A Model Fusion Approach for Generative AI Authorship Verification

Notebook for PAN at CLEF 2024

Rui Qin, Haoliang Qi* and Yusheng Yi

[1]*Foshan University, Foshan, China*

Abstract

This paper aims to outline our method for distinguishing between text generated by humans and generative AI models with a model fusion approach. Our approach consists of three steps: first, we extend the competition dataset of PAN at CLEF 2024 with an external dataset from a famous data science and machine learning competition platform Kaggle and apply the Levenshtein distance algorithm to correct misspelled words. Datasets of text pairs are then formed based on a shared theme and split into training, validation, and testing datasets. Second, we train a fine-tuned BERT as the base model and a BERT with the R-Drop method to mitigate the overfitting issue. Last, the two models are combined using an ensemble learning technique with a voting strategy. Our experimental results show that the fusion model achieves an ROC-AUC metric of 0.932, representing a 5.6% improvement over the baseline model Fast-DetectGPT (Mistral).

Keywords

PAN 2024, Generative AI Authorship Verification, BERT, Ensemble Learning

1. Introduction

Recently, the development of natural language generation technology has significantly improved the diversity[1], controllability, and quality of AI-generated text, making it difficult to distinguish whether the text is generated by a machine or a human. However, the advancement of producing human-like text at high efficiency also raises concerns about detecting and preventing misuse of LLMs. These concerns about the misuse of LLMs have hindered the NLG application in important domains such as media and education. The ability to accurately detect LLM-generated text is critical for realizing the full potential of NLG while minimizing serious consequences[2].

PAN at CLEF 2024 [3] [4] introduces a task that involves presenting pairs of texts on the same topic—one written by a human and the other by a machine—and requiring participants to identify the human-written text. The Generative AI Authorship Verification Task [5] is organized in collaboration with the Voight-Kampff Task ELOQUENT Lab in a builder-breaker format. PAN participants will develop systems to differentiate between human-written and AI-generated texts, while ELOQUENT participants will explore novel text generation and obfuscation techniques to evade detection [6].

Our method consists of three steps: first, we apply the Levenshtein distance algorithm to correct misspelled words in the text dataset consisting of an external AI-generated text dataset and the PAN 2024 text dataset. We form two texts as a pair of texts if they describe the same topic and treat the pair (*t1*,*t2*) as the sample of model training. Second, we develop a fine-tuned BERT model to serve as our base model to verify generative AI authorship with the ability of powerful pre-trained contextual embeddings. To address and mitigate the issue of overfitting, we also implement a BERT with an R-Drop method model. Both models generate a score to evaluate the probability that the text *t1* is an AI-generated text. The R-Drop method is a regularization technique designed to reduce the variance and improve the robustness of the BERT model by ensuring that the predictions of models are consistent across multiple

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*Corresponding Author

qrcarry@163.com (R. Qin); haoliangqi163@163.com (H. Qi); yiys@fosu.edu.cn (Y. Yi)

0009-0009-3863-3587 (R. Qin); 0000-0003-1321-5820 (H. Qi); 0009-0006-7098-3681 (Y. Yi)

dropout masks. Finally, to enhance the overall performance and robustness of our system, we generate the final score with the ensemble learning[7].

2. Method

To enhance the model performance, we train and fuse two models: a fine-tuned BERT-base model and a BERT model fine-tuned with the R-Drop method.

2.1. BERT Model

We introduce the BERT-based model for AI-generated text detection in this part. The process of fine-tuning using the BERT model is illustrated as shown in Figure 1. In the model training process for a classification task, the two texts are first preprocessed and tokenized to generate corresponding input embeddings, then the embedding sequences are fed into the pre-trained BERT model to obtain contextual representations, and features are extracted from the `[CLS]` token's output for classification, ultimately producing the prediction results. This targeted training is designed to refine the BERT-base model's ability to perform precise binary classification on our dataset. Fine-tuning in the dataset yielded a probability that the generated *t1* was generated by AI.

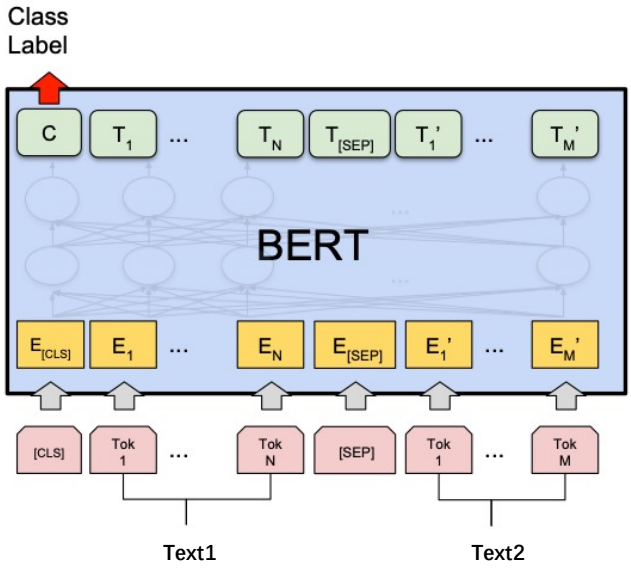

Figure 1: A fine-tuned BERT-base model.

2.2. BERT Model With R-Drop

We introduce the R-Drop method into the training process in Figure 2. We employ an encoder architecture that consists of two separate towers to handle the feature encoding for *t1* and *t2*. By integrating these features, we aim to capture the nuances of the text pairs more effectively. The training process begins with dropout sampling, which generates a subset of the model's output distribution, denoted as p. We then utilize the KL divergence loss[8], $\mathcal{L}_{\mathcal{KL}}$, to guide the optimization process, allowing us to update the model parameters in a way that minimizes the divergence between the actual and the predicted distributions. Ultimately, this leads to the generation of the final output, which is a refined representation of the text pairs relationship as understood by the model. This function encompasses both the cross-entropy loss and the KL divergence loss. The cross-entropy loss continues to drive the

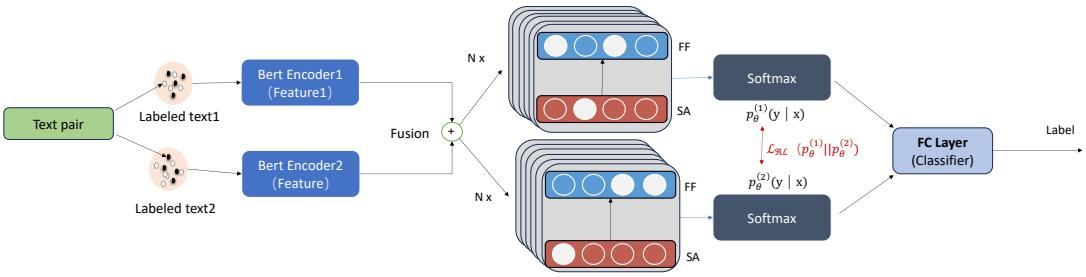

Figure 2: A BERT model fine-tuned with R-Drop method.

model towards accurate classification[9], while the KL divergence loss introduces a regularization effect. By training the model to minimize the KL divergence between its outputs When processing text pairs and applying dropout, we increase the consistency and robustness of the model to some extent in our predictions.

KL Divergence Loss Formula:

$$\mathcal{L}_{\mathcal{KL}} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{KL}(p_i^1 || p_i^2) \tag{1}$$

where p_i^1 and p_i^2 are different prediction probability distributions for the same input.

Total Loss Formula:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\mathcal{CE}} + \lambda \cdot \mathcal{L}_{\mathcal{KL}} \tag{2}$$

where $\mathcal{L}_{\mathcal{CE}}$ is the cross-entropy loss, a common loss function used for classification tasks, and λ is a hyperparameter that balances its weight against the KL divergence loss. We dynamically adjust$\mathcal{L}_{\mathcal{KL}}$ by using a learning rate decay strategy, which gradually decreases λ as training progresses.

2.3. Ensemble Learning

In essence, while the the fine-tuned BERT model is more conventional and focuses solely on classifying accuracy, the BERT model with the R-Drop method through the integration of the R-Drop method, seeks to enhance the model's performance by reducing overfitting and promoting a more stable and reliable prediction across varying inputs. The incorporation of the R-Drop method significantly enhances the model's classification accuracy while also bolstering its generalization capabilities[10]. Considering the above situation, we choose to fuse these two models through the method of Ensemble learning[11]. This improvement is achieved by ensuring that the model's predictions remain reliable and stable.

For each sample in the test set, we use the trained models (One experiment will be conducted without utilizing the R-Drop method, while another will incorporate the R-Drop method) to make predictions, resulting in two probability distributions $p_{\text{no R-Drop}}$ and $p_{\text{R-Drop}}$. Each model evaluates the test data and outputs a probability score. For two models, let the outputs be $p_{\text{no R-Drop}}$ and $p_{\text{R-Drop}}$ Compute the average probability score for each sample:

$$p_{\text{average}} = \frac{p_{\text{no R-Drop}} + p_{\text{R-Drop}}}{2} \tag{3}$$

Determine the final classification based on the average probability score. For binary classification, a threshold (typically 0.5) is used:

$$\text{Final Prediction} = \begin{cases} 1 & \text{if } p_{\text{average}} \geq 0.5 \\ 0 & \text{if } p_{\text{average}} < 0.5 \end{cases} \tag{4}$$

3. Experiments

3.1. Datasets

To execute this task, we leverage datasets from two sources: the competition dataset and an additional external dataset. The competition dataset, provided by the PAN at CLEF 2024 organization, comprises a collection of real and fake news articles. These articles encapsulate a variety of headlines from U.S. news in 2021. The structure of the dataset is detailed in Table 1. The dataset encompasses 1087 distinct topics. For each topic, it includes 1 human-written text file and 13 AI-generated text files. Each of these 13 files is generated by a different AI model, as enumerated in Table 2.

Table 1
Data style of the competition dataset.

id	text
gemini-pro/news-2021-01-01-2021-12-31-kabulairportattack/art-081	generated text content
gemini-pro/news-2021-01-01-2021-12-31-colinpowelldead/art-082	generated text content
...	...
gemini-pro/news-2021-01-01-2021-12-31-colonialpipelinehack/art-012	generated text content

Table 2
The data source and number of topics of the competition dataset.

Data Source	**Number**
alpaca-7b	1087
bigscience-bloomz-7b1	1087
chavinlo-alpaca-13b	1087
gemini-pro	1087
gpt-3.5-turbo-0125	1087
gpt-4-turbo-preview	1087
meta-llama-llama-2-70b-chat-hf	1087
meta-llama-llama-2-7b-chat-hf	1087
mistralai-mistral-7b-instruct-v0.2	1087
mistralai-mixtral-8x7b-instruct-v0.1	1087
qwen-qwen1.5-72b-chat-8bit	1087
text-bison-002	1087
vicgalle-gpt2-open-instruct-v1	1087

Except for the competition dataset, we introduce an external dataset Kaggle's Dataset: DAIGT V2 Train [1] to enhance the generalization of our model. The Kaggle dataset consists of 44,868 items, 58% of the data comes from persuade corpus, 5% came from mistral7binstruct_v2, and the remaining 42% could have been generated by other AI or written by the author himself. The data source and several topics from the Kaggle dataset are detailed in Table4. The fields of the dataset are shown in the Table 3.

Table 3
The field interpretation of Kaggle dataset.

Field name	text	label	prompt_name	source
Description	essay text	1 for AI generated, 0 for human	Represents text topic	data source

[1]The dataset is available at https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset

Table 4
The data source and number of topics of the Kaggle's dataset.

Data Source	Number
persuade_corpus	25996
chat_gpt_moth	2421
llama2_chat	2421
mistral7binstruct	4842
original_moth	2421
train_essays	1378
llama_70b_v1	1172
falcon_180b_v1	1055
darragh_claude	2000
radek_500	500
NousResearch/Llama-2-7b-chat-hf	400
mistralai/Mistral-7B-Instruct-v0.1	400
cohere-command	350
palm-text-bison1	349
radekgpt4	200

3.1.1. Construct Dataset

Texts in both datasets are labeled with a topic. To fine-tune a model to distinguish AI-generated text, we first construct a dataset of text pairs with the same topic. In the competition dataset illustrated in Table 1, the 'id' field contains three parts: the name of the AI model, the name of the news, and the serial number of the article. First, we treat the name of the news as the topic and collect all texts with the same topic. Then, we form a sample text pair by selecting each human-written text and each AI-generated text. Finally, we have constructed 14,131 text pairs from the competition dataset. Since the number of human-written texts is much smaller than the number of AI-generated texts with a ratio of 1 to 13, the unbalanced ratio is more likely to lead to an overfitting problem.

To provide more data to the model, we collected the relevant dataset DAIGT V2 Train Dataset from the Kaggle competition platform, including 44 texts. We compose 15,235 text pairs from the Kaggle dataset with the same method above.

Therefore, the two datasets from Kaggle and competition generate 29,366 pairs of texts. We divide these pairs into training sets, evaluation sets, and test sets according to a ratio of 60%, 30%, and 10%, respectively.

3.1.2. Clean Text Data

For the collected dataset, it is inevitable that some words are spelled incorrectly. To solve this problem, we use a Python library[2] that created a structure for quickly searching for words within a given Levenshtein distance. The Levenshtein distance, also known as edit distance, is an algorithm used to measure the difference between two strings. Then we use the Levenshtein Distance algorithm to correct misspelled words in the data set. This method can balance the data between human and machine as much as possible, while ensuring the accuracy of word spelling in the data set, restoring the text semantics completely and improving the quality of the data set. Specific implementation steps are as follows:

1. Search for a distance 1, and when there is one corrected word, correct it.
2. Count all suggested edits (inserts, deletions, replacements of specific letters) for all words within distance "1".
3. Get the most frequent one, and if its frequency is more than 6% of all words in the document, then search again for the Levenshtein distance but with custom cost specification: the most frequent

[2]The source code is available at https://github.com/pkoz/leven-search.

edit cost being 10% of the default edit cost. That way, we fixed most of the "letter obfuscations" mentioned in the discussions.

Table 5
Dataset partitioning.

	Train	Validation	Test
Dataset	17619	8809	2936

3.2. Experimental Setup

To achieve optimal experimental results, we adhere to common experimental configurations as follows: We employ the BERT-base model as our encoder, leveraging its 12-layer transformer architecture, 768 hidden sizes, and 12 attention heads. For hyperparameter optimization, we set the batch size to 32 and the learning rate to 3×10^{-5}. The model underwent training for 10 epochs with a maximum input sequence length capped at 512 tokens. This configuration ensures a balance between computational efficiency and model performance, laying a solid experimental foundation for our study.

Additionally, to incorporate the R-Drop method, we introduce the following parameter settings:

- **R-Drop method Hyperparameter** λ: This hyperparameter controls the weight of the KL divergence loss.
- **Dropout Rate**: For the R-Drop method, we no longer use the traditional fixed dropout rate but employ the dynamic dropout strategy of the R-Drop method, which dynamically adjusts the dropout rate during training. Specifically, we set the initial dropout rate to 0.5 to enable the R-Drop method.

By utilizing the above formula and incorporating the KL divergence loss into the loss function, we aim to reduce the distributional discrepancy between different dropout instances, thereby mitigating overfitting:

$$\text{Loss} = \text{BinaryCrossEntropy}(y, \hat{y}) + \lambda \cdot \mathcal{L}_{\mathcal{KL}}$$

3.3. Result

Table 6
Overview of the accuracy in detecting if a text is written by a human in Voight-Kampff Generative AI Authorship Verification task on PAN 2024. We report the ROC-AUC, Brier, C@1, F_1, $F_{0.5u}$ and their mean.

Approach	**ROC-AUC**	**Brier**	**C@1**	**F_1**	**$F_{0.5u}$**	**Mean**
BERT-base	0.891	0.891	0.891	0.893	0.885	0.89
BERT-base with R-Drop	0.91	0.902	0.898	0.9	0.884	0.899
qinnnruiii	**0.932**	**0.916**	**0.907**	**0.904**	**0.903**	**0.913**
Baseline Binoculars	0.972	0.957	0.966	0.964	0.965	0.965
Baseline Fast-DetectGPT (Mistral)	0.876	0.8	0.886	0.883	0.883	0.866
Baseline PPMd	0.795	0.798	0.754	0.753	0.749	0.77
Baseline Unmasking	0.697	0.774	0.691	0.658	0.666	0.697
Baseline Fast-DetectGPT	0.668	0.776	0.695	0.69	0.691	0.704
95-th quantile	0.994	0.987	0.989	0.989	0.989	0.990
75-th quantile	0.969	0.925	0.950	0.933	0.939	0.941
Median	0.909	0.890	0.887	0.871	0.867	0.889
25-th quantile	0.701	0.768	0.683	0.657	0.670	0.689
Min	0.131	0.265	0.005	0.006	0.007	0.224

The results in Table6. It shows that our fine-tuned BERT-base and BERT-base with R-Drop methods achieved **ROC-AUC** values of 0.891 and 0.91 respectively, demonstrating some competitiveness. It's worth noting that the fusion of the BERT model and BERT with the R-Drop method outperformed both individually, scoring higher across all metrics, with an average score of 0.913. Particularly, the **ROC-AUC** value reached 0.932, which falls between the 75th quantile and the Median of all participants, surpassing Baseline Fast-DetectGPT (Mistral) but falling short of Baseline Binoculars.

From the perspective of the fused model, there was some improvement, though not entirely satisfactory. Upon reviewing my scores across various datasets, I notice particularly low scores on two German-related datasets, dragging down my overall performance. I speculate this might be due to the limited size of my dataset, or perhaps certain flaws introduced during training with textual pairs. The exact reasons remain uncertain. Moving forward, I plan to conduct further experiments to validate these hypotheses and identify the underlying causes.

4. Conclusion

The results demonstrate that fine-tuning the BERT-base alone yields lower performance compared to the model enhanced with the R-Drop method. Furthermore, the ensemble voting approach, combining the outputs of both models, surpasses the performance of either individual model, highlighting the effectiveness of our proposed method. The integration of the R-Drop method with the BERT-base and the subsequent ensemble learning strategy improves classification performance. This approach demonstrates the potential for combining regularization techniques and ensemble methods to enhance model robustness and accuracy in text classification tasks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.62276064)

References

[1] R. Tang, Y.-N. Chuang, X. Hu, The science of detecting llm-generated text, Commun. ACM 67 (2024) 50–59. URL: https://doi.org/10.1145/3624725. doi:10.1145/3624725.

[2] V. S. Sadasivan, V. Sankar, A. Kumar, et al., Can ai-generated text be reliably detected?, arXiv preprint arXiv:2303.11156 (2023).

[3] J. Bevendorff, M. Wiegmann, J. Karlgren, et al., Overview of the "Voight-Kampff" Generative AI Authorship Verification Task at PAN and ELOQUENT 2024, in: G. Faggioli, N. Ferro, P. Galuščáková, A. G. S. de Herrera (Eds.), Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[4] A. A. Ayele, N. Babakov, J. Bevendorff, et al., Overview of PAN 2024: Multi-Author Writing Style Analysis, Multilingual Text Detoxification, Oppositional Thinking Analysis, and Generative AI Authorship Verification, in: L. Goeuriot, P. Mulhem, G. Quénot, D. Schwab, L. Soulier, G. M. D. Nunzio, P. Galuščáková, A. G. S. de Herrera, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2024.

[5] J. Bevendorff, X. B. Casals, B. a. a. Chulvi, Overview of the Voight-Kampff Generative AI Authorship Verification Task at PAN 2024, in: Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, CEUR-WS.org, 2024.

[6] M. Fröbe, M. Wiegmann, N. Kolyada, et al., Continuous Integration for Reproducible Shared Tasks with TIRA.io, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on

IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. doi:10.1007/978-3-031-28241-6_20.

[7] X. Dong, Z. Yu, W. Cao, et al., A survey on ensemble learning, Frontiers of Computer Science 14 (2020) 241–258.

[8] T. Kim, J. Oh, N. Kim, et al., Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation, arXiv preprint arXiv:2105.08919 (2021).

[9] x. liang, J. Li, Y. Wang, et al., R-drop: Regularized dropout for neural networks, in: M. Ranzato, A. Beygelzimer, e. a. Y. Dauphin (Eds.), Advances in Neural Information Processing Systems, volume 34, Curran Associates, Inc., 2021, pp. 10890–10905. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/5a66b9200f29ac3fa0ae244cc2a51b39-Paper.pdf.

[10] R. Polikar, Ensemble learning, Ensemble machine learning: Methods and applications (2012) 1–34.

[11] T. G. Dietterich, et al., Ensemble learning, The handbook of brain theory and neural networks 2 (2002) 110–125.