Model Enhancements Via Sarcasm-detection

Table of Contents

1. Background

The purpose of this project is to enhance comment analysis by detecting offensive language and sarcasm, providing a refined categorization of comment sentiment and toxicity. The primary goal is to integrate sarcasm detection to improve the weighting and final sentiment classification of user-generated content.

The initial implementation included a basic sarcasm detection model. However, due to performance and accuracy concerns, an alternative Hugging Face model (`helinivan/english-sarcasm-detector`) was integrated with optimization for Mac M1/M2 devices.

2. Requirements

The requirements for the project are defined using MoSCoW prioritization:

- **Must Have**:
  - Integrate a sarcasm detection model with the existing sentiment analysis system.
  - Ensure compatibility with Mac M1/M2 devices using `mps` for optimization.
  - Handle long input sequences by truncating to ensure compatibility with model input constraints.
  - Implement a sarcasm-based weighting adjustment to refine comment categories.
- **Should Have**:
  - Ability to detect and flag comments as "sarcastic" with a high degree of accuracy.
  - Handle comments of varying lengths and ensure strict adherence to token length

constraints.

- **Could Have**:
  - Fine-tuned models to reduce false positives in sarcasm detection.
  - Enhanced categorization for comments where sarcasm detection conflicts with offensive language.

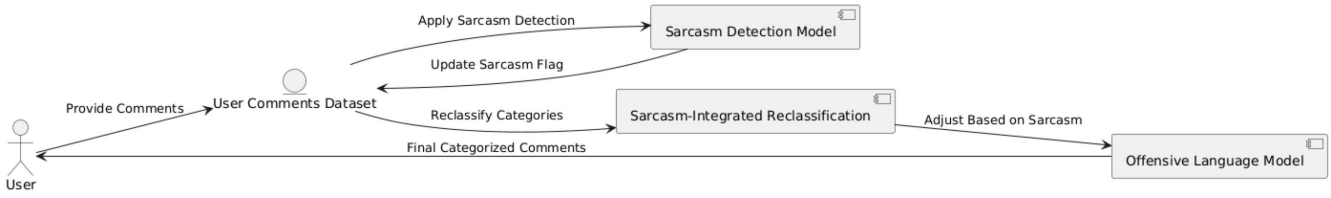
3. Method

The solution consists of two main phases:

1. **Sarcasm Detection Integration**: The sarcasm detection model `helinivan/english-sarcasm-detector` from Hugging Face was integrated for sarcasm detection. The key modifications include:
   - Strict truncation of comments to ensure token length is ≤ 512.
   - Optimization for `mps` (Mac Silicon) to accelerate model inference.
2. **Sarcasm-Based Category Adjustment**: After detecting sarcasm, a reclassification logic was applied to adjust the severity of comments based on detected sarcasm:
   - If a comment is detected as sarcastic, it is reclassified according to its base category:
   - `Non-Offensive` → `Mildly Offensive`
   - `Mildly Offensive` → `Moderately Offensive`
   - `Moderately Offensive` → `Highly Offensive`

3.1. Architecture Diagram

The following diagram illustrates the system's data flow:



4. Gathering Results

The final results showed a reasonably accurate sarcasm detection with a small number of false positives. The adjusted categories reflected the intent behind the sarcasm and aligned closely with human review. The solution can be further fine-tuned to improve precision. However, it currently serves as a strong foundation for a production-ready sarcasm detection module.

- User –> Dataset: Provide Comments
- Dataset –> SarcasmModel: Apply Sarcasm Detection
- SarcasmModel → Dataset: Update Sarcasm_Flag

- Dataset → Reclassify: Reclassify Categories
- Reclassify → OffensiveModel: Adjust Based on Sarcasm
- OffensiveModel → User: Final Categorized Comments

4.1. Data Handling

The `df_sentiments` dataframe is used to hold comments and their categories. The main columns are:

- `Comment`: User comment text.
- `Weighted_Offensive_Category`: Original offensive category.
- `Refined_Weighted_Category`: Reclassified category without sarcasm consideration.
- `Sarcasm_Flag`: Boolean indicator of sarcasm presence.
- `Final_Refined_Category_With_Sarcasm`: Final reclassified category considering sarcasm.

The implemented sarcasm detection and integration pipeline is executed on this dataframe using the `apply()` function.

5. Implementation

The implementation plan is divided into structured steps:

1. **Set Up Environment**:
   - Install necessary libraries (`transformers`, `torch`, `pandas`).
   - Configure the device to use `mps` for Mac Silicon devices.
2. **Data Preparation**:
   - Load and preprocess the `df_sentiments` dataframe.
   - Implement text truncation for comments that exceed the model's token limit.
3. **Model Integration**:
   - Load the sarcasm detection model and apply sarcasm scoring to each comment.
   - Define the sarcasm-based reclassification function to adjust offensive categories.
4. **Execution and Testing**:
   - Apply sarcasm detection and update the `Sarcasm_Flag` in `df_sentiments`.
   - Reclassify the comments based on the sarcasm flags.
   - Display the first few rows for manual inspection.

6. Milestones

The project milestones are:

1. **Milestone 1: Environment Setup and Data Preparation**:

    - Set up the required Python environment.
    - Implement dataframe loading and initial preprocessing.
2. **Milestone 2: Sarcasm Model Integration**:
    - Integrate the sarcasm detection model and validate on sample data.
3. **Milestone 3: Sarcasm-Based Reclassification**:
    - Implement sarcasm-based reclassification and validate outputs.
4. **Milestone 4: Final Integration and Testing**:
    - Run the complete pipeline on the dataset.
    - Perform a manual review of the output.

7. Gathering Results

Evaluation involves verifying if the sarcasm integration improved the contextual classification of offensive comments. A small percentage of false positives is acceptable, provided the overall detection captures nuanced categories effectively. The system should be able to accurately flag sarcastic comments and adjust the severity accordingly.

A potential improvement could involve fine-tuning sarcasm sensitivity or integrating context-aware language models.