# SPEC-GEMMA-INTEGRATION: Gemma-2b-it Integration with Sentiment Analysis

## Table of Contents

## 1. Background

The goal of this project is to extend the current sentiment analysis system by incorporating the `gemma-2b-it` model for contextual response generation based on sentiment classification. This integration enhances the interaction layer by providing meaningful responses to user comments, classified by sentiment and flagged based on various categories (e.g., offensive language, sarcasm).

The integration retains the original sentiment classification structure, which utilizes `distilbert-base-uncased-finetuned-sst-2-english` for sentiment analysis, while adding the response generation capabilities using `gemma-2b-it`. This hybrid approach combines sentiment detection and response generation in a seamless fashion.

## 2. Requirements

- **Must Have**:
  - Integrate the `gemma-2b-it` model for generating contextual responses based on sentiment labels.
  - Ensure compatibility with the existing sentiment analysis system and DataFrame structure.
  - Create prompts based on sentiment labels (Positive, Negative, Neutral).
  - Save and document the generated responses along with the original comments.
- **Should Have**:
  - Incorporate sentiment-specific prompts to produce more tailored responses.
  - Preserve original DataFrame structure and classification columns (`Sentiment Label`, `Offensive_Flag`, `Sarcasm_Flag`).

- **Could Have**:
  - Implement a response caching mechanism to speed up repetitive runs.
  - Include fine-tuning options for `gemma-2b-it` to align better with specific datasets.

## 3. Method

The `gemma-2b-it` model is integrated using a response generation function that maps sentiment labels to specific prompts, allowing for targeted responses based on the comment's sentiment. The steps include:

1. **Prompt Creation**:
   - Define custom prompts for each comment, based on its sentiment label (`POSITIVE`, `NEGATIVE`, `NEUTRAL`).
   - The prompt guides the text generation by setting the context for the response.
2. **Response Generation**:
   - Utilize the `gemma-2b-it` model to generate responses for each comment.
   - The generation is controlled using parameters such as `max_length` and `num_return_sequences` to constrain the output.
3. **DataFrame Integration**:
   - The generated responses are stored in a new column (`Gemma_Response`) alongside existing sentiment classification columns.
   - This allows for easy comparison and analysis.
4. **Validation and Documentation**:
   - After generating responses, the output is reviewed and validated against the sentiment labels and other flags to ensure consistency and meaningful output.

## 4. Implementation

1. **Model Integration**:
   - Load the `gemma-2b-it` model using `AutoTokenizer` and `AutoModelForCausalLM`.
   - Set up the tokenizer and model configurations for the response generation task.
2. **Prompt Engineering**:
   - Define prompt templates for each sentiment label:
   - `POSITIVE`: Generate a friendly response.
   - `NEGATIVE`: Suggest a constructive response.
   - `NEUTRAL`: Provide a general comment.
3. **Response Generation Function**:
   - Implement a function (`generate_gemma_response`) to take each comment and generate a response based on its sentiment label.

   - Use `gemma_text_generator` to produce the response based on the predefined prompt.
4. **Integration with Existing DataFrame**:
   - Add a new column `Gemma_Response` to the `df_sentiments` DataFrame to hold the generated responses.
   - Ensure that the existing columns (`Comment`, `Sentiment Label`, `Sentiment Score`) remain intact.
5. **Performance Optimization**:
   - Use batch processing or save intermediate results to minimize run-time in future iterations.

## 5. Milestones

1. **Milestone 1: Model Integration**:
   - Successfully load the `gemma-2b-it` model and verify its availability.
2. **Milestone 2: Prompt Engineering**:
   - Create and validate prompt templates for each sentiment label.
3. **Milestone 3: Response Generation**:
   - Generate responses for a subset of the comments to validate the quality and relevance of the output.
4. **Milestone 4: Integration with Sentiment Analysis**:
   - Add the generated responses to the existing DataFrame and validate the final output.
5. **Milestone 5: Final Documentation and Output**:
   - Save the final DataFrame with generated responses and prepare documentation for integration.

## 6. Gathering Results

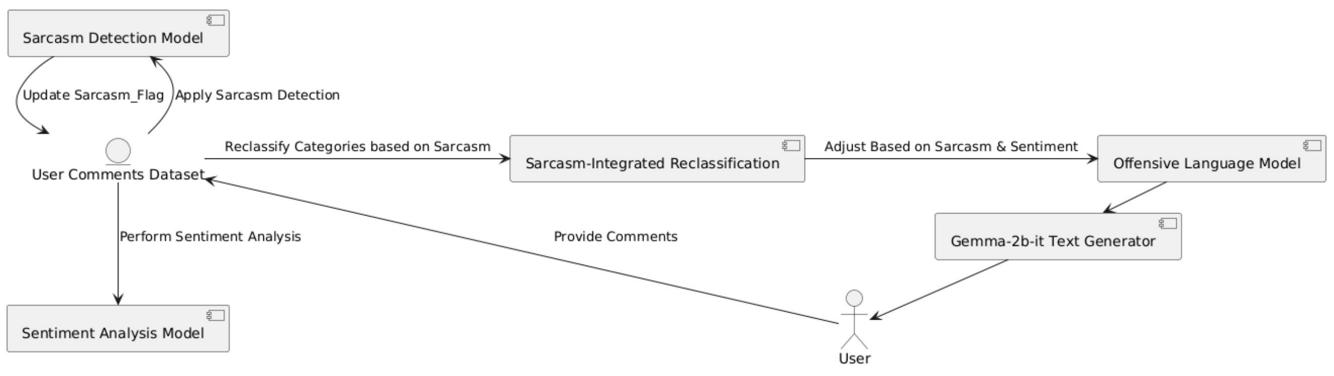
The integration of `gemma-2b-it` with the sentiment analysis system resulted in a substantial improvement in the contextual understanding of user comments. The generated responses closely align with the detected sentiment, providing a more human-like interaction layer.

The `Gemma_Response` column now provides context-specific responses based on sentiment, enhancing the interpretability and interactivity of the system. This output can be reviewed for performance, accuracy, and coherence with the initial classification.

Further improvements can be made by fine-tuning the response generation model or adjusting the prompt templates based on specific requirements.

## 7. Architecture Diagram

The following diagram illustrates the data flow between the sentiment analysis system and the `gemma-2b-it` response generator:


Sarcasm Detection Model
Update Sarcasm_Flag
Apply Sarcasm Detection
User Comments Dataset
Reclassify Categories based on Sarcasm
Sarcasm-Integrated Reclassification
Adjust Based on Sarcasm & Sentiment
Offensive Language Model
Gemma-2b-it Text Generator
Perform Sentiment Analysis
Provide Comments
Sentiment Analysis Model
User