PROJECT REPORT ON

Project Report: Emotional Tone Detector for Journaling Apps

SUBMITTED TO MIT SCHOOL OF COMPUTING, LONI, PUNE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

BACHELOR OF TECHNOLOGY

(INFORMATION TECHNOLOGY)

Ayush Athare	MITU23BTITD003
Karthik Reddy	MITU22BTIT0040
Mahima Kela	MITU22BTIT0044
Parth Datar	MITU22BTIT0056

Prof. Kalyani Lokhande

Abstract

The project “Simple Emotional Tone Detector for Journaling Apps” is a Python-based system that analyzes the emotional tone of journal entries. It detects whether a user's written content expresses positive, negative, or neutral emotions using a predefined list of emotional keywords. The system stores all entries in an SQLite database along with the detected tone, allowing users to track their emotional patterns over time. The project demonstrates how basic Natural Language Processing (NLP) and data storage techniques can be applied to enhance personal journaling and emotional awareness.

Aim

To design and develop a journaling application that automatically detects and categorizes the emotional tone of user entries as **Positive**, **Negative**, or **Neutral**, helping users gain insights into their emotional well-being.

Methodology

1. **Keyword-Based Sentiment Analysis:**
   - The system maintains two predefined lists: positive_words and negative_words.
   - Each journal entry is converted to lowercase, and the program counts occurrences of positive and negative words.
   - Based on these counts, the tone is classified as *Positive*, *Negative*, or *Neutral*.
2. **Database Implementation (SQLite):**
   - An SQLite database (journal_simple.db) stores all journal entries with:
     - Timestamp
     - Original text
     - Sanitized text
     - Detected emotional tone
3. **Data Sanitization:**
   - All entries are sanitized using HTML escaping to prevent script injection or data corruption.
4. **Interactive Journaling Interface:**
   - Users can type journal entries through a simple console interface.
   - The system provides instant feedback on the emotional tone and saves the entry to the database.

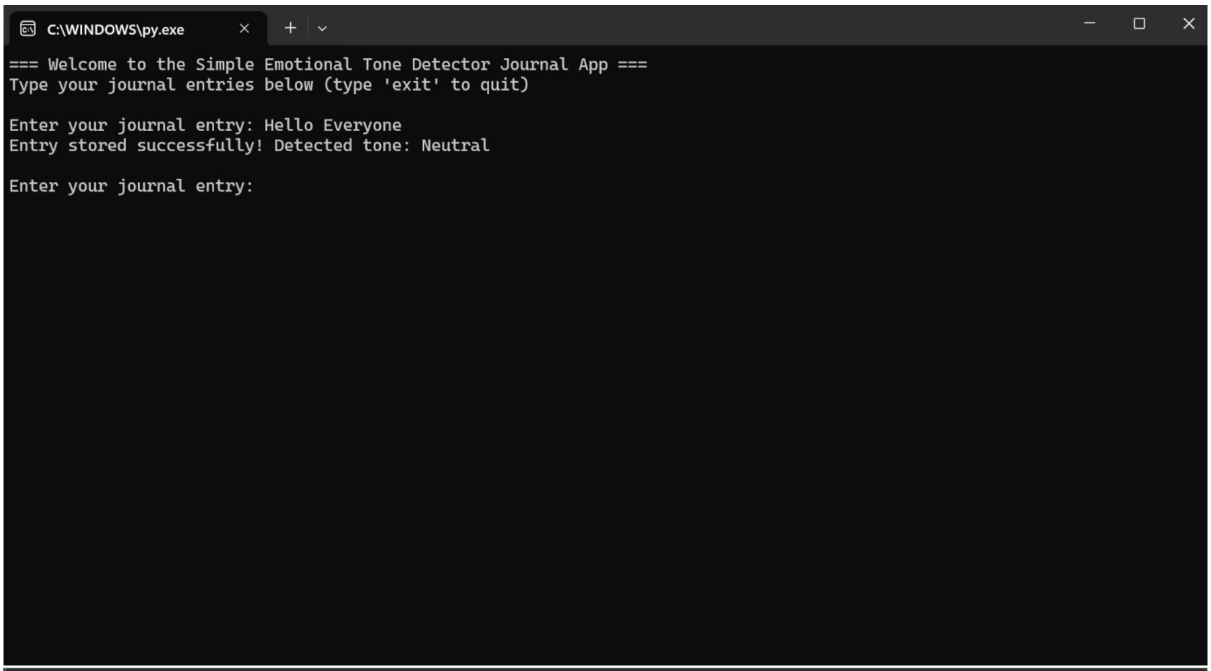
5. **Data Persistence:**
    - Each journal entry is timestamped and stored permanently, enabling emotional tracking over time.

Results

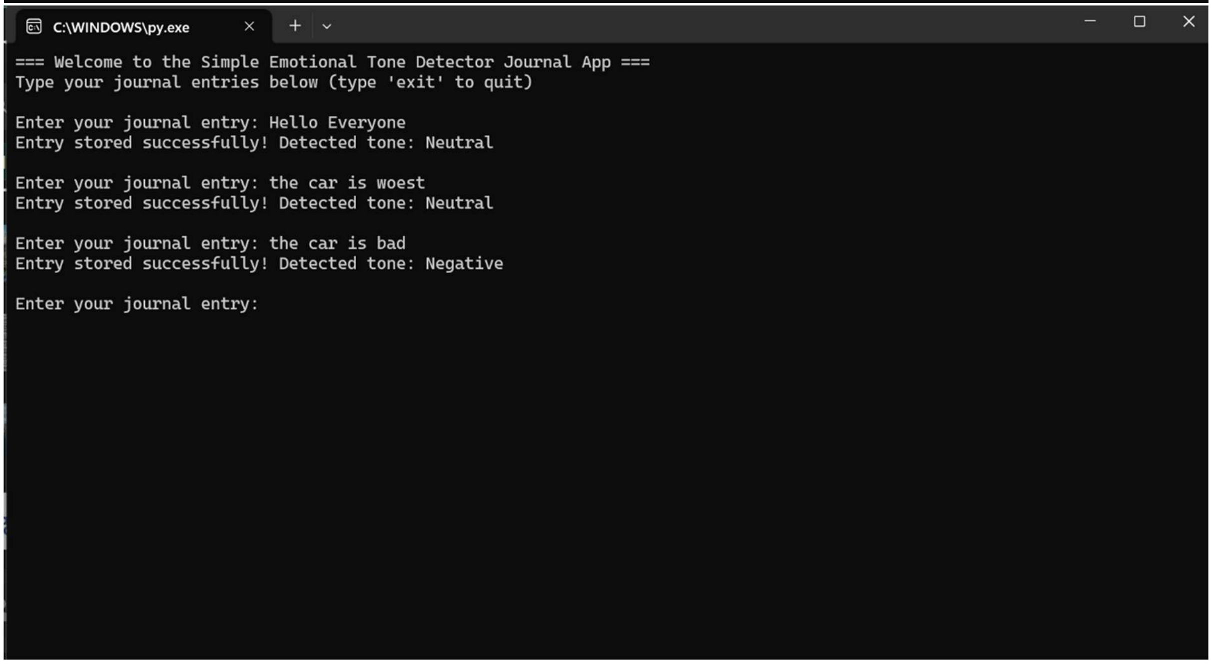
The system effectively:

- Categorized journal entries into *Positive*, *Negative*, or *Neutral* tones based on keyword analysis.
- Stored all entries with proper sanitization and timestamps in the SQLite database.
- Provided immediate emotional feedback for each journal input.
- Created a secure and structured journaling environment suitable for emotion tracking or self-reflection tools.

```
"""
Simple Emotional Tone Detector for Journaling Apps
"""

import sqlite3
import time
import html

# ---------- Configuration ----------
DB_FILE = "journal_simple.db"
MAX_ENTRY_LENGTH = 2000

# ---------- Database ----------
def init_db(conn):
    cur = conn.cursor()
    cur.execute("""
        CREATE TABLE IF NOT EXISTS journal (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            ts INTEGER NOT NULL,
            entry TEXT NOT NULL,
            sanitized TEXT NOT NULL,
            tone TEXT NOT NULL
        )
    """)
    conn.commit()

def store_entry(conn, entry, sanitized, tone):
    ts = int(time.time())
    cur = conn.cursor()
    cur.execute(
        "INSERT INTO journal (ts, entry, sanitized, tone) VALUES (?, ?, ?, ?)",
        (ts, entry, sanitized, tone)
    )
    conn.commit()


# ---------- Emotional Tone Analysis ----------
positive_words = ["happy", "joy", "love", "good", "great", "excited", "wonderful"]
negative_words = ["sad", "angry", "tired", "stress", "bad", "worried", "upset"]


def analyze_tone(entry):
    entry_lower = entry.lower()
    pos_count = sum(word in entry_lower for word in positive_words)
    neg_count = sum(word in entry_lower for word in negative_words)

    if pos_count > neg_count:
        return "Positive"
    elif neg_count > pos_count:
        return "Negative"
    else:
        return "Neutral"


def sanitize_for_display(text):
    return html.escape(text)


# ---------- Main Journal Loop ----------
def journaling_app():
    conn = sqlite3.connect(DB_FILE)
    init_db(conn)

    print("=== Welcome to the Simple Emotional Tone Detector Journal App ===")
    print("Type your journal entries below (type 'exit' to quit)\n")

    while True:
        entry = input("Enter your journal entry: ").strip()
        if entry.lower() == "exit":
            break
        if not entry:
            print("Empty entry ignored.")
            continue
        if len(entry) > MAX_ENTRY_LENGTH:
            print(f"Entry too long. Maximum {MAX_ENTRY_LENGTH} characters allowed.")
            continue

        tone = analyze_tone(entry)
        sanitized = sanitize_for_display(entry)

        store_entry(conn, entry, sanitized, tone)

        print(f"Entry stored successfully! Detected tone: {tone}\n")

    conn.close()
    print("Thank you for journaling. Goodbye!")


if __name__ == "__main__":
    journaling_app()
```



```
=== Welcome to the Simple Emotional Tone Detector Journal App ===
Type your journal entries below (type 'exit' to quit)

Enter your journal entry: Hello Everyone
Entry stored successfully! Detected tone: Neutral

Enter your journal entry:
```



```
=== Welcome to the Simple Emotional Tone Detector Journal App ===
Type your journal entries below (type 'exit' to quit)

Enter your journal entry: Hello Everyone
Entry stored successfully! Detected tone: Neutral

Enter your journal entry: the car is woest
Entry stored successfully! Detected tone: Neutral

Enter your journal entry: the car is bad
Entry stored successfully! Detected tone: Negative

Enter your journal entry:
```

Conclusion

The Simple Emotional Tone Detector successfully demonstrates the integration of natural language processing and data storage in a journaling context. By analyzing emotional tone using predefined word lists, the project offers an accessible way for users to reflect on their moods and mental well-being. It can serve as a foundational step toward more advanced emotional analytics and mental health applications.

Recommendations

- **Expand Word Lists:** Incorporate a larger and more context-aware emotional vocabulary for improved accuracy.
- **Machine Learning Integration:** Implement sentiment analysis using ML models (e.g., Naive Bayes, BERT) for deeper contextual understanding.
- **Graphical User Interface (GUI):** Develop a desktop or mobile interface for better user interaction.
- **Data Visualization:** Add charts to display mood trends over time for self-analysis.
- **Cloud Synchronization:** Store data securely online to allow users to access their journals from multiple devices.