Team Asdkklkk at PAN 2026:ModernBERT-Based Source Detection for Mathematical Reasoning Trajectories at PAN 2026

Notebook for the PAN Lab at CLEF 2026

Zhankeng Liang[1], Zhongyuan Han[1,*], Xianbing Duan[1], Zhengqiao Zeng[1] and Chang Liu[1]

[1]Foshan University, Foshan, China

Abstract

This paper presents our approach to Subtask 1 of the PAN CLEF 2026 Reasoning Trajectory Detection task, which aims to determine whether a mathematical solution was written by a human or generated by an AI system. We formulate the task as binary text classification and fine-tune ModernBERT-base using only the `solution` field. Our pipeline includes label derivation, text cleaning, supervised fine-tuning, threshold calibration, and grouped evaluation. On the validation set, ModernBERT achieves a macro F1 of 0.9249 with the default threshold and 0.9428 after threshold calibration. Grouped analysis shows perfect accuracy for gemini-3-flash, gpt-5-nano, and k2think-v2, high accuracy for deepseek-r1, and an accuracy of 0.89 for the human category. We use the calibrated threshold to generate the final test submission. The results indicate that solution-only modeling can capture useful differences between human and AI reasoning texts, although human-written solutions remain the main source of errors.

Keywords

PAN 2026, Reasoning Trajectory Detection, Text Source Detection, AI-Generated Text Detection, ModernBERT, macro F1

1. Introduction

In recent years, large language models have demonstrated increasingly strong reasoning capabilities in mathematical reasoning, code generation, and complex question-answering tasks. Their generated texts have appeared at scale in both open-domain and specialized domains, posing serious challenges to information authenticity and content safety [1]. Many models not only output final answers but also generate intermediate reasoning steps, i.e., reasoning trajectories. While reasoning trajectories can improve the interpretability of answers, they also introduce new risks: models may produce seemingly plausible but actually incorrect reasoning steps, or arrive at final answers through unsafe, misleading, or logically invalid paths. Therefore, identifying the source of reasoning trajectories is of great significance for understanding large language model behavior, evaluating model reliability, and improving AI safety.

The PAN CLEF 2026 Reasoning Trajectory Detection task focuses on reasoning trajectories and follows the broader PAN evaluation tradition of studying text analysis and generative AI detection problems [2], with the 2026 edition introducing reasoning trajectory detection [3]. The task consists of two subtasks: Subtask 1 focuses on source detection, requiring the determination of whether the reasoning trajectory and final answer in a given sample are generated by an AI system or written by a human; Subtask 2 focuses on safety detection, requiring the determination of whether the reasoning process and final answer are safe. This paper mainly describes our system design and experimental results for Subtask 1.

Subtask 1 can be regarded as a text authorship classification problem. However, Sadasivan et al. point out that as large language models continue to improve, the boundary between AI-generated text and human-written text is becoming increasingly blurred, making reliable detection become a severe

CLEF 2026 Working Notes , 21– 24 September 2026, Jena, Germany

*Corresponding author.

oooo1390329007@163.com (Z. Liang); hanzhongyuan@gmail.com (Z. Han); 15007473346@163.com (X. Duan); zhengqiaozeng@163.com (Z. Zeng); lc965024004@gmail.com (C. Liu)

challenge [4]. Unlike general AI text detection, the texts in this task mainly come from mathematical solution scenarios, typically containing formulas, step-by-step reasoning, symbolic expressions, and lengthy explanations. The differences between human and AI solutions may be reflected in language templates, reasoning unfolding styles, redundancy levels, formatting conventions, and local expression habits. Based on this characteristic, we fine-tune ModernBERT-base for binary classification using only the solution text as input, and analyze how well this simple input scheme can discriminate between human and AI authors.

After training, we calibrate the decision threshold by grid search on the validation set to improve classification performance, instead of relying on the default 0.5 threshold. The main contributions of this paper are as follows:

1. We formulate PAN 2026 Reasoning Trajectory Detection Subtask 1 as a binary classification task based on ModernBERT.
2. We adopt a solution-only input construction approach, using only the solution text to complete source detection.
3. We evaluate the model using macro F1 and balanced accuracy on the official validation set, conduct generator-grouped error analysis, and apply threshold calibration via grid search to select the optimal decision boundary.
4. Using the optimal threshold found on the validation set, we generate the test set submission and report the official TIRA evaluation results. We further analyze the generalization gap and discuss directions for future improvement.

2. Related Work

Text authorship classification and AI-generated text detection have long been of interest in the field of natural language processing. Early methods typically relied on manual features, such as lexical statistics, syntactic patterns, punctuation usage, text length, and readability metrics, combined with traditional machine learning models such as support vector machines, naive Bayes, logistic regression, or decision trees for classification. These methods are simple to implement and relatively interpretable, but they are heavily dependent on feature engineering and struggle to fully capture deep semantics and reasoning structures in complex texts.

With the development of deep learning, models such as RNN, LSTM, GRU, and CNN have been applied to text classification tasks. Uchendu et al. conducted a systematic study on authorship attribution for neural text generation, pointing out the limitations of traditional statistical features in capturing the style of deep generative model texts [5]. These methods can automatically learn local or sequential features, reducing manual feature design compared to traditional machine learning methods. However, reasoning trajectories typically contain long-distance dependencies, mathematical symbols, step-by-step structures, and long contexts, and traditional deep models still have limitations in long text modeling and capturing complex semantic relationships.

In recent years, pre-trained language models based on the Transformer architecture have become the mainstream approach for text classification tasks. The self-attention mechanism proposed by Vaswani et al. laid the foundation for the Transformer architecture [6], enabling models to capture long-distance dependencies in parallel. BERT proposed by Devlin et al. [7] performs bidirectional pre-training through masked language modeling and next sentence prediction tasks, initiating the widespread application of the pre-training-fine-tuning paradigm in natural language understanding tasks. Liu et al. proposed RoBERTa [8] through robust optimization of BERT's training recipe, further improving the performance of encoders on understanding tasks. For AI-generated text detection tasks, such models can learn discriminative clues from language style, expression templates, reasoning organization, and local patterns. This paper adopts ModernBERT-base as the base model, focusing on the impact of the solution text itself on model performance in the PAN 2026 reasoning trajectory source detection task.

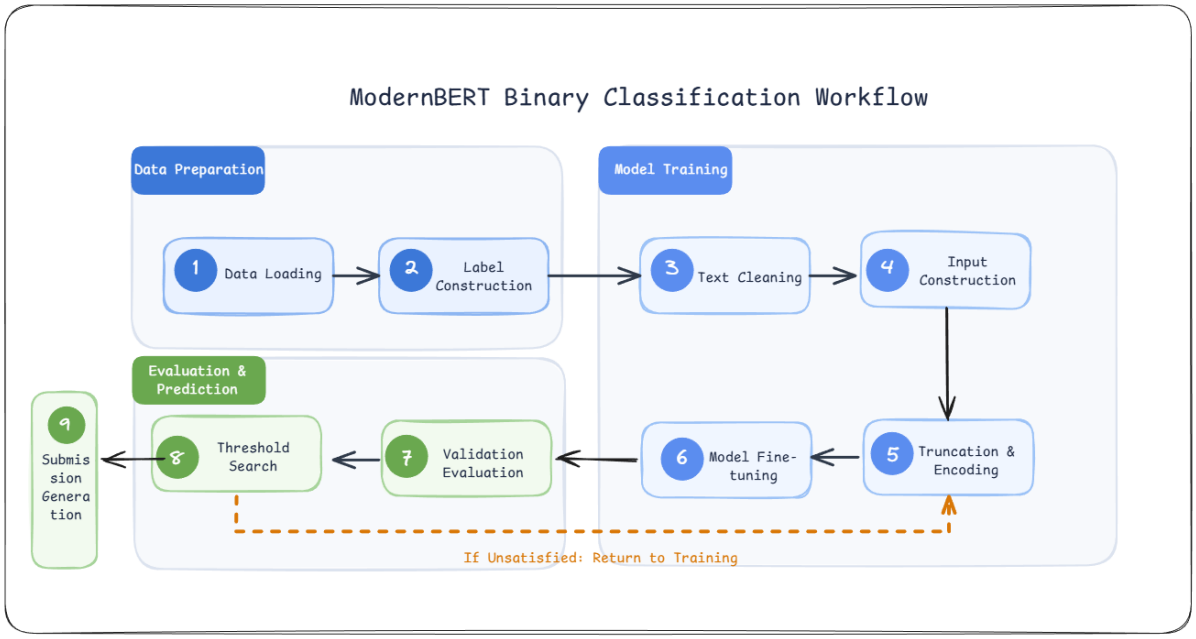

Figure 1: Training and inference pipeline for source detection.

3. Methodology

3.1. Overall Pipeline

We formulate the source detection task as a text binary classification problem. Given an input text, the model outputs the probability that it belongs to the AI category, and then classifies the sample as `human` or `ai` based on a threshold selected on the validation set. The overall pipeline includes data reading, label construction, input text construction, tokenization, model fine-tuning, validation set evaluation, threshold search, and test set inference.

The overall pipeline of the method is shown in Figure 1.

3.2. Model Selection and Initialization

This experiment uses ModernBERT-base as the base model [9]. The choice of this model is mainly based on the following considerations:

- Subtask 1 is essentially a text discrimination task, requiring the model to capture differences in text style, reasoning structure, formatting patterns, and language distribution. ModernBERT performs stably on natural language understanding and classification tasks.
- The `base` size offers a balanced trade-off between training cost and effectiveness, making it suitable for experiments with local resources.
- Hugging Face Transformers provides comprehensive support for ModernBERT's tokenizer, sequence classification model, and training pipeline, facilitating implementation and reproducibility.
- The model can be directly used for binary classification fine-tuning and seamlessly integrated with the threshold calibration pipeline.

We use the ModernBERT-base model and initialize the binary classifier with AutoModelForSequenceClassification. The label mapping is `human` = 0 and `ai` = 1.

3.3. Data Preprocessing

The training and validation data are both in JSONL format. The preprocessing steps are as follows:

1. Read the `problem`, `solution`, `generator`, and `detailed_generator` fields.
2. Filter out samples with empty solutions to ensure valid input text.
3. Map `human` samples to 0 and various AI generator samples to 1.
4. Construct input text containing only the `solution` field.
5. Use the tokenizer to truncate or pad the text to a fixed length of 512 tokens.

Since reasoning trajectory texts may be long, this experiment adopts a simple strategy of keeping the first 512 tokens of each text.

3.4. Input Construction Scheme

This scheme is designed to use only the `solution` text as model input. The design goal is to allow the model to focus on the language style, reasoning unfolding style, formatting habits, and templated expressions of the solution itself. Since the task ultimately judges the source of the solution rather than its correctness, using only the `solution` more directly corresponds to the source detection objective.

3.5. Threshold Calibration

The model outputs logits for two categories. In implementation, softmax is first applied to the logits to convert the output corresponding to the AI category into a probability; then, given a candidate threshold, when the AI probability is greater than or equal to this threshold, the sample is predicted as `ai`, otherwise it is predicted as `human`.

The default binary classification threshold is 0.5, but since the data category distribution is not completely balanced, a fixed threshold may not be optimal. Therefore, we perform threshold calibration using grid search on the validation set. The specific implementation steps are as follows:

1. Use the trained model to predict on the validation set, obtaining the probability that each sample belongs to the AI category.
2. Uniformly generate 31 candidate thresholds in the range of 0.2 to 0.8.
3. For each candidate threshold, convert the probabilities into binary classification labels.
4. Calculate the macro F1 on the validation set for that threshold.
5. Select the threshold with the highest macro F1 as the final inference threshold.

The optimal threshold obtained in this experiment is 0.72, corresponding to a validation set calibration macro F1 of 0.9428. The test set inference stage uses this threshold to convert the model's probability outputs into `human` or `ai` labels.

4. Experiments

4.1. Experimental Requirements Overview

PAN CLEF 2026 Reasoning Trajectory Detection Subtask 1 requires building a binary classification system to determine whether a given reasoning trajectory or solution text comes from a human author or an AI system. The official primary evaluation metric is macro F1, which calculates the F1 value for the human category and the AI category separately and then averages them:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{1}$$

$$\text{Macro } F_1 = \frac{F_1(\text{human}) + F_1(\text{AI})}{2}. \tag{2}$$

Since the validation set category distribution is not completely balanced, macro F1 better reflects the model's comprehensive recognition ability for both categories than accuracy.

Table 1
Dataset statistics.

Dataset	Sample Count	Description
Training Set	87,556	Valid solution-only training samples after filtering empty texts; human texts account for 34.71%, and AI texts account for 65.29%.
Validation Set	497	Contains human, deepseek-r1, gemini-3-flash, gpt-5-nano, and k2think-v2; human texts account for 20.12%, and AI texts account for 79.88%.
Test Set	2,617	Unlabeled samples used to generate the final submission file.

4.2. Data Source

The Subtask 1 data is in JSONL format, mainly containing the following fields:

- `problem`: The mathematical problem text, which may contain problems of different difficulty levels and complexities.
- `solution`: The solution text for the given problem, which is also the main input field used in this experiment.
- `generator`: The source of the solution, which can be `human` or `llm`.
- `detailed_generator`: More fine-grained source information, which can be `human` or a specific model name.

According to the task description and our data statistics, the training and validation set sizes are shown in Table 1.

In the local training set, the number of AI samples is higher than that of human samples. The training set contains 531 deepseek-r1 samples, 24,655 gemini-3-flash samples, 30,301 gpt-5-nano samples, 30,392 human samples, and 1,677 k2think-v2 samples. The validation set contains 100 human samples and 397 AI samples in total. After grouping the validation set by generator, there are 100 samples each for deepseek-r1, gemini-3-flash, gpt-5-nano, and human, and 97 samples for k2think-v2.

4.3. Program Design

4.3.1. Training Program

The training program is responsible for reading the training and validation data, constructing labels and input texts, calling the tokenizer to encode the texts, and using ModernBERT-base for binary classification fine-tuning. The training stage uses macro F1 as the optimal model selection metric and employs early stopping to control overfitting.

The training pipeline is as follows:

1. Read the official training and validation JSONL files.
2. Derive binary classification labels based on `label`, `generator`, or `detailed_generator`.
3. Use only the `solution` field to construct the input text.
4. Use the tokenizer to encode the text to a fixed length.
5. Perform forward propagation to compute the cross-entropy loss between the predicted logits and the ground-truth labels. Then backpropagate the loss and update the model parameters using the AdamW optimizer.
6. Evaluate the model on the validation set and save the model with the optimal macro F1.
7. Perform threshold search on the validation set and save the best threshold.

4.3.2. Inference Program

The inference program first loads the optimal model parameters saved during the training stage and the corresponding tokenizer, and preprocesses the test set samples using text cleaning, truncation, and

Table 2
Experimental parameters.

Parameter	Value	Parameter	Value
Base Model	ModernBERT-base	Max Length	512
Batch Size	24	Epochs	5
Learning Rate	5e-6	Weight Decay	0.01
Warmup Ratio	0.1	Inner Valid Size	0
Early Stopping Patience	1	Optimal Model Metric	macro F1
Threshold Search Metric	macro F1	Threshold Search Range	0.2–0.8
Threshold Search Steps	31		

Table 3
Validation results under default and calibrated thresholds.

Evaluation Setting	Threshold	Accuracy	Balanced Accuracy	Macro F1
Default Threshold	0.50	0.9537	0.9074	0.9249
Calibrated Threshold	0.72	0.9638	0.9362	0.9428

encoding methods consistent with the training stage. Then, the encoded test samples are fed into the model for batch forward inference to obtain the probability output for each sample belonging to the AI category. Next, the optimal threshold determined through threshold search on the validation set is used to convert the probability results into final prediction labels: when the AI probability is greater than or equal to the threshold, it is classified as AI, otherwise as human. Finally, the test sample IDs and prediction results are organized into a CSV file according to the official requirements for final submission.

4.4. Experimental Parameters

The training parameters used in this experiment are shown in Table 2.

4.5. Experimental Results and Analysis

4.5.1. Overall Results

This experiment recorded the results under the default threshold and the calibrated threshold on the validation set. Before threshold calibration, the model directly uses the default threshold for classification; after threshold calibration, the model uses the optimal threshold 0.72 obtained from validation set grid search for classification.

As shown in Table 3, after threshold calibration, the macro F1 improves from 0.9249 to 0.9428, and the balanced accuracy also improves from 0.9074 to 0.9362, indicating that threshold search based on the validation set can further improve the final discrimination effect.

4.5.2. Grouped Analysis by Generator

To further understand the performance on different model sources, we calculated the accuracy for each generator group on the validation set. Fraser et al. systematically analyzed the detectability differences of current detection methods under different conditions, pointing out that generator type, text length, and domain adaptation are key factors affecting discrimination accuracy [10]. Based on this, this experiment conducts grouped evaluation for each source, with results shown in Table 4.

The solution-only approach achieves perfect accuracy for gemini-3-flash, gpt-5-nano, and k2think-v2, and an accuracy of 0.9300 for deepseek-r1, indicating that the model can effectively capture stylistic or structural features in these AI-generated solutions. The main weakness is the human category, with an accuracy of only 0.8900, indicating that some human solutions may be formally similar to AI

Table 4
Grouped validation accuracy by generator.

Group	Sample Count	Accuracy
deepseek-r1	100	0.9300
gemini-3-flash	100	1.0000
gpt-5-nano	100	1.0000
human	100	0.8900
k2think-v2	97	1.0000

Table 5
Ablation study on the validation set. All results are reported after threshold calibration.

Input	Model	Max Len	Truncation	Cal. Thresh.	Macro F1	Human Acc
solution	ModernBERT-base	512	head-only	0.72	0.9428	0.89
solution	DeBERTa-v3-base	512	head-only	0.64	0.9372	0.87
solution	DeBERTa-v3-base	256	head-only	0.32	0.8138	0.55
problem+solution	DeBERTa-v3-base	256	head-only	0.20	0.6371	0.85

solutions, or that the model over-relies on certain AI-style cues, thus misclassifying more standardized and complete human solutions as AI.

4.5.3. Ablation Study

To assess the contribution of each design choice, we conduct ablation experiments on the validation set comparing different input schemes, model architectures, context lengths, and truncation strategies. The results are summarized in Table 5.

Several findings emerge. First, extending the context length from 256 to 512 tokens improves DeBERTa's macro F1 from 0.8138 to 0.9372 and more than doubles human accuracy (0.55 $\rightarrow$ 0.87), confirming that longer context is critical for this task. Second, the problem+solution scheme performs substantially worse (macro F1 0.6371) than any solution-only variant under the same 256-token budget, showing that the shared problem text introduces noise rather than discriminative signal. This justifies our solution-only design. Third, under identical settings, ModernBERT slightly outperforms DeBERTa-v3.

4.5.4. Error Analysis

Under the calibrated threshold (0.72), the confusion matrix shows 11 human samples misclassified as AI (false positive rate 11%) and 7 AI samples misclassified as human (false negative rate 1.8%). Human false positives are thus the dominant error mode. These likely correspond to well-structured solutions with step-by-step enumeration, LaTeX-style formatting, or polished language—patterns that appear predominantly in AI-generated training data, causing the model to learn them as spurious AI indicators. The 7 false negatives are concentrated in the deepseek-r1 group, consistent with its lower group accuracy (0.93, Table 4). DeepSeek-R1 is known for producing human-like reasoning patterns, which may weaken the discriminability of surface-level stylistic features.

4.5.5. Test Set Inference Results

Based on the validation set performance, we finally adopt the calibrated model for test set inference. The test set prediction distribution is shown in Table 6.

The official test set scoring results are generated by the TIRA evaluation platform [11], as shown in Table 7. The table reports the metrics returned by the official platform; we denote them as macro-averaged metrics to align them with the official evaluation setting.

Table 6
Prediction distribution on the test set.

Method	Test Samples	Predicted Human	Human Ratio	Predicted AI	AI Ratio
ModernBERT solution-only	2,617	577	22.05%	2,040	77.95%

Table 7
Official test set results.

Method	Macro F1	Macro Recall	Macro Precision	Rank
ModernBERT solution-only	0.7071	0.6938	0.7507	6

The test set prediction distribution (77.95% AI) is close to the validation calibrated positive rate (80.68%). However, the official macro F1 of 0.7071 represents a 0.2357 absolute drop from the calibrated validation score of 0.9428, ranking 6th. We attribute this substantial generalization gap to three factors:

(1) **Unseen generators**: the model achieves near-perfect per-generator accuracy on seen generators (Table 4), suggesting it learned generator-specific stylistic shortcuts. Unfamiliar test-set generators break these shortcuts.

(2) **Domain shift**: test problems may differ in mathematical topics, difficulty, or solution length, causing surface-level features to not transfer.

(3) **Input sensitivity**: as the ablation study shows (Table 5), even small changes to input construction substantially alter performance, indicating the model's reliance on brittle surface cues rather than robust authorship representations.

These factors point to a fundamental challenge: the known training generators encourage learning generator-specific patterns, while the hidden test set may contain arbitrary new generators. Future work should prioritize generator-agnostic features and domain-invariant representations.

5. Conclusion

This paper presents our ModernBERT fine-tuning method for PAN CLEF 2026 Reasoning Trajectory Detection Subtask 1. We formulate the task as a binary classification problem between human and AI, and adopt a solution-only scheme using only the `solution` field. Validation set experiments show that this scheme achieves a macro F1 of 0.9249; after threshold calibration, the macro F1 further improves to 0.9428. Grouped analysis by generator shows that the model performs well on multiple AI generators and achieves an accuracy of 0.89 for the human category. Ablation experiments confirm that solution-only input outperforms problem+solution input, and that longer context (512 tokens) substantially improves over shorter context (256 tokens). Error analysis reveals that human false positives are the dominant failure mode, likely caused by well-structured human solutions that superficially resemble AI-generated text. We submitted our final results using the validation-calibrated threshold and obtained a test macro F1 of 0.7071, indicating that generalization to the hidden test distribution remains challenging. We analyze this gap in terms of unseen generators, domain shift, and input sensitivity.

Future work will focus on the following directions: to address the insufficient recall for the human category, we will explore class weights, resampling, focal loss [12], and more fine-grained threshold calibration; we will also investigate longer input lengths to better capture key information in long texts; furthermore, we will compare other pre-trained models and attempt multi-model ensembles; finally, we will design more robust source detection methods through grouped error analysis for different generators.

Acknowledgments

This work is supported by the Guangdong Province Basic and Applied Basic Research Fund (Guangdong-

Foshan Joint Funds, 2024A1515140026).

Declaration on Generative AI

During the preparation of this work, the author utilized Kimi K2.6 and GPT-5.5 to accomplish drafting content, text translation, and content enhancement tasks. After employing these tools, the author reviewed and edited the content as necessary and took appropriate measures to assume full responsibility for the content of the publication.

References

[1] E. N. Crothers, N. Japkowicz, H. L. Viktor, Machine-generated text: A comprehensive survey of threat models and detection methods, IEEE Access 11 (2023) 70977–71002.

[2] J. Bevendorff, M. Fröbe, A. Greiner-Petter, A. Jakoby, M. Mayerl, P. Nakov, H. Plutz, M. Potthast, B. Stein, M. N. Ta, Y. Wang, E. Zangerle, in: M. Hagen, M. Potthast, B. Stein, P. Schaer, E. Zangerle, A. Barrón-Cedeño, A. G. S. de Herrera, E. S. Salido, S. MacAvaney, J. M. Struß (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2026.

[3] M. N. Ta, K. Wan, Y. Lin, Y. Wang, P. Nakov, Overview of the first Reasoning Trajectory Detection Task at PAN 2026, in: A. Barrón-Cedeño, A. G. S. de Herrera, E. S. Salido, S. MacAvaney, J. M. Struß (Eds.), Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2026.

[4] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, S. Feizi, Can ai-generated text be reliably detected?, 2023. arXiv:2303.11156.

[5] A. Uchendu, Z. Ma, A. Le, T. Le, R. Zhang, D. Lee, Authorship attribution for neural text generation, in: Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 2020, pp. 8384–8395.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[7] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pp. 4171–4186.

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, 2019. arXiv:1907.11692.

[9] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, N. Cooper, G. Adams, J. Howard, I. Poli, Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, 2024. arXiv:2412.13663.

[10] K. C. Fraser, H. Dawkins, S. Kiritchenko, Detecting ai-generated text: Factors influencing detectability with current methods, Journal of Artificial Intelligence Research 82 (2025) 2233–2278.

[11] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous integration for reproducible shared tasks with TIRA.io, in: Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, 2023, pp. 236–241. doi:10.1007/978-3-031-28241-6_20.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.