Team FOSU at PAN 2026: Generative AI Text Detection via Part-of-Speech Style Injection Pre-training

Notebook for the PAN Lab at CLEF 2026

Yihao Jia, Junlang Liu and Leilei Kong*

Foshan University, Foshan, China

Abstract

This paper proposes a generative AI text detection method named INSA (INjected Style Augmented training) based on part-of-speech style injection pre-training. The method first constructs a textual style representation using the distribution of part-of-speech tags, then discretises the continuous style space into style tokens via K-Means clustering. Subsequently, during the continued pre-training phase with Masked Language Modeling (MLM), style tokens are injected into the input sequence as special tokens to guide the model toward learning conditional style representations, thereby enhancing its ability to model syntactic patterns of generated text. Finally, the model is fine-tuned on the downstream generative AI text detection task. Experiments are conducted on the training and validation sets of the PAN 2026 Generative AI Detection dataset, and evaluated using the official PAN metrics. The proposed method achieves a Mean score of 0.979 on the validation set, outperforming multiple official baselines, which verifies the effectiveness of the part-of-speech style conditional modeling strategy in generated text detection.

Keywords

generative text detection, stylometry, pre-training, BERT

1. Introduction

In recent years, with the development of large generative language models such as ChatGPT and DeepSeek, AI-generated text has achieved high quality in grammar, semantics, and discourse structure, and is widely used in scenarios such as news generation, educational assistance, intelligent customer service, and content creation [7]. However, the rapid advancement of generative models has also brought problems such as misinformation flooding, automated content manipulation, and academic integrity [11]. Therefore, how to effectively detect AI-generated text has become an important research direction in natural language processing. As large language models continuously improve, the differences in semantic fluency and discourse structure between generated and human-written texts gradually diminish, making the detection task more challenging. The PAN 2026 lab provides shared evaluation settings for this and related authorship and provenance tasks [1]; our submission addresses the third "Voight-Kampff" Generative AI / LLM Detection Task [2].

This task is typically formalised as a text provenance attribution problem, i.e., determining whether an input text is "human-written" or "model-generated" [11, 12].

Existing detection methods fall into three main categories. The first category is based on language model probability distributions, such as GLTR [3], which performs detection by analysing token probability ranks and entropy, but such methods often rely on a specific generative model and have limited generalisation to unknown models [8]. DetectLLM [13] further utilises token log-rank information for zero-shot detection, improving the robustness of probabilistic features.

The second category is based on perturbation analysis. The representative work DetectGPT [4] discriminates by comparing the probability curvature difference between the original text and perturbed

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

*Corresponding author.

15237443323@163.com (Y. Jia); liujunlang2015@gmail.com (J. Liu); kongleilei@fosu.edu.cn (L. Kong)

0009-0003-7352-4489 (Y. Jia); 0009-0009-1578-0867 (J. Liu); 0000-0002-4636-3507 (L. Kong)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

versions. Although robustness is improved, multiple forward passes are required, leading to high computational cost [4].

The third category uses supervised classification based on pre-trained language models, such as fine-tuning BERT or RoBERTa for binary classification [12]. These methods fully leverage contextual semantics and are efficient in inference. However, most of them focus on semantic features and under-utilise implicit syntactic style information (e.g., part-of-speech distribution, function word usage) [5, 6]. Nevertheless, none of the above methods explicitly model syntactic style features during the pre-training stage, which limits their ability to distinguish between generated and human-written texts.

To address these issues, this paper proposes INSA (INjected Style Augmented training), a generative AI text detection method based on part-of-speech style injection pre-training. Compared with existing methods, INSA does not rely on an extra generator or complex perturbation strategies, explicitly introduces syntactic style information during the pre-training stage, and has low inference cost. The core idea is to incorporate stable syntactic style features from traditional stylometry into the MLM continued pre-training process as conditional tokens, thereby achieving style-conditioned language representation learning.

The main contributions of this paper are summarised as follows:

1. A discrete style token construction method based on part-of-speech distribution clustering is proposed to characterise textual syntactic style.
2. A style-injected MLM continued pre-training framework is proposed, allowing the pre-trained language model to learn conditional style representations during the language modelling phase.
3. Experimental results show that explicit syntactic style modelling complements language modelling and thus improves generative AI text detection performance.

2. Related Work

2.1. Probability Distribution Based Methods

Early generative text detection methods mainly analyse language model probability distributions. Such methods typically use perplexity or token probability distribution differences to determine the source of a text [3, 10]. Because generative models tend to produce high-probability, low-entropy text, there are usually differences between generated and human-written texts in the probability space.

Perplexity-based methods perform detection by analysing the generation probability of a language model [8, 9]. These methods are simple to implement and require no additional training, but their detection capability often depends on a specific generative model. When the generative model is updated or the decoding strategy changes, detection performance may degrade. Moreover, when the detection model and the generative model have similar architectures, probability features may degenerate [8].

Existing studies have shown that generative models exhibit stable statistical patterns in part-of-speech usage, function word distribution, and syntactic structure [5, 6]. However, probability distribution based methods mainly focus on generation likelihood and do not explicitly model syntactic style information during pre-training. Therefore, relying solely on probability space differences is insufficient to stably capture style patterns in generated text.

2.2. Perturbation Analysis Based Methods

To improve detection robustness, researchers have proposed perturbation-based methods. These methods apply random substitution, masking, or rewriting perturbations to the original text and analyse the trend of probability changes before and after perturbation for detection. The representative method DetectGPT argues that generated text usually lies in high-curvature regions of the model's probability landscape and can thus be identified by perturbation stability [4].

Compared with simple probability-based methods, perturbation analysis can improve generalisation to some extent. However, such methods typically require multiple forward passes, incurring high

inference cost and low efficiency for long texts. Also, different perturbation strategies may affect detection results [4]. Therefore, the inference efficiency of these methods in real-world deployment remains limited.

2.3. Supervised Learning Based Methods

Supervised classification using pre-trained language models is a common approach for binary text classification. Such methods typically use BERT, RoBERTa, DeBERTa, etc., as encoders and fine-tune them for AI text detection. Because pre-trained models have strong contextual representation abilities, these methods achieve good performance on several public datasets [12].

However, most supervised classification methods focus on semantic representations and underutilise stable stylistic features at the syntactic level. Moreover, existing research mostly incorporates handcrafted features only at the classification stage, rather than explicitly modelling style conditioning during pre-training. In contrast, the proposed INSA introduces part-of-speech style conditioning from the pre-training stage, avoiding the limitation of simply concatenating features at the classification layer.

2.4. Stylometry and Style Modelling

Stylometry research has shown that features such as part-of-speech distribution, function word frequency, and syntactic structure can effectively reflect an author's writing style [5, 6]. Among these, part-of-speech distribution is stable and generalisable across domains because it is weakly correlated with semantic content.

Generative language models optimised for next-token likelihood tend to produce stable, high-frequency, low-entropy syntactic organisation patterns, thus exhibiting strong regularities in part-of-speech usage frequency and function word distribution. Some studies have attempted to combine stylistic features with deep learning models, e.g., by concatenating word frequency statistics or syntactic features into a classifier [7]. However, these methods typically fuse style features only at the classification stage, lacking a mechanism for conditional style modelling during language model pre-training. Hence, how to effectively incorporate style information into pre-training remains a problem worthy of further investigation.

In summary, existing methods either rely on specific generative models, have high inference costs, or do not fully utilise syntactic style information. The proposed INSA method explicitly models syntactic style patterns of text through style-injected MLM pre-training while maintaining efficient inference.

3. Method

3.1. Overall Framework

The proposed INSA framework consists of three stages:

1. Part-of-speech (POS) style feature extraction from text;
2. Style token construction based on clustering;
3. Style-injected MLM pre-training and classification fine-tuning.

First, each text in the dataset is POS-tagged, and the normalised distribution of POS tags is computed to form a style vector. Then, K-Means clustering is applied to discretise the continuous style space, and each text sample is assigned a corresponding style token. Finally, the same style token is used in two different training stages. In the continued MLM pre-training stage, it is prepended to the masked input sequence so that the encoder learns embeddings that associate syntactic style clusters with token prediction. In the classification fine-tuning stage, it is again prepended to the full input text, but the objective changes from masked-token prediction to binary provenance classification.

Given an input text x, INSA aims to learn a mapping function:

$$f(x) \rightarrow \{\text{human}, \text{AI}\}$$

where the model learns syntactic style representations of the text through style token pre-training.

Unlike traditional BERT classification frameworks, INSA explicitly introduces style conditioning information during continued pre-training, rather than only concatenating features at the classification stage. Figure 1 summarises this left-to-right data flow: POS statistics are extracted first, style clusters are converted into special tokens, and the style-conditioned encoder is then used for downstream classification.

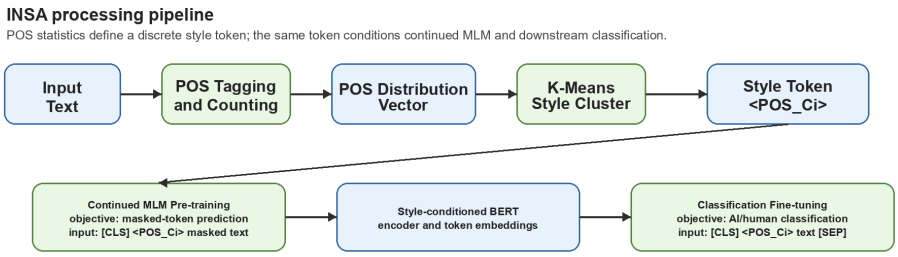

Figure 1: Overall framework of INSA. Arrows indicate the actual processing order from POS feature extraction to style-token construction, continued MLM pre-training, and downstream classification.

3.2. Part-of-Speech Distribution Feature Extraction

3.2.1. Definition of POS Distribution

The part-of-speech (POS) distribution refers to the statistical distribution of frequencies of different POS tags in a text. In stylometry, POS distribution can reflect an author's syntactic organisation habits and linguistic style. Figure 2 shows the POS distribution on the dataset.

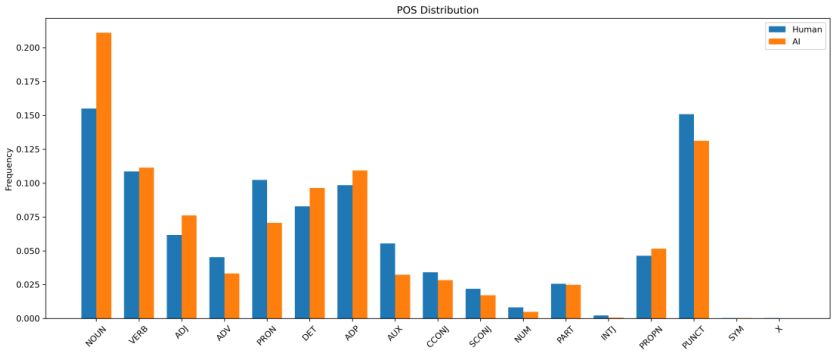

Figure 2: Statistical plot of POS distribution.

This paper adopts the 17 basic POS tags from the Universal Dependencies (UD) standard, including NOUN, VERB, ADJ, ADV, PRON, ADP, etc.

3.2.2. Single Sample Stylistic Feature Statistics

For each text sample x_i in the dataset, we first obtain the POS tag of each word using a POS tagger. Then we count the occurrence of different tag categories, where C_i denotes the count of the i-th POS category:

$$C_i = \{c_{\text{noun}}, c_{\text{verb}}, c_{\text{adj}}, \dots\}$$

Then normalise the POS counts:

$$p_k = \frac{\text{count}(k)}{\sum_{j \in \text{Tags}} \text{count}(j)}$$

Finally, we obtain the POS distribution vector for the text sample:

$$P_i = [p_1, p_2, \ldots, p_{17}]$$

This vector can describe, to some extent, the syntactic structural features and stylistic features of the text.

3.3. Style Clustering and Token Construction

To transform the continuous style space into learnable conditional representations, we apply K-Means clustering to the POS distribution vectors of all text samples. K-Means discretises the continuous style space into a finite number of categories, which facilitates integration with the Transformer special token mechanism. Let the number of clusters be K; then each sample is assigned a style token according to its cluster:

$$\texttt{<POS\_C1>}, \texttt{<POS\_C2>}, \ldots, \texttt{<POS\_CK>}$$

Subsequently, the style token is injected into the input sequence as a special token. The insertion position is fixed immediately after [CLS]:

$$[\text{CLS}]\ \texttt{<POS\_Ci>}\ \text{Text}$$

where <POS_Ci> denotes the style token corresponding to the cluster of the current text.

Compared to directly concatenating continuous statistical features, discrete style tokens are easier to integrate with the input structure of a pre-trained language model and can leverage the Transformer's representation learning ability for special tokens to model conditional information. The operation is illustrated in Figure 3; no original word token is removed, and the additional style token simply shifts the text tokens one position to the right within the maximum sequence length.

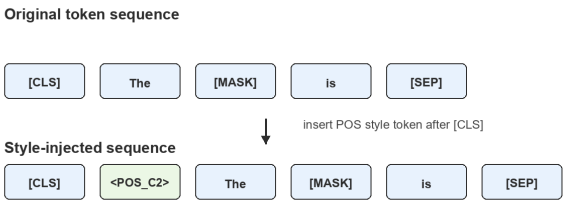

Figure 3: Schematic diagram of style token injection. The POS cluster token is inserted after [CLS], followed by the original text tokens and the final [SEP] token.

3.4. Style-Injected MLM Pre-training

During the continued pre-training phase, we employ the Masked Language Modeling (MLM) task. This step updates the BERT encoder and the newly added style-token embeddings using unlabeled training texts. Unlike traditional MLM, we prepend the style token to the input sequence, allowing the model to perceive both contextual content and style conditions when predicting masked tokens. The training target in this stage is only masked-token reconstruction, not AI/human classification.

For an input text:

$$X = [\text{CLS}]\ \texttt{<POS\_Ci>}\ w_1, w_2, \ldots, w_n$$

The model learns language representations under different style conditions through the POS token during training.

In implementation, we first add new special tokens to the tokenizer:

$$\{\texttt{<POS\_C1>}, \ldots, \texttt{<POS\_CK>}\}$$

For example:

$$[\texttt{<POS\_C2>}]\ \text{The [MASK] is important.}$$

Then we resize the token embeddings to enable the model to learn embeddings for the new tokens. The remaining text tokens are processed by the original BERT tokenizer.

We adopt a dynamic masking strategy during training with an MLM probability of 15%. Early stopping is used to prevent overfitting.

Compared with traditional continued pre-training, our method explicitly introduces style conditioning information in the MLM phase, thereby enhancing the model's ability to model syntactic features of generated text.

3.5. Classification Fine-tuning

After completing style-injected continued pre-training, we use a BERT-based classification framework for the downstream generative AI text detection task. The input format remains [CLS] <POS_Ci> text [SEP], so the model can reuse the style-conditioned representation learned during MLM. In this stage, however, the objective is supervised classification: the [CLS] vector is used as the overall representation of the text, and a fully connected classification layer outputs the class probability. The cross-entropy loss is used during training, and the best model parameters are selected based on the validation F1 score.

3.6. Algorithm Flow

```
Algorithm INSA Training Pipeline
Input: training text set D = {x1,x2,...,xn}
Output: AI text detection model M

Step1: POS tagging for training texts;
Step2: Compute POS distribution vector for each text;
Step3: Apply K-Means clustering on POS distribution vectors;
Step4: Assign style token <POS_Ci> to each text;
Step5: Inject style token into input sequence;
Step6: Continue pre-training with MLM task;
Step7: Fine-tune with classification task;
Step8: Output final AI text detection model.
```

3.7. Complexity Analysis

INSA requires only a single forward pass at inference time, does not rely on an extra generator or multiple perturbation samples, so its inference complexity is essentially the same as that of a standard BERT classifier, as shown in Table 1.

Table 1
Complexity comparison.

Method	Multi-forward	Extra Generator
DetectGPT	✓	✓
Binoculars	✓	✓
INSA (ours)	×	×

4. Experiments

4.1. Dataset

Our experiments are conducted on the PAN 2026 Generative AI Detection dataset released for the PAN 2026 shared task [2]. The dataset contains human-written texts and texts generated by various generative models. The task is to determine whether an input text is produced by a generative model.

We split the dataset into training, validation, and test sets. Since the test set labels are invisible during evaluation, our experimental analysis is mainly performed on the validation set.

4.2. Evaluation Metrics

We adopt the official PAN evaluation metrics:

- ROC-AUC;
- Brier Score;
- C@1;
- F1;
- F0.5u.

The final result uses the mean of the above five metrics (Mean) as the comprehensive evaluation indicator. The F0.5u metric further evaluates the model's stability in high-confidence prediction scenarios by filtering out low-confidence samples.

4.3. Experimental Settings

We use `bert-base-uncased` as the base encoder from HuggingFace Transformers. Continued MLM pre-training is performed to build the INSA model. Main experimental parameters are as follows:

- MLM pre-training batch size: 8;
- Classification fine-tuning batch size: 4;
- Learning rate: 2e-5;
- Maximum input length: 512;
- Number of clusters K: 5;
- Training epochs: 10;
- MLM mask probability: 15%;
- Random seed: 42.

Weight decay and early stopping are used to improve training stability.

4.4. Experimental Results

To verify the effectiveness of our method, we compare it with the official baselines provided by PAN:

- **Linear SVM with TF-IDF**: A traditional machine learning text classification method that extracts TF-IDF features and uses a linear SVM classifier.
- **PPMd Compression-based Cosine**: A compression-based detection method that uses the PPMd compression algorithm to measure information similarity between texts and then uses cosine similarity for comparison.
- **Binoculars**: A training-free detection method that compares the perplexity difference between two large language models to determine the source of a text.

The official baselines and our experimental results are shown in Table 2.

The ROC-AUC reaches 0.998, indicating that the model can distinguish between positive and negative samples well. The F1 and F0.5u are 0.980 and 0.972 respectively, showing stable performance under high-confidence prediction scenarios on the validation set. Overall, the conditional pre-training strategy based on part-of-speech style injection enhances the model's ability to model stylistic features of generated texts.

Table 2
Experimental results of INSA and baselines on the PAN 2026 validation set.

Model	ROC-AUC	Brier	C@1	F1	F0.5u	Mean
Binoculars	0.918	0.867	0.844	0.872	0.882	0.877
PPMd Compression-based Cosine	0.786	0.799	0.757	0.812	0.778	0.786
Linear SVM with TF-IDF	0.996	0.951	0.984	0.980	0.981	0.978
INSA (ours)	**0.998**	**0.975**	**0.973**	**0.980**	**0.972**	**0.979**

4.5. Ablation Study

To validate the effectiveness of each component of INSA, we design an ablation study. We remove the POS style token (POS Token) and the continued MLM pre-training (Continue MLM) modules respectively, and evaluate on the official dataset.

Because the test set labels are unavailable, the ablation study is performed on the validation set for modular analysis. Therefore, the results are only used to observe the contribution trends of the modules and are not directly comparable with the official evaluation results. As the validation set is also used for parameter selection and model analysis, the reported metrics may be higher than those in the official evaluation.

Four settings are compared:

- **BERT_Baseline**: Directly use the original BERT classification model.
- **Continue_MLM**: Only apply MLM continued pre-training.
- **POS_Token**: Only inject POS clustering labels (without continued MLM).
- **INSA**: Use both POS style token and MLM continued pre-training.

The results are shown in Table 3.

Table 3
Ablation study results on the validation set.

Model	ROC-AUC	Brier	C@1	F1	F0.5u	Mean
BERT_Baseline	0.998	0.989	0.987	0.991	0.991	0.991
Continue_MLM	0.998	0.990	0.987	0.991	0.993	0.992
POS_Token	0.998	0.974	0.971	0.979	0.970	0.978
INSA	**0.999**	**0.991**	**0.990**	**0.993**	**0.992**	**0.993**

From the experimental results we observe:

1. Continued MLM pre-training brings stable but moderate gains. Compared with BERT_Baseline, Continue_MLM improves the Mean score from 0.991 to 0.992 and F0.5u from 0.991 to 0.993, while ROC-AUC, C@1, and F1 remain nearly unchanged. This suggests that domain continued pre-training mainly improves calibration and high-confidence prediction stability.
2. Using only the POS style token yields limited effectiveness. POS_Token decreases the Mean score from 0.991 to 0.978, C@1 from 0.987 to 0.971, F1 from 0.991 to 0.979, and F0.5u from 0.991 to 0.970 compared with BERT_Baseline. This indicates that explicit style-token injection alone is insufficient; without continued pre-training, the model cannot reliably associate the new tokens with meaningful syntactic style information.
3. INSA joint modelling achieves the best overall performance. Compared with BERT_Baseline, INSA improves ROC-AUC from 0.998 to 0.999, Brier from 0.989 to 0.991, C@1 from 0.987 to 0.990, F1 from 0.991 to 0.993, and Mean from 0.991 to 0.993. Compared with POS_Token alone, INSA raises the Mean score by 0.015, showing that continued MLM pre-training is important for making the injected POS style tokens useful.

4. POS style information and language modelling are complementary. The gap between POS_Token and INSA suggests that style tokens should not be treated as simple labels appended at inference time. Instead, they become effective only when the encoder is first adapted through MLM to model the relationship between style clusters and contextual token distributions.

5. Conclusion

This paper proposes a generative AI text detection method named INSA (INjected Style Augmented training) based on part-of-speech style injection pre-training. The method first constructs a textual style representation using the distribution of part-of-speech tags, generates discrete style tokens via K-Means clustering, then injects the style token into the input sequence during the MLM pre-training phase to guide the model toward learning conditional style representations, and finally performs AI-generated text detection on the downstream classification task. Experimental results on the PAN 2026 Generative AI Detection dataset demonstrate the effectiveness of the proposed method, confirming that explicit syntactic style conditioning and semantic representations from language modelling are complementary. Future work includes exploring finer-grained stylistic representations and investigating the cross-model and cross-domain generalisation ability of the method.

Acknowledgments

The authors would like to thank the PAN organisers and the providers of the PAN 2026 Generative AI Detection dataset.

Declaration on Generative AI

The authors have not employed any Generative AI tools in the preparation of this work.

References

[1] J. Bevendorff, M. Fröbe, A. Greiner-Petter, A. Jakoby, M. Mayerl, P. Nakov, H. Plutz, M. Potthast, B. Stein, M. N. Ta, Y. Wang, and E. Zangerle. Overview of PAN 2026: Voight-Kampff Generative AI Detection, Text Watermarking, Multi-Author Writing Style Analysis, Generative Plagiarism Detection, and Reasoning Trajectory Detection. In *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, 2026.

[2] J. Bevendorff, R. R. Gunti, J. Karlgren, M. Fröbe, M. Potthast, and B. Stein. Overview of the third "Voight-Kampff" Generative AI / LLM Detection Task at PAN and ELOQUENT 2026. In *Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum*, CEUR Workshop Proceedings. CEUR-WS.org, Jena, Germany, 2026.

[3] S. Gehrmann, H. Strobelt, A. M. Rush. GLTR: Statistical Detection and Visualization of Generated Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, 2019.

[4] E. Mitchell, Y. Lee, A. Khazatsky, et al. DetectGPT: Zero-Shot Machine-Generated Text Detection Using Probability Curvature. In *Proceedings of the 40th International Conference on Machine Learning*, PMLR, 2023, pages 24950–24962.

[5] A. Uchendu, T. Le, K. Shu, et al. Authorship Attribution for Neural Text Generation. *arXiv preprint*, arXiv:2001.06775, 2020.

[6] T. Fagni, F. Falchi, A. Gabrielli, et al. TweepFake: About Detecting Deepfake Tweets. *arXiv preprint*, arXiv:2008.00036, 2020.

[7] I. Solaiman, M. Brundage, J. Clark, et al. Release Strategies and the Social Impacts of Language Models. *arXiv preprint*, arXiv:1908.09203, 2019.

[8] D. Ippolito, Y. Du, O. Levy, et al. Automatic Detection of Generated Text is Easiest when Humans are Fooled. *arXiv preprint*, arXiv:1911.00650, 2019.

[9] C. Zerva, P. Zervas, D. Koutsomitropoulos, et al. Benchmarking Human versus Machine-Generated Text Detection Models. *arXiv preprint*, arXiv:2211.06860, 2022.

[10] E. Andersson. AI or Human? Detecting Machine-Generated Text Through Perplexity. Master's thesis, Uppsala University, 2024.

[11] Y. Yuan, D. F. Wong, L. Liu, et al. A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions. *Computational Linguistics*, 51(1):1–48, 2025.

[12] B. Guo, Z. Pan, Z. Liu, et al. Machine-Generated Text Detection: A Survey on Methods and Datasets. *arXiv preprint*, arXiv:2403.12350, 2024.

[13] J. Su, T. Y. Zhuo, Q. Ai, et al. DetectLLM: Leveraging Log Rank Information for Zero-Shot Detection of Machine-Generated Text. *arXiv preprint*, arXiv:2305.19201, 2023.