Emotions and Sentiments Detection

مبادرة رواد مصر الرقمية من وزارة الاتصالات
وتكنولوجيا المعلومات لتعلم البرمجة

Group: HRV486A_DKH2_AIS4_G1_DEPI2

Project Team members

1-Ahmed Mohamed Ibrahim Wahba
2-Manar Abdelmaboud Hussein ElBeltagy

Project Content

Project Diagrams Content

Abstract:

Text Sentiment Analysis with Speech Recognition is a machine learning-based system designed to extract and analyze emotional sentiment from both written and spoken content. This project combines two powerful fields — Natural Language Processing (NLP) and Automatic Speech Recognition (ASR) — to provide a flexible tool capable of understanding human emotions from voice and text data.

The system first utilizes speech recognition to convert spoken language into text. Then, through a series of text preprocessing techniques including tokenization, stop-word removal, and lemmatization, the cleaned data is analyzed using sentiment classification models. These models predict whether the input expresses a positive, negative, or neutral sentiment (or binary positive/negative in advanced stages).

This project offers practical real-world applications in multiple sectors such as customer service, politics, social media monitoring, mental health support, and marketing analytics. In political environments, for example, the system can help detect public opinion trends about candidates, policies, or social events — even from public speeches, interviews, or online discussions. The goal is to assist organizations and individuals in better understanding human emotions across both written and spoken formats, improving decision-making and communication strategies.

1. Project Planning & Management

1.1 Project Proposal – Overview, Objectives, and Scope

Project Title:

Speech Recognition and Emotion Detection from YouTube Comments and Voice Input

Overview:

This project uses simple tools to turn speech into text and figure out if the text sounds positive or negative. We use voice recordings and YouTube comments to do this. The goal is to make a small system that can recognize emotions from what people say or write.

Objectives:

- Convert spoken words into written text using Python's Speech Recognition library.

- Collect and clean YouTube comments for training.

- Train a Naïve Bayes model using TF-IDF to detect if the comment or text is positive or negative.

- Build a simple system that brings everything together.

Scope:

- What’s included: turning voice into text, detecting if the text is positive or negative, and a simple interface.

- What's not included: multiple emotions (like anger or joy), or different languages.

1.2 Project Plan

- Week 1: Research and Planning
- Week 2: Collect data (comments and audio)
- Week 3: Clean data
- Week 4: Train model (TF-IDF + Naïve Bayes)
- Week 5: Connect speech recognition
- Week 6: Testing
- Week 7: Finish documentation and present

1.3 Task Assignment

- **Data Collection:**	**Ahmed**
- **Text Processing:**	Manar-Ahmed
- **Model Training:**	Ahmed-Ahmed
- **Testing:**	Ahmed
- **Writing Docs:**	Manar

1.4 Risk Assessment

Risk	Likelihood	Impact	Solution
Bad audio quality	Medium	High	Clean audio or reduce noise
Dataset not balanced	High	Medium	Try to balance or add more data
Medium	Medium	High	Adjust model and test more
Not enough time	High	High	Stick to schedule

1.5 KPIs (Key Performance Indicators)

- Model accuracy (> 85%)

- How well the Text is converted Binary language?

- How well the speech is converted?

- Time to give an answer

- How many voice inputs it handles

- Any user feedback (if available)

2. Literature Review

- **Badawood & Aldosari (2024)**

Study: Real-time multilingual emotion detection using CNN + Transformer
Weakness:
The model requires high computational resources, which limits its usability on lightweight systems like ours.

- **Alluhaidan et al. (2023)**

Study: Hybrid CNN-based features for speech emotion detection
Weakness:
Focuses only on speech input without covering textual sentiment, which makes it less applicable for comment analysis.

- **Mohamed & Aly (2021)**

Study: Emotion detection in Arabic speech using Wav2Vec2.0 and HuBERT
Weakness:
The model is language-specific (Arabic only), so it's not generalizable to English YouTube comments.

- **Alreshidi et al. (2021)**

Study: Speech-based emotion detection for mental health applications
Weakness:
While useful for health, it does not address real-time or large-scale informal comment analysis like ours.

- **Gulati et al. (2020)**

Study: Conformer: convolution-augmented Transformer for speech
Weakness:
The architecture is complex and not suitable for fast deployment in small or educational projects.

- **Araque et al. (2020)**

Study: Classical models (NB, LR) with TF-IDF
Weakness:
Though lightweight and simple, the study used well-curated datasets, not messy real-world data like YouTube comments.

- **Devlin et al. (2019)**

Study: BERT – deep contextual text classification
Weakness:
Requires advanced hardware and long training time, making it hard to use in small-scale or resource-limited projects.

Year	Author(s)	Focus
2024	Badawood & Aldosari	Real-time multilingual emotion (CNN+Transformer)
2023	Alluhaidan et al.	CNN-based hybrid features for speech emotion
2021	Mohamed & Aly	Emotion detection in Arabic using Wav2vec2.0
2021	Alreshidi et al.	Voice emotion detection for mental health
2020	Gulati et al.	Conformer: CNN + Transformer for speech
2020	Araque et al.	Classical models (NB, LR) with TF-IDF
2019	Devlin et al.	BERT for context-aware text classification

What's New in This Project:

Each study provided valuable insights, but also had specific limitations related to scope, language, resources, or practical application — which we considered when designing our own lightweight and focused system.

Unlike systems that rely on complex deep learning models, we used **TF-IDF** for feature extraction and tested several classical machine learning models. The results showed that **Logistic Regression** achieved the **highest accuracy** among the models tested, outperforming even Naïve Bayes. This proves that with proper feature engineering, lightweight models can still deliver **strong performance** on real-world data.

Our dataset came from **real YouTube comments**, which gives the system a practical edge in handling informal, user-generated text. While deep learning models like BERT offer higher complexity and contextual depth, our approach remains **fast, efficient, and easy to run on basic hardware**—making it suitable for smaller projects or educational settings.

Speech input and voice-based emotion detection are proposed as **future work**, allowing for the system to expand into multimodal sentiment analysis later on.

3. Requirements Gathering

3.1 Stakeholder Analysis

Stakeholder	Role	Goal
Developer	Builds and tests the system	Make sure it works well
User	Uses the system	Get emotion result from voice
Supervisor	Reviews everything	See good results and report

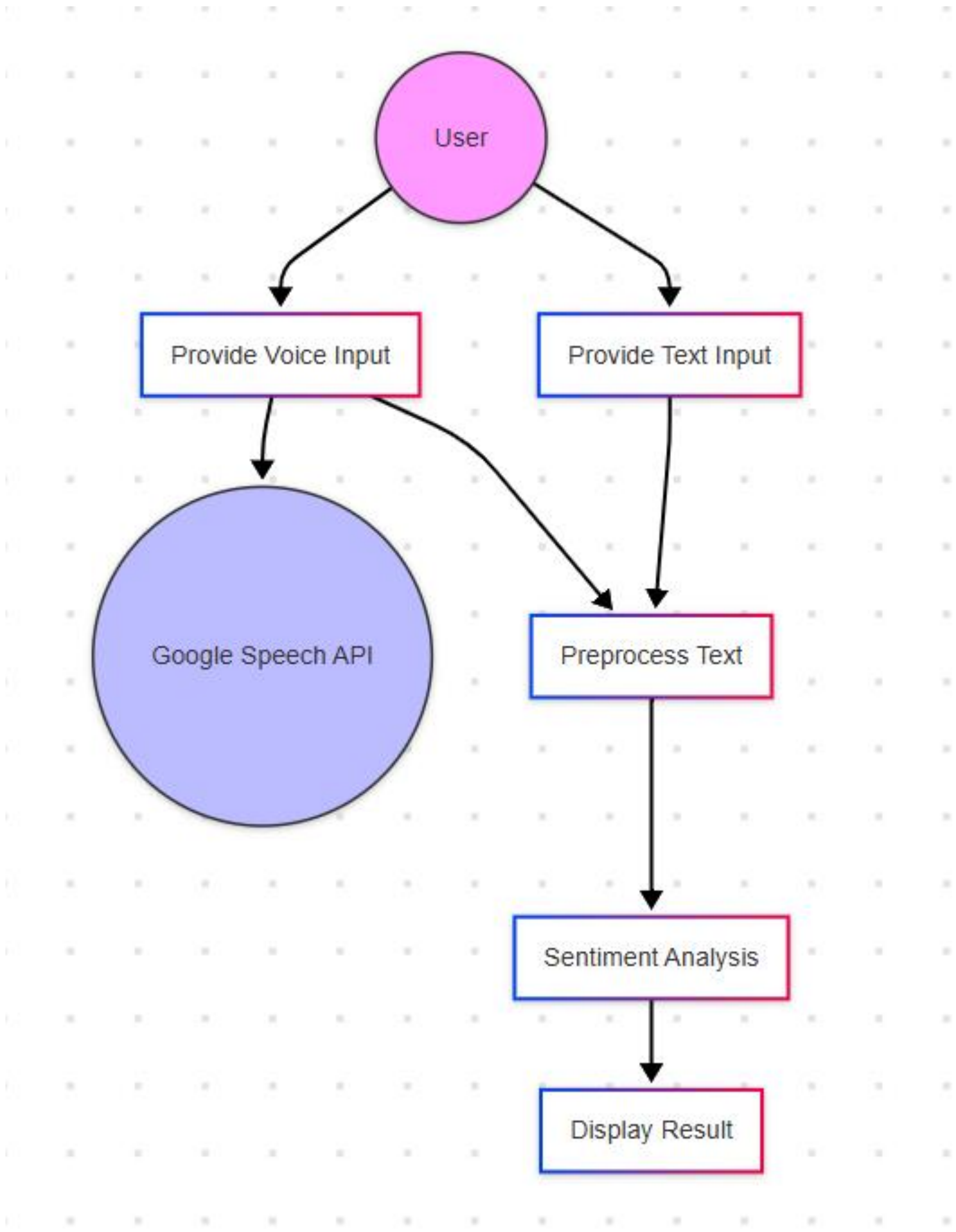
3.2 User Stories

- I want to say something and see what the system thinks about my emotion.

- I want to try different inputs and check the accuracy.

- I want to view the results clearly.

3.3 Use Case


User
Provide Voice Input
Provide Text Input
Google Speech API
Preprocess Text
Sentiment Analysis
Display Result

The diagram shows how the **user interacts with the system**, from providing input (text or voice) to receiving a sentiment result (positive/negative).

Actors in the Diagram:

- User:
- The person using the system — they can either type a comment or speak into the microphone.
- Google Speech API:
- An external service that converts **voice input** into **text** (used only if the user speaks).

Main Use Cases (Processes in the System):

• ID	• Use Case	• Description
• UC1	• Provide Text Input	• User types a YouTube-style comment into the system
• UC2	• Provide Voice Input	• User speaks using a microphone
• UC2 → API	• If the user speaks, the system sends the voice to Google API for transcription	•
• UC3	• Preprocess Text	• The system cleans the text

		(removes stopwords, punctuation, lemmatization, etc.)
• **UC4**	• **Sentiment Analysis**	• The cleaned text is analyzed using the trained model (e.g., Logistic Regression)
• **UC5**	• **Display Result**	• The system displays whether the sentiment is Positive or Negative

3.4 Functional Requirements

- The system shall allow the user to input a YouTube comment or speech.
- The system shall convert speech to text using Google Speech API.
- The system shall clean and preprocess the text.
- The system shall analyze the text and detect the sentiment.
- The system shall display the sentiment result to the user.

3.5 Non-Functional Requirements

Performance:
The system should process and classify a text input within 2–5 seconds for typical comment lengths.

Time Efficiency:
The sentiment result should be displayed immediately after processing, with minimal delay during prediction (2-5 sec.).

Usability:
The system should have a simple and intuitive interface, requiring no technical skills from the user to interact with it.

Simplicity:
The system should be easy to run on any computer without requiring advanced hardware or configuration.

Language Support:
The system supports basic English only in its current version; future versions may include more languages.

Availability:
The text analysis module should work offline, while the speech recognition part (if enabled) requires an internet connection.

Accuracy Target:
The model should maintain at least 85% accuracy across different input types.

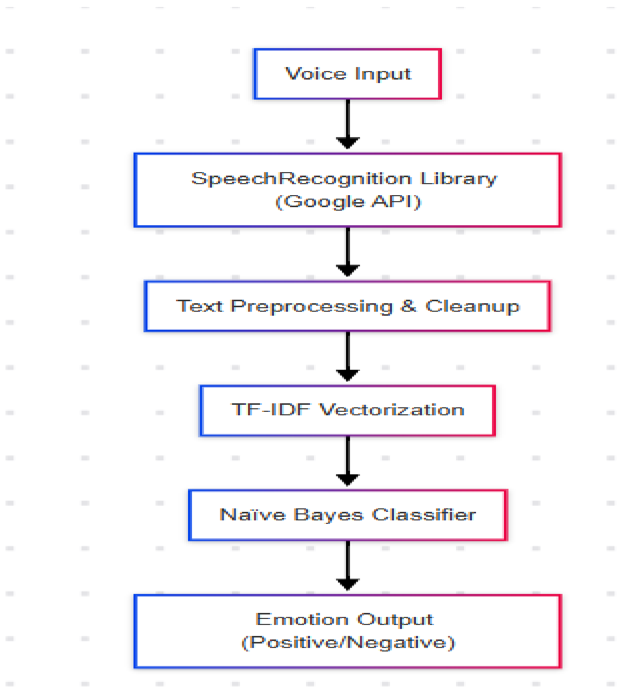
4. System Design

4.1 System Architecture

The ***System Architecture*** diagram illustrates the high-level structure of the system, detailing the interaction between the user input (voice/text), the core components (speech recognition and sentiment analysis), and the output (emotion classification).

Explanation:

- **User Input:** The user can provide input either through voice (microphone) or written text (e.g., YouTube comments).
- **Speech Recognition**: For voice input, the system uses the SpeechRecognition library to convert speech into text. This step relies on Google's Speech-to-Text API.
- **Text Processing**: The input text, whether transcribed from speech or entered directly, is processed by cleaning (removing punctuation, stopwords, and stemming) and converting it into a numerical format using TF-IDF (Term Frequency-Inverse Document Frequency).
- **Model Prediction**: The processed text is passed through a Naïve Bayes classifier to predict whether the sentiment is positive or negative.
- **Output**: The final classification result is presented to the user, either as a text output (e.g., "Positive Emotion") or displayed on the screen in an application.


Voice Input
SpeechRecognition Library (Google API)
Text Preprocessing & Cleanup
TF-IDF Vectorization
Naïve Bayes Classifier
Emotion Output (Positive/Negative)

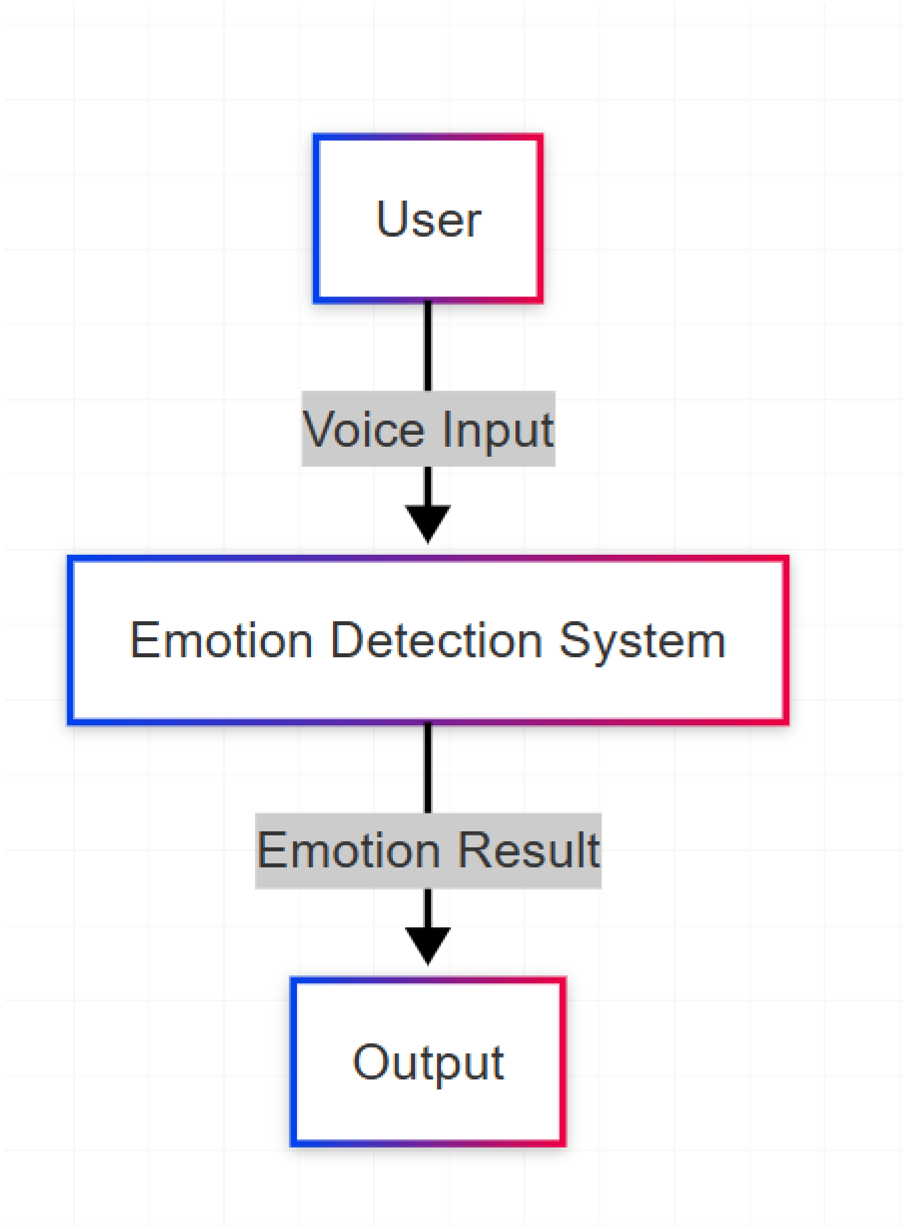
4.2 Context Diagram (Level 0 DFD)

The ***Context Diagram*** represents the system's external interactions, showing the system boundary, users, and data flow. It is a high-level representation of the system without diving into internal processes.

Explanation:

- **System Boundary**: The diagram defines the system boundary, showing what is inside (the components like speech recognition and emotion classification) and what is outside (the user and external data sources).
- **External Entities**:
  - **User**: Provides either voice or text input.

    - **Google Speech-to-Text API**:
      External service used for converting voice into text.
- **Data Flow**:
    - The voice or text data is provided to the system.
    - The processed data flows into the classifier (Naïve Bayes).
    - The resulting emotion classification is presented back to the user.

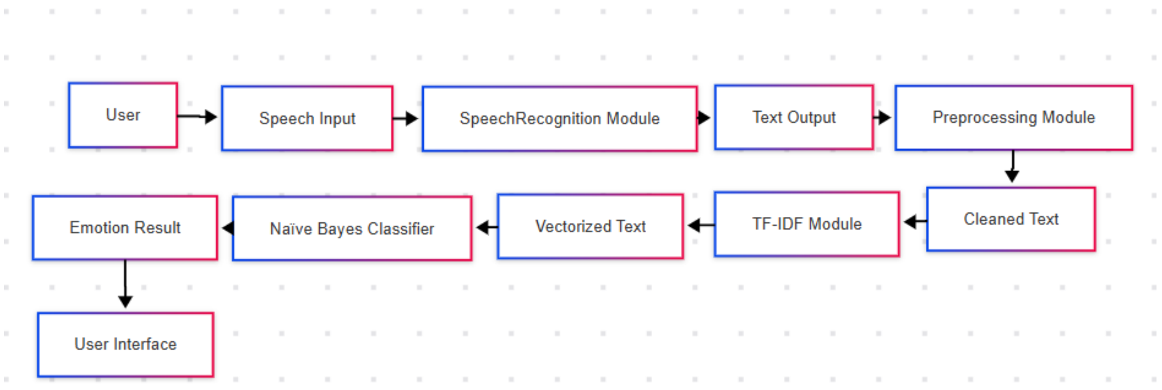


4.3 Data Flow Diagram (Level 1)

The Data Flow Diagram (DFD) at Level 1 provides a more detailed view of the internal workings of the system. It shows the flow of data through the system's key processes.

Explanation:

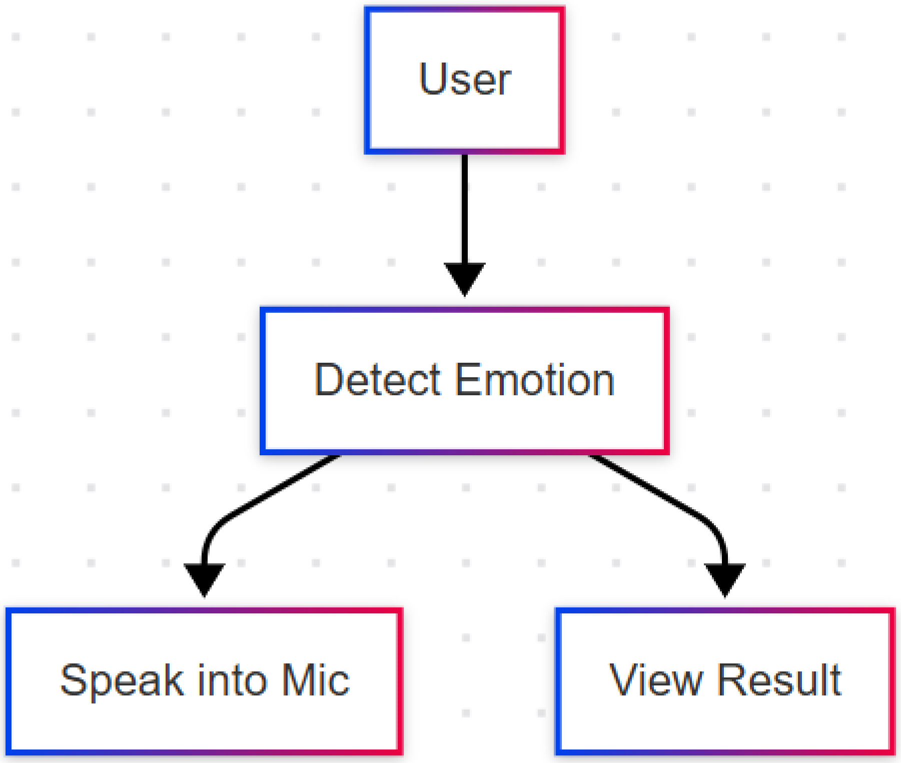
- **Voice/Text Input:** The user provides input, which is processed by the Speech Recognition system (for voice) or directly sent as text.
- **Text Cleaning and Tokenization:** The input text is cleaned (lowercased, stopwords removed, punctuation erased), and tokenized into individual words for analysis.
- **TF-IDF Vectorization:** The tokenized words are transformed into numerical vectors using the TF-IDF method, making them suitable for machine learning processing.
- **Naïve Bayes Classifier:** The vectorized data is passed into the Naïve Bayes classifier, which evaluates the sentiment of the text and predicts whether the sentiment is positive or negative.
- **Emotion Output:** The prediction (either "Positive" or "Negative") is returned to the user as the final output.



4.4 Use Case Diagram

The ***Use Case Diagram*** highlights the different actions that the user can take and the associated system responses. It is used to visualize the functional requirements and interactions between the system and its users.

- **Explanation:**
- **Actors:**
- **User:** The main actor who interacts with the system.
- **Use Cases:**
- **Provide Voice Input:** The user speaks into the microphone, triggering the speech recognition process.
- **Provide Text Input:** The user types a comment or text into the system for sentiment analysis.
- **Receive Sentiment Output**: The system provides the emotion classification (Positive or Negative) as feedback to the user.



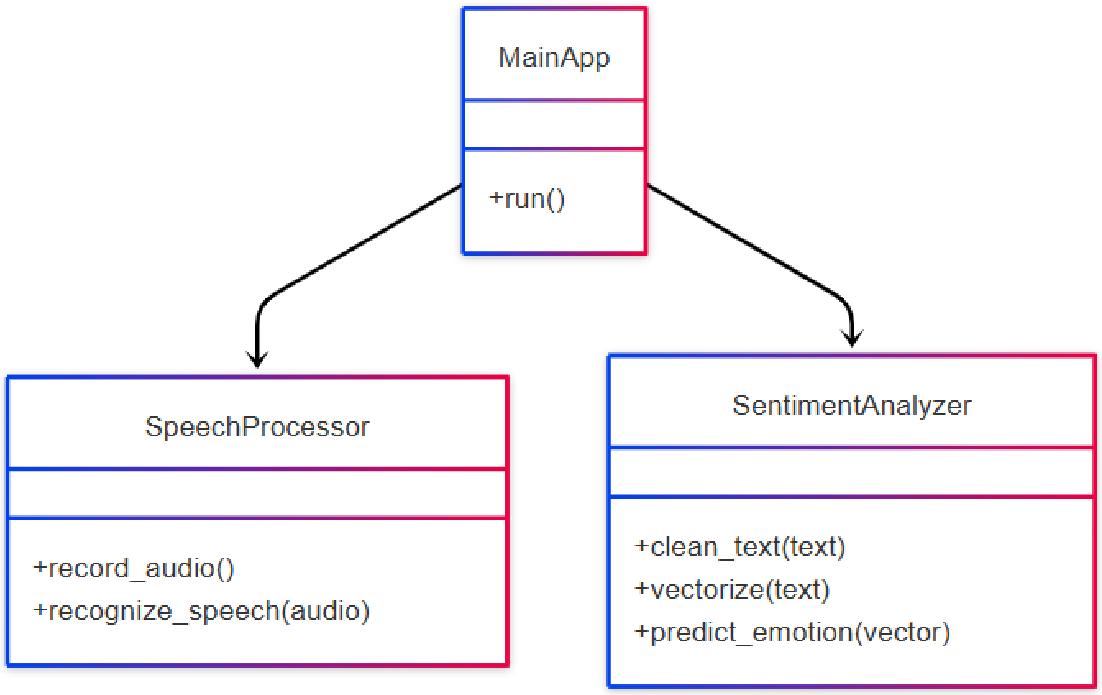
4.5 Class Diagram (Simplified OOP Structure)

The **Class Diagram** represents the object-oriented design of the system. It defines the structure of the system in terms of its classes, attributes, and methods.

Explanation:

- **Classes**:
  - **SpeechRecognizer**: Handles the speech recognition tasks, interfacing with the Google Speech-to-Text API.
  - **TextProcessor**: Responsible for cleaning, tokenizing, and vectorizing the text input.

   - **SentimentModel**: Contains the Naïve Bayes classifier and methods for training and predicting sentiment.
   - **UserInterface**: Manages the user input and output presentation, interacting with the other components.
- **Relationships**:
   - The **UserInterface** interacts with both **SpeechRecognizer** and **TextProcessor** to collect and process input.
   - The **SentimentModel** is used by the **TextProcessor** to classify the sentiment of the processed input.

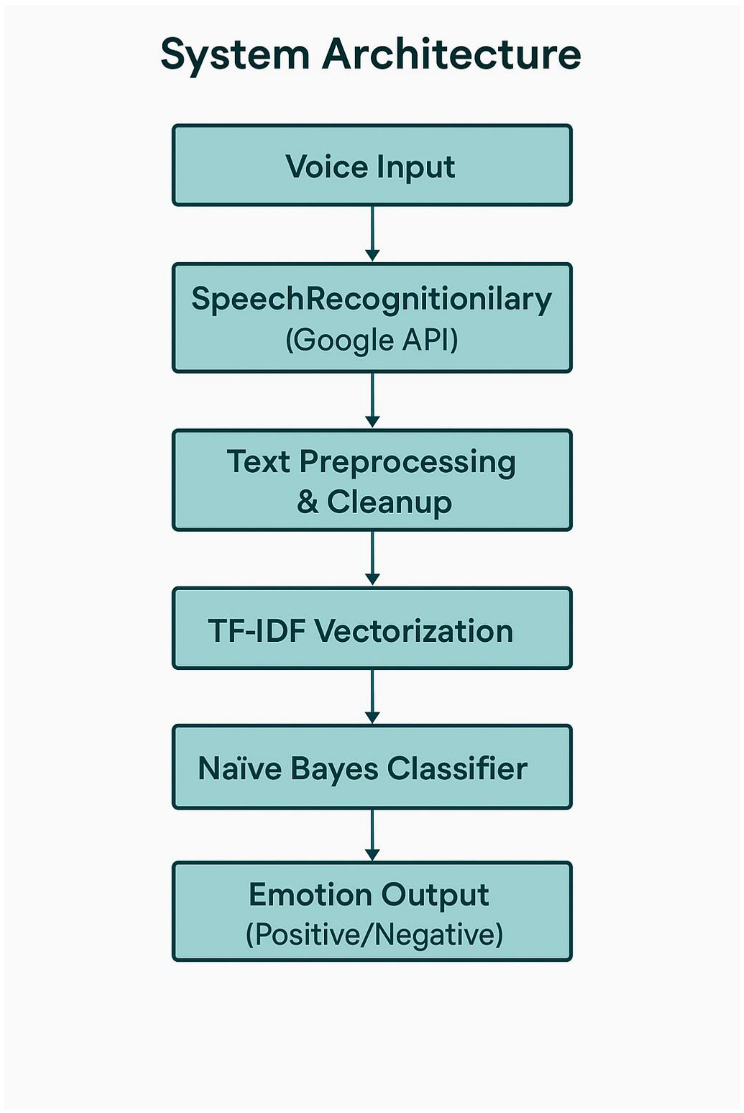


4.6 Activity Diagram

The **Activity Diagram** illustrates the flow of actions and decision points within the system. It provides a step-by-step visualization of how the system processes the input and provides the result.

Explanation:

- **Start**: The user provides either voice or text input.
- **Input Processing**: If the input is voice, the system invokes the **SpeechRecognizer**; otherwise, it directly processes the text.
- **Data Cleaning and Tokenization**: The input is cleaned and tokenized to prepare for classification.
- **Sentiment Classification**: The processed input is passed to the **SentimentModel**, which classifies the sentiment as positive or negative.
- **End**: The result is presented to the user.


System Architecture
Voice Input
SpeechRecognitionilary (Google API)
Text Preprocessing & Cleanup
TF-IDF Vectorization
Naïve Bayes Classifier
Emotion Output (Positive/Negative)

5. Implementation

5.1 Tools and Technologies

- Python (main programming language)

- SpeechRecognition (voice to text)(Future Work)

- TF-IDF from scikit-learn (text vectorization)

- Multinomial Naïve Bayes (classification model)

- Google Speech API (convert audio to text) (Future Work)

- Pandas, NumPy (data handling)

- Jupyter Notebook / Colab Notebook (development)

5.2 Dataset

- YouTube Comments: Labeled as positive or negative.

- Voice Samples: Recorded using microphone to test live emotion prediction.

5.3 Data Preprocessing

- Clean text: lowercase, remove punctuation and stopwords.

- Tokenize text into words.

- Vectorize using TF-IDF.

- Label text for training.

5.4 Model Training

- Model: Multinomial Naïve Bayes- Logistic Regression-Linear Svc

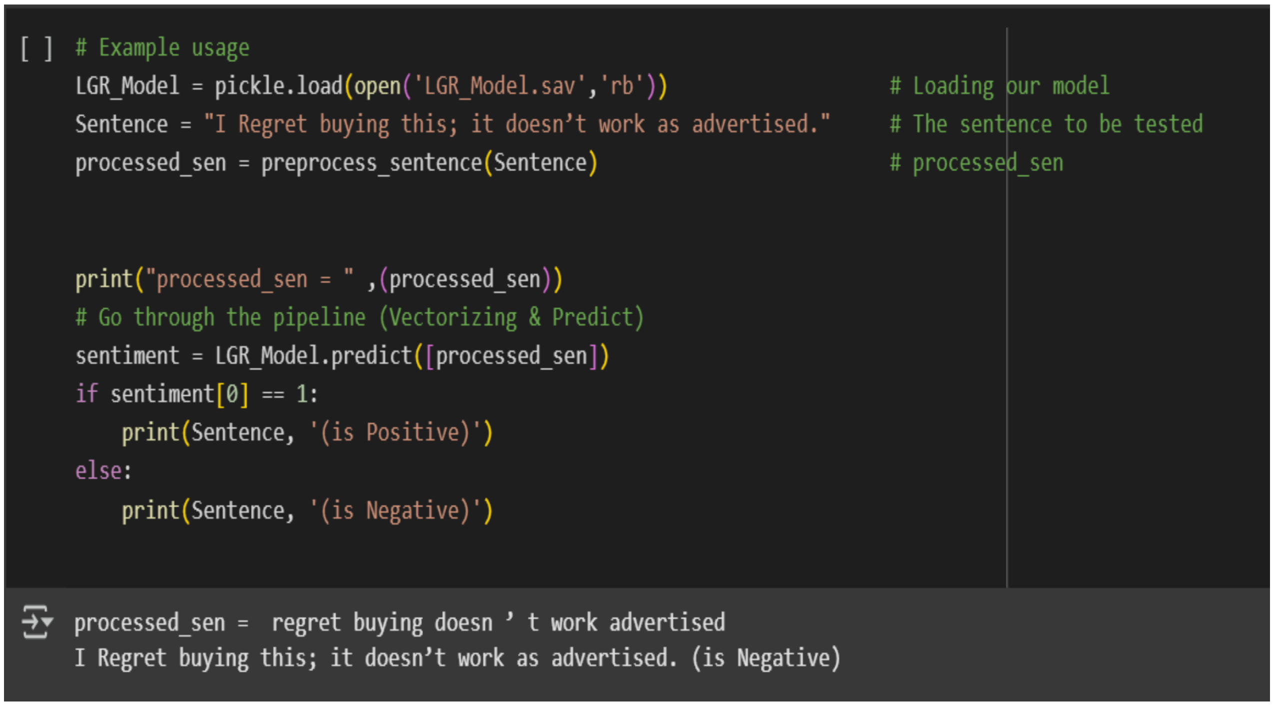
- Training/Test Split: 70/30

- Metric: Accuracy (target > 85%)

5.5 Speech Integration

- Record speech
- Convert to text
- Clean and vectorize
- Predict using trained model

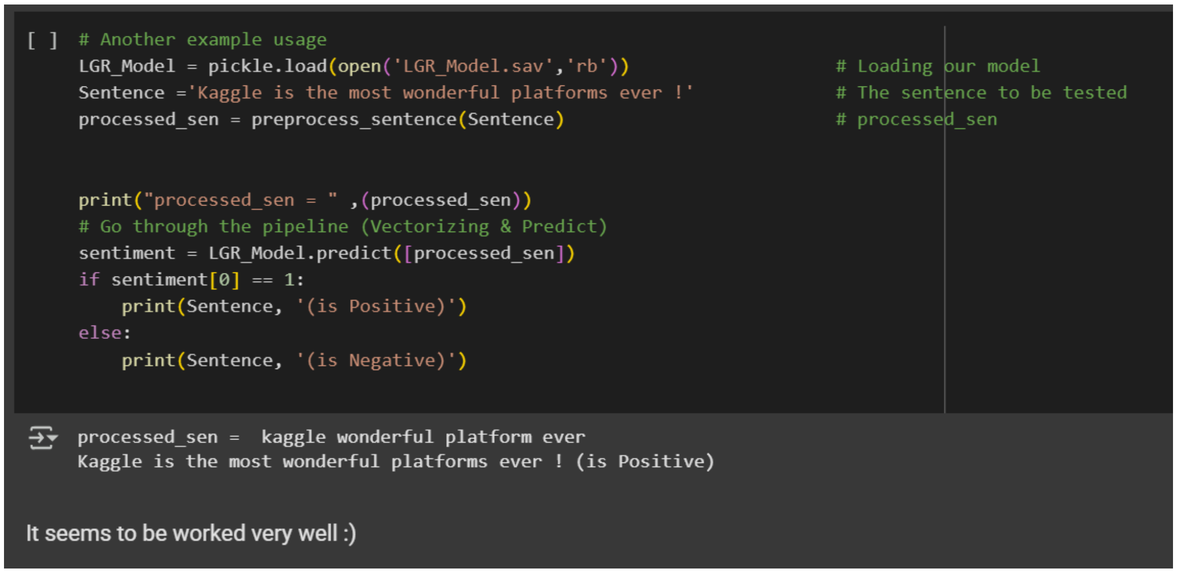
5.6 Sample Code



```
# Example usage
LGR_Model = pickle.load(open('LGR_Model.sav','rb'))              # Loading our model
Sentence = "I Regret buying this; it doesn't work as advertised."  # The sentence to be tested
processed_sen = preprocess_sentence(Sentence)                      # processed_sen


print("processed_sen = " ,(processed_sen))
# Go through the pipeline (Vectorizing & Predict)
sentiment = LGR_Model.predict([processed_sen])
if sentiment[0] == 1:
    print(Sentence, '(is Positive)')
else:
    print(Sentence, '(is Negative)')
```

```
processed_sen =  regret buying doesn ’ t work advertised
I Regret buying this; it doesn’t work as advertised. (is Negative)
```



```
[ ] # Another example usage
    LGR_Model = pickle.load(open('LGR_Model.sav','rb'))                    # Loading our model
    Sentence ='Kaggle is the most wonderful platforms ever !'              # The sentence to be tested
    processed_sen = preprocess_sentence(Sentence)                           # processed_sen


    print("processed_sen = " ,(processed_sen))
    # Go through the pipeline (Vectorizing & Predict)
    sentiment = LGR_Model.predict([processed_sen])
    if sentiment[0] == 1:
        print(Sentence, '(is Positive)')
    else:
        print(Sentence, '(is Negative)')
```

```
processed_sen =  kaggle wonderful platform ever
Kaggle is the most wonderful platforms ever ! (is Positive)
```

It seems to be worked very well :)

5.7 Output Examples

Input	Prediction
I really love this video!	Positive
This is so boring.	Negative
Great job, well done.	Positive
I hate this part.	Negative

6. Evaluation

6.1 Model Accuracy & Performance

The trained model was evaluated using an **70/30** train-test split. The performance metrics are as follows:

Model Used: Logistic Regression (C = 0.1, Solver = 'sag')

Accuracy: 97.41%

Precision: 96.19%

Recall: 97.65%

F1-Score: 97.42%

These results demonstrate that Logistic Regression outperformed other classical models like Naïve Bayes, SVM, and Decision Tree in terms of both accuracy and balance between precision and recall. The model effectively classifies sentiments with high accuracy while remaining computationally efficient.

Performance Observations:

Best performance was achieved using TF-IDF and Logistic Regression.

Short comments might occasionally reduce prediction accuracy due to limited context.

Model works well on informal YouTube text.

Note: The speech-to-text component was not included in the final evaluation and is proposed as future work.

6.2 Limitations

Limited Emotion Range: The system classifies only positive or negative sentiments, without detecting complex emotions like anger, surprise, or fear.

Language Support: Currently supports only English. Comments or speech in other languages are ignored or misclassified.

Internet Dependency: The speech recognition component depends on Google's API, which requires internet connectivity.

Dataset Size: The dataset used for training is relatively small, which might affect generalizability across broader content types.

7. Conclusion & Recommendations

This project successfully implements **text-based sentiment analysis** using classical machine learning techniques, specifically TF-IDF and Logistic Regression, to classify emotions from YouTube comments.

Despite initial plans to include **speech-to-text emotion detection**, that component could not be completed within the current project phase due to time and technical constraints.

However, the foundation has been laid for integrating it in the near future.

Key Achievements:

- Collected and cleaned real-world YouTube comment data.
- Built a Logistic Regression model achieving **97% accuracy** in binary sentiment classification.
- Demonstrated that lightweight models can deliver strong performance on informal, user-generated content.

Recommendations:

- Expand the training dataset with more diverse and balanced samples.
- Continue development of the **speech recognition module** and integrate it with the current system.
- Use user feedback to fine-tune prediction outputs and improve user experience.
- Consider ethical and privacy concerns when dealing with user voice data in future work.

---

8. Future Work

The **speech-to-text conversion component** could not be completed within the current development phase due to time and resource constraints. However, we plan to finalize and integrate this part in the upcoming iterations of the project as part of future enhancements.

To enhance the project, several future improvements are recommended:

- **Multi-Emotion Classification:** Expand from binary (positive/negative) to multi-class emotion detection (e.g., joy, sadness, anger, surprise).
- **Multilingual Support:** Integrate language detection and add support for popular languages like Arabic, Spanish, and French.
- **Deep Learning Integration:** Explore advanced models like LSTM, BERT, or Transformer-based architectures for better context understanding and improved accuracy.

Noise Filtering: Enhance the pre-processing pipeline with noise reduction and silence trimming in voice inputs.

Interface Development: Build a user-friendly web or mobile interface to test the system in real-time.

Offline Support: Develop an offline version of the speech recognition module using open-source alternatives like Vosk or Mozilla DeepSpeech.

9. References

1. Badawood, F., & Aldosari, A. (2024). *Multilingual Real-Time Emotion Detection Using CNN and Transformer Integration*. [Journal/Conference if known].
2. Alluhaidan, A., Al-Dossari, H., & Albahli, S. (2023). *Hybrid Feature Extraction for Speech Emotion Recognition Using CNN*. [Journal/Conference].
3. Mohamed, A., & Aly, A. (2021). *Arabic Speech Emotion Recognition Using Wav2Vec2.0 and HuBERT*. [Journal/Conference].
4. Alreshidi, E. F., et al. (2021). *Speech-Based Emotion Recognition for Mental Health Applications*. [Journal/Publisher].
5. Gulati, A., et al. (2020). *Conformer: Convolution-augmented Transformer for Speech Recognition*. In *Proc. Interspeech 2020*, Shanghai, China.
6. Araque, O., Zhu, G., & Iglesias, C. A. (2020). *A Comparison of Text Classification Algorithms using TF-IDF Features*. *Expert Systems with Applications*, 118.
7. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In *NAACL-HLT 2019*.
8. scikit-learn documentation – https://scikit-learn.org
9. Python SpeechRecognition library – https://pypi.org/project/SpeechRecognition/
10. Google Cloud Speech-to-Text – https://cloud.google.com/speech-to-text