Toxic Comment Classification
—Deep Learning Project Report

Guangjie Li, Xuecheng Wu

January 2025

1 Introduction

The content of the course project is the classification of toxic comments. In our daily lives, toxic comments can harm social morals and cause divisions between different groups. With the development of the social media, toxic comments are becoming increasingly common in our lives, making the task of identifying and classifying the comments more and more important.
Classifying toxic comments has several challenges: First, there is the problem of embedding the raw data of comments. We need to find the suitable embeddings for the classification task. Second, the toxic comments are quite different from each other, so we need to improve the generalization ability of our model, in order to handle different types of comments. In our work, we use a method that combines BERT and adversarial training together. We achieve an accuracy of 0.79036 on the test set on Kaggle.

2 Dataset

The datasets include the training set, the validation set and the test set. Specifically, the training set consists of 269 038 comments, each contains textual contents and corresponding labels. The labels include several categories, such as male, female, etc. Besides, there is also a final label y indicating whether the comment is toxic or not.
The validation set contains 45 180 comments and each of them has the same format as the training set. The test set only includes the textual content of 133 782 comments without labels. It will be used to evaluate the performance of our model.

3 Methods

3.1 BERT

3.1.1 Archtecture

In our work, we use BERT model [Dev18] for the sequence classification task. BERT stands for Bidirectional Encoder Representations from Transformers, which is a pretrained transformer-based language model. Specifically, the model we used is BertForSequenceClassification. It is designed for text classification tasks.

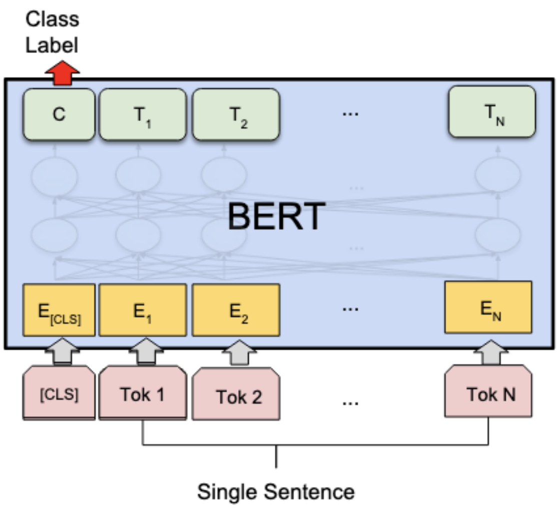


Figure 1: BERT: Single Sentence Classification task [Dev18]

3.1.2 Tokenization

In our work, the text is tokenized by the pre-trained tokenizer of the BERT model. Each input will be processed into three parts:

- Input IDs: The token IDs of the input text generated by the tokenizer.
- Attention Masks: The masks that identify the input parts that the model should focus on.
- Labels: The ground-truth labels for each input for the classification task.

3.2 Adversarial training

To enhance the robustness of our model, we introduce a adversarial training [Zhu+19] module during our training. It is a commonly used method in both NLP and CV and it's essentially a process of attack and defense. Through repeated attacks on the model, the model strengthens its defensive abilities, thereby improving its noise resistance and finally its generalization performance.
Traditionally, attacking a model involves generating adversarial examples to attack the input. For example,

- In NLP, this might involve randomly inserting, deleting, replacing words in a sentence.
- In CV, this might include rotating, scaling, cropping images.

These attacks could serve as a kind of data augmentation, which enhances the model's robustness by exposing it to a wide variety of different inputs.
The adversarial training method we use here focuses on attacking the embeddings directly. This allows us not to manually design data augmentation strategies. The attack and defense happens naturally during the training and it works as follows:

1. Attack: Perform gradient ascent on the input embeddings to increase the loss.

2. Defense: Perform gradient descent on the parameters to decrease the loss.

This process happens during all the training steps. As a result, our model becomes more and more robust and better deal with the noise in the input.

4 Experiment and Result

In the experiment, we used the cross-entropy loss function, and the adversarial perturbation epsilon for the FGM module in adversarial training was set to 0.1.
For the learning rate, we used two approaches. The first one is using the 'get linear schedule with warmup' learning rate scheduler. 'get linear schedule with warmup' gradually increases the learning rate during the warmup phase, then linearly decays it throughout the training. This helps stabilize training and improve convergence. We ran for three epochs.
The second approach is using the 'Cosine Annealing Warm Restarts' learning rate scheduler. Due to limited computing resources, we only obtained results after running for one epoch. For the parameters inside 'Cosine Annealing Warm Restarts', we set T_0 to 10, meaning the learning rate decay cycle for the first restart was 10 epochs, T_mult to 2, meaning the cycle length doubles after each restart, and eta_min to 1e-6, meaning the minimum learning rate was set to 1e-6. 'Cosine Annealing Warm Restarts' reduces the learning rate using a cosine function and restarts it to a higher value after each cycle. The cycle length increases with each restart, helping the model explore the parameter space and avoid local minima.
Since we used a learning rate scheduler, we chose the AdamW optimizer.

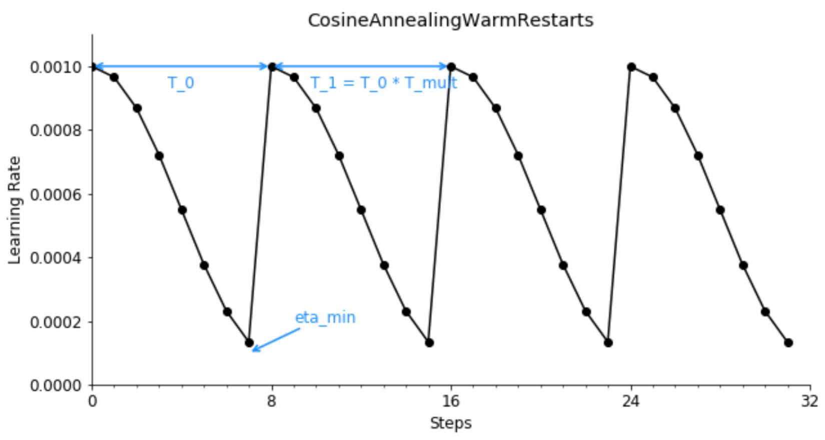
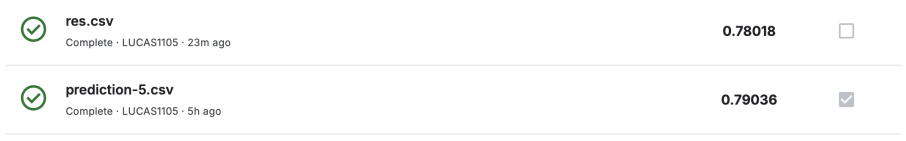


Figure 2: 'Cosine Annealing Warm Restarts' [Mon22]



res.csv
Complete · LUCAS1105 · 23m ago
0.78018

prediction-5.csv
Complete · LUCAS1105 · 5h ago
0.79036

Figure 3: Two results

The accuracy of our first approach on the test set was 0.79036, while the second approach achieved an accuracy of 0.78018.

References

[Dev18] Jacob Devlin. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[Mon22] Leonie Monigatti. *A Visual Guide to Learning Rate Schedulers in PyTorch*. Accessed: 2025-01-17. 2022. URL: https://towardsdatascience.com/a-visual-guide-to-learning-rate-schedulers-in-pytorch-24bbb262c863.

[Zhu+19] Chen Zhu et al. “Freelb: Enhanced adversarial training for natural language understanding”. In: *arXiv preprint arXiv:1909.11764* (2019).