Toxic Comment Classification Report

Introduction

This project aims to tackle the challenge of identifying toxic comments on the web by building and comparing three different deep learning models: **LSTM, GRU, and DistilBERT.** The models are trained to classify comments from Wikipedia into six toxicity categories: **toxic**, **severe_toxic**, **obscene**, **threat**, **insult**, and **identity_hate**. Since the dataset is imbalanced, we prioritize the **macro-averaged F1 score** to ensure that each category is treated equally, regardless of how common or rare it is.

Dataset

We use the **Jigsaw Toxic Comment Classification Challenge** dataset, available on Kaggle. The dataset comes in two main files:

train.csv: The train.csv file contains **159,571 comments** and their corresponding labels. This data was used for training and was further split into an 80% training set and a 20% validation set to evaluate model performance during training. The dataset exhibits a significant class imbalance, with the 'toxic' category being the most prevalent, while other toxicity categories have much lower representation. The class distribution is as follows:

- toxic: 9.58% positive
- severe_toxic: 1.00% positive
- obscene: 5.29% positive
- threat: 0.30% positive
- insult: 4.94% positive
- identity_hate: ~0.88% positive

test.csv: This file contains 153,164 comments but does not include labels, and it’s used to test our models to make predictions on unseen data.

Visualizations

Various visualizations were utilized to understand the data's structure and patterns:

- **Label Frequency Analysis:** A bar chart illustrates the frequency of each toxicity label, revealing class imbalances. Toxic label has the highest frequency, and threat has the lowest frequency.
- **Correlation Analysis:** A cross-correlation matrix was used to visualize the relationships between different toxicity labels, highlighting potential co-occurrences. A strong positive correlation was observed between obscene and insult labels, suggesting that obscene comments are likely to be insulting as well.

- **Word Cloud Visualization:** Word clouds were created to visualize the most common words associated with each toxicity label, providing insight into the language patterns of toxic comments.

Data Examples

Specific examples of toxic and non-toxic comments were examined to provide a qualitative understanding of the data. This step helped in grasping the nature and context of toxic language in Wikipedia talk page comments.

Word Cloud Visualization

Word clouds were generated for each toxicity label to visualize the most common words associated with each class:

- Toxic comments were associated with words like "s.ck" and "f.ck".
- Threat comments were mostly related to words like "d.e", "will k.ll", "going to k.ll".
- Identity hate comments included nationalities and specific groups based on their religion, ethnicity, sexual orientation, or gender identity.

These visualizations provided valuable insights into the language patterns of toxic comments.

Models

LSTM Model

The **LSTM (Long Short-Term Memory)** model is designed to capture long-range dependencies in the text. By using a **bidirectional LSTM**, the model can process the text both forwards and backwards, giving it a better understanding of the context. We stack multiple LSTM layers to improve the depth of feature extraction. The training process uses **AdamW optimization** with **learning rate scheduling and dropout** , and we monitor both the **loss** and **Macro-F1 score**.

GRU Model

The **GRU (Gated Recurrent Unit)** model is similar to LSTM but uses fewer parameters, making it more efficient. Like the LSTM, it uses a **bidirectional architecture** to capture contextual information and includes **layer normalization** to stabilize training. The model also uses **AdamW optimization**, **learning rate scheduling and dropout** with a focus on improving both loss and **Macro-F1 score**.

DistilBERT Model

DistilBERT is a small, fast, and light Transformer model based on the BERT architecture. It is trained using knowledge distillation, where a smaller model (DistilBERT) learns to mimic the behavior of a larger, pre-trained BERT model. In this project, we freeze all the layers of the base DistilBERT model and only fine-tune the classification head to our task. DistilBERT is fine-tuned for multi-label classification.DistilBERT is fine-tuned for multi-label classification. It helps preserve the pre-trained knowledge and prevents catastrophic forgetting, where the model loses its general language understanding during fine-tuning on a smaller, task-specific dataset and keeping the base model's weights fixed. We use **AdamW optimization and learning rate scheduling**, with the Macro-F1 score as our evaluation metric as well.

Training and Evaluation

Training Process

Each model was trained using the train.csv dataset, with an 80/20 split for training and validation. During training, we use 5 epochs and we aim to minimize the **loss** and maximize the **Macro-F1 score**. The following strategies were employed during training:

- **Early stopping** ensures training halts when the model's performance plateaus or worsens on the validation set.
- **AdamW Optimizer and Learning Rate Scheduling:** The AdamW optimizer was utilized with ReduceLROnPlateau learning rate scheduling for dynamic adjustment and improved convergence.
- **Dropout Regularization:** Dropout was implemented in both the LSTM and GRU models to further mitigate overfitting by randomly dropping units during training. This technique helps prevent the models from relying too heavily on specific features and encourages them to learn more robust representations.
- **Model checkpoints** allow us to save and later restore the best-performing versions of our models. Each model has a unique folder for outputs and the saved models can be found in the ***checkpoints*** subfolder.
- **Learning curves** for macro-f1 and loss are plotted to track the training process and help visualize progress and they can be in the ***learning_curves*** subfolder for each model.

Testing

After training, each model is tested on the **test.csv** data to make predictions. Due to the imbalanced dataset, where 'toxic' labels are prevalent, models tend to predict accordingly.

Error Analysis

After the models are trained, we conduct an **error analysis** to understand where they make mistakes. We review **false positives** (comments that were incorrectly labeled as toxic) and **false negatives** (toxic comments that were missed). This helps us identify weaknesses in the models and gives us insights into how they can be improved. We can find the results in the ***error_analysis*** folder for each model.

Findings from the Error Analysis

- **False positives**: These occur when the model falsely classifies a non-toxic comment as toxic. This could be due to the model overreacting to certain keywords or phrases.
- **False negatives**: These happen when the model misses a toxic comment. This suggests that the model needs to be more sensitive in identifying toxic language.

We also manually annotated **10 comments** and analyzed the results, the toxic labels were also predominant.

Results and Comparison

Model Comparison

The primary evaluation metric used was the macro-averaged F1 score, which gives equal weight to each toxicity category and is suitable for imbalanced datasets.

Key Insights

- **LSTM** achieved a lower performance compared to both GRU and DistilBERT. LSTMs have a more complex architecture compared to GRUs, making them prone to overfitting, especially with limited or imbalanced data. LSTMs can also struggle to capture long-range dependencies crucial for understanding context.
- **GRU** outperformed LSTM and DistilBERT in terms of the macro-averaged F1 score on the validation set. GRU's efficient architecture, incorporating bidirectional processing, layer normalization, and attention, may have contributed to its superior performance in capturing contextual information and sequential dependencies in the toxic comments.
- **DistilBERT,** despite being a powerful pre-trained model, did not achieve the highest performance with the limited fine-tuning strategy employed. Freezing the base model layers in DistilBERT might have restricted its ability to adapt fully to the nuances of the toxic comment dataset, leading to lower performance compared to GRU.

Conclusion

In this project, we built and compared three different deep learning models for toxic comment classification. **GRU** achieved the highest macro-averaged F1 score, outperforming both LSTM and DistilBERT. However, due to the **imbalanced nature of the dataset**, all models exhibited a tendency to predict the 'toxic' class more frequently. To mitigate overfitting and optimize training, techniques such as **early stopping, model checkpoints, dropout, the AdamW optimizer, and learning rate scheduling were employed.**

Further research could explore more advanced methods for addressing data imbalance to improve overall classification performance and try different fine-tuning techniques and training with different parameters.