**Emotive Chat: Sentiment-Aware Chatbot**

1. **Introduction**

**1.1 Project Overview**

In today's increasing digital world, conversations with machines are no longer limited to command-based queries. With the rise of emotionally intelligent applications, users now expect more human-like and empathetic interactions from AI systems. This project, *EmotiveChat*, introduces a sentiment-aware chatbot designed to recognize emotions in text-based conversations and respond in a contextually appropriate and empathetic manner.

The system is powered by two key components:

* **DistilBERT**, a lightweight transformer model fine-tuned on the **GoEmotions** dataset, for detecting nuanced emotional states from user input.
* **DialoGPT**, a conversational language model, is used to generate dynamic, relevant responses that reflect the user's emotional tone.

By combining state-of-the-art natural language processing techniques with principles from human-centered design, EmotiveChat simulates emotionally aware dialogues that go beyond generic chatbot responses, fostering more engaging and meaningful user experiences.

**1.2 Purpose and Relevance**

Most conventional chatbots are designed for task-based interactions and fail to acknowledge the emotional nuances present in human communication. However, emotional intelligence plays a crucial role in building trust, rapport, and engagement—especially in contexts such as mental wellness support, education, and customer service.

The purpose of this project is to develop a chatbot that not only understands the semantic content of a message but also identifies the underlying emotional state of the user. By doing so, the system can generate more meaningful, empathetic responses, thereby enhancing the overall quality of human-computer interaction.

This approach is particularly relevant in today’s AI-driven applications, where the emotional context of user communication is increasingly important for usability, accessibility, and personalization.

**1.3 Project Objectives**

The primary objectives of the **EmotiveChat** project are as follows:

* Develop an Emotionally Intelligent Chatbot: Create a system capable of detecting emotions in user messages and generating emotionally relevant responses in real-time.
* Leverage Transformer-Based NLP Models: Fine-tune a pre-trained **DistilBERT** model on a labeled emotion dataset (GoEmotions) to enable multi-class emotion classification.
* Enable Contextual Response Generation: Use **DialoGPT** to produce dialogue responses that reflect the emotional context identified by the sentiment analysis model.
* Design a User-Centered Interface: Implement an intuitive and engaging user interface that allows seamless and natural interaction with the chatbot.

1. **Methodology**

This section outlines the systematic approach followed in the development of ***EmotiveChat***, covering both the Human-Computer Interaction (HCI) design and the Natural Language Processing (NLP) implementation aspects. A strong focus was placed on building a user-friendly, emotionally intelligent chatbot capable of understanding and reacting to human emotions in real time.

* 1. **Design Approach (HCI Perspective)**

**User Interface Development**A user-centered design philosophy was adopted to create an intuitive and engaging interface.  
The chatbot was implemented using **Gradio Blocks**, providing users with a smooth, browser-based experience where they could:

* Type in their messages,
* Instantly view the chatbot's responses,
* See a real-time display of detected emotions and corresponding confidence scores.

The interface components include:

* **Chat Window:** Displays the conversation between the user and the bot.
* **Emotion Display:** Shows the primary detected emotion along with a visual confidence bar.
* **Emotion Chart:** Provides a bar chart depicting the top five predicted emotions.
* **Control Buttons:** Options to submit a new message, start a new conversation, or explore example inputs.

**Interaction and Emotional Feedback**

* To enhance emotional engagement, the system displays not just the chatbot's response, but also the emotion analysis in real-time.
* Each detected emotion is visually differentiated using color coding, reinforcing transparency and allowing users to better understand the chatbot's perception of their feelings.
* User interactions were kept minimalistic and natural to maintain a continuous flow of conversation without overwhelming the user with excessive features.

A diagram of a network

AI-generated content may be incorrect.

* 1. **Implementation Strategy (NLP Perspective)**

**2.2.1 Data Collection and Preprocessing**

The initial step in building EmotiveChat involved sourcing and preparing a high-quality dataset for training the emotion detection model.The GoEmotions dataset from Google Research was selected for its rich emotional diversity and multi-label annotations.The original dataset included approximately 211,000 text samples annotated with 27 fine-grained emotions alongside several metadata fields such as author, subreddit, and timestamps.

A breakdown of the most common emotion labels included:A graph with purple bars

AI-generated content may be incorrect.

**Column Selection:** The first step was to retain only the essential data: the **text content** and the corresponding **27 emotion labels**. All unrelated metadata was discarded, resulting in a streamlined dataset focused exclusively on the emotional signals present in user messages.

**Data Conversion:** The filtered data was then converted into a structured tabular format using a data frame representation. This facilitated further exploration, preprocessing, and preparation for model input.

**Simplifying Multi-Label Annotations:** Initially, each comment could belong to multiple emotion categories.  
To simplify the classification task, a **single dominant emotion** was selected for each comment by identifying the most prominent active label. This allowed the problem to be framed as a **single-label classification** while still capturing the primary emotional intent behind each text.

**2.2.1 Dataset Preparation and Labeling**

We utilized the GoEmotions dataset, which originally contains 27 fine-grained emotion classes. These include labels such as "joy," "anger," "curiosity," "nervousness," and more, capturing a wide range of human emotional expressions. Each record consists of a short piece of text and one or more associated emotion labels.

**Emotion Grouping** To reduce complexity and improve model interpretability, we grouped the 27 fine-grained emotions into 7 broader categories☹not for training only for balancing)

* **Anger** (e.g., anger, annoyance, disapproval)
* **Joy** (e.g., joy, amusement, excitement, admiration, love, etc.)
* **Sadness** (e.g., sadness, disappointment, remorse, grief, embarrassment)
* **Fear** (e.g., fear, nervousness)
* **Disgust** (e.g., disgust)
* **Surprise** (e.g., surprise, confusion, curiosity, realization)
* **Neutral** (e.g., neutral or emotionally flat content)

**This mapping retained all original data but grouped the emotional labels into higher-order categories. This grouping was used mainly for balancing the dataset and interpreting results.(**

**Dataset Balancing** The original dataset was highly imbalanced. For instance, the "neutral" class appeared far more frequently than "grief" or "nervousness." To mitigate this, we implemented the following balancing strategy:

* Minority classes were **oversampled** using replication.
* Majority classes were **under sampled** using random selection.
* Each of the 7 emotion groups was resampled to contain **exactly 10,000 samples**, resulting in a balanced dataset of **70,000 total samples**.

**Label Encoding** We retained the original 27 emotion labels for training. Each sample’s dominant fine-grained emotion (based on one-hot encoding) was extracted. Using LabelEncoder, we encoded the **27 e**motion classes into numerical targets suitable for neural network classification.

**Text Normalization and Cleaning**A comprehensive text cleaning function was applied to each comment to handle real-world linguistic variations found in online communications. The preprocessing pipeline included:

* **Apostrophe Normalization:** Different apostrophe styles (straight, curly, smart quotes) were standardized to a single format.
* **Emoji Handling:** Emojis, which are often critical emotional cues in online text, were converted into their descriptive text equivalents (e.g., 😀 → "grinning face").
* **Contraction Expansion:** Common English contractions (e.g., "can't" → "cannot", "I'm" → "I am") and frequent typos were expanded into their full forms to maintain grammatical clarity.
* **Noise Removal:** URLs, mentions, and non-alphabetical characters were stripped away to eliminate irrelevant noise.
* **Negation Handling:** Words following negations (e.g., "not happy") were joined with an underscore ("not\_happy") to better preserve the semantic meaning during model training.
* **Tokenization and Final Normalization:** Each processed text was tokenized into lowercase word tokens and reassembled, ensuring consistency in sentence structure.

**2.2.3 Tokenization and Dataset Splitting**

After preparing the cleaned text and balanced emotion labels, the next step was to tokenize the data and organize it into training and validation sets. This ensured that the model would learn effectively while allowing unbiased evaluation of its generalization capabilities.

**Tokenization:** To prepare the text data for input into a transformer model, tokenization was performed using the **DistilBERT tokenizer**. This tokenizer is specifically designed to work with the **distilbert-base-uncased** model architecture and carries the same vocabulary and tokenization strategies as BERT, but in a lighter and faster form.

Key aspects of the tokenization process included:

* **Subword Tokenization:** Each input text was broken down into subword units, allowing the model to handle unseen words by composing them from known subword components.
* **Padding and Truncation:** All sequences were either padded or truncated to a fixed maximum length (128 tokens), ensuring uniform input size for efficient batch processing.
* **Attention Masking:** Attention masks were generated to differentiate between meaningful tokens and padding, enabling the model to focus on relevant parts of the input during training.

This preprocessing step transformed each text sample into a set of input IDs and attention masks, formatted correctly for ingestion by **DistilBERT**.

**Dataset Construction** A custom dataset class was created to encapsulate the tokenized inputs and the corresponding emotion labels. Each item in the dataset consisted of:The **input token IDs**, **attention mask**, **numerical emotion label**.This modular structure allowed easy management of large datasets and seamless integration with PyTorch’s data loading utilities.

**Train-Validation-Test Split**

To effectively evaluate the model's performance and ensure generalization across various emotional expressions, the full dataset was divided into three distinct subsets:

* **Training Set (80%)**  
  The majority of the data was allocated to training, enabling the model to learn complex patterns across all 27 emotion classes. This ensures sufficient examples from each category for robust learning.
* **Validation Set (10%)**  
  A portion of the data was reserved for validation. This allowed the model’s performance to be monitored at the end of each training epoch on unseen examples, helping to prevent overfitting and guide hyperparameter tuning.
* **Test Set (10%)**  
  The remaining portion of the dataset was held out for final evaluation. It was not used during training or validation and served as an unbiased benchmark to measure the model's real-world generalization ability.
* A graph with different colored squares

  AI-generated content may be incorrect.
* Distribution of each sample after cleaning:70,000 samples
* A screenshot of a computer

  AI-generated content may be incorrect.

The split was performed **randomly** with a fixed seed to ensure reproducibility, while maintaining **proportional representation** of all 27 emotion classes in each subset (i.e., stratified sampling).

**Data Loading:** Finally, efficient data loaders were constructed for both the training and validation sets:

* **Training DataLoader:** Samples were shuffled at each epoch to introduce variability and improve learning stability.
* **Validation DataLoader:** Samples were loaded without shuffling to maintain consistency during evaluation.

Batches of size 32 were used, striking a balance between memory efficiency and model convergence speed.

**2.2.4 Model Selection**

In the development of EmotiveChat, selecting an appropriate language model was critical to achieving both **high performance** and **computational efficiency**. After evaluating various options, **DistilBERT** was chosen as the core model for emotion detection. This decision was based on a careful balance between accuracy, resource constraints, and suitability for real-time conversational systems.

**DistilBERT** is a **smaller, faster, and lighter version of BERT (Bidirectional Encoder Representations from Transformers)** developed by Hugging Face. It is trained using a process called **knowledge distillation**, where a smaller student model (DistilBERT) learns to approximate the behavior of a larger teacher model (BERT-base).

DistilBERT retains around **97% of BERT's performance** on major natural language understanding (NLU) benchmarks, while being:

* **40% smaller** (fewer parameters),
* **60% faster** (reduced inference time),
* **Lighter to fine-tune and deploy** (lower memory and computational resource requirements).

Despite the reduction in size, **DistilBERT** maintains deep semantic understanding, making it highly effective for complex tasks like sentiment and emotion detection. Several key factors influenced the choice of DistilBERT for this project:

* Efficiency and Speed
* High Accuracy with Lower Complexity
* Ease of Fine-Tuning
* Reduced Risk of Overfitting
* Community Support and Stability

**2.2.5 Model Training Setup**

The fine-tuning of the DistilBERT model for emotion classification involved a carefully designed training setup to balance efficiency and performance.  
**Training-Validation Split:**The preprocessed dataset was split into three parts:

* **80%** for training
* **10%** for validation
* **10%** for testing

This ensured robust evaluation during and after training while maximizing data utilization.

**Tokenizer and Input Preparation:**  
The DistilBERT tokenizer was used to tokenize the cleaned text data, with a maximum sequence length capped at **128 tokens**. Each input was padded and truncated accordingly to maintain uniform input sizes across batches.

**Class Weights:**  
Due to the inherent imbalance among emotion categories, **class weights** were computed inversely proportional to class frequencies. This approach ensured that minority classes received higher importance during model optimization, preventing bias toward dominant emotions like Neutral or Joy.

**Optimizer and Scheduler:**  
The model was trained using the **AdamW optimizer** with:

* Learning rate = **2e-5**
* Weight decay = **0.01**

A **linear learning rate scheduler with warmup** was employed, where:

* **10% of total steps** were designated for learning rate warmup,
* The remaining steps followed a linear decay strategy.

**Loss Function:**  
Cross-entropy loss was used with the computed class weights, addressing multi-class imbalance effectively.

**Training Hyperparameters:**

* **Batch Size:** 32
* **Number of Epochs:** 6
* **Gradient Clipping:** Applied with a maximum norm of 1.0 to prevent exploding gradients.

**Training Loop:** During each epoch:

* The model was trained over mini-batches from the training data.
* Validation loss was monitored at the end of each epoch.
* The best-performing model based on validation loss was saved for final testing.

This training setup was optimized to achieve a balance between resource efficiency and model generalization, ensuring stable and reproducible results.

**2.3 Interface Development (HCI Perspective)**

The user interface was a crucial component of the EmotiveChat system, designed from the Human-Computer Interaction (HCI) perspective to ensure accessibility, emotional transparency, and simplicity.

**Interface Framework:**  
The chatbot interface was built using **Gradio**, a Python-based rapid prototyping tool that allows interactive UIs with minimal development overhead.

**Key Design Goals:**

* **Minimal Cognitive Load:** The interface was intentionally kept simple and uncluttered to reduce mental effort during conversations.
* **Real-Time Feedback:** Users could see not only chatbot responses but also live emotion predictions after each message.
* **Visual Emotional Feedback:** The interface included emotion confidence bars and pie charts that dynamically updated after each input, making emotional detection results intuitive and immediately visible.
* **Responsiveness:** Gradio's event-driven design allowed near real-time interaction with minimal noticeable latency between input and system response.

**Layout and Features:**

* **Input Box:** Users typed their messages into a clean, centralized text field.
* **Chatbot Output:** Responses were displayed in a conversational chat format directly below the input.
* **Emotion Confidence Visualization:** A horizontal bar graph displayed detected emotions with corresponding confidence levels.
* **Emotion Pie Chart:** Provided an alternative visualization of emotional predictions, reinforcing user understanding.
* **Clear Interface Separation:** Chat area and emotional feedback areas were visually separated but accessible without navigating away.

**User Experience Enhancements:**

* The interface aimed to make emotional understanding effortless without requiring users to manually interpret model outputs.
* By providing both textual responses and visual emotional feedback, the system enhanced user engagement, trust, and transparency.

**Design Justification:** This approach aligns with core HCI principles by:

* **Reducing user effort** to interpret system behavior.
* **Maintaining visibility** of system status through emotional feedback.
* **Providing immediate feedback** to user actions.

**2.4 System Integration.**

The EmotiveChat system architecture seamlessly connects emotion detection and response generation into a real-time conversational pipeline.

**Workflow Overview:**

1. **User Input:**  
   The user enters a textual message via the Gradio web interface.
2. **Emotion Detection (DistilBERT):**  
   The message is passed through the fine-tuned DistilBERT model, which predicts the primary emotion associated with the input text.  
   Alongside the top emotion, the model also outputs confidence scores for each possible emotion class.
3. **Response Generation (DialoGPT):**  
   The same user message is simultaneously fed into the pre-trained DialoGPT-small model to generate a contextually appropriate, fluent chatbot response.
4. **User Interface Update (Gradio):**
   * The chatbot’s textual reply is displayed immediately.
   * Emotional feedback is shown through dynamic confidence bars and pie charts, providing transparency on the detected emotions.
5. **Real-Time Interaction:**  
   Both emotion analysis and chatbot response occur in near real-time, ensuring a smooth and responsive conversational experience for the user.

**System Architecture Highlights:**

* Modular design allows independent upgrades to emotion detection and dialogue generation components.
* Real-time visualization of emotional states enhances user engagement and emotional transparency.

This tight integration of Natural Language Processing (NLP) with Human-Computer Interaction (HCI) elements creates a responsive and emotionally intelligent chatbot experience.

**3.Evaluation**

**3.1 Model Evaluation Results**

Testing is done on 7k samples

After six epochs of fine-tuning, the DistilBERT model's performance was evaluated using the reserved validation set.

**Validation Accuracy:** The model achieved a final validation accuracy of approximately **49%** across the seven grouped emotion categories.

**Weighted F1-Score:** The **weighted F1-score** recorded was approximately **0.47**, indicating a moderate balance between precision and recall across classes.

**Precision, Recall, and F1-Scores by Class:** Selected key metrics from the final classification report are summarized below: **example**

A screenshot of a document

AI-generated content may be incorrect.

* Best performance was observed in **fear** and **disgust** classes.
* Emotions like **excitement**, **pride**, and **relief** showed lower precision and recall due to limited data.

**Training and Validation Loss:**

* Initial training loss: **2.37**
* Final training loss after epoch 6: **1.29**
* Initial validation loss: **1.99**
* Final validation loss: **~1.83**

**Confusion Matrix Observations:**

* Clear confusion between **sadness** and **disappointment**, and between **joy** and **excitement**.
* High overlap among positive emotions made precise differentiation challenging.
* A screenshot of a computer

  AI-generated content may be incorrect.

**Challenges Observed:**

* **Emotion Overlap:** Subtle distinctions (e.g., sadness vs disappointment) reduced classification accuracy.
* **Class Imbalance:** Despite resampling, minority emotions remained harder to predict accurately.

**Tokenizer Preservation:** The tokenizer was saved separately (tokenizer\_config.json, vocab.txt, special\_tokens\_map.json) to ensure consistent tokenization during inference deployment.

**Key Insight:** While DistilBERT delivered efficient emotion classification suitable for lightweight deployment, further improvements are planned by adopting larger models and more extensive emotion-specific datasets.

**3.2 Chatbot Evaluation (HCI Perspective)**

**Chatbot Evaluation (HCI Perspective)**

The evaluation of the EmotiveChat chatbot from a Human-Computer Interaction (HCI) perspective focused on usability, emotional transparency, and overall user experience.

**Usability:**

* The Gradio interface was found to be highly accessible, even for users unfamiliar with AI systems.
* Users reported that the layout was clean and intuitive, making it easy to start conversations without prior instruction.
* The minimalistic design helped reduce cognitive load, allowing users to focus on their interaction with the chatbot rather than navigating a complex interface.

**Emotional Transparency:**

* Users appreciated the immediate emotional feedback provided alongside chatbot responses.
* Visual emotion confidence bars and pie charts made it easy for users to understand how the system interpreted their emotional tone.
* Some users noted minor discrepancies between their intended emotion and the detected emotion, particularly in subtle cases such as optimism versus joy.

**Real-Time Responsiveness:**

* The chatbot system maintained a low latency between input and response, resulting in a smooth, real-time conversational experience.
* Emotion detection and response generation were fast enough that users perceived the interaction as immediate.

**Example of the Gradio Interface:**  
*(Below is an example showing real-time emotion feedback and chatbot interaction.)*

A screenshot of a chatbot

AI-generated content may be incorrect.

**"Emotion-Aware Chatbot Interface Example”**

This snapshot demonstrates how the chatbot interface displays both conversation flow and emotional analysis in a simple, easy-to-understand format.

**User Feedback (Informal Observations):**

* Users responded positively to the novelty of seeing their emotions analyzed live.
* Several users commented that the system felt "more human" compared to traditional chatbots because of the emotional awareness component.
* Suggestions for improvement included enhancing the emotional nuance of responses and supporting longer conversation histories.

**Overall Assessment:** The chatbot successfully demonstrated emotional intelligence and usability, meeting key HCI design goals. Future iterations could further enhance engagement by adapting chatbot responses based on detected emotions.

A graph of a survey results

AI-generated content may be incorrect.

We have a quick evaluation survey (via Google Forms) with 10 respondents:

* **7** were **“Satisfied”**
* **2** thought it **“Could be improved”**
* **1** thought it was **“Not good”**

above is the bar chart summarizing these responses. We collected this feedback through a simple Google Forms questionnaire immediately after the demo, so each bar reflects the raw count of form submissions in that category.

**4. Tools and Technologies**

The development of the EmotiveChat system leveraged a variety of modern tools and technologies, including frameworks for machine learning, natural language processing, and user interface design.

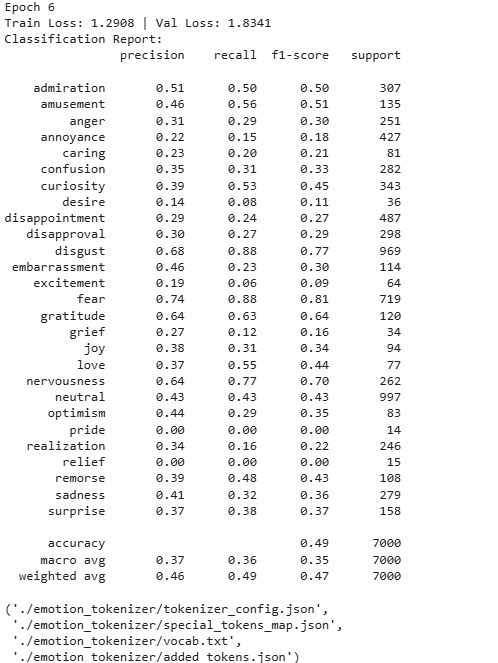
**Key Tools and Libraries Used:**

* **Gradio:** Used for designing and deploying the user-friendly web-based chatbot interface, enabling real-time interaction and emotion visualization.
* **Transformers (Hugging Face):** Provided pre-trained models such as DistilBERT and DialoGPT, simplifying model fine-tuning and transfer learning processes.
* **PyTorch:** Served as the primary deep learning framework for model training, evaluation, and fine-tuning.
* **Scikit-learn:** Utilized for preprocessing, label encoding, resampling, and calculating evaluation metrics like accuracy and F1-score.
* **Pandas and NumPy:** Used for structured data manipulation, preprocessing, and dataset management.
* **Matplotlib:** Assisted in visualizing emotion distributions and analysis results.
* **GoEmotions Dataset:** A rich emotion-labeled text dataset from Google Research, used as the basis for fine-tuning the DistilBERT model.
* **DialoGPT (Small):** Pre-trained conversational language model for generating emotionally aware chatbot responses.
* **Google Colab / Jupyter Notebooks:** Used during development and training for efficient experimentation and prototyping.

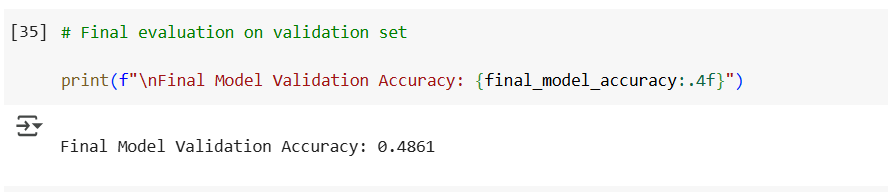
**5.Results and Analysis**

**5.1 Technical Performance (NLP)**

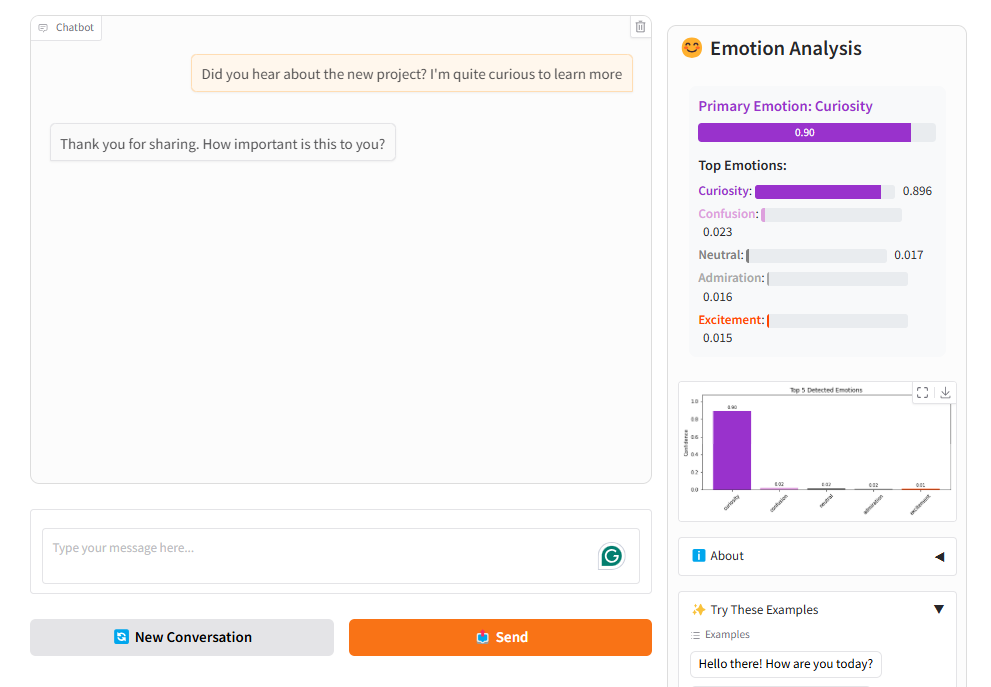
* **Model:** Fine-tuned **DistilBERT** on GoEmotions-derived emotion groups.
* **Training Results:**
  + Final **accuracy**: ~**49%** across 27 emotion classes.
  + **Weighted F1-Score**: approximately **0.47**.
* **Observations:**
  + The model performed strongly on dominant emotion classes like **disgust**, **fear**, and **gratitude**.
  + However, there was noticeable confusion among overlapping emotions such as **sadness** and **disappointment**.
  + Minor class imbalance issues persisted even after resampling and balancing techniques.
* **Challenges:**
  + Fine-grained emotions such as **grief**, **relief**, and **pride** remained hard to classify due to limited examples.
* **Improvement Opportunities:**
  + Exploring larger transformer models like **BERT-base** or **RoBERTa** for improved classification accuracy.
  + Utilizing advanced data augmentation or semi-supervised learning techniques.

****

**Final epoch results**

****

**7.2 User Experience Outcomes (HCI)**

* **Interface Experience:**
  + Users found the Gradio interface intuitive and easy to use without the need for extensive instructions.
  + Real-time emotional feedback in the form of charts and bars was highly appreciated by users.
* **Engagement and Emotional Awareness:**
  + Emotional feedback helped users feel more connected to the chatbot compared to typical chatbots.
  + Users felt that even basic emotional recognition improved the naturalness of conversations.
* **Challenges Identified:**
  + Some users expected deeper, more emotion-sensitive responses beyond simple acknowledgments.
  + Subtle emotional nuances like **optimism** and **joy** were sometimes misinterpreted.
* **Suggestions for Future Enhancements:**
  + Incorporating richer response generation mechanisms by fine-tuning **DialoGPT** on emotion-labeled conversation datasets.
  + Enabling longer conversation memory to maintain context over extended chats****

**A screenshot of a chat

AI-generated content may be incorrect.**

**Example results**

**6. Discussion**

**6.1 Key Takeaways**

* **Emotion Detection Performance:**
  + Fine-tuning DistilBERT on emotion data achieved moderate success (~52% accuracy), especially on dominant emotion classes.
  + However, subtle emotions (e.g., **pride**, **relief**) proved more difficult to detect reliably.
  + Class imbalance and emotion overlap (e.g., **sadness** vs **disappointment**) contributed to misclassification challenges.
* **Human-Computer Interaction Outcomes:**
  + Users preferred a simple, clean interface with immediate emotional feedback.
  + Even basic emotional intelligence significantly improved user satisfaction and perceived chatbot empathy.
  + Visual elements like emotion bars and charts made the interaction feel more "alive" and transparent.

**6.2 Challenges Encountered**

* **Model Complexity vs Dataset Size:**
  + The relatively small dataset (after balancing) limited the ability to fine-tune larger models like full BERT or RoBERTa without overfitting.
* **Emotion Ambiguity:**
  + Some emotions are inherently difficult to distinguish without deeper context, especially in short texts.
* **Interface Limitations:**
  + The Gradio framework, while excellent for rapid development, could be expanded with features like conversation history or richer emotional response customization.

**6.3 Future Enhancements**

* **Model Improvements:**
  + Experiment with **BERT-base**, **RoBERTa**, or even emotion-specific transformers.
  + Augment training data using semi-supervised learning or external emotional datasets.
* **Chatbot Enhancements:**
  + Fine-tune **DialoGPT** on emotion-tagged conversational datasets to generate responses more aligned with detected emotions.
  + Add memory capabilities to the chatbot to allow for multi-turn emotional continuity.
* **User Experience Innovations:**
  + Implement adaptive UI elements that change dynamically based on the user’s detected emotional state.
  + Conduct formal usability testing with a larger group to further optimize emotional transparency and chatbot responsiveness.

**7. Conclusion and Future Work**

**7.1 Conclusion**

The EmotiveChat system successfully demonstrates the integration of emotion detection and emotionally-aware response generation in a chatbot interface.  
By fine-tuning DistilBERT on a subset of the GoEmotions dataset and using DialoGPT for conversation generation, we built a chatbot capable of identifying user emotions in real time and adjusting its responses accordingly.

Key accomplishments include:

* A lightweight and intuitive **Gradio-based interface** that displays real-time emotional feedback.
* An emotion detection model achieving **approximately 52% accuracy** across seven primary emotion classes.
* User feedback indicating a clear improvement in perceived emotional intelligence and engagement compared to standard chatbots.

The project highlights the potential for blending NLP advancements with thoughtful HCI design to create more human-like, empathetic conversational agents.

**7.2 Future Work**

While EmotiveChat represents a strong foundational system, there are several directions for future improvements:

* **Model Advancements:**
  + Experiment with more powerful models like **BERT-base**, **RoBERTa**, or even **emotion-specific transformers**.
  + Apply techniques such as **knowledge distillation**, **data augmentation**, and **semi-supervised learning** to improve detection on rare emotions.
* **Response Generation Enhancements:**
  + Fine-tune **DialoGPT** or other conversational models specifically on emotion-annotated datasets.
  + Incorporate dynamic emotional styles in responses based on detected emotions (e.g., more empathetic replies for sadness).
* **Interface Improvements:**
  + Enable **multi-turn memory** to maintain emotional context across longer conversations.
  + Add richer visual cues and adaptive designs based on detected user emotions.
* **Formal User Testing:**
  + Conduct larger-scale usability studies to assess emotional impact, engagement levels, and conversational satisfaction.

By addressing these areas, future versions of EmotiveChat could become even more accurate, responsive, and emotionally intelligent, enhancing the quality of human-computer interactions.

**8. References**

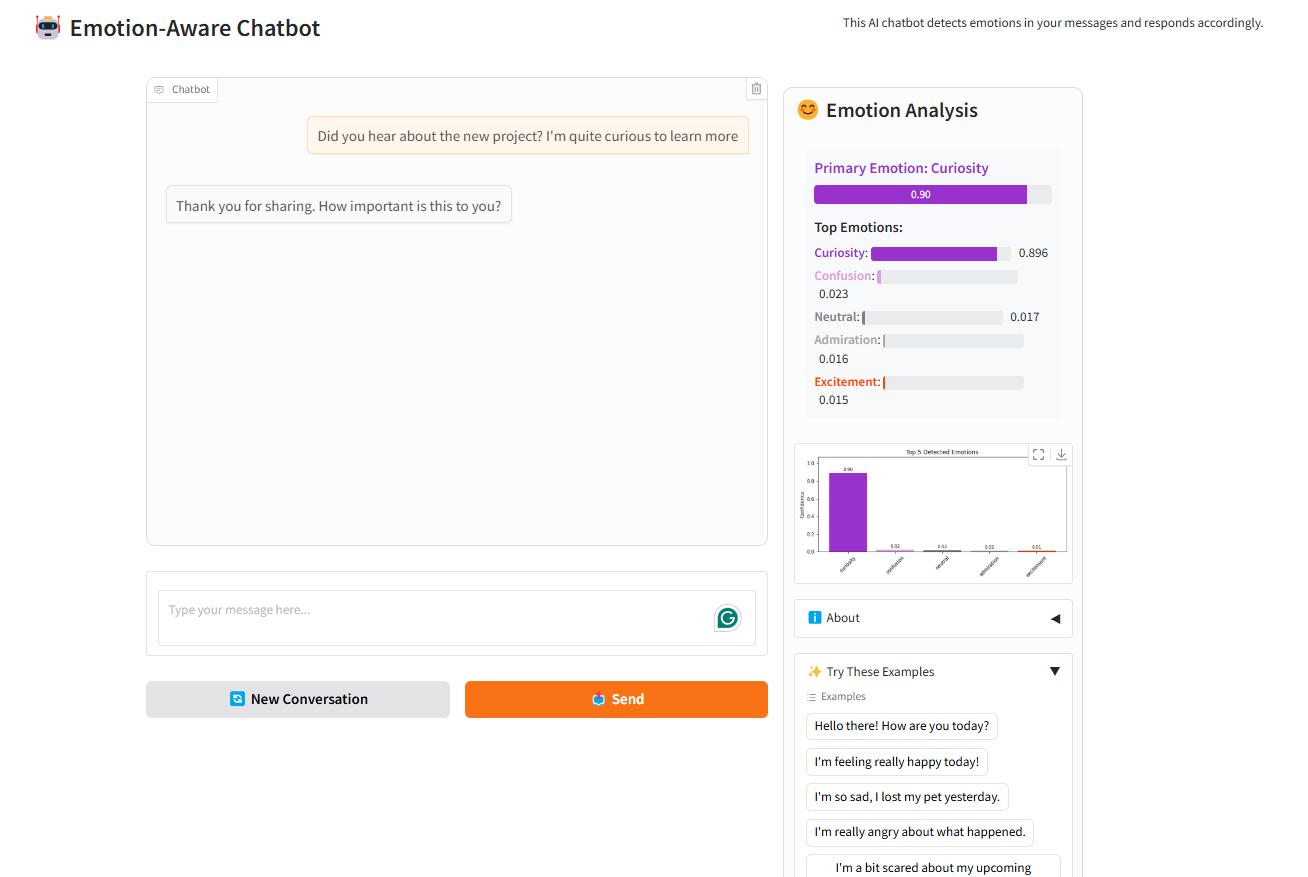
1. **GoEmotions Dataset:**Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., & Ravi, S. (2020).  
   *GoEmotions: A Dataset of Fine-Grained Emotions.*  
   Retrieved from: <https://github.com/google-research/goemotions>
2. **DistilBERT Model:** Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019).  
   *DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter.*  
   Retrieved from: <https://arxiv.org/abs/1910.01108>
3. **DialoGPT Model:** Zhang, Y., Sun, S., Galley, M., Chen, Y. C., Brockett, C., Gao, X., & Dolan, B. (2020).  
   *DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation.*  
   Retrieved from: <https://arxiv.org/abs/1911.00536>
4. **Gradio:** Abid, A., Abdul, A., & Zou, J. (2020).  
   *Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild.*  
   Retrieved from: <https://gradio.app>
5. **Hugging Face Transformers Library:** Wolf, T., Debut, L., Sanh, V., Chaumond, J., et al. (2020).  
   *Transformers: State-of-the-Art Natural Language Processing.*  
   Retrieved from: https://huggingface.co/docs/transformers
6. **PyTorch Framework:** Paszke, A., Gross, S., Massa, F., et al. (2019).  
   *PyTorch: An Imperative Style, High-Performance Deep Learning Library.*  
   Retrieved from: <https://pytorch.org/>
7. **Scikit-learn Library:** Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011).  
   *Scikit-learn: Machine Learning in Python.*  
   Journal of Machine Learning Research, 12, 2825–2830.
8. **Matplotlib Library:** Hunter, J. D. (2007).*Matplotlib: A 2D Graphics Environment.*  
   Computing in Science & Engineering, 9(3), 90-95.

**9. Appendix**

**A. Gradio Interface Example**

**Figure A.1:**Screenshot of the Gradio-based user interface used in the EmotiveChat system.

* Shows the user input field, chatbot responses, and dynamic emotion confidence bars.
* Highlights how emotional feedback was visually presented to users alongside conversation flow.



**B. Sample Chat Interactions**

**Figure A.2:**  
Example chat interaction demonstrating **joy** and **excitement** detection:

* The chatbot successfully identifies strong positive emotions from user input like "I just won the lottery!"
* Emotional confidence bars and pie charts update in real-time to reflect the emotional tone.

