Team ChatOnly at PAN 2026: Hybrid N-Gram and DeBERTa AI Detection

Notebook for the PAN Lab at CLEF 2026

Tomas Nagy[1], Lukas Skopar[1]

[1]Technical University of Kosice, Slovakia

Abstract

This notebook describes the Team ChatOnly submission to the PAN 2026 Voight-Kampff Generative AI Detection task. The task requires a standalone Dockerized system that receives newline-delimited JSON records containing text documents and returns calibrated confidence scores indicating whether each text was written by a machine. Our final system combines two complementary supervised detectors: a high-dimensional TF-IDF N-gram logistic regression model and a fine-tuned DeBERTa-v3-small sequence classifier. The N-gram model captures lexical and character-level regularities, while the transformer captures semantic and contextual patterns. At inference time both models are loaded from serialized artifacts inside the Docker image and their probabilities are combined by weighted soft voting. The system is designed to satisfy the PAN/TIRA submission constraints: no training on TIRA, no network access during evaluation, independent processing of each test case, and exactly one JSONL output file containing one probability score per input identifier. On the provided validation split, the final blend reaches a mean validation score of 0.9945 across the official PAN-style metrics.

Keywords

generative AI detection, authorship verification, TF-IDF, DeBERTa, PAN, TIRA

1. Introduction

The PAN 2026 lab comprises several shared tasks on generated-content analysis and related authorship problems [1]. Within this lab, the Voight-Kampff Generative AI Detection task asks participants to distinguish human-written texts from machine-authored texts in a binary authorship verification setting [2, 3]. The task is intentionally more difficult than ordinary generated-text detection because the generated texts may be obfuscated and may imitate the style of specific human authors. Furthermore, the test data may contain unseen models and previously unknown obfuscation strategies. A robust detector therefore needs to avoid relying on a single brittle signal.

The submitted Team ChatOnly system is a hybrid detector. It combines a high-capacity sparse lexical model with a compact transformer classifier. The sparse model is a logistic regression classifier trained on word and character TF-IDF N-grams. This component is fast, compact, and very strong on the supplied validation data. The transformer component is based on DeBERTa-v3-small [4], fine-tuned for binary sequence classification. It provides a semantically richer signal that is less tied to surface features. The two scores are combined by weighted averaging.

The implementation is packaged as a Docker-compatible submission for TIRA [5]. The repository contains the inference script, the serialized N-gram model, the fine-tuned DeBERTa checkpoint, and the Dockerfile.

2. Task and Data

The task input consists of newline-delimited JSON objects. The training and validation splits contain an identifier, the text, a model field, the binary label, and a genre field. A label of 0 denotes human-written text and a label of 1 denotes AI-written text. The official test split has the same JSONL structure but

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

tomas.nagy@student.tuke.sk (T. Nagy); lukas.skopar@student.tuke.sk (L. Skopar)

contains only the `id` and `text` fields [2]. Consequently, the submitted system must not depend on labels, genre metadata, or model identifiers at inference time.

The local development data used in this work contained 23,707 training instances and 3,589 validation instances. The validation set contained 1,277 human-written texts and 2,312 machine-written texts. The genre distribution in the validation split was 665 essays, 1,837 fiction texts, and 1,087 news texts. These figures were used only for development and reporting; the inference script ignores all fields except `id` and `text`.

PAN submissions are evaluated with ROC-AUC, the complement of the Brier score, C@1, F1, $F_{0.5u}$, their arithmetic mean, and a confusion matrix from which false-positive and false-negative rates are derived [2, 3]. Scores below 0.5 indicate human authorship, scores above 0.5 indicate machine authorship, and exactly 0.5 denotes an undecided answer. Our submitted system always emits continuous probabilities in the range [0, 1] and does not intentionally abstain.

3. System Overview

Figure 1 summarizes the system architecture. During development, the N-gram classifier and the DeBERTa classifier are trained on the provided training split and validated on the provided validation split. Before submission, both models are serialized. During TIRA evaluation, the Docker container loads the serialized artifacts and applies them to each input record. The output is a single JSONL file named `prediction.jsonl`.

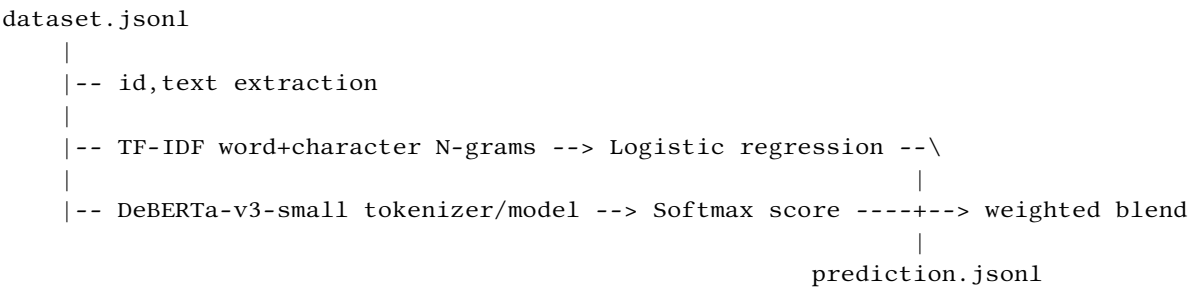

Figure 1: High-level inference pipeline of the Team ChatOnly system.

The final predictor uses weighted soft voting. If the serialized N-gram model is available, its probability receives weight 0.9 and the DeBERTa probability receives weight 0.1. This choice was selected because the N-gram model had the best local validation performance, while the transformer contributed a small additional contextual signal. If the N-gram artifact is not available, the script falls back to DeBERTa-only prediction. The submitted Docker image contains both artifacts.

4. N-Gram Logistic Regression

The first component is a sparse supervised classifier implemented with scikit-learn [6]. Its feature extractor is a `FeatureUnion` of two TF-IDF vectorizers:

- word N-grams of order 1–3 with 50,000 maximum features;
- character-boundary N-grams of order 3–5 with 50,000 maximum features.

Both vectorizers use sublinear term-frequency scaling. The concatenated sparse feature representation is passed to a logistic regression classifier with `C=10`, the `liblinear` solver, `max_iter=1000`, and a fixed random seed. The complete pipeline is serialized as `models/ngram_pipeline.joblib`. This makes inference independent of the training data and avoids retraining on TIRA.

This component is motivated by the observation that generated text often contains subtle lexical, punctuation, and character-pattern regularities even when high-level meaning appears fluent. Character

N-grams are useful for capturing punctuation habits, whitespace-adjacent patterns, suffixes, and micro-style effects, while word N-grams capture phrase-level preferences and topic-independent formulaic expressions.

5. DeBERTa Classifier

The second component is a fine-tuned transformer classifier based on DeBERTa-v3-small. DeBERTa improves upon BERT-style encoders through disentangled attention and enhanced decoding pre-training [4]. In our setup, the model is fine-tuned as a two-class sequence classifier using the Hugging Face Transformers library [7]. Texts are tokenized with truncation to 512 tokens. Training uses two epochs, a learning rate of 2×10^{-5}, weight decay 0.01, mini-batches of size 8, gradient accumulation over 2 steps, and mixed precision on GPU.

For submission, the selected checkpoint is stored in the final DeBERTa checkpoint directory. The Docker build copies only the inference-relevant files: the model configuration, the model weights, and the tokenizer vocabulary and metadata. Optimizer states, scheduler states, and other training-only files are excluded from the submission image.

6. Inference and Submission Packaging

The inference entry point is `predict.py`. It accepts two positional arguments: the path to an input JSONL file and the path to an output directory. The script reads all non-empty input lines, validates that each line contains `id` and `text`, and ignores any additional fields. This is important because validation data includes labels and metadata, whereas the test data does not.

The DeBERTa model is loaded from `/app/models/deberta` inside the container. The N-gram model is loaded from `/app/models/ngram_pipeline.joblib`. DeBERTa predictions are computed in batches, using CUDA when available and CPU otherwise. N-gram predictions are computed with the serialized scikit-learn pipeline. The final probability is clipped to the range [0, 1] before writing.

The Dockerfile is based on the official PAN generative-authorship baseline image and installs the small additional tokenizer dependencies required by the DeBERTa checkpoint. It then copies the inference script, validation script, serialized N-gram model, and DeBERTa checkpoint into `/app`. The container entry point is:

```
ENTRYPOINT ["python3", "/app/predict.py"]
```

The model artifacts are stored with the software package. Large binary model weights are kept separate from the source code representation using Git LFS, so that the repository remains usable while still containing the full inference checkpoint needed to reproduce the submitted container.

7. Validation Results

Table 1 reports local validation results. The official hidden test set was not available during development, so these numbers should be interpreted as development-set evidence rather than final PAN or ELOQUENT test scores. The official task overview paper compiles the final submitted-system results for this task [3].

The N-gram model is the strongest single component according to the validation mean. DeBERTa attains slightly higher ROC-AUC, suggesting that its ranking of examples is excellent, but it is less well calibrated around the 0.5 decision threshold on this split. The final blend improves the mean score only marginally, from 0.9940 for the N-gram model alone to 0.9945 for the 0.9/0.1 blend, and slightly reduces the false-positive rate. Thus, the transformer is best interpreted as a small auxiliary signal rather than a major independent improvement in this configuration. The reduction in false positives is still useful for the task because false accusations of human-authored texts are undesirable in practical AI-detection settings.

Table 1
Local validation results. Higher is better for all listed aggregate metrics. FPR and FNR are reported as error rates and are therefore lower-is-better.

Model	AUC	Brier	C@1	F1	$F_{0.5u}$	Mean	FPR	FNR
N-gram LR	0.9989	0.9923	0.9916	0.9935	0.9935	0.9940	0.0117	0.0065
DeBERTa	0.9994	0.9867	0.9850	0.9884	0.9839	0.9887	0.0352	0.0039
Final blend	0.9990	0.9928	0.9925	0.9942	0.9943	0.9945	0.0102	0.0061

8. Error Analysis and Robustness Considerations

The main strength of the system is the complementarity between sparse surface-form features and contextual transformer features. The N-gram model is highly effective when generated texts preserve recurring lexical and character-level traces. It is also very fast and stable at inference time. Its main weakness is that it may overfit to generation artifacts present in the training set and may be less robust to strong paraphrasing or deliberate style transfer.

The transformer model provides a more semantic signal and may help when surface forms are deliberately altered. However, the fine-tuned DeBERTa model is computationally heavier and can be sensitive to the distribution of fine-tuning data. The weighted blend is therefore intentionally conservative: the sparse classifier dominates the final score, while DeBERTa acts as a secondary signal.

The task rules require test cases to be processed independently. The submitted system satisfies this requirement because each text receives a score based only on its own content. Batching is used only for computational efficiency in the DeBERTa forward pass; no score depends on labels, metadata, neighboring examples, or aggregate statistics from the test file.

9. Reproducibility

The relevant repository files are:

- `train_custom_model.py`: trains and serializes the N-gram pipeline;
- `train_transformer.py`: fine-tunes DeBERTa-v3-small;
- `predict.py`: standalone inference script;
- `validate_submission.py`: format and ID validator;
- `Dockerfile`: Docker image definition.

The software can be executed locally with Docker:

```
docker build -t pan26-subtask1 .
docker run --rm \
  -v "$PWD/data/val.jsonl:/input/dataset.jsonl" \
  -v "$PWD/submission_out:/out" \
  pan26-subtask1 /input/dataset.jsonl /out
```

The produced output can be checked with:

```
docker run --rm \
  -v "$PWD/data/val.jsonl:/input/dataset.jsonl" \
  -v "$PWD/submission_out:/out" \
  --entrypoint python3 \
  pan26-subtask1 /app/validate_submission.py \
  /input/dataset.jsonl /out/prediction.jsonl
```

10. Conclusion

Team ChatOnly submitted a standalone hybrid detector for the PAN 2026 Voight-Kampff Generative AI Detection task. The system combines a compact and highly effective word/character N-gram logistic regression classifier with a fine-tuned DeBERTa-v3-small transformer. The final Docker image contains all required inference artifacts and requires no external network access during evaluation. On the provided validation split, the final soft-voting blend achieved a mean score of 0.9945 across the official PAN-style metrics.

Acknowledgments

We thank the PAN organizers for providing the task, data, baselines, and TIRA infrastructure. We also acknowledge the authors and maintainers of scikit-learn, Hugging Face Transformers, PyTorch, and the DeBERTa model family.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-5.5 in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

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, 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: 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, p. to appear.

[2] PAN, Voight-Kampff Generative AI Detection 2026, https://pan.webis.de/clef26/pan26-web/generated-content-analysis.html, 2026. Accessed: 2026-05-27.

[3] J. Bevendorff, R. R. Gunti, J. Karlgren, M. Fröbe, M. Potthast, B. Stein, Overview of the third "Voight-Kampff" Generative AI / LLM Detection Task at PAN and ELOQUENT 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, p. to appear.

[4] P. He, X. Liu, J. Gao, W. Chen, DeBERTa: Decoding-enhanced BERT with disentangled attention, International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=XPZIaotutsD.

[5] 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: 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. URL: https://link.springer.com/chapter/10.1007/978-3-031-28241-6_20. doi:10.1007/978-3-031-28241-6_20.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, Journal of Machine Learning Research 12 (2011) 2825–2830. URL: https://jmlr.org/papers/v12/pedregosa11a.html.

[7] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, 2020, pp. 38–45. doi:10.18653/v1/2020.emnlp-demos.6.