Milestone 1 Report: Multilingual Sentiment and Toxicity Analysis

Lingsong Zeng[*]
arnozeng@outlook.com
lingsong

Yuchen Li[*]
yuchenli.cn@gmail.com
yyyuchen

COLX_565_final_project

Abstract

In this report, we describe the approach and implementation of a combined sentiment analysis and toxicity detection task using large-language models (LLM). The task involves classifying text into positive, negative, or mixed sentiment and identifying whether text is toxic or non-toxic with explanation. We describe the selection of the model, the integration of tasks and implementation details, followed by the results of the evaluation and the observed challenges.

1 Overall Approach

Our approach involves using a pre-trained causal language model for both sentiment analysis and toxicity detection. The architecture is as follows.

1. Use a pre-trained model `granite-3.0-2b-instruct` for text classification and generation tasks.

2. Define two distinct prompts: one for sentiment analysis and the other for toxicity detection.

3. Build a Langchain framework to process each sentence in the data set with the model, generating both the classification label and an explanation for the label.

4. Use evaluation metrics such as accuracy, precision, recall, and F1 score to measure model performance.

2 Integration of Sentiment and Toxicity Detection

To integrate the sentiment and toxicity detection tasks, we used similar structures for both tasks. For each task, we create a prompt that instructs the model to classify the sentence and provide an explanation in a structured format.

For sentiment analysis, the prompt template was:

> Question: Explain why the following sentence is classified as positive, negative, or mixed: {sentence}. Please give me your class: positive, negative, or mixed and provide your explanation within 50 words as followed structure: 'The sentence is ... (positive, negative, or mixed). ... (your explanation)'

For toxicity detection, the prompt template was modified as follows:

> Question: Explain why the following sentence is classified as toxic or non-toxic: {sentence}. Please give me your class: toxic or non-toxic and provide your explanation within 50 words as followed structure: 'The sentence is ... (toxic or non-toxic). ... (your explanation)'

Each task was processed by iterating through the sentences, generating model predictions, and extracting the relevant information from the generated output using regular expressions.

3 Implementation Details

The code for sentiment and toxicity detection was implemented using the following libraries:

- `transformers` for loading pre-trained models and tokenizers.
- `langchain` for prompt management.
- `sklearn` for evaluation metrics.
- `pandas` for handling data.
- `tqdm` for progress tracking.
- `google.colab` for mounting Google Drive.

The environment was set up in Google Colab with the necessary libraries installed using the following commands:

[*] These authors contributed equally to this project and are listed in alphabetical order by first name.

```
!pip install transformers langchain
    sklearn tqdm
```

The model was initialized with the following code:

```
from transformers import
    AutoModelForCausalLM,
    AutoTokenizer

model = AutoModelForCausalLM.
    from_pretrained("ibm-granite/
    granite-3.0-2b-instruct",
    device_map="cuda")

tokenizer = AutoTokenizer.
    from_pretrained("ibm-granite/
    granite-3.0-2b-instruct")
```

The dataset was loaded and processed as follows:

```
import pandas as pd
df = pd.read_csv("/content/drive/
    MyDrive/565FinalProject/
    eng_sentiment_test_solutions.csv")
```

After generating results, we evaluated the model's performance using accuracy, precision, recall, and F1-score. The following code was used for evaluation:

```
from sklearn.metrics import
    accuracy_score, precision_score,
    recall_score, f1_score
accuracy = accuracy_score(merged_df["
    class-label"], merged_df["
    sentiment_label"])
precision = precision_score(merged_df
    ["class-label"], merged_df["
    sentiment_label"], average="
    weighted")
recall = recall_score(merged_df["class
    -label"], merged_df["
    sentiment_label"], average="
    weighted")
f1 = f1_score(merged_df["class-label
    "], merged_df["sentiment_label"],
    average="weighted")
```

4 Evaluation and Results

The model was evaluated on the provided datasets for both sentiment analysis and toxicity detection tasks using the metrics described above. Preliminary results for each task are as follows:

Sentiment Analysis:

Accuracy: 0.8400
Precision: 0.8764
Recall: 0.8400
F1 Score: 0.8407

Toxicity Detection:

Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00

These results are promising, though there is room for improvement in handling more nuanced sentence structures and cases with mixed sentiment or subtle toxicity.

5 Challenges and Limitations

Some challenges and limitations encountered during the project include:

- Running the model locally was constrained by insufficient memory, while cloud-based execution was limited by quota restrictions, making testing and development challenging. This led us to integrate the Granite model from Hugging Face into a local LangChain-based workflow.
- The model's accuracy was suboptimal, requiring improvements through model selection, prompt engineering, and temperature adjustments. Modifying the prompt structure helped provide clearer context for classification, reducing ambiguity, while adjusting the temperature influenced the balance between deterministic and diverse responses, improving overall reliability.
- Sentences with ambiguous sentiment or toxicity labels were often misclassified.
- The reliance on regular expressions for extracting labels and explanations sometimes led to inaccuracies in retrieving the correct explanations.

In future iterations, more advanced techniques such as fine-tuning the model or further refining the prompt structure could be explored to enhance classification performance.

6 Conclusion

This project demonstrates the feasibility of using large language models for combined sentiment and toxicity detection tasks. By using pre-trained models and prompt-based classification, we were able to classify sentences and provide explanations efficiently. Further refinement of the model and evaluation methods will be needed to address the challenges faced during this work.

7 Code

The code for this project can be found in our github: `COLX_565_final_project`, which run end-to-end on the provided datasets.