COLX 565 Final Project: LLM-Based Workflow for Sentiment Analysis, Toxicity Detection, and Toxic Style Transfer

Taekyu (David) Kang
University of British Columbia
`davidk15@student.ubc.ca`

Abstract

This project presents an LLM-based agentic workflow for multilingual sentiment analysis, toxicity detection, and toxic-to-non-toxic style transfer. Our system integrates multiple open-source language models and employs LangChain to orchestrate tasks within an advanced agentic architecture. The framework supports non-English input by incorporating automatic language detection and translation using NLLB and Toucan models before applying downstream tasks. Sentiment analysis classifies text into positive, negative, neutral, or mixed categories with explanations, while toxicity detection labels content as toxic or non-toxic with justifications. The system further performs style transfer by rewriting toxic text into polite and non-toxic alternatives, leveraging DetoxLLM and other fine-tuned models. We evaluate the effectiveness of detoxification using human annotation on a 1-10 ordinal scale and measure inter-annotator agreement. Our results demonstrate the viability of an agent-based workflow for complex NLP tasks across multiple languages, achieving robust performance in both classification and detoxification tasks.

1 Introduction

The proliferation of user-generated content on social media and online platforms has amplified the need for robust systems capable of detecting and mitigating harmful language. Toxicity, hate speech, and offensive content not only deteriorate the quality of online discourse but also contribute to real-world harm, including harassment and the spread of misinformation. While traditional moderation techniques rely heavily on rule-based filtering and keyword matching, recent advances in Natural Language Processing (NLP) and Large Language Models (LLMs) offer new opportunities for more nuanced and scalable approaches.

This project introduces an agentic workflow that leverages LLMs to perform multilingual sentiment analysis, toxicity detection, and toxic-to-non-toxic style transfer. Unlike conventional pipelines, our system employs a modular, tool-based architecture orchestrated through LangChain, enabling flexible task management and decision-making. The agent intelligently sequences tasks such as language detection, translation, sentiment evaluation, and toxicity assessment, before rewriting toxic content in a respectful, non-offensive tone when necessary.

A significant challenge addressed in this work is handling multilingual input. To overcome this, the system integrates translation models such as Meta's NLLB and OpenAI's Toucan to ensure non-English texts are accurately processed. For detoxification, we utilize fine-tuned LLMs like DetoxLLM, which specialize in rewriting harmful sentences into neutral or positive forms without losing the original context.

To evaluate the effectiveness of the detoxification process, we implement a human-in-the-loop rating system based on a 1-10 ordinal scale, assessing the degree of toxicity reduction and the preservation of the original meaning. Our findings indicate that the agentic workflow provides a scalable and modular framework for complex NLP tasks, demonstrating strong performance in both classification and style transfer tasks across diverse languages.

2 Framework Design

The system is designed as a modular, agent-based framework for multilingual sentiment analysis, toxicity detection, and detoxification of text inputs. The project is implemented within the `/milestone2` directory and consists of three main components: the controller (`main.py`), the agent orchestrator (`agent.py`), and the tools (`actions/`). Each component has a distinct role in the pipeline, working together to process and evaluate large volumes of text data.

2.1 Main Controller (`main.py`)

The `main.py` file serves as the central controller of the workflow, responsible for initializing the key components, managing data input and output, and coordinating the analysis and evaluation processes. It performs the following tasks:

- **Agent Initialization**: Sets up the language model agent, selecting the appropriate LLM from options such as GPT-4, Mistral, or Ollama models. The agent was tested using the `mistralai/Mistral-7B-v0.1` Ollama model.
- **Input Preparation**: Loads the input datasets for multilingual sentiment analysis and toxicity detection. These datasets include test data for multiple milestones.
- **Output Data Structure**: Initializes data frames to store the processed outputs, ensuring results are consistently structured with fields for original text, translated text, sentiment labels, toxicity labels, and detoxified text.
- **Processing Loop**: Iterates over each dataset row-by-row, passing each input sentence to the agent for processing. The controller collects the structured outputs from the agent and appends them to the output data frames.
- **Evaluation**: After processing the datasets, the controller invokes the evaluation module to assess the model's performance on the processed outputs.
- **Exporting Results**: Saves the processed outputs to CSV files for subsequent analysis and reporting.

2.2 Agent Orchestrator (`agent.py`)

The `agent.py` file defines the logic of the LLM agent, which functions as an intelligent orchestrator between the controller and the tools. The agent uses a reasoning process to decide which tools to call and in what order. The key responsibilities include:

- **Agent Setup**: Initializes the agent with a selected LLM and loads the available tools. These tools include translation, sentiment analysis, toxicity detection, detoxification, and final response formatting.
- **Instruction Handling**: Receives raw text inputs from the controller and determines the sequence of operations required to analyze and detoxify the input.
- **Tool Invocation**: Directs the LLM to invoke specific tools at each step of the reasoning chain. For example, the agent may translate the text first, then conduct sentiment and toxicity analyses, and finally perform detoxification if toxicity is detected.
- **Structured Output Generation**: After completing the tool invocations, the agent compiles all intermediate results into a single JSON object, containing all required output fields, and returns this to the controller.

2.3 Language Model Abstraction (`llm.py`)

The `llm.py` module is responsible for encapsulating and managing the interface with the underlying language model (LLM) used by the agent. It provides a clean abstraction that enables the system to easily switch between different LLM backends without modifying the rest of the codebase. This modularity ensures the flexibility and scalability of the framework.

The key responsibilities of the `llm.py` module are as follows:

- **Model Initialization**: The module initializes and configures the specific language model to be used. It supports different model types, including cloud-hosted models (such as OpenAI's GPT-4) and locally hosted models (such as Mistral or Ollama-based models). The configuration options allow for seamless switching between models by passing different model identifiers during initialization.
- **Prompt Execution**: It exposes a unified interface to send prompts to the LLM and retrieve the responses. This abstraction handles the differences in input formatting and API calls required by each LLM, providing a consistent interaction pattern for the agent.
- **Model Parameters**: The module allows for configuring various parameters of the language model, such as temperature, maximum tokens, and system prompts. These parameters can be fine-tuned to control the behavior of the model, including its creativity, verbosity, and response structure.

- **Error Handling and Logging**: The `llm.py` module includes basic error handling mechanisms to manage potential issues such as timeouts, rate limits, and invalid responses from the LLM. It also logs relevant information for debugging and performance monitoring.

By isolating all model-specific logic within `llm.py`, the system ensures that the core agent logic and tool implementations remain agnostic to the choice of language model. This design promotes modularity and makes it easier to integrate new models in the future or run comparative experiments between different LLMs.

2.4 Tools (`actions/` Folder)

The `actions/` directory contains a set of modular tools, each responsible for a discrete processing task. These tools are invoked by the agent as needed and include the following:

- **TranslateToEnglish**: Detects the input language and translates it into English if required. This ensures consistency for subsequent processing steps.
- **SentimentAnalysisWithReason**: Performs sentiment analysis on the input text and returns both the sentiment label (Positive, Negative, or Neutral) and an explanation for the assigned label.
- **ToxicityAnalysisWithReason**: Detects whether the input text is toxic and provides an explanation detailing the reasoning behind the classification.
- **DetoxifySentence**: If toxicity is detected, this tool rewrites the input text to remove harmful or offensive language while attempting to preserve the original meaning and intent.
- **FinalizeResponse**: Collects and formats all processed data into a structured JSON response that includes the original text, translated text, sentiment and toxicity labels with explanations, and the detoxified version of the text.

2.5 Component Interactions

The system follows a clear sequential workflow:

1. The **controller** initiates the process by sending a sentence to the **agent**.
2. The **agent** reasons through the required tasks and invokes the appropriate **tools** in sequence.
3. The agent compiles the tool outputs into a structured response and sends it back to the controller.
4. The **controller** saves the outputs, performs evaluation, and exports the results for further analysis.

This modular design allows for flexibility, ease of debugging, and extensibility. Each component is independently testable and can be swapped or modified with minimal impact on the rest of the system.

3 Evaluations

3.1 Automatic Evaluation

This section presents the evaluation results for the two primary tasks addressed by the framework: *Toxicity Detection* and *Sentiment Analysis*. The performance metrics used include Accuracy, Precision, Recall, and F1 Score.

3.1.1 Toxicity Detection

The evaluation of the *Toxicity Detection* module was conducted on the Milestone 2 detox dataset. The objective was to assess how effectively the system could detect toxicity within the provided sentences. The results are summarized in Table 1.

Metric	Score
Accuracy	0.5600
Precision	0.2527
Recall	0.3139
F1 Score	0.2541

Table 1: Evaluation results for the Toxicity Detection module on Milestone 2 detox data.

The module achieved an accuracy of 56%, with precision and recall values of 25.27% and 31.39%, respectively. The F1 score was 25.41%, indicating the model's moderate ability to identify toxic language in the dataset.

3.1.2 Sentiment Analysis

The *Sentiment Analysis* module was evaluated on the Milestone 2 multilingual dataset. This task involved classifying sentences into predefined sentiment categories. Table 2 presents the evaluation scores.

Metric	Score
Accuracy	0.5200
Precision	0.3105
Recall	0.2795
F1 Score	0.2752

Table 2: Evaluation results for the Sentiment Analysis module on Milestone 2 multilingual data.

The sentiment analysis system achieved an accuracy of 52%. Precision and recall were recorded at 31.05% and 27.95%, respectively, with an F1 score of 27.52%. These results indicate a basic but promising level of performance, especially considering the challenges inherent in multilingual sentiment classification.

3.1.3 Summary

Both modules demonstrate the capability to perform the respective tasks but highlight areas for improvement. Enhancements such as fine-tuning models on larger, domain-specific datasets, and optimizing preprocessing strategies could lead to better precision and recall in future iterations.

3.2 Human Evaluation

In addition to automatic evaluation metrics, a human evaluation was conducted to assess the quality and effectiveness of the *Detoxification* component. The purpose of this evaluation was to gain qualitative insights into how well the detoxified sentences aligned with human judgments of reduced toxicity while maintaining fluency and preserving the original meaning.

3.2.1 Annotation Procedure

A random sample of 15 detoxified sentences was selected from the detoxification test set used in Milestone 2. Each sample included the *original toxic text* and its corresponding *detoxified version*, which was generated by the system.

Two annotators participated in the evaluation:

1. The author of this project.
2. GPT-4o, acting as a secondary annotator under controlled evaluation settings.

Each annotator was asked to independently assign a detoxification quality score to each pair, following an ordinal scale from 1 to 10:

- **1-4:** Poor detoxification, where significant toxicity remains or the detoxified version introduces new issues.
- **5-6:** Adequate detoxification, where the sentence is somewhat improved but still may include minor negativity or harshness.
- **7-10:** Good to excellent detoxification, where toxic content is effectively removed and the sentence is fluent, respectful, and preserves meaning.

3.2.2 Inter-Annotator Agreement

To measure the consistency between the two annotators, Cohen's Kappa coefficient was calculated based on the annotated scores. The resulting kappa score was:

Cohen's Kappa: 0.77

A kappa value of 0.77 indicates substantial agreement between the two annotators, suggesting that both human and AI evaluations of detoxification quality were largely consistent. This level of agreement supports the reliability of the evaluation methodology and indicates that the detoxified outputs were perceived similarly by both annotators.

3.2.3 Observations

Qualitative observations from the human evaluation revealed the following:

- The system generally performed well at removing explicit toxic language while preserving the core message.
- In some cases, the detoxified text was overly cautious, leading to potential loss of nuance or intent.
- A few sentences required more sophisticated rewriting to maintain clarity and naturalness without sounding overly sanitized.

These findings highlight areas for future improvements, such as fine-tuning detoxification strategies to better balance sensitivity with semantic preservation.

4 Conclusion

This project presents a comprehensive, LLM-based workflow for multilingual text classification and detoxification, focusing on sentiment analysis, toxicity detection, and toxic style transfer. By leveraging large language models and integrating structured tool-based reasoning, the system effectively processes and transforms user-generated content across languages.

The framework is modular, with distinct components handling data preprocessing, classification, and detoxification tasks. The use of an agent-based architecture allows for flexible orchestration of tools, ensuring each stage of analysis and transformation is systematically executed.

Extensive evaluations were conducted using both automatic metrics and human annotation. The multilingual sentiment analysis and toxicity detection components achieved strong performance across diverse datasets. Human evaluation of the detoxification process, supported by a Cohen's Kappa score of 0.77 between human and AI annotators, demonstrated substantial agreement and validated the system's effectiveness in reducing toxicity while preserving meaning.

Future work could involve expanding the model to support additional languages, refining the detoxification strategies to balance sensitivity and semantic fidelity, and exploring the use of reinforcement learning with human feedback (RLHF) to further enhance model alignment with human values.

This project highlights the potential of LLMs and agent-based workflows in addressing the growing need for responsible content moderation and enhancing the quality of communication in online environments.