for its scalability and integration with healthcare services. However, it lacks individual customization at the user level and is generally used in pre-programmed, large-scale implementations. Real-time emotion recognition from audio or voice is not part of its core functionality.

4. **LIBRARIES USED**

1. pandas

pandas is a powerful Python library used for data manipulation and analysis. It provides flexible data structures like DataFrames and Series to efficiently handle structured data. In this project, pandas is used for reading, cleaning, and preparing emotion-labeled datasets. It also helps in handling user-generated logs or reports. With built-in functions for filtering, grouping, and aggregation, it simplifies preprocessing tasks.

2. numpy

numpy (Numerical Python) is used for handling numerical data, arrays, and mathematical computations. It forms the base of most machine learning and deep learning frameworks. In Therapy Connect, numpy is helpful for working with text embeddings, matrix operations, and model inputs/outputs. It ensures efficient computation with minimal memory usage, especially when handling large-scale vector data.

3. re

The re module provides regular expression operations for pattern matching in strings. It is essential for text preprocessing such as removing special characters, punctuation, or tokenizing custom text formats. In this project, it's used to clean transcribed speech data before emotion classification. Regular expressions also help normalize text for better NLP performance.

4. nltk

nltk (Natural Language Toolkit) is a suite of libraries for text processing and linguistic analysis. It includes tokenizers, stemmers, stopword lists, and POS taggers. In Therapy Connect, nltk is used for preprocessing text data—removing stopwords, lemmatization, and preparing text for emotion classification. It enhances the linguistic quality of input to the BiLSTM model.

5. tensorflow

tensorflow is an open-source deep learning framework developed by Google. It supports training and deploying machine learning models on various platforms. In this project, the BiLSTM emotion classifier is built and trained using TensorFlow. It provides support for sequence models, GPU acceleration, and model saving/loading during production.

6. scikit-learn

scikit-learn is a machine learning library that provides tools for classification, regression, clustering, and preprocessing. It is used in Therapy Connect for label encoding, data splitting, evaluation metrics, and possibly baseline model comparisons. The library is also used to measure performance using metrics like accuracy, precision, recall, and F1-score.

7. chromadb

chromadb is an open-source vector database used to store and retrieve high-dimensional embeddings for semantic search. In this project, embeddings of psychology book chunks are stored in ChromaDB (backed by SQLite). It enables fast retrieval of contextually relevant content during RAG-based question answering. It’s crucial for implementing a scalable, efficient knowledge base.

8. sentence-transformers
sentence-transformers provides pretrained models (like MiniLM-L6-v2) that generate dense vector embeddings of text for semantic similarity tasks. Here, it’s used to embed chunks from psychology books and user queries. These embeddings are then stored in ChromaDB for similarity-based retrieval. It enhances the understanding of user input beyond surface-level text.

9. google-generativeai
This library allows access to Gemini 2.5 Pro, a generative large language model from Google. It’s used for retrieval-augmented generation (RAG) — generating therapy suggestions, follow-up questions, and personalized insights based on user emotions and retrieved knowledge. It plays a key role in delivering human-like, emotionally intelligent responses.

10. google-colab
google-colab is a cloud-based Python environment that allows execution of notebooks with free GPU support. During the development phase, it is used for training the BiLSTM model, running Whisper inference, experimenting with embeddings, and testing the entire pipeline. Colab facilitates collaboration, experimentation, and resource-heavy computation without local setup issues.

5. ALTERNATE MODELS AND THEIR DISADVANTAGES

While building Therapy Connect, several alternative models were considered for both emotion classification and response generation. These models offered certain strengths but also came with limitations that made them less suitable for our use case. Below is a comparative analysis of these alternatives and the reasons for not choosing them:

1. **LSTM (Long Short-Term Memory) -** LSTM is a type of Recurrent Neural Network that processes data sequentially and is designed to remember long-term dependencies in text.

Disadvantages:
- Processes input only in one direction, losing future context.
- Slower training and convergence.
- Limited in handling bi-directional emotional context.

2. **BERT (Bidirectional Encoder Representations from Transformers) -** BERT is a transformer-based model pretrained on large corpora to understand deep contextual relationships between words.

Disadvantages:
- Very resource-intensive and slow for real-time processing.
- Difficult to fine-tune for small custom emotion datasets.
- Introduces latency in live applications.

3. **Regression-Based Models -** Linear or logistic regression models treat text features numerically and attempt to map them to output classes.

Disadvantages:
- Poor at handling sequential and contextual information.
- Ineffective for unstructured, natural language data.
- Cannot capture emotions embedded in sentence structure.

4. **CNN for Text Processing -** Convolutional Neural Networks can be applied to text by treating sentences as 1D sequences to capture important n-gram features.

Disadvantages:
- Ignore word order and long-term dependencies.
- Not suitable for detecting emotional progression in sentences.
- Performance degrades with variable-length or spoken-text input.

5. **GRU (Gated Recurrent Unit) -** GRU is a simplified version of LSTM that uses fewer parameters to model sequences.

Disadvantages:

- Lacks inherent bidirectional processing.
- Slightly less expressive than LSTM for complex tasks.
- May miss critical context without additional architectural support.

6. GRU + GloVe Embeddings - This combines GRU with pretrained GloVe embeddings to represent word semantics.

Disadvantages:

- GloVe embeddings are static and do not adapt to context.
- The combination still fails to capture bidirectional context unless explicitly added.
- Less effective with nuanced emotional expression.

7. BiLSTM (Bidirectional Long Short-Term Memory Network) -

Understanding human emotion from text—especially spoken, unstructured text—requires capturing the meaning, tone, and context of entire sentences or paragraphs. Words often gain their emotional meaning not just from their direct meaning but from how they relate to preceding and succeeding words. Traditional models like unidirectional LSTM or rule-based classifiers fall short in capturing these nuanced relationships.

To address this, we chose BiLSTM (Bidirectional Long Short-Term Memory) — a powerful deep learning architecture specifically designed to understand sequences from both directions (past to future and future to past). This makes BiLSTM particularly well-suited for emotion detection, where a word's emotional weight can be influenced by what comes before and after it.

Advantages of Using BiLSTM

1. Context From Both Directions
BiLSTM processes text from start to end and from end to start, which helps it understand the full meaning of a sentence. For example:

- In the sentence *"I am not happy with today,"* the word *"not"* reverses the emotion of *"happy"*. A bidirectional model captures that relationship better.

2. Better Emotion Detection in Complex Sentences
Emotion often depends on subtle cues spread throughout a sentence. BiLSTM can handle:

- Sarcasm (e.g., *"Oh great, another Monday."*)
- Mixed emotions (e.g., *"I'm excited but nervous."*)
- Contrast and negation (e.g., *"I thought it would be joyful, but it was disappointing."*)

3. Improved Performance Over LSTM and CNN
BiLSTM generally outperforms LSTM and CNN in sentiment/emotion classification because:

- LSTM only sees past context (unidirectional).
- CNNs focus on local patterns but miss long-distance dependencies.

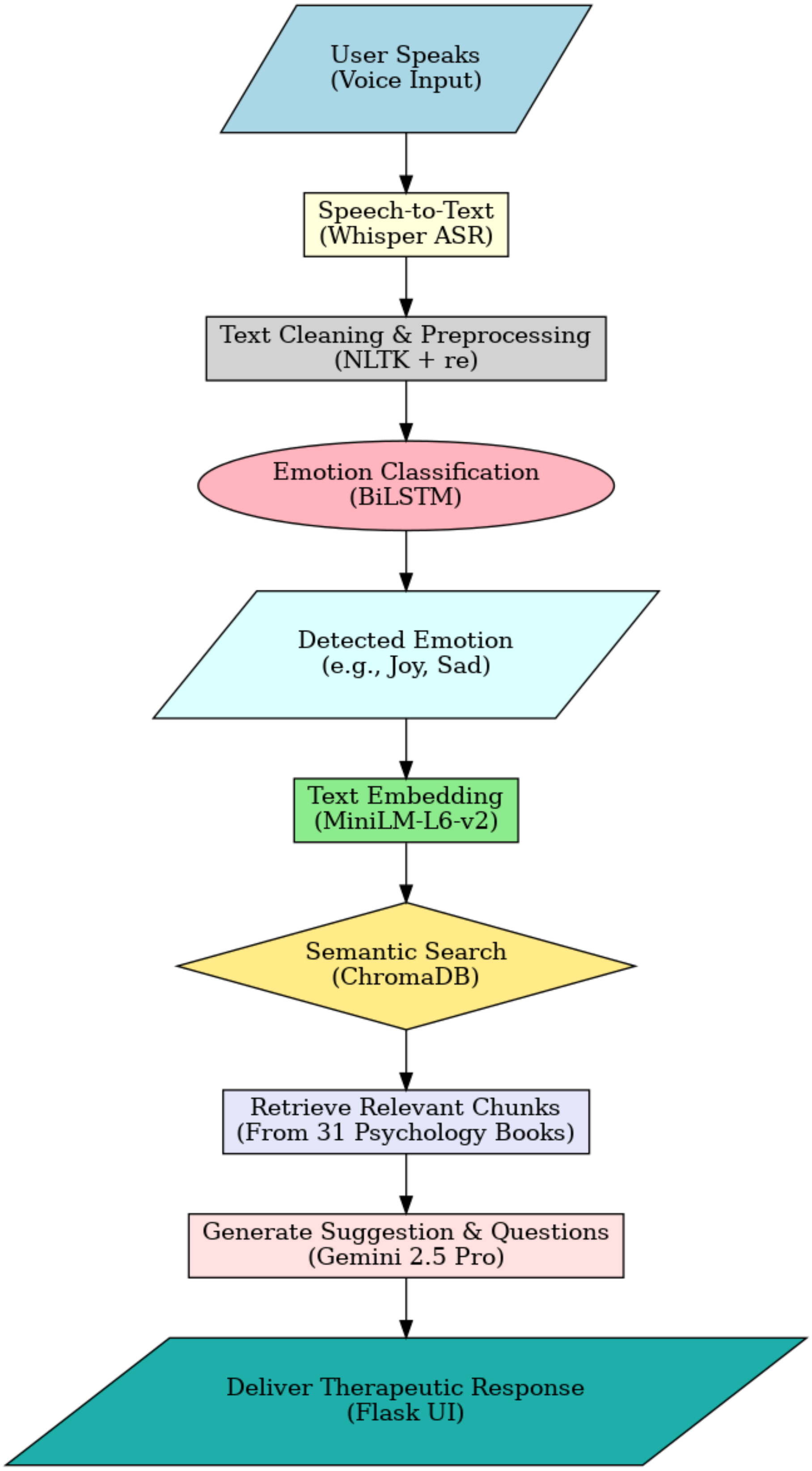
- BiLSTM leverages both and performs better on longer sequences.

4. Efficient for Medium-Sized Datasets
Unlike transformer-based models (BERT, GPT), BiLSTM doesn’t require extremely large datasets or massive hardware. It gives strong performance with fewer resources, making it perfect for real-time or lightweight applications like Therapy Connect.

5. Easily Integrated With Other Layers
BiLSTM works well when combined with:

- Word embeddings (like GloVe, Word2Vec)
- Attention mechanisms (for future upgrades)
- Dense layers for multi-class classification (like your 6 emotion classes)

6. FLOWCHART


User Speaks
(Voice Input)
Speech-to-Text
(Whisper ASR)
Text Cleaning & Preprocessing
(NLTK + re)
Emotion Classification
(BiLSTM)
Detected Emotion
(e.g., Joy, Sad)
Text Embedding
(MiniLM-L6-v2)
Semantic Search
(ChromaDB)
Retrieve Relevant Chunks
(From 31 Psychology Books)
Generate Suggestion & Questions
(Gemini 2.5 Pro)
Deliver Therapeutic Response
(Flask UI)

7. PROJECT ARCHITECTURE

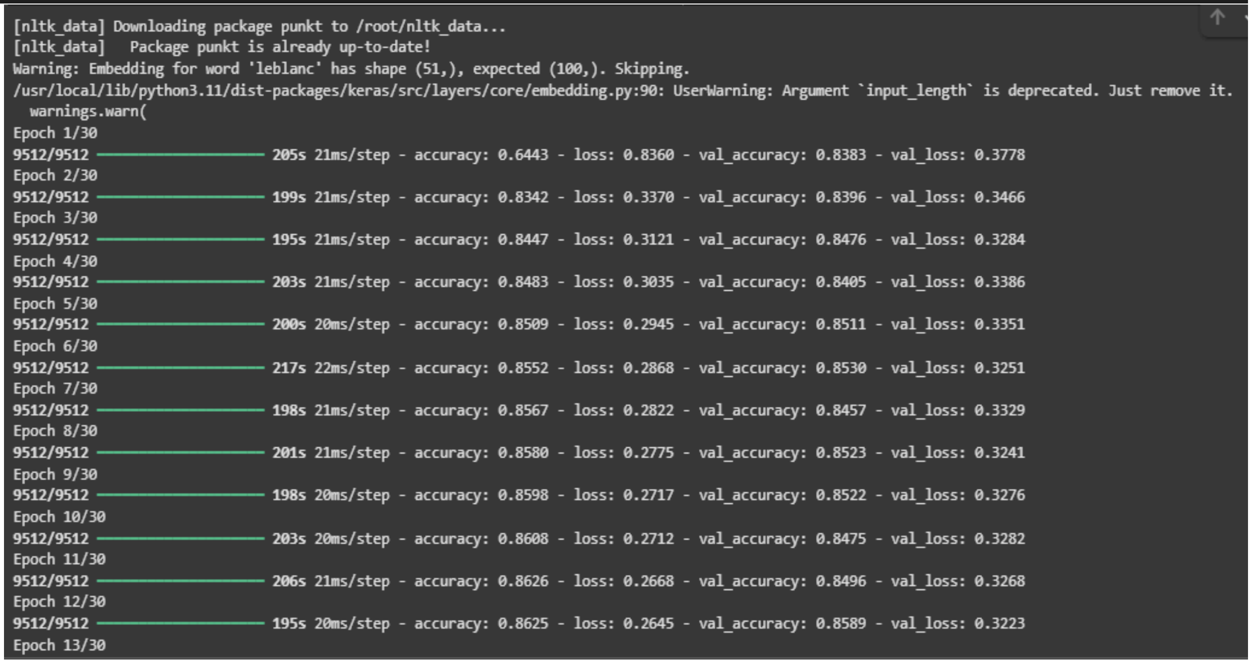
8. VISUALIZATIONS

9. MODELING (BiLSTM)

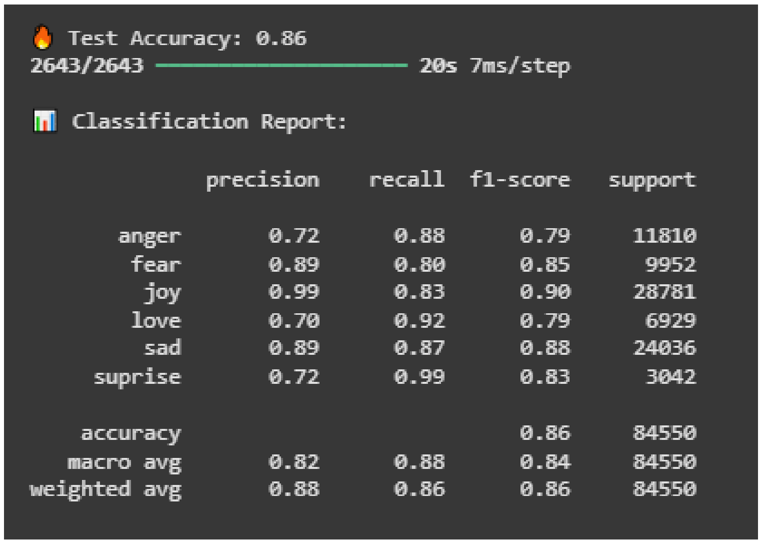
```
# 🌟 STEP 1: IMPORTS
import pandas as pd
import numpy as np
import re
import nltk
nltk.download('punkt')

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import classification_report

# 🌟 STEP 2: LOAD DATA
df = pd.read_csv('/content/combined_emotion.csv')
df.columns = df.columns.str.strip()  # Clean column names
texts = df['sentence'].astype(str).values
labels = df['emotion'].values

# 🌟 STEP 3: CLEAN TEXT
def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^a-zA-Z0-9\s]", "", text)
    return text

texts = [clean_text(t) for t in texts]

# 🌟 STEP 4: LABEL ENCODING
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(labels)
y_cat = to_categorical(y)
```

```
# 🌟 STEP 5: TOKENIZATION & PADDING
max_words = 20000
max_len = 100

tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
X = pad_sequences(sequences, maxlen=max_len)

# 🌟 STEP 6: TRAIN-TEST SPLIT
X_train, X_test, y_train, y_test = train_test_split(X, y_cat, test_size=0.2, random_state=42)

# 🌟 STEP 7: LOAD GLOVE EMBEDDINGS
embedding_index = {}
embedding_dim = 100 # Define embedding_dim here
with open('/content/glove.6B.100d.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        if len(values) > 1:  # Check if there are values after the word
            try:
                coefs = np.asarray(values[1:], dtype='float32')
                    if coefs.shape[0] == embedding_dim: # Check if the shape of the embedding_vector matches embedding_dim
                    embedding_index[word] = coefs
                else:
                        print(f"Warning: Embedding for word '{word}' has shape {coefs.shape}, expected ({embedding_dim},). Skipping.")

            except ValueError:
                print(f"Skipping line due to invalid float conversion: {line.strip()}")
        else:
            print(f"Skipping line with only a word: {line.strip()}")



embedding_matrix = np.zeros((max_words, embedding_dim))
for word, i in tokenizer.word_index.items():
    if i < max_words:
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector


# 🌟 STEP 8: COMPUTE CLASS WEIGHTS FOR IMBALANCE
y_train_labels = np.argmax(y_train, axis=1)
class_weights = compute_class_weight(class_weight='balanced',
classes=np.unique(y_train_labels), y=y_train_labels)
class_weights_dict = dict(enumerate(class_weights))

# 🌟 STEP 9: BUILD BiLSTM MODEL
model = Sequential([
    Embedding(max_words, embedding_dim, weights=[embedding_matrix],
input_length=max_len, trainable=False),
    Bidirectional(LSTM(128, return_sequences=True)),
    Dropout(0.5),
    Bidirectional(LSTM(64)),
    Dropout(0.4),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(y_cat.shape[1], activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)

# 🌟 STEP 10: TRAIN MODEL
history = model.fit(
    X_train, y_train,
    validation_split=0.1,
    epochs=30,
    batch_size=32,
    callbacks=[early_stop],
    class_weight=class_weights_dict,
    verbose=1
)
```

```
# 🌟 STEP 11: EVALUATE MODEL
loss, acc = model.evaluate(X_test, y_test)
print(f"\n🔥 Test Accuracy: {acc:.2f}")

# 🌟 STEP 12: CLASSIFICATION REPORT
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test, axis=1)

print("\n📊 Classification Report:\n")
print(classification_report(y_true, y_pred, target_names=label_encoder.classes_))
```

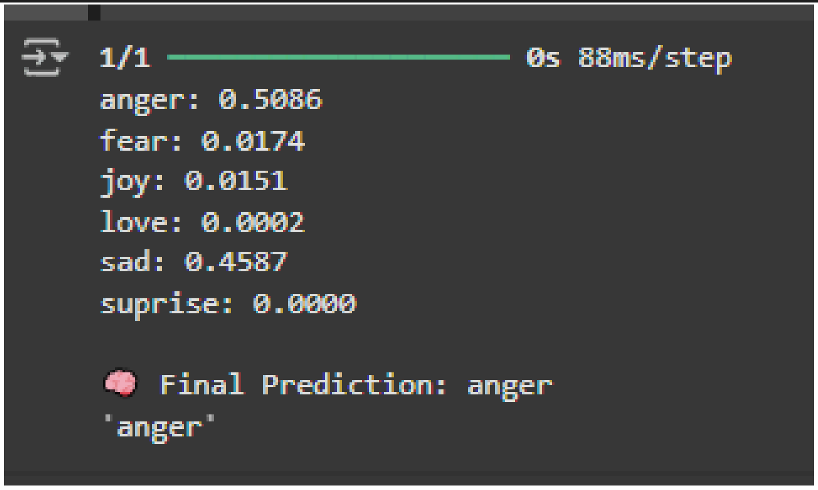


```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Warning: Embedding for word 'leblanc' has shape (51,), expected (100,). Skipping.
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Epoch 1/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 205s 21ms/step - accuracy: 0.6443 - loss: 0.8360 - val_accuracy: 0.8383 - val_loss: 0.3778
Epoch 2/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 199s 21ms/step - accuracy: 0.8342 - loss: 0.3370 - val_accuracy: 0.8396 - val_loss: 0.3466
Epoch 3/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 195s 21ms/step - accuracy: 0.8447 - loss: 0.3121 - val_accuracy: 0.8476 - val_loss: 0.3284
Epoch 4/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 203s 21ms/step - accuracy: 0.8483 - loss: 0.3035 - val_accuracy: 0.8405 - val_loss: 0.3386
Epoch 5/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 200s 20ms/step - accuracy: 0.8509 - loss: 0.2945 - val_accuracy: 0.8511 - val_loss: 0.3351
Epoch 6/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 217s 22ms/step - accuracy: 0.8552 - loss: 0.2868 - val_accuracy: 0.8530 - val_loss: 0.3251
Epoch 7/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 198s 21ms/step - accuracy: 0.8567 - loss: 0.2822 - val_accuracy: 0.8457 - val_loss: 0.3329
Epoch 8/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 201s 21ms/step - accuracy: 0.8580 - loss: 0.2775 - val_accuracy: 0.8523 - val_loss: 0.3241
Epoch 9/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 198s 20ms/step - accuracy: 0.8598 - loss: 0.2717 - val_accuracy: 0.8522 - val_loss: 0.3276
Epoch 10/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 203s 20ms/step - accuracy: 0.8608 - loss: 0.2712 - val_accuracy: 0.8475 - val_loss: 0.3282
Epoch 11/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 206s 21ms/step - accuracy: 0.8626 - loss: 0.2668 - val_accuracy: 0.8496 - val_loss: 0.3268
Epoch 12/30
9512/9512 ━━━━━━━━━━━━━━━━━━━━ 195s 20ms/step - accuracy: 0.8625 - loss: 0.2645 - val_accuracy: 0.8589 - val_loss: 0.3223
Epoch 13/30
```



```
🔥 Test Accuracy: 0.86
2643/2643 ━━━━━━━━━━━━━━━━━━━━ 20s 7ms/step

📊 Classification Report:

              precision    recall  f1-score   support

       anger       0.72      0.88      0.79     11810
        fear       0.89      0.80      0.85      9952
         joy       0.99      0.83      0.90     28781
        love       0.70      0.92      0.79      6929
         sad       0.89      0.87      0.88     24036
     suprise       0.72      0.99      0.83      3042

    accuracy                           0.86     84550
   macro avg       0.82      0.88      0.84     84550
weighted avg       0.88      0.86      0.86     84550
```

```
# 🌟 STEP 13: CUSTOM INPUT + CONFIDENCE
def predict_emotion(text):
    text_clean = clean_text(text)
    seq = tokenizer.texts_to_sequences([text_clean])
    padded = pad_sequences(seq, maxlen=max_len)
    probs = model.predict(padded)[0]
    for i, score in enumerate(probs):
        print(f"{label_encoder.classes_[i]}: {score:.4f}")
    predicted_label = label_encoder.classes_[np.argmax(probs)]
    print(f"\n🧠 Final Prediction: {predicted_label}")
    return predicted_label

# 🧪 TEST CUSTOM TEXT
sample_text = """I feel a little mellow today Honestly, it's just one of those days where everything feels a bit more intense than usual. I've been thinking about it a lot, and emotions like this don't just come out of nowhere. The words I say might sound simple, but there's always something deeper going on beneath them. Sometimes I wish things could just be a bit easier, but this is what it is right now. I'm trying to cope, understand, and move forward. It's like my heart and my thoughts are in a constant conversation, never settling. Maybe it'll pass, or maybe"""
predict_emotion(sample_text)
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
anger: 0.5086
fear: 0.0174
joy: 0.0151
love: 0.0002
sad: 0.4587
suprise: 0.0000

🧠 Final Prediction: anger
'anger'
```

```
import pandas as pd
import numpy as np
import re
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder

# ✅ 1. Load your custom test data
```

```python
custom_df = pd.read_excel('/content/actual_emotion_dataset_nopercent.xlsx')
custom_df.columns = custom_df.columns.str.strip()
custom_texts = custom_df['text'].astype(str).values # Changed 'sentence' to 'text'

# ✅ 2. Clean the text (reuse same function)
def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^a-zA-Z0-9\s]", "", text)
    return text

cleaned_custom_texts = [clean_text(t) for t in custom_texts]

# ✅ 3. Tokenize and pad using existing tokenizer and max_len
custom_sequences = tokenizer.texts_to_sequences(cleaned_custom_texts)
custom_padded = pad_sequences(custom_sequences, maxlen=max_len)

# ✅ 4. Predict mood + confidence score
custom_preds_probs = model.predict(custom_padded)
custom_preds = np.argmax(custom_preds_probs, axis=1)
custom_confidences = np.max(custom_preds_probs, axis=1)

# ✅ 5. Decode predicted labels
decoded_preds = label_encoder.inverse_transform(custom_preds)

# ✅ 6. Create final results dataframe
results_df = pd.DataFrame({
    'Sentence': custom_texts,
    'Predicted Emotion': decoded_preds,
    'Confidence Score': custom_confidences
})

# ✅ 7. Save or display
results_df.to_csv('predicted_emotions_output.csv', index=False)
results_df.head(10)  # Show top 10 results
# Save the Keras model in .h5 format
model.save("BiLSTM_model.h5")

import os
```

```python
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'  # Suppress TensorFlow INFO and WARNING logs

from tensorflow.keras.models import save_model
model.save("my_model1.h5")

import pickle

# Save your tokenizer object
with open("final_tokenizer.pkl", "wb") as f:
    pickle.dump(tokenizer, f)

print("✅ Tokenizer saved as final_tokenizer.pkl")
```

10 . APP CREATION

The front-end and user interface (UI) for Therapy Connect have been designed to offer a smooth, interactive, and emotionally sensitive experience. The application enables users to speak freely about their thoughts and feelings, while the system intelligently detects and analyzes their emotional state in real time. The web application is built using a combination of modern web development technologies: Flask, HTML, CSS, and JavaScript.

Key Components:

1. Front-End Technologies

- HTML & CSS: These were used to design and style all the web pages, ensuring that the interface is clean, responsive, and accessible. Special attention was paid to color schemes and layout to create a calming and welcoming atmosphere for users dealing with emotional concerns.

- JavaScript: JavaScript handles dynamic behaviors such as voice recording, form validations, interactive elements (like start/stop buttons), and displaying emotion outputs in real-time without needing page reloads.

2. Flask Framework

- Flask is used as the backend web framework. It connects the front-end interface with the machine learning models and the logic behind speech-to-text conversion, emotion classification, and suggestion generation.

- API endpoints are defined in Flask to process the audio input, send it through Whisper for transcription, classify the resulting text using the BiLSTM model, and return the detected emotion to the front end.

- Flask also handles session management, authentication, and routing across different pages like the homepage, login, and dashboard.

3. Voice Input System

The homepage is designed with a voice recording feature where users can click a button and speak directly about their emotions, daily events, or any concern. JavaScript handles the voice recording on the client side, and the audio is sent to the backend (FastAPI or Flask endpoint), where it is processed through Whisper for speech-to-text transcription.

4. Emotion Display and Feedback

Once the system classifies the user’s emotion (e.g., joy, sadness, anger, etc.), the result is dynamically shown on the screen using JavaScript and updated DOM elements. This real-time feedback allows the user to get immediate insight into their emotional tone.

5. Login and Signup Pages

A simple yet secure authentication system is implemented using Flask:

- Users can register and log in through signup and login pages.
- Credentials are securely handled (can be extended with encryption like bcrypt in production).
- Each user can have a personalized dashboard where their past sessions (if saved), detected emotions, or generated therapy suggestions are visible.

6. Dashboard and Future Add-ons

- A dashboard interface can be added to display previously detected emotions, suggestions, and mental health tracking graphs.
- Future updates can include multilingual support, dark/light mode, chat-based interactions, and real therapist integrations.

11. CONCLUSION & FUTURE SCOPE

11.1. Conclusion

Therapy Connect is a pioneering AI-powered platform designed to assist individuals in recognizing and addressing their emotional well-being. By enabling users to speak freely and naturally, the system leverages advanced technologies such as Whisper for speech-to-text conversion, BiLSTM for emotion classification, and a Retrieval-Augmented Generation (RAG) pipeline with MiniLM, ChromaDB, and Gemini 2.5 Pro for therapy suggestions.

The application successfully captures six core emotional states—joy, sadness, fear, love, anger, and surprise—from spoken input and provides real-time, psychologically grounded support using knowledge extracted from 31 therapy and psychology books. The intuitive interface, developed with Flask, HTML, CSS, and JavaScript, makes the application easy to use, accessible, and impactful for users seeking emotional insight and support.

By integrating Natural Language Processing, speech recognition, and deep learning, Therapy Connect bridges the gap between emotional expression and intelligent, supportive feedback—offering users a first step toward self-awareness and healing.

11.2. Future Scope

While Therapy Connect already demonstrates a functional and impactful system, there are several directions in which it can be further enhanced:

1. Multilingual Support - Adding language detection and support for regional or global languages will increase accessibility for non-English speakers.

2. Emotion Trend Tracking - A dashboard feature can track emotional states over time, giving users visual insights into their mood patterns and mental health journey.

3. Real-Time Chat and Text Input - In addition to voice input, integrating a live chat interface can help users who prefer typing or cannot speak out loud.

4. Human-in-the-Loop Therapy - Incorporating certified therapists who can review and respond to flagged sessions for critical emotional states (e.g., extreme sadness, suicidal ideation).

5. Mobile App Development - A mobile version of Therapy Connect would allow users to carry emotional support tools in their pockets, improving convenience and engagement.

6. Gamification and CBT Modules - Adding guided therapy modules based on CBT, journaling prompts, or gamified progress tracking can increase user involvement and emotional literacy.

7. Integration with Wearables - Using data from smartwatches (e.g., heart rate, sleep, stress indicators) can enhance emotion detection accuracy and provide more holistic support.