Adept: AI-Generated Text Detection Based on Phrasal Category N-Grams

Notebook for PAN at CLEF 2025

Felix Völpel[1,*], Oren Halvani[1]

[1]Fraunhofer Institute for Secure Information Technology SIT, Rheinstraße 75, 64295 Darmstadt, Germany

Abstract

With the advent of large language models (LLMs), the generation of artificial text has become remarkably accessible and is increasingly integrated into everyday applications. As the use of LLMs to produce content becomes more widespread, the ability to distinguish between AI-generated and human-written texts has grown in importance. This year's PAN competition focuses on this specific challenge: Based on a text, participants must determine whether it was written by a human or generated by an AI system (more specifically, an LLM). We propose a classification approach called Adept, which explicitly leverages constituent trees to model the grammatical structure of texts. For each sentence, we generate a constituent tree and represent the entire text by aggregating the distribution of syntactic n-grams, defined as paths of a fixed length within these trees. Using these structural representations, we train a multilayer perceptron (MLP) to classify authorship. Adept achieves a mean score of 0.843 on the test dataset, evaluated by the organizers of the competition. This ranks us on rank 16 out of 24 with a score difference of 0.056 to the first place and 0.036 to the third place.

Keywords

PAN 2025, Subtask 1: Voight-Kampff AI Detection Sensitivity, Constituent Tree n-grams, AI-generated Text Detection

1. Introduction

The development and integration of LLMs into various aspects of our daily personal and professional life has rapidly increased over the last years. From support in writing emails or reports to assistance in generating source code or solving mathematical tasks. The number of use cases that are involved is manifold. Also, ever since the release of ChatGPT in 2022 multiple LLMs of various sizes, shapes and costs have emerged. Therefore, there exist plenty of possibilities for users to apply any LLM. However, while the advantages are given, there are clearly downsides of this development too. Examples are the extensive generation and dissemination of false information or the violation of plagiarism. Being able to distinguish whether a text has been AI generated or is an original of a human is therefore crucial to have full transparency. To achieve this, academic research has to adapt and be stimulated to develop effective classifiers. This motivates this year's PAN-Webis competition in Voight-Kampff Generative AI Detection [1] [2]. Two subtasks, focused on the separation of AI-generated texts from human written originals, have been hosted. We participated in *Subtask 1: Voight-Kampff AI Detection Sensitivity* [1], which deals with the following challenge. Given a text, that has either been written entirely by a **human** or entirely been generated by a **LLM**, a model has to predict which one is true [2].

There has already been carried out substantial research in the field of AI-generated text detection. Ghosal et al. [3], for example, provide a survey of recent methods and challenges to detect AI-generated texts. Prominent approaches base on a watermarking method to embed signals in AI-generated texts to identify them later [4]. Other methods could be post-hoc detectors that involve techniques like zero-shot detection. This might be by using pre-trained language models or fully separate statistical approaches that calculate entropy, perplexity measures or n-gram frequencies. The authors also discuss

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain

felix.voelpel@sit.fraunhofer.de (F. Völpel); oren.halvani@sit.fraunhofer.de (O. Halvani)

0009-0005-7688-7036 (F. Völpel); 0000-0002-1460-9373 (O. Halvani)

[1]https://pan.webis.de/clef25/pan25-web/generated-content-analysis.html

existing challenges, that make the detection of AI-generated content difficult. As such they mention paraphrasing attacks, in which AI-generated texts can be rephrased to evade detection. Other examples mentioned are the generative attack or the spoofing attack. Open questions that remain, are, according to the authors, how detectors could be standardized or if watermarking should be regulated to prevent misuse. Other aspects include the improvement of fairness and robustness of text detectors among others.

In last year's PAN competition, the following task was hosted, i. e., given two texts, among which one text was AI-generated and one was human-authored, the participants had to develop a model, that was capable of identifying the human-written one [5]. The organizers summarized, that most solutions based on a BERT-based classification with some additional modifications like PU-loss or R-Drop. There were also approaches, that developed classifiers such as LSTMs or XGBoost based on engineered features like perplexity, token distributions of general stylometrics. The best performing solution achieved an AUC of 0.961 and a $c@1$ score of 0.912 [6]. The authors implemented a special kind of ensemble, involving two LLMs fine-tuned with a classification head and the Binoculars scores. Binoculars, which is also selected among the baselines of this year's competition, is a zero-shot detection method, that employs two pre-trained LLMs [7]. It contrasts the perplexity, given by one LLM for the available text to the cross-perplexity between both LLMs. These scores allow for excellent separation as is shown in their publication.

At its core, Adept relies on **constituent trees** extracted from the text. A constituent tree (also called a phrase structure or parse tree) is a hierarchical, tree-like representation of the syntactic structure of a sentence according to a phrase structure grammar (such as Context-Free Grammar). It shows how a sentence is composed of nested constituents (phrases) such as noun phrases (**NP**), verb phrases (**VP**) and prepositional phrases (**PP**), each of which can be further broken down into smaller constituents or individual words (terminals). For each sentence in a text, we generate a constituent tree. We then convert the complete set of trees into a numerical vector representation by computing the relative frequencies of syntactic paths of a fixed length across all trees in the text. These vectorized representations are used as input to a neural network classifier. In our case, we use a simple MLP trained to distinguish between AI-generated and human-authored texts. By incorporating the grammatical structure of the text into the classification process, this approach achieves a **mean score of 0.843** on the final test dataset.

This paper is organized as follows. Section 2 introduces the datasets made available by the organizers of the competition. In Section 3, we describe our solution to the task, followed by Section 4, which reports the results obtained on the final test dataset. We then discuss, in Section 5, the evaluation framework of the competition and our proposed method and highlight possible improvements. Finally, Section 6 presents the conclusions.

2. PAN Dataset

The organizers of the PAN competition provide a comprehensive training and validation dataset comprising a mix of AI-generated and human-authored texts. Figures 1, 2, and 3 present an exploratory analysis of the training data. In total there are 23,707 texts in the training set. Among these there are 14,606 AI-generated texts, which amounts to a ratio of 61.6%, and 9,101 human-authored texts, which relates to a share of 38.4%. In this regard, it is important to emphasize that this dataset is highly **unbalanced** with respect to the ratio of AI-generated to human-authored texts. The same holds for the validation dataset. In total there are 2,312 (64.4%) AI-generated and 1,277 (35.6%) human-written texts. We elaborate on the effects of this distribution in Section 5. The number of applied AI-models used to generate the texts is versatile, as can be seen in Figure 1. This is beneficial, as it enables greater generalizability of the developed classification model. Furthermore, the respective texts have a varying length (Figure 2). While AI-generated and human-written texts are mostly between 500 and 1,000 words long, the human-authored texts are more often longer. There are also some extremes, as can be seen,

since there seem to be both very short and very long texts available. The plots further reveal that any text belongs to one of three possible genres with fiction writing being represented the most (Figure 3).

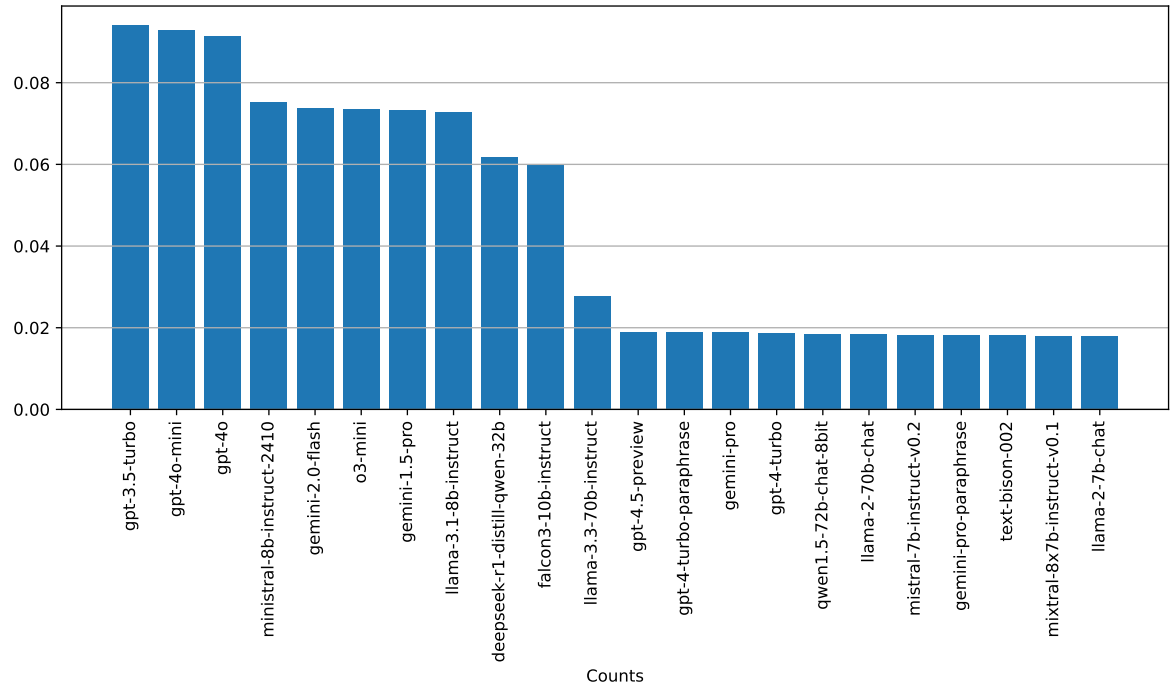


Figure 1: Ratio of the respective LLM used to generated texts in the training corpus.

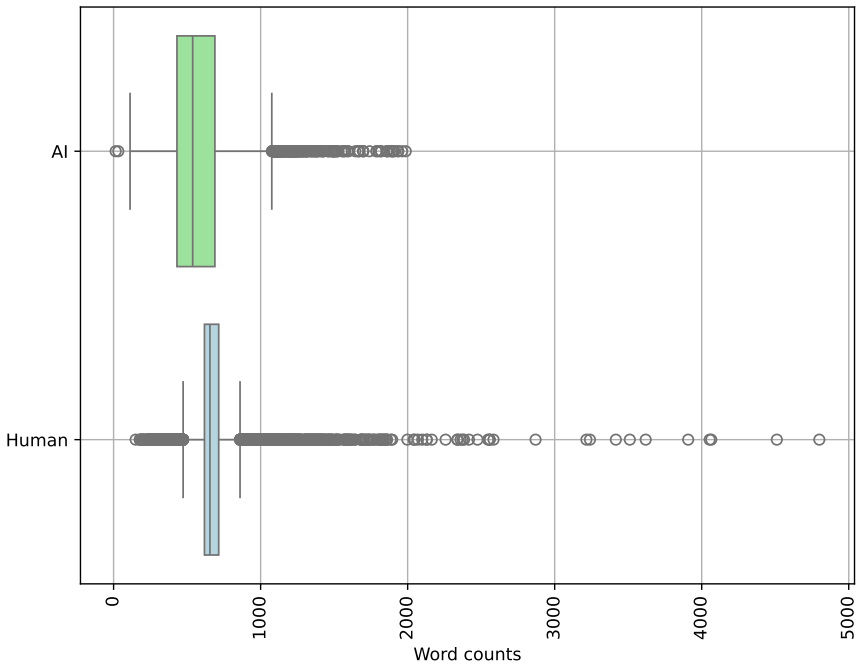


Figure 2: Distribution of number of words conditioned on authorship (AI vs. Human) in the training corpus.

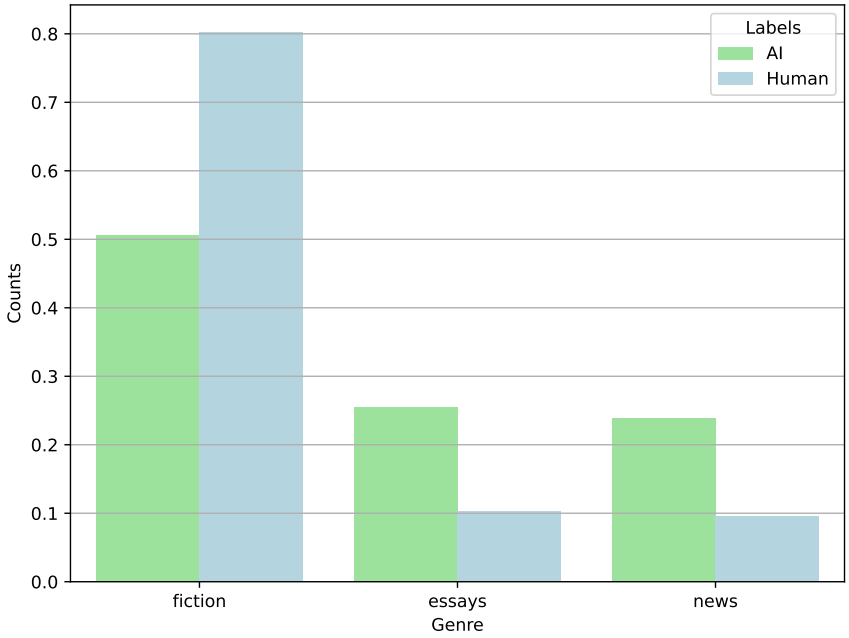


Figure 3: Number of AI-written and human-written texts conditioned on genre in the training corpus.

3. Proposed Solution

3.1. Algorithm

Our proposed approach bases inherently on **capturing the grammatical structures** of the sentences of a text. Our main assumption lies in the hypothesis, that the complexity and structural composition of sentences contains discriminative information to differentiate AI-generated from human-authored texts. To do so, the central element is the generation of constituent trees. Constituent trees have a long tradition in linguistics. A constituent tree is a hierarchical representation of the grammatical units of a sentence. A constituent is either a word or a compound object, that represents a sub-phrase of a sentence. Each constituent, except for single words, consists of other constituents again, which is why a tree graph structure evolves. A comprehensive introduction into constituent trees can be found in [8]. An illustration of a constituent tree (also known as tree diagram [8]) can be seen in Figure 4. In that case, the sentence consists of a noun phrase and a verb phrase on the highest level. In our approach, we deal with a slightly different version of a constituent tree, i. e., leave nodes, which originally represent single words, are replaced by their respective part of speech tag. A constituent tree gives a profound insight into the grammatical complexity of a sentence. The height of such trees, the number of unique phrase-types are, for example, entities that contain relevant information.

Our method is composed of three successive steps, which are summarized in the Figures 5, 6, 7 and described in what follows.

Step-1: In the first step (Figure 5), each text is segmented into sentences and **for each sentence of a text** the respective constituent tree is constructed. As a result, one receives for each text a set of constituent trees. In our approach, the leave nodes of any constituent tree are part of speech tags of the respective word that is being replaced.

Step-2: In the second step (Figure 6) these sets of constituent trees are transformed into a numeric

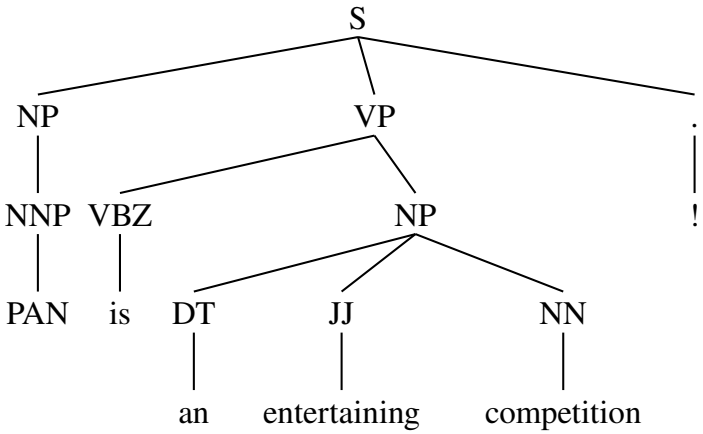


Figure 4: Example showing the resulting constituency tree for the sentence: *PAN is an entertaining competition!*.

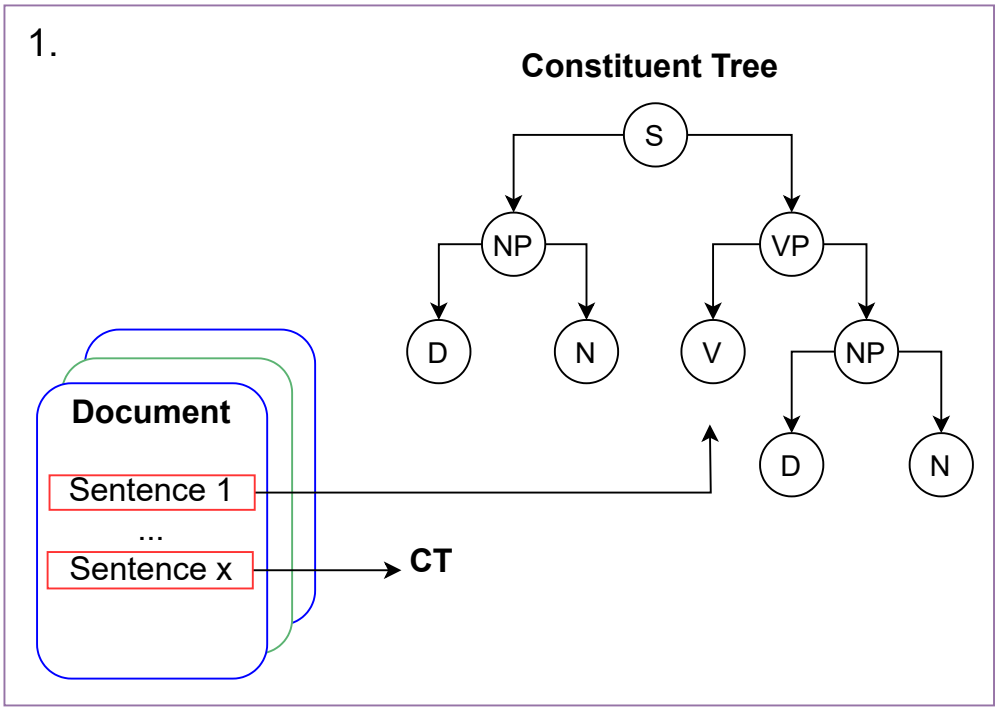


Figure 5: First step of our algorithm is the construction of a constituent tree for each sentence of a text.

representation. To do so, for each tree, we count the occurring n-grams. By this, we mean, **counting the frequencies of unique paths of length** N, starting at any node in the tree and following the directions of the edges. We interpret any edge in the tree to be directed and pointing downwards in the tree. This is done for all numbers $N = 1, \ldots, 7$, at which $N = 1$ simply represents the counts of occurring node types. Afterwards, the **frequencies are aggregated** for each N **across all trees belonging to the same text**. Subsequently, these frequencies are normalized for each N, such that the corresponding vector represents relative frequencies. As a result of this step, we obtain for each text seven vectors (one for each N), that represents the relative frequencies of unique paths of a specific length across all trees belonging to the same text. (We ensure that these vectors have the same length for each text, such that unique paths that do not occur in one text are still represented by 0.) These seven vectors are concatenated and finally represent the original text. The final vector is an interpretable representation of the grammatical structure of a text.

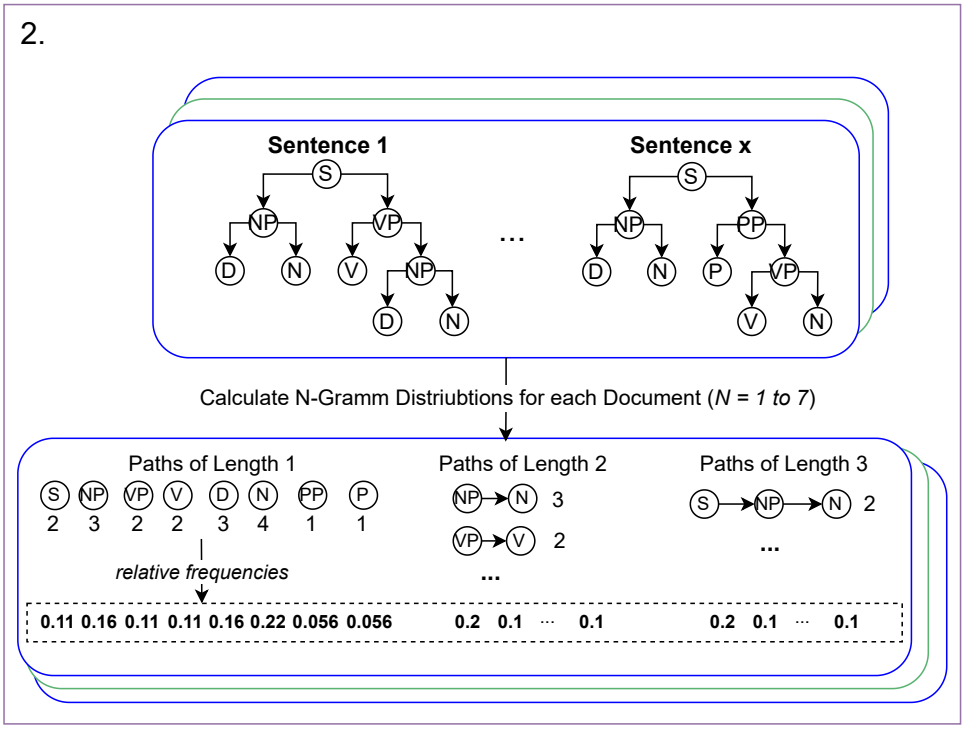


Figure 6: Second step of our algorithm is counting all n-grams across all trees belonging to the same text. These frequencies are normalized to relative ones.

Step-3: The final step (Figure 7) consists of **training a classifier** based on the generated dataset, received after step-2. To do so we repeat step-1 and step-2 for both the training and validation dataset. As a classifier, we have implemented a simple MLP. For the optimization we used the Adam method with a fixed learning rate of 0.001. The MLP was composed of 3 hidden layers, each with a dimension of 1000 neurons. Together with the validation dataset we manually tuned the number and size of the hidden layers. We executed the training for 30 epochs and subsequently plotted the curve of the accuracy on the validation set to select the best possible model.

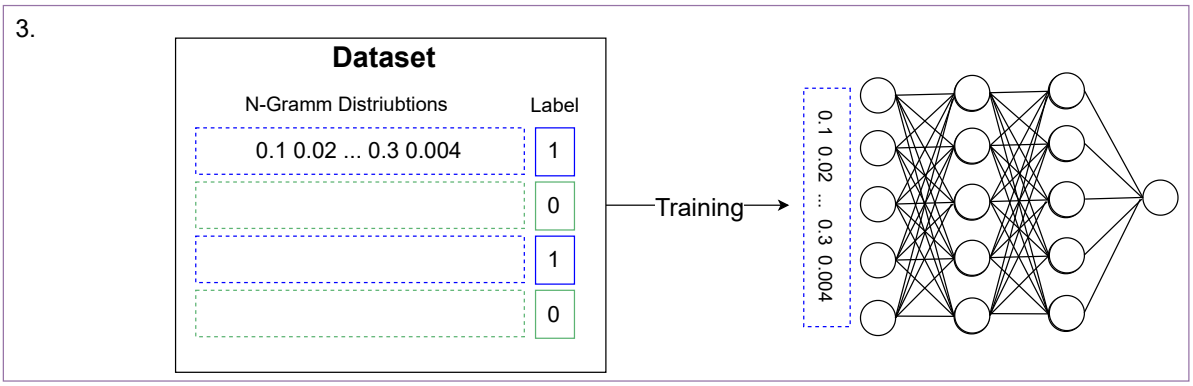


Figure 7: Third step of our algorithm is the training of a MLP to classify whether a text, represented by the distributions of n-grams of the respective constituent trees, was AI-generated or human-written.

3.2. Implementation Details

The implementation was carried out in Python. We used the *Constituent Treelib (CTL) package*[2] [9] for the generation of the constituent trees and torch to develop the neural network. The CTL package only provides a bracketed string representation of a constituent tree. Therefore, we had to develop a parser, that processes such a string and transforms it into a list of edges and a dictionary, that maps each node index to the respective node type of the constituent tree. Furthermore, it should be noted that the computation of a constituent tree for a given sentence takes several seconds. This does not scale very well for large corpora and long texts. To speed up the computation we parallelized the processing of different texts across multiple CPU cores on our cluster. In total, step-1 took ≈ 9 hours for the training corpus to complete and ≈ 1.5 hours for the validation data. The final evaluation on the test dataset, which was for obvious reasons inaccessible to us, was executed by the organizers of the competition. The solution was uploaded to Tira, an environment that provided access to computing resources and was used to manage the submitted solutions [10].

4. Results

In Table 1, the final results of all evaluated solutions, as published by the organizers of the competition, are shown [2]. As can be seen, we obtain a good $c@1$ score of 0.816, which equals in our case accuracy, since our method does not produce any non-predictions, and an overall **mean score of 0.843**[3]. This beats two out of three baselines and is also close to the best performing baseline, i. e., linear SVM with TF-IDF. Overall, the performances are rather close to each other. Our approach has a difference of only 0.02 to the top-10 method and a score difference of 0.056 to the winning team. This shows, that the submitted models all perform within a similar range.

5. Discussion

In the following, we will scrutinize two aspects, i. e., on the one hand, the framework used for evaluation and on the other hand our method as such and how it could be advanced.

5.1. Discussion on the evaluation

The overall score in the evaluation is computed as the average of five different metrics. However, it is generally not meaningful to compute an average across heterogeneous performance metrics such as AUC, F_1, $c@1$ (which, in our case, is equivalent to Acc) and the (inverted) Brier score. Each metric captures a distinct aspect of the model's performance. The F_1 score quantifies the trade-off between precision and recall. The AUC measures the model's ability to rank positive instances ahead of negative ones. The Brier score, which is essentially the standard Mean Squared Error (MSE), evaluates the accuracy of predicted probabilities, that is, the model's calibration.

Besides this, there are other challenges as well. Since our approach always returns a prediction, the $c@1$ measure degenerates to the standard accuracy measure. This can be proven easily. According to [11], this measure is defined as follows:

$$c@1 = \frac{1}{n}\left(n_c + \frac{n_c \cdot n_u}{n}\right)$$

Here, n denotes the total number of instances in the dataset, n_c is the number of correct predictions (i. e., true positives and true negatives, or TP + TN) and n_u is the number of non-predictions, that is,

[2]The package is available under: https://github.com/Halvani/Constituent-Treelib
[3]Note, the Brier score is actually the inverted value, i. e., 1-Brier, such that big values correspond to better performance.

Table 1
Final results of the submitted solutions on the test dataset, as evaluated and published by the PAN organizers. The last two columns show the balanced accuracy and the corresponding rank, that was calculated by us based on the published FPR and FNR values.

rk	Team	AUC	Brier	$c@1$	F_1	$F_{0.5u}$	**Mean**	FPR	FNR	BAC	rk_{BAC}
1	mdok	0.853	0.896	0.894	0.898	0.903	**0.899**	0.108	0.094	0.899	1
2	steely	0.842	0.879	0.877	0.865	0.881	**0.880**	0.151	0.100	0.875	8
3	nexus-interrogators	0.865	0.874	0.870	0.860	0.881	**0.879**	0.159	0.083	0.879	6
4	yangjlg	0.845	0.878	0.871	0.856	0.881	**0.877**	0.172	0.062	0.883	4
5	cnlp-nits-pp	0.825	0.873	0.873	0.854	0.882	**0.874**	0.176	0.050	0.887	2
6	unibuc-nlp	0.828	0.885	0.864	0.845	0.876	**0.872**	0.187	0.052	0.881	5
7	moadmoad	0.822	0.866	0.865	0.855	0.882	**0.871**	0.175	0.058	0.884	3
8	iimasnlp	0.838	0.868	0.856	0.851	0.877	**0.869**	0.171	0.077	0.876	7
9	bohan-li	0.848	0.858	0.852	0.847	0.870	**0.866**	0.174	0.092	0.867	12
10	advacheck	0.802	0.855	0.855	0.854	0.879	**0.863**	0.169	0.084	0.874	9
11	hello-world	0.838	0.871	0.836	0.827	0.862	**0.856**	0.153	0.128	0.860	13
x	Baseline TF-IDF	0.838	0.871	0.836	0.827	0.862	**0.856**	0.153	0.128	0.856	x
12	xlbniu	0.794	0.847	0.847	0.840	0.869	**0.854**	0.188	0.077	0.868	10
13	shushantatud	0.823	0.850	0.840	0.831	0.862	**0.852**	0.203	0.093	0.852	14
14	ds-gt-pan	0.803	0.844	0.844	0.835	0.867	**0.851**	0.195	0.070	0.868	11
15	styloch	0.793	0.866	0.821	0.823	0.853	**0.844**	0.201	0.131	0.834	19
16	**felix-volpel**	0.815	0.854	0.816	0.822	0.855	**0.843**	0.212	0.115	0.837	18
17	sinai-inta	0.811	0.841	0.807	0.818	0.860	**0.838**	0.222	0.079	0.850	15
18	pindrop	0.782	0.854	0.814	0.815	0.853	**0.835**	0.211	0.115	0.837	17
19	diveye	0.786	0.828	0.806	0.823	0.862	**0.831**	0.211	0.104	0.843	16
20	s-titze	0.797	0.848	0.798	0.807	0.849	**0.827**	0.243	0.131	0.813	20
x	Baseline Binoculars	0.760	0.835	0.793	0.802	0.831	**0.818**	0.206	0.200	0.80	x
21	iunlp	0.734	0.799	0.799	0.829	0.850	**0.814**	0.178	0.210	0.806	21
22	hiwiy	0.765	0.806	0.771	0.830	0.791	**0.807**	0.000	0.636	0.682	24
23	team-a	0.603	0.783	0.783	0.824	0.801	**0.788**	0.049	0.457	0.747	23
x	Baseline PPMd	0.636	0.795	0.735	0.763	0.771	**0.758**	0.129	0.499	0.686	x
24	asdkklkk	0.718	0.739	0.739	0.726	0.781	**0.753**	0.308	0.110	0.791	22

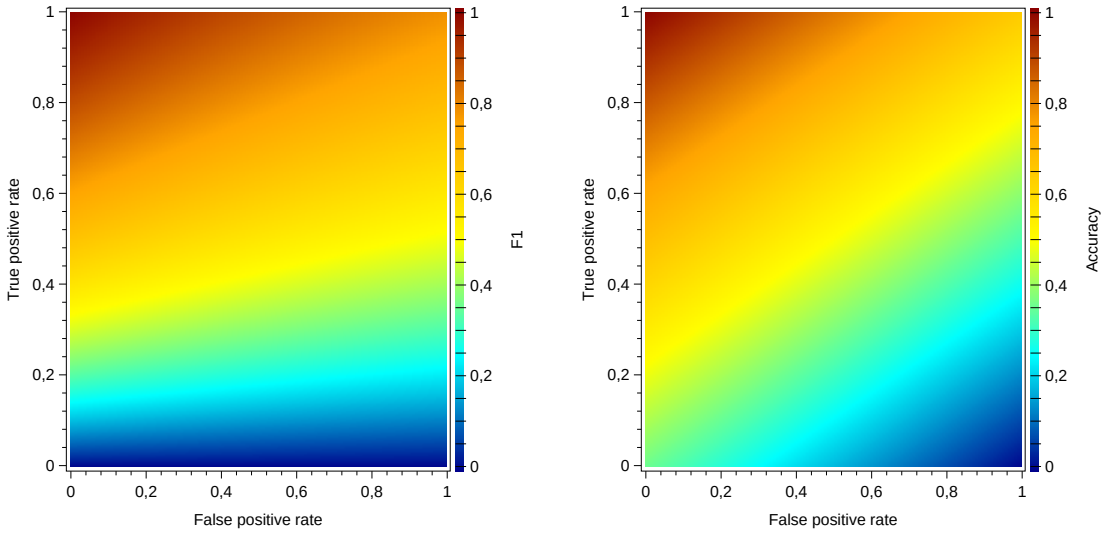
instances for which no prediction was provided. Since our approach always produces a prediction, $n_u = 0$, causing the term $\frac{n_c \cdot n_u}{n}$ to vanish. What remains is:

$$\frac{n_c}{n} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \text{Accuracy}$$

Although, we had no insight into the test dataset, the training, as well as the validation dataset, were unbalanced regarding the ratio of AI-generated and human-authored texts, as was presented in Section 2. This might indicate that the test dataset was assembled with a similar imbalance. In the context of machine learning, it is widely acknowledged that using accuracy as an evaluation metric on unbalanced test sets is not advisable. This is because accuracy can be overly influenced by the majority class, potentially masking poor performance on the minority class. Consequently, the use of $c@1$ (which is equivalent to accuracy) is of limited value and **does not provide a reliable assessment of model performance**.

Beyond the limitations of $c@1$, there are additional concerns within the evaluation framework that merit attention in the context of this year's competition. Notably, F_1 introduces another drawback, as this measure is **not symmetric with respect to the class labels**. In other words, F_1 depends on the designation of the positive class, which in the context of the competition is defined as "AI-generated texts." In this case, the positive class also constitutes the majority class, which leads to counterintuitive effects when interpreting the F_1 score. Figure 8 contrasts the F_1 score with the accuracy across the

full space of possible confusion-matrix outcomes. Each metric is visualized as a heat-map surface in ROC space, where the x-axis is the false-positive rate and the y-axis is the true-positive rate; the underlying class counts arise from human- and AI-generated texts in the validation data, treated as positives and negatives, respectively. When the positive class predominates, the two surfaces become almost indistinguishable: the scarcity of negatives makes the false-positive term negligible, so both metrics depend almost entirely on the true-positive rate. In this high-prevalence regime the F_1 score collapses to a mildly non-linear yet very close approximation of accuracy, and the usual distinction between *overall correctness* and *precision–recall balance* effectively disappears.



Isometric plot for F_1. Isometric plot for Accuracy.

Figure 8: A comparison between the F_1 score and accuracy, based on all possible confusion matrix configurations, presented using heatmaps plotted in ROC space (with respect to the number of human- and AI-generated texts i. e., **positives** and **negatives**).

As a **simple countermeasure**, we propose to use **balanced accuracy** instead, which is defined as [12]:

$$\mathrm{BAC} = \frac{\mathrm{TPR} + \mathrm{TNR}}{2} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}}\right)$$

In Table 1 we have appended the values for the balanced accuracy and the corresponding ranking. We can use the deviance between this and the original ranking as indicator of how strong the imbalance in the test dataset might be and, as a consequence, how reliable the published results are. A more in-depth analysis reveals, that the average absolute rank difference is 2.4 positions. It can be seen that, for example, the top-10 approaches remain mostly in the top-10. The biggest decrease is regarding the (originally) second approach that ranks at place 8 according to the balanced accuracy. Our approach stays about the same with a slight decrease from rank 16 to 18.

5.2. Discussion on the method

The proposed method, Adept, is easy to implement and bases on an intuitive logic which leads to fairly good results. This backs our assumption, that constituent trees contain relevant information about the grammatical structure that is unique for an author and can therefore be used for the given classification task. There are two lines, along which we can envision straightforward improvements to be implemented.

The generation of the constituent trees does not scale well for long texts. To counteract this behavior one has to either search for a more efficient implementation or apply parallelization. Another, straight-

forward, workaround is to **sample sentences of a text at random** and only compute the trees for this sample. Obviously, this goes hand in hand with a loss of information.

Another aspect regards the constituent tree representation. The considered path lengths of 1 up to 7, to generate the tree n-grams, were chosen a-priori. However, we didn't optimize the model considering subsets or even longer paths. Furthermore, one might think of involving **graph neural networks** to classify the constituent trees or to generate some kind of embeddings. However, it remains an open question **how to aggregate the information of multiple constituent trees** into one common representation of an entire text. In [13], Vinyal et al. developed an attention-enhanced sequence-to-sequence model for syntactic constituency parsing based on a LSTM encoder-decoder framework. To process constituency trees, they used a linearized representation, based on a depth-first traversal order. Hence, they have presented an efficient way to process as well as to generate constituency trees, something that might be applicable to the here presented use case too.

6. Conclusion

In this paper, we presented our solution to this year's PAN competition about AI-generated text detection. Given a document, which was either AI-generated or human-written, a model has to decide the correct case. Core element of our approach is the concept of the constituent tree. A constituent tree is a hierarchical representation of grammatical sub-phrases of a sentence. The leave nodes represent the single words, although in our implementation they have been replaced with the respective words' part of speech tags. By generating such a tree for each sentence of a text we receive a rich pool of information, that captures the grammatical and syntactical complexity of the sentences of a text. We represent such a set of trees by the distribution of occurring paths of a specified length in any tree belonging to the same text. The received dataset of path distributions, representing each text of the corpus, was used to train a neural network to classify whether a text was AI-generated or human-written. In the final evaluation, we obtain fairly good results, achieving an overall mean score of 0.843. Possible improvements of this approach encompass the computational effort of constructing the constituent trees as well as other ideas to numerically represent such trees, potentially by incorporating other neural network structures such as graph neural networks.

7. Acknowledgments

This research work was supported by the National Research Center for Applied Cybersecurity ATHENE. ATHENE is funded jointly by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research and the Arts.

Declaration on Generative AI

During the preparation of this paper, the authors used no GenAI tools.

References

[1] J. Bevendorff, D. Dementieva, M. Fröbe, B. Gipp, A. Greiner-Petter, J. Karlgren, M. Mayerl, P. Nakov, A. Panchenko, M. Potthast, A. Shelmanov, E. Stamatatos, B. Stein, Y. Wang, M. Wiegmann, E. Zangerle, Overview of PAN 2025: Voight-Kampff Generative AI Detection, Multilingual Text Detoxification, Multi-Author Writing Style Analysis, and Generative Plagiarism Detection, in: J. C. de Albornoz, J. Gonzalo, L. Plaza, A. G. S. de Herrera, J. Mothe, F. Piroi, P. Rosso, D. Spina, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction.

Proceedings of the Sixteenth International Conference of the CLEF Association (CLEF 2025), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2025.

[2] J. Bevendorff, Y. Wang, J. Karlgren, M. Wiegmann, M. Fröbe, A. Tsivgun, J. Su, Z. Xie, M. Abassy, J. Mansurov, R. Xing, M. N. Ta, K. A. Elozeiri, T. Gu, R. V. Tomar, J. Geng, E. Artemova, A. Shelmanov, N. Habash, E. Stamatatos, I. Gurevych, P. Nakov, M. Potthast, B. Stein, Overview of the "Voight-Kampff" Generative AI Authorship Verification Task at PAN and ELOQUENT 2025, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS.org, 2025.

[3] S. S. Ghosal, S. Chakraborty, J. Geiping, F. Huang, D. Manocha, A. S. Bedi, Towards Possibilities & Impossibilities of AI-generated Text Detection: A Survey, 2023. URL: https://arxiv.org/abs/2310.15264. arXiv:2310.15264.

[4] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, T. Goldstein, A watermark for large language models, 2024. URL: https://arxiv.org/abs/2301.10226. arXiv:2301.10226.

[5] A. A. Ayele, N. Babakov, J. Bevendorff, X. B. Casals, B. Chulvi, D. Dementieva, A. Elnagar, D. Freitag, M. Fröbe, D. Korenčić, M. Mayerl, D. Moskovskiy, A. Mukherjee, A. Panchenko, M. Potthast, F. Rangel, N. Rizwan, P. Rosso, F. Schneider, A. Smirnova, E. Stamatatos, E. Stakovskii, B. Stein, M. Taulé, D. Ustalov, X. Wang, M. Wiegmann, S. M. Yimam, E. Zangerle, 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, G. M. D. Nunzio, L. Soulier, P. Galuscakova, A. G. S. Herrera, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. 15th International Conference of the CLEF Association (CLEF 2024), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2024.

[6] E. Tavan, M. Najafi, Marsan at pan: Binocularsllm, fusing binoculars' insight with the proficiency of large language models for machine-generated text detection, in: Conference and Labs of the Evaluation Forum, 2024. URL: https://api.semanticscholar.org/CorpusID:271801058.

[7] A. Hans, A. Schwarzschild, V. Cherepanova, H. Kazemi, A. Saha, M. Goldblum, J. Geiping, T. Goldstein, Spotting llms with binoculars: Zero-shot detection of machine-generated text, 2024. URL: https://arxiv.org/abs/2401.12070. arXiv:2401.12070.

[8] C. Anderson, B. Bjorkman, D. Denis, J. Doner, M. Grant, N. Sanders, A. Taniguchi, Essentials of Linguistics, 2nd Edition, Open textbook library, eCampusOntario, 2022. URL: https://ecampusontario.pressbooks.pub/essentialsoflinguistics2.

[9] O. Halvani, Constituent Treelib - A Lightweight Python Library for Constructing, Processing, and Visualizing Constituent Trees., 2024. URL: https://github.com/Halvani/constituent-treelib. doi:10.5281/zenodo.10951644.

[10] 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, Berlin Heidelberg New York, 2023, pp. 236–241.

[11] E. Stamatatos, W. Daelemans, B. Verhoeven, M. Potthast, B. Stein, P. Juola, M. Sanchez-Perez, A. Barrón-Cedeño, Overview of the Author Identification Task at PAN 2014, in: L. Cappellato, N. Ferro, M. Halvey, W. Kraaij (Eds.), Working Notes Papers of the CLEF 2014 Evaluation Labs, volume 1180 of *Lecture Notes in Computer Science*, 2014. URL: https://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-StamatosEt2014.pdf.

[12] K. H. Brodersen, C. S. Ong, K. E. Stephan, J. M. Buhmann, The balanced accuracy and its posterior distribution, in: 2010 20th International Conference on Pattern Recognition, 2010, pp. 3121–3124. doi:10.1109/ICPR.2010.764.

[13] O. Vinyals, L. u. Kaiser, T. Koo, S. Petrov, I. Sutskever, G. Hinton, Grammar as a foreign language, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 28, Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/277281aada22045c03945dcb2ca6f2ec-Paper.pdf.