AI System Design Document: SMS Grief Support

Prepared by Pankti Shah

Abstract

This document presents the design and development of an AI-powered system for processing SMS replies in a grief support service. The system classifies responses based on Sentiment Analysis, Response Necessity, and Crisis Detection to ensure timely and appropriate interventions. It follows a modular, scalable, and real-time processing architecture, leveraging transformer-based NLP models (e.g., BERT, DistilBERT) alongside a vector database can be applied for semantic search and context-aware classification.

Key components include an event-driven pipeline using Kafka, a FastAPI ingestion service, and an on-premise deployment strategy to ensure privacy, security, and compliance. Additionally, the system integrates human-in-the-loop interventions for handling high-risk messages and continuous learning mechanisms using MLflow and DVC. The document outlines system design, data processing, model training, decision-making strategies, deployment, and ethical considerations for building a robust and effective grief support AI system. A Streamlit-based SMS Classification tool is also being developed for overview of the task.

Table of contents

List of figures

Task 1: System Design Approach

1. How would you design this system?

The proposed system is designed to process incoming SMS replies to grief support messages, categorizing them based on **Sentiment Analysis, Response Necessity, and Crisis Detection**. The architecture follows a modular, scalable, and distributed approach, ensuring real-time processing, robustness, and adaptability for future improvements.

The system ingests structured data (metadata such as timestamps, message history, and user interactions) and unstructured data (raw text from SMS messages) through an event-driven pipeline using **Kafka** for real-time message streaming. **FastAPI** serves as the ingestion endpoint, preprocessing text through **standard NLP** techniques for pre-processing of data. The raw text undergoes preprocessing using standard NLP techniques, and structured metadata is stored in **PostgreSQL** to allow efficient retrieval and analysis. While unstructured data (SMS messages and processed features) could be stored in **NoSQL databases like MongoDB or Elasticsearch** for flexible schema handling and fast text search capabilities.

To enhance context-aware classification and crisis detection, the system can integrate a **vector database** (FAISS, Qdrant, or Weaviate) for semantic search and similarity matching. SMS messages are converted into dense vector embeddings using Sentence-BERT (SBERT) or FastText, which are then stored in the vector database. This enables efficient retrieval of similar past messages, improving response necessity classification and detecting high-risk crisis patterns by comparing new messages against a repository of historically flagged crisis texts.

The classification pipeline consists of three independently deployed **deep learning models**, each focusing on distinct tasks: sentiment analysis using a model such as fine-tuned DistilBERT model, response necessity classification via a model such as BERT-based binary classifier, and crisis detection with a model such as hybrid LSTM-Transformer model. These models operate concurrently, leveraging the distributed computing capabilities of **Ray to achieve parallel inference**, optimizing speed and resource utilization. If any classification result falls below a confidence threshold, the system triggers a human-in-the-loop intervention, ensuring critical cases receive immediate attention while also contributing to model improvement.

For model training and continuous learning, **MLflow** is used to track experiments, while **DVC (Data Version Control)** ensures dataset reproducibility. A feedback loop is implemented where low-confidence predictions are flagged for human review, improving model accuracy over time. The system is deployed using **Docker and Kubernetes**, ensuring fault tolerance, auto-scaling, and efficient resource management. **Model drift** is monitored using **Prometheus and Grafana**, triggering automated retraining workflows if performance degradation is detected.

The system prioritizes real-time crisis detection, ensuring immediate intervention when necessary while maintaining **data privacy and security** in an on-premise infrastructure. Future scalability is enabled through distributed computing frameworks such as Ray, supporting high-throughput processing and adaptive learning mechanisms.

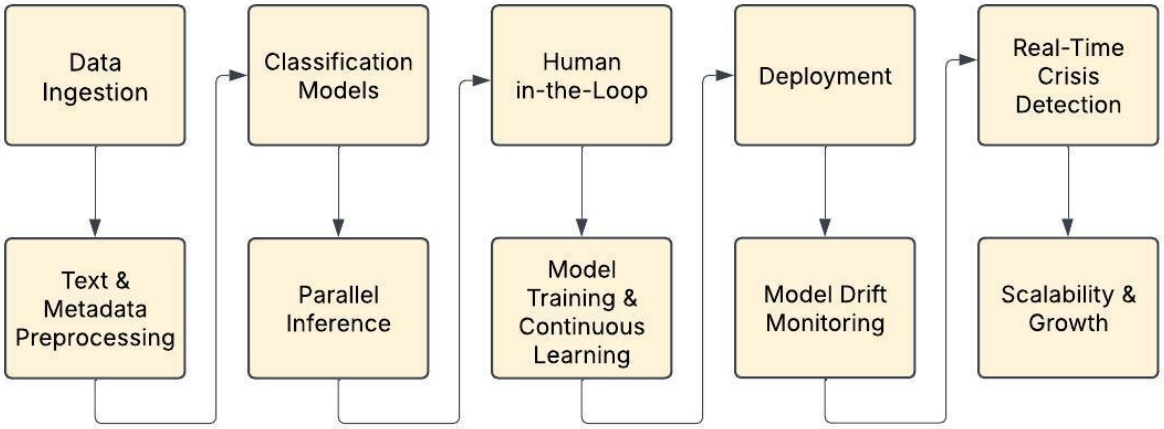

Fig 1. AI Grief support System Design diagram

2. What AI/ML techniques would you use?

The system leverages a combination of **advanced deep learning, hugging face models and classical machine learning techniques** to ensure accurate, efficient, and reliable classification of SMS responses. Transformer-based models such as BERT and DistilBERT utilize self-attention mechanisms to capture contextual dependencies in text, making them well-suited for sentiment analysis and response necessity classification. For crisis detection, sequence modeling techniques such as bidirectional LSTMs and GRUs process sequential user interactions, enabling the system to identify escalating distress patterns over time. Additionally, **anomaly detection** methods like autoencoders and one-class SVMs help recognize messages that deviate significantly from normal communication, flagging them for further review.

To enhance model reliability and ensure robust decision-making, Bayesian deep learning methods such as Monte Carlo Dropout estimate prediction uncertainty, facilitating human-in-the-loop interventions when confidence levels are low. Hybrid ensemble approaches integrate transformer embeddings **with classical classifiers** like Random Forests and XGBoost, improving the accuracy of response necessity classification while maintaining interpretability. **Self-supervised learning** and **contrastive learning techniques** enable models to adapt to domain-specific language in grief-support conversations without requiring extensive labeled datasets.

To maintain efficiency and scalability**, federated learning** techniques can be explored for training models across distributed environments while preserving user privacy in an on-premises setup. Classical machine learning models such as logistic regression and SVMs serve as lightweight alternatives for cases where **interpretability, explainability, and lower latency** are required. The combination of these AI/ML techniques ensures that the system remains adaptable, interpretable, and effective in processing real-time SMS responses.

3. Would you fine-tune an existing model or train a custom one from scratch? Why?

The system would adopt a fine-tuning approach using pre-trained transformer models rather than training a model from scratch. Pre-trained architectures like BERT, DistilBERT, and RoBERTa have already been trained on extensive corpora, equipping them with robust language understanding capabilities. Fine-tuning these models on domain-specific grief-support conversations enables

them to capture the nuances of empathetic communication and crisis-related language without requiring massive amounts of labeled data.

Fine-tuning is significantly more efficient in terms of computational resources and training time compared to training a model from scratch. It allows the system to leverage existing linguistic knowledge while adapting to task-specific requirements with relatively small datasets. Additionally, fine-tuned models generalize well and can be optimized for on-premises deployment, ensuring scalability and performance within real-time SMS processing pipelines. Training from scratch, on the other hand, would demand a high-quality, large-scale dataset, substantial GPU resources, and extensive hyperparameter tuning, making it impractical for this application. By fine-tuning pre-trained models, the system achieves high accuracy, faster development cycles, and reduced infrastructure costs while maintaining adaptability for real-world deployment.

4. How would you structure the pipeline to classify sentiment, response necessity, and crisis detection efficiently?

The classification pipeline is designed to process incoming SMS messages efficiently by structuring the workflow into distinct stages: preprocessing, feature extraction, model inference, and post-processing. The first stage involves **text normalization, tokenization, and stopword removal to standardize** input data while preserving essential context. **Feature extraction** is handled using transformer-based contextual embeddings, enabling deep semantic understanding of the text.

The pipeline employs three independent yet parallel classification models. Sentiment classification utilizes fine-tuned transformer models like BERT or LSTM networks to analyze message tone. Response necessity detection functions as a binary classification task, incorporating transformer-based models with rule-based heuristics to identify whether an SMS requires a reply. Crisis detection relies on deep learning-based anomaly detection techniques, leveraging attention-based models and keyword recognition to identify high-risk messages that deviate from typical conversation patterns. Each model is evaluated using metrics such as accuracy, precision, recall, and F1-score to ensure reliability.

The inference stage runs in parallel using a **distributed framework like Ray**, allowing efficient real-time processing. Post-processing logic determines the appropriate response, with low-confidence classifications automatically flagged for human review. The system is deployed on-premise, ensuring **scalability, compliance, and adaptability**, while continuous retraining mechanisms update models based on new data to improve classification accuracy over time.

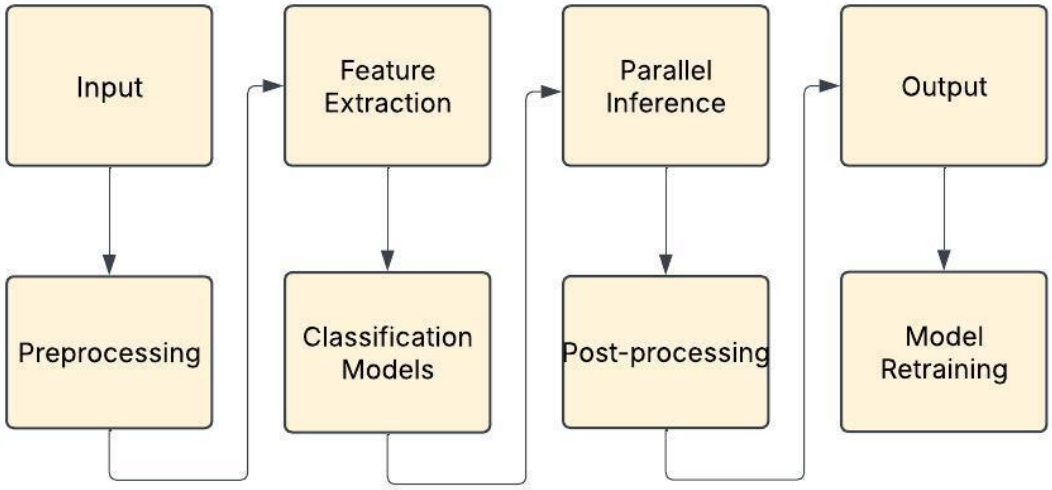

Fig 2. Classification pipeline

5. **How would you handle uncertainty in classifications?**

   To handle uncertainty in classifications, the system integrates **threshold-based classification, ensemble learning, and human-in-the-loop review**. **Confidence scores** generated by each model determine the reliability of a prediction, with low-confidence cases automatically flagged for manual intervention. Threshold tuning ensures that ambiguous classifications do not lead to incorrect automated actions, particularly in sensitive scenarios such as crisis detection.

   Ensemble methods enhance robustness by aggregating predictions from multiple models, reducing bias and improving overall reliability. A hybrid approach combining **transformer-based models** with traditional classifiers (such as Random Forests or XGBoost) further refines decision-making, leveraging both deep learning and interpretable ML techniques.

   For high-risk situations, such as messages indicating a crisis, the system enforces automatic escalation to human reviewers, ensuring critical interventions are prioritized. The feedback from human interventions is incorporated into continuous learning pipelines, enhancing model adaptability and refining classification accuracy over time. This balanced approach maximizes **automation efficiency** while maintaining the necessary safeguards for **sensitive communications.**

Task 2: Data Processing & Feature Engineering

1. **How would you prepare and preprocess SMS messages for analysis? What text-cleaning steps would you apply (handling typos, abbreviations, emojis, etc.)?**

   To ensure high-quality input for analysis, SMS messages undergo a structured text preprocessing pipeline that includes multiple NLP techniques. First, text normalization is performed by **converting all characters to lowercase, followed by tokenization using subword-based models (WordPiece, Byte-Pair Encoding)** to handle rare words effectively. **Spell correction** and abbreviation expansion are applied using contextual embeddings, which help normalize informal

text (e.g., “idk” → “I don’t know”). **Emojis and symbols** are mapped to their textual equivalents to preserve sentiment-related information (e.g., 😊 → “happy”). **Stopword removal** and **lemmatization** are performed using spaCy or NLTK, ensuring that only meaningful words are retained. Additionally, the system incorporates **named entity recognition (NER**) to extract important contextual details such as names, locations, or events. To handle code-switching, messages containing multiple languages are detected and processed separately, ensuring accurate interpretation. This robust preprocessing pipeline ensures that the text is clean, structured, and optimized for downstream classification models.

2. **Would you incorporate message history for better context? If so, how?**

Yes, incorporating message history significantly improves classification accuracy by providing conversational context. To achieve this, the system maintains a rolling window of past messages (e.g., the last 3-5 exchanges) to help the model understand sentiment shifts and evolving intent. Contextual embeddings are generated using transformer-based sequential encoders, which allow the model to retain information over multiple turns in a conversation. Additionally, memory-augmented architectures like Attention-based LSTMs and GRUs enhance historical context retention for crisis detection. In cases where deeper conversation tracking is needed, graph-based approaches can be used to model the relationships between users and their message interactions over time. By integrating previous responses and sentiment trends, the system ensures that messages are classified not in isolation but as part of a larger conversation flow, improving the detection of subtle distress signals and response necessity.

3. **How would you extract meaningful linguistic features for classification?**

To improve classification accuracy, the system extracts various types of linguistic and contextual features. **Lexical features** include word frequency, sentence length, punctuation, and n-grams (bigrams and trigrams) to identify patterns in text. **Semantic features** use **pre-trained word embeddings** (BERT, RoBERTa, word2vec, FastText) to understand the meaning of words in context. **Sentiment and emotion-based features** are derived from tools like **VADER, NRC Emotion Lexicon, and LIWC**, helping detect emotional tone and distress signals. **TF-IDF scores** highlight important words, improving response necessity classification. For **crisis detection**, techniques like **autoencoders and outlier detection** identify messages that differ significantly from normal conversations. **Temporal features** such as message timestamps, response times, and frequency of replies help determine urgency, particularly for crisis situations. By combining these features, the system ensures accurate and reliable message classification.

Task 3: Model Training & Evaluation

1. **How would you approach training and validating your model(s)?**

The training process follows a **supervised fine-tuning approach** using pre-trained transformer models adapted for each classification task—**sentiment analysis, response necessity, and crisis detection.** The dataset is **preprocessed, tokenized, and split into training, validation, and test sets** (typically 80-10-10). **Data augmentation** techniques such as back-translation, synonym replacement, and paraphrasing help improve model generalization. **Stratified sampling ensures a**

balanced representation of all classes. The model is trained using **cross-entropy loss, and early stopping prevents overfitting**. Hyperparameters such as **learning rate, batch size, and dropout rate are optimized using grid search or Bayesian optimization**. The validation set helps tune the model, while the test set evaluates its generalization performance. **K-fold cross-validation** further enhances model robustness.

2. What kind of dataset would you need?

The dataset must include a diverse collection of real-world SMS messages related to grief support with human-labeled classifications for sentiment (positive, neutral, negative), response necessity (reply needed, no reply needed), and crisis level (no crisis, possible crisis, crisis detected). Data sources include anonymized **grief support conversations, publicly available mental health datasets** (e.g., Crisis Text Line datasets, Suicide Risk datasets), and **synthetic data generated**. Metadata such as timestamp, sender type (human vs. automated response), and prior message history should also be included to improve classification accuracy.

3. How would you handle class imbalance (e.g., crisis messages being rare)?

Since crisis messages are much rarer than non-crisis messages, the system uses **data balancing techniques** to improve accuracy. **Oversampling** (SMOTE) creates synthetic crisis messages, while **undersampling** reduces the number of non-crisis messages to avoid bias. **Cost-sensitive learning** adjusts the model's focus on crisis cases by giving them more weight during training. **Focal loss** helps the model pay more attention to difficult-to-classify crisis messages. Additionally, **data augmentation** (e.g., paraphrasing and adversarial text generation) increases crisis message variations, making the model more robust.

4. What evaluation metrics would you track to measure model performance?

- **Sentiment Analysis & Response Necessity**: The model is evaluated using **Accuracy, Precision, Recall, and F1-score** to measure classification performance. Additionally, the **Average Confidence Score per Class** is tracked to assess the certainty of predictions, ensuring that the model provides reliable outputs for automated decision-making.
- **Crisis Detection**: Given the critical nature of crisis identification, **Recall (Sensitivity) is prioritized** to minimize false negatives and prevent missed crisis cases. **Precision and F1-score** ensure a balance between false positives and false negatives. Furthermore, **Confidence Thresholding** is applied—messages with a confidence score below **80%** are flagged for **human-in-the-loop intervention** to prevent misclassification of high-risk cases.
- **Overall Model Robustness**:
    - **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)** evaluates classification confidence across different decision thresholds, providing insights into the model's discriminatory power.
    - **PR-AUC (Precision-Recall Area Under Curve)** is particularly useful for handling class imbalances, ensuring that the model maintains high sensitivity in detecting rare crisis cases.

- **Confidence Score Calibration** is monitored to ensure that predicted probabilities align with actual classification accuracy, enhancing trust in automated decisions.

By integrating **confidence-aware evaluation metrics**, the system ensures that predictions are not only **accurate and reliable** but also **transparent and interpretable**, enabling informed decision-making and effective crisis intervention.

Task 4: Decision-Making & Tradeoffs

1. If you had to choose between different model architectures, how would you decide?

The choice between models depends on a tradeoff between **accuracy, interpretability, latency, and computational cost**. Transformers (e.g., BERT, RoBERTa, DistilBERT) are preferred for high accuracy in sentiment and crisis classification due to their context-awareness and deep semantic understanding. However, if **lower latency is required for real-time processing**, lighter models like DistilBERT or ALBERT may be chosen. For crisis detection, **hybrid models** (LSTMs combined with Transformers) **might be more effective in capturing sequential dependencies**. Benchmarking different models on validation loss, inference time, and deployment constraints helps determine the optimal choice.

2. How do you weigh accuracy vs. computational efficiency?

The tradeoff is managed by considering the real-time constraints of SMS classification and on-premise deployment limitations. **Model quantization** (e.g., using ONNX Runtime or TensorRT) reduces memory footprint and speeds up inference without significantly sacrificing accuracy. Distilled models like DistilBERT provide a good balance of speed and performance, whereas heavier models such as full-sized BERT or GPT-based models may be reserved for offline batch processing. A cascading model approach can be used where a lightweight model first filters critical messages, and only high-risk cases are passed to a more computationally expensive model.

3. How would you optimize the system to run inference efficiently on-premise?

Optimizing inference involves **model compression, parallel processing, and hardware acceleration**. Quantization and pruning reduce model size without major performance loss. Batch inference processing allows multiple messages to be processed simultaneously, reducing redundant computations. **Ray for distributed parallel** execution ensures efficient multi-core processing, while Triton Inference Server enables multi-model deployment. ONNX Runtime and TensorRT optimize model execution, ensuring low-latency predictions on CPUs and GPUs. Additionally, using edge AI hardware such as NVIDIA Jetson for inference acceleration may further improve performance while keeping deployment cost-effective.

4. Would you use rule-based methods in combination with ML? Why or why not?

Yes, a **hybrid rule-based + ML approach** enhances model interpretability and ensures safety in critical classifications. Rule-based keyword detection (e.g., detecting phrases like "I want to end it all") provides an instant fail-safe mechanism for crisis alerts. Combining lexicon-based sentiment analysis with ML-based predictions improves accuracy for short, ambiguous messages. Rules help reduce false negatives in crisis detection, while ML models generalize better across varying sentence structures. This combination ensures robust, explainable, and fail-safe decision-making, making it ideal for real-world deployment in grief support messaging systems.

Task 5: Handling Edge Cases & Uncertainty

1. **How would you handle ambiguous responses?**

   Ambiguous responses are handled using a combination of **context-aware embeddings, sentiment lexicons, and uncertainty estimation techniques**. The model leverages **historical message context** to infer meaning, utilizing **vector-based similarity search (via FAISS or Qdrant)** to compare messages with previously classified cases. Additionally, **Monte Carlo Dropout-based uncertainty estimation** ensures that responses with high variance across multiple inference runs are flagged for **human review**.

   Example #1: A message simply says, "thanks"—is this positive, neutral, or disengaging?

   The model evaluates the **preceding conversation history** to determine intent. If the **prior message was supportive**, "thanks" is classified as **positive**. If it follows an **informational or automated response**, it may be **neutral** or **disengaging**. In cases of uncertainty (confidence < threshold), a **rule-based heuristic or fallback response mechanism** (e.g., prompting for clarification) can be applied.

   Example #2: "I just want to be with him"

   This statement is **contextually dependent and potentially indicative of crisis**. The system uses **entity recognition, and similarity search in a crisis message vector database** to determine the likelihood of distress. If classified as **high-risk with low confidence**, the message is escalated for **human intervention** to prevent misclassification.

2. **What confidence thresholds would you set before escalating a message for human review?**

   - **Sentiment Analysis**: Escalate if model confidence is **below 60%** and sentiment ambiguity is detected.
   - **Response Necessity**: Escalate if **confidence < 50%**, ensuring uncertain cases are reviewed for follow-up.
   - **Crisis Detection**: **High recall is prioritized**, meaning messages with **confidence < 80% or high-risk words** (suicide, harm, depression) are automatically flagged.
   - A **hybrid rule-based + ML approach** ensures **safe failovers** for critical cases.

Task 6: Ethical Considerations & Bias Mitigation

1. **How would you ensure fairness and minimize bias in the model?**

   Bias is minimized by **curating diverse training datasets**, including messages from **different age groups, ethnic backgrounds, and dialects**. **Fairness-aware loss functions (e.g., adversarial debiasing, reweighting strategies)** ensure that **underrepresented groups receive equal model performance**. Regular **bias audits** using explainability tools like **LIME and SHAP** highlight disparities, prompting model retraining if necessary. It is also necessary to follow **Responsible AI** principles.

2. **How do you prevent misclassification of crisis messages?**

   To **minimize false negatives**, the system prioritizes **high recall** by combining **ML-based classification with rule-based keyword detection**. Additionally, **human-in-the-loop verification** for ambiguous crisis messages ensures no urgent case is missed. **Ensemble methods** (e.g., merging BERT predictions with LIWC-based emotion analysis) improve robustness.

3. **How do you handle language variations across different demographics?**

   A **multilingual NLP pipeline** using **XLM-R, mBERT, or language-adapted transformer models** enables accurate classification across different linguistic patterns. The model is **fine-tuned on dialect-specific datasets**, and **code-switching detection** (e.g., Spanglish, Hinglish) ensures messages are correctly interpreted.

4. **How do you ensure user privacy and data security in an on-premise setup?**

   - **Data Anonymization**: All messages are stripped of personally identifiable information (PII) using **Named Entity Recognition (NER)** before processing.
   - **End-to-End Encryption**: Messages are **encrypted at rest and in transit** using AES-256.
   - **Access Control & Auditing**: **Role-based access control (RBAC)** ensures only authorized personnel can review flagged messages.
   - **On-Premise Vector Database**: Storing message embeddings locally in **Qdrant or FAISS** prevents external data exposure.

Task 7: Deployment & Continuous Improvement

1. **How would you maintain and improve model performance over time?**

   A **continuous learning pipeline** is implemented where **newly flagged messages** are manually reviewed and added to a **data versioning system (DVC)**. **Active learning techniques** identify **low-confidence cases**, prioritizing them for human annotation and retraining. **Periodic fine-tuning** ensures the model adapts to evolving language patterns.

2. **What infrastructure would you use to run inference on-premise efficiently?**

   The system is deployed using **Docker and Kubernetes**, ensuring **scalability and fault tolerance**. **Inference is optimized with ONNX Runtime and TensorRT**, reducing computational overhead. **Triton Inference Server** allows multiple models to run concurrently with minimal latency.

3. **How would you monitor model drift and retrain the model?**

- **Prometheus & Grafana** track real-time model performance (e.g., confidence distribution, misclassification trends).
- **Concept drift detection** (e.g., KL divergence monitoring) triggers alerts when the **distribution of incoming messages significantly differs from training data**.
- **Automated retraining workflows** using **MLflow and DVC** ensure continuous model improvement.

4. **What kind of feedback loop would you implement to refine the system?**

   The feedback loop integrates **human review of flagged messages**, where low-confidence predictions are **annotated and used for retraining**. Additionally, **user feedback** (e.g., "Was this response helpful?") refines sentiment and response necessity models. **Ensemble-based retraining strategies** incorporate past feedback for continuous model enhancement.

Task 8: Streamlit-based SMS Classification tool - Sample Output

The provided Python script is a **Streamlit-based SMS classification tool** using **DistilBERT** models to analyze messages. It performs three key tasks:

1. **Sentiment Analysis** – Classifies messages as Negative, Neutral, or Positive.
2. **Response Necessity** – Determines whether a reply is needed.
3. **Crisis Detection** – Assesses if a message indicates a crisis.

Key Features:

- **Pretrained Models**: Uses DistilBERT for classification.
- **Text Preprocessing**: Cleans (text, emojis) and tokenizes messages.
- **Confidence Scores**: Provides probability distributions for each classification.
- **Flagging System**: Flags messages for review if neutral confidence is low or crisis probability is high.
- **Streamlit UI**: Accepts user input and displays results in a structured table.

Metrics:

Sentiment Confidence: A list containing probabilities for each sentiment category (Negative, Neutral, Positive).

- Example: [0.1, 0.7, 0.2] → 10% Negative, 70% Neutral, 20% Positive.

Response Confidence: A list with probabilities for response necessity classification (No Reply Needed, Reply Needed).

- Example: [0.3, 0.7] → 30% No Reply, 70% Reply Needed.

Crisis Confidence: A list with probabilities for crisis detection (No Crisis, Possible Crisis, Crisis Detected).

- Example: [0.6, 0.3, 0.1] → 60% No Crisis, 30% Possible Crisis, 10% Crisis Detected.

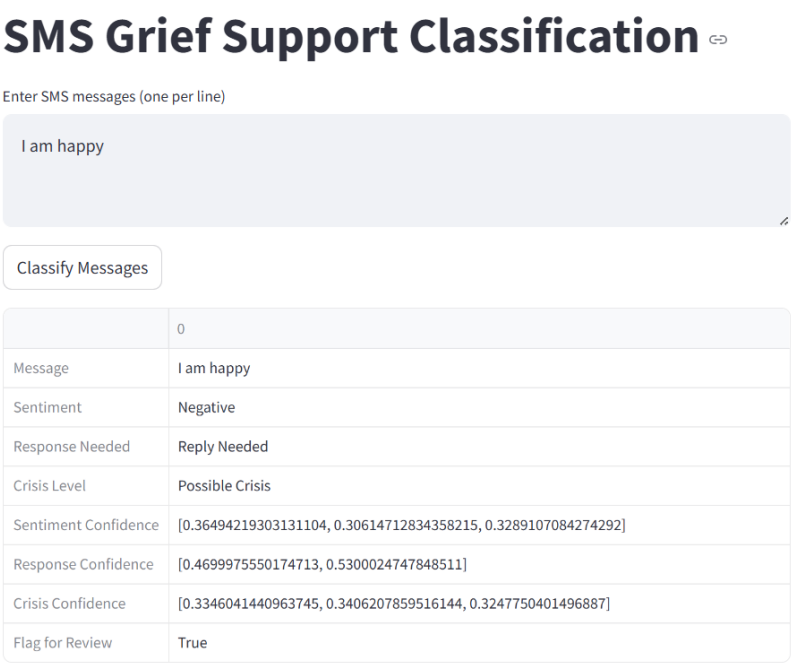

Fig 3. Example 1 of Support Classification model

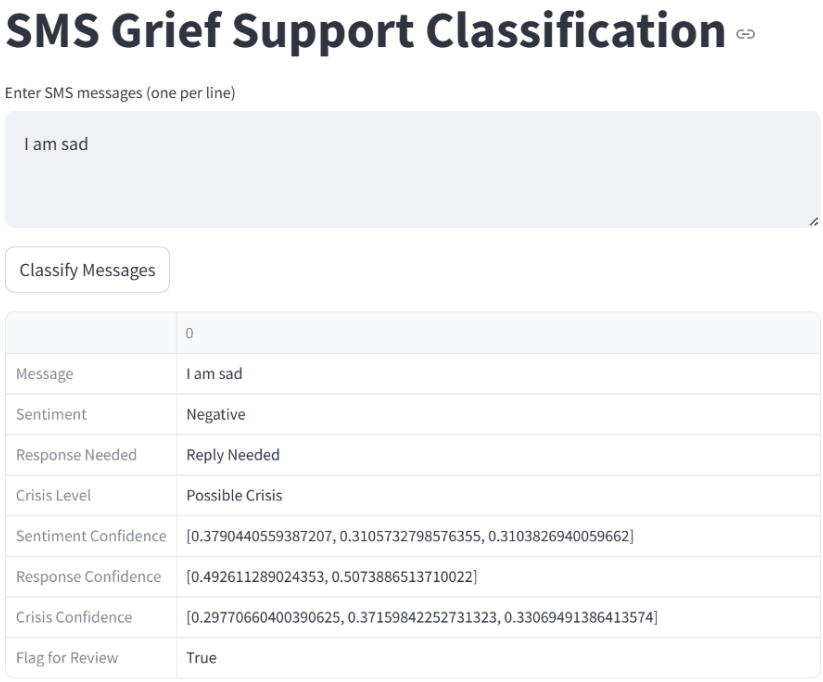

Fig 4. Example 2 of Support Classification model

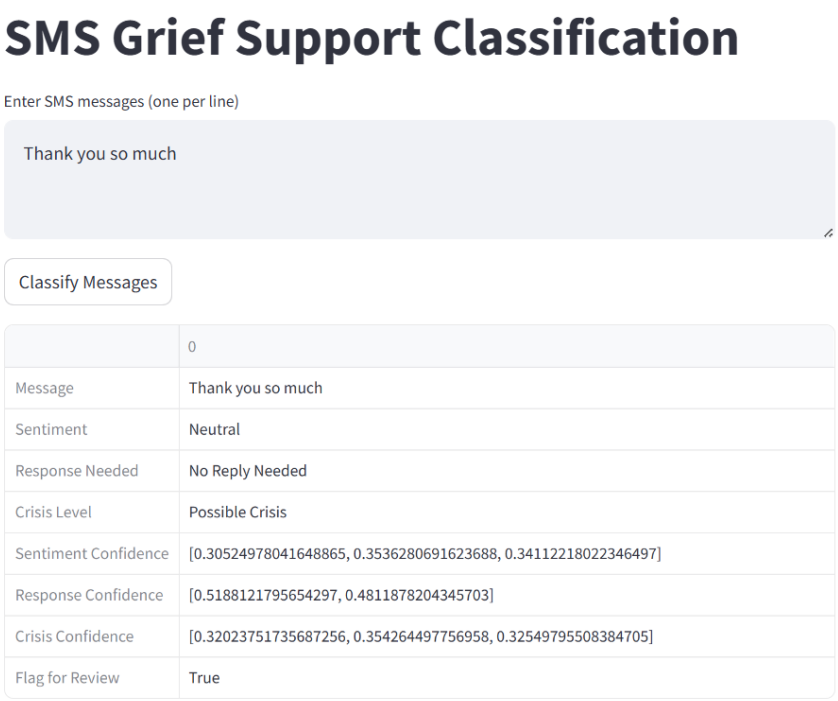

Fig 5. Example 3 of Support Classification model

Conclusion

This document presents a comprehensive technical design for an AI-powered SMS grief support system, integrating sentiment analysis, response necessity classification, and crisis detection. The architecture is modular, scalable, and optimized for real-time processing, ensuring privacy through an on-premise deployment. By leveraging transformer-based NLP models, vector databases for semantic search, and a human-in-the-loop mechanism, the system ensures both efficiency and reliability in handling sensitive messages.

The proposed solution balances accuracy, computational efficiency, and ethical considerations, incorporating bias mitigation strategies and continuous learning mechanisms. With robust monitoring, model retraining workflows, and secure deployment strategies, the system remains adaptive to evolving user interactions while maintaining high precision in crisis detection. Future work could explore further enhancements in context retention, real-time inference optimizations, and multilingual support to expand accessibility and effectiveness.