Unibuc - NLP at "Voight-Kampff" Generative AI Detection PAN 2025

Notebook for PAN at CLEF 2025

Teodor Marchitan[1,3], Claudiu Creanga[2,3] and Liviu P. Dinu[1,3]

[1]*Faculty of Mathematics and Computer Science, University of Bucharest, Romania*
[2]*Interdisciplinary School of Doctoral Studies, University of Bucharest, Romania*
[3]*HLT Research Center, Bucharest, Romania*

Abstract
This paper describes the approach of the Unibuc - NLP team in tackling the "Voight-Kampff" Generative AI Detection task from PAN at CLEF 2025 [1, 2]. In the Subtask 1: Voight-Kampff AI Detection Sensitivity we have explored different approaches of leveraging embeddings extracted from LLMs, using either classical ML models or further fine-tuning LLMs with a classification head for the downstream binary classification task. Our experiments showed that using a majority vote ensemble of classical ML models (Light GBM, XGB, Logistic Regression and SVM) is comparable with fine-tuning for downstream task achieving around 99.4% F1 score. For the Subtask 2: Human-AI Collaborative Text Classification we explored a way of combining features at different layers extracted using transformer backbone with layer-wise projection and attentive pooling which proved modest results.

Keywords
PAN 2025, Voight-Kampff Generative AI Detection 2025, LLM, Classical ML, NLP, Machine-Generated Text Detection

1. Introduction

As artificial intelligence continues to advance and generative AI technologies are more and more popular, distinguishing between machine-generated and human-authored texts has become very difficult. The "Voight-Kampff" Generative AI Detection task from PAN at CLEF 2025 [1, 2] aims to advance the research in the field and find ways to develop real systems capable of detecting not only AI generated texts but also machine-altered ones. The need for such systems is becoming a necessity day by day in order to maintain the intergrity and authenticity of the content from the internet. Such tools can help mitigate the risks associated with social manipulation, deepfakes, fake news, propaganda, misinformation and so on.

For the Subtask 1: Voight-Kampff AI Detection Sensitivity we experimented various methods of utilizing the embeddings from large language models (LLMs), including classical machine learning approaches and fine-tuning classification head for the binary classification task. The results indicate that a majority vote ensemble of classical models - specifically LightGBM, XGBoost, Logistic Regression and SVM - offers a slightly better performance and faster training times and resources compared to the fine-runing approach.

For the Subtask 2: Human-AI Collaborative Text Classification, we have tried a different approach by fusing together layers of a transformer backbone model with layer-wise projection and attentive pooling strategies.

2. Related work

The detection of AI-generated text has become a rapidly evolving research area, particularly with the advent of large language models. Recent shared tasks such as the PAN 2024 "Voight-Kampff"

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain
teodor.marchitan@s.unibuc.ro (T. Marchitan); claudiu.creanga@s.unibuc.ro (C. Creanga); ldinu@fmi.unibuc.ro (L. P. Dinu)

Generative AI Authorship Verification challenge have provided valuable benchmarks and insights into the capabilities and limitations of current detection systems [3, 4]. Early approaches to authorship verification, such as one-class classification [5] and compression-based models [6] have been extended to address the challenges posed by neural text generation [7, 8]. More recent work has focused on zero-shot detection methods, leveraging statistical properties of model outputs [9]. Despite these advances, studies have shown that both human and automated heuristics can be unreliable, especially as generative models improve in mimicking human style [10, 11]. Our work builds on these foundations, evaluating both classical and transformer-based approaches for robust detection of machine-generated and collaboratively authored texts.

3. Dataset

3.1. Subtask 1: Voight-Kampff AI Detection Sensitivity

The first subtask is formulated as a binary classification problem and the dataset contains either human-written texts (label 0) or ai-written texts (label 1). The genre of the text is also available and it can be one of the following: essays, news or fiction. Some of the news texts were sampled from the last year task dataset [12].

Overall the train split has $23,707$ samples and dev split has $3,589$. The distribution of the labels in each split is shown in Table 1. The predominant genre for both labels is fiction, followed by essays and news. As for the models used to generate text, gpt [13] comes first followed by mistral [14], gemini [15], llama [16] and deepseek [17] (more info in section 7 and Figure 2).

Table 1
Label distribution in train and dev splits for task 1 dataset.

	human-written (0)	ai-written (1)	Total
Train	9,101	14,606	23,707
Dev	1,277	2,312	3,589

3.2. Subtask 2: Human-AI Collaborative Text Classification

The second subtask is a multi-class classification problem where the goal is to classify texts into six distinct categories based on the nature of human and machine contributions:

- **Fully human-written:** The document is entirely authored by a human without any AI assistance.
- **Human-initiated, then machine-continued:** A human starts writing, and an AI model completes the text.
- **Human-written, then machine-polished:** The text is initially written by a human but later refined or edited by an AI model.
- **Machine-written, then machine-humanized (obfuscated):** An AI generates the text, which is later modified to obscure its machine origin.
- **Machine-written, then human-edited:** The content is generated by an AI but subsequently edited or refined by a human.
- **Deeply-mixed text:** The document contains interwoven sections written by both humans and AI, without a clear separation.

Overall the train split has $288,919$ samples and dev split has $72,661$. The distribution of the labels in each split is shown in Table 2.

In the train split, models used for text rewriting and generation are mostly from mistral [14], gpt [13], gemini [15] and llama [16] (more info in Figure 3). Most of the texts that are fully human-written, machine-written then machine-humanized or human-written then machine-polished come from the

LLM-DetectAIve dataset [18]. The next big source of the texts is TriBERT [19] where ChatGPT is mainly used to produce deeply-mixed texts and human-initiated then machine-continued.

On the other hand, in the dev split llama [16] is mainly used for text generation followed by gpt [13] and mistral [14], gemini [15] being less used in this split. While in this split a good amount of fully human-written, machine-written then machine-humanized or human-written then machine-polished texts come from the LLM-DetectAIve dataset [18], there is a huge amount of human-written and human-initiated then machine-continued texts that are sampled from m4gt-bench [20]. Also in dev set there are more samples from RoFT [21] than in the train set (more information in Figure 4).

Table 2
Label distribution in train and dev splits for task 2 dataset.

Label category	Train	Dev
Fully human-written (0)	14, 910	225
Human-initiated, then machine-continued (3)	1, 368	510
Human-written, then machine-polished (1)	10, 740	37, 170
Machine-written, then machine-humanized (obfuscated) (2)	75, 270	12, 330
Machine-written, then human-edited (5)	95, 398	12, 289
Deeply-mixed text (4)	91, 232	10, 137
Total	288, 919	72, 661

4. System overview

We explored 3 different ways of laveraging LLM embeddings for either binary or multi-class classification tasks. For subtask 1 we conducted experiments with systems described in sections 4.1 and 4.2. For the second subtask we did our experiments with systems from sections 4.2 and 4.3.

4.1. Frozen LLM embeddings with classical machine learning models

This approach is using the last layer embeddings extracted from Qwen3-0.6B [22] to train different classic machine learning classifiers:

- Light GBM [23]
- XGBoost [24]
- Logistic Regression [25]
- SVM [26]

In the final approach we used an ensemble of the above models combining predictions in a majority voting manner. So the final prediction is the class voted by most of the models in the ensemble. The embeddings model was kept frozen and used only to extract embeddings from the last layer.

4.2. LLM fine-tuning with classification head for downstream task

The next approach also laverages the rich representational capacity of large language models, which can capture both complex linguistic patterns and semantic relationships through their high-dimensional vector space representations, and used as input for the final classification head. Experiments were conducted with embeddings extracted using differnet LLM backbones: Qwen2.5-0.5B [1], Qwen3-0.6B [2], Mistral7B-v0.1 [3], Llama-3.1-8B [4] and fine-tuning them with a classification head on top of the last token.

[1]https://huggingface.co/Qwen/Qwen2.5-0.5B
[2]https://huggingface.co/Qwen/Qwen3-0.6B
[3]https://huggingface.co/mistralai/Mistral-7B-v0.1
[4]https://huggingface.co/meta-llama/Llama-3.1-8B

4.3. Combine transformer-based embeddings at different layers with layer-wise projection and attentive pooling

We experimented this approach only for subtask 2. It uses embeddings from transformer-based models, combines them using frame-wise projection, pooling strategies and a lightweight classification head.

4.3.1. Backbone and Feature Extractor

The model uses a pretrained transformer as a backbone to extract contextualized token embeddings. This part of the model is always kept frozen during the training process.

4.3.2. Layer Aggregation Block

The information across multiple hidden layers of the transformer are aggregated using a learnable, softmax-normalized weighted scheme. The weighted sum of hidden states is normalized via layer normalization (without element-wise affine tranformation), producing a single sequence of contextual embeddings for each input.

4.3.3. Frame-wise Processing Block

The aggregated embeddings are further processed by a configurable frame-wise module, which can be either a simple linear projection or a small neural network. This step reduces the dimensionality and introduces additional non-linearity, preparing the features for temporal pooling.

4.3.4. Time Pooling Strategies

To summarize the sequence of embeddings into a fixed-size representation, the model supports several time pooling strategies:

- **Statistical Pooling (SP):** Concatenates the mean and standard deviation across the sequence.
- **Attentive Statistical Pooling (ASP):** Computes attention-weighted statistics using a learnable attention mechanism.

5. Experimental setup

For both subtasks, we kept 20% of the training set for validation and hyperparameter tuning (taking into consedireation the label distribution as well), and the dev split was used to test the final models. For the dataset splits and implementation for classical machine learning models (4.1) we used scikit-learn [27] [5], xgboost [6] and lightgbm [7] whereas for LLM fine-tuning (4.2) and transformer-based embeddings combined (4.3) we used PyTorch [28] [8].

In order to find the best parameters for each of the classical models from 4.1 we did hyperparameter tuning using optuna [29] [9] and the final configuration is reported in Table 6.

For the second approach 4.2 we loaded the backbone LLMs using QLoRA [30] (full parameters in Table 7) in order to be able to fine-tune the entire model (including the LLM backbone) for the downstream task [10]. All hyperparameters used for training in Table 8.

The backbone transformer model used for the last approach 4.3 is RoBERTa [31] [11]. The final model we used a hidden dimension of 128 for the frame-wise processing block and an attentive statistical

[5]https://scikit-learn.org/stable/
[6]https://xgboost.readthedocs.io/en/release_3.0.0/
[7]https://lightgbm.readthedocs.io/en/stable/
[8]https://pytorch.org
[9]https://optuna.org
[10]All experiments were conducted using 1 Nvidia RTX 4090 GPU
[11]https://huggingface.co/FacebookAI/roberta-base

pooling with 4 heads and 256 hidden dimension with ReLU activation. All other training parameters are available in Table 9.

The evaluation platform used for this challenge was TIRA [32].

6. Results

6.1. Subtask 1: Voight-Kampff AI Detection Sensitivity

The metrics used to evaluate the system are the following:

- **ROC-AUC:** The area under the ROC (Receiver Operating Characteristic) curve.
- **Brier:** The complement of the Brier score (mean squared loss).
- **C@1:** A modified accuracy score that assigns non-answers (score = 0.5) the average accuracy of the remaining cases.
- **F1:** The harmonic mean of precision and recall.
- **F0.5u:** A modified F0.5 measure (precision-weighted F measure) that treats non-answers (score = 0.5) as false negatives.
- **Mean:** The arithmetic mean of all the metrics above.

The overall results of the first approach 4.1 are available in Table 3 and those using the second approach 4.2 are available in Table 4.

Looking at Figure 6 it seems like the biggest problem for the model is that is classifies fiction texts generated with gemini-1.5-pro and llama-3.1-8b-instruct as human-written fiction texts. In the embeddings visualization plot Figure 5 we can see that there are some red squares (ai-written fiction) that are really close to the blue squares (human-written fiction).

Table 3
Reported metrics for the classical ML models on the dev set.

Model	ROC-AUC	Brier	C@1	F1	F0.5u	Mean
XGBoost	0.999	0.991	0.989	0.991	0.992	0.992
Light GBM	0.999	0.992	0.989	0.991	0.992	0.993
Logistic Regression	0.999	0.994	0.994	0.995	0.996	0.996
SVM	0.999	0.994	0.992	0.994	0.996	0.995
Ensemble	0.999	0.994	0.991	0.993	0.995	0.995

Table 4
Reported metrics for the fine-tuning LLM approach on the dev set.

Model	F1	Accuracy	Precision	Recall
Qwen2.5-0.5B	0.978	0.972	0.980	0.977
Qwen3-0.6B	0.993	0.991	0.993	0.993
Llama-3.1-8B	0.997	0.996	0.999	0.996
Mistral-7B-v0.1	0.995	0.993	0.997	0.992

6.2. Subtask 2: Human-AI Collaborative Text Classification

The metrics used to evaluate the system are the following:

- **Recall (Macro):** Macro recall is the average recall computed independently for each class, measuring the proportion of true positives identified out of all actual instances for each class, and then averaging across all classes.

- **F1 (Macro):** Macro F1 is the average of the F1 scores calculated for each class, where the F1 score is the harmonic mean of precision and recall, providing a balanced measure of a model's accuracy across all classes.
- **Accuracy:** Accuracy is the proportion of all predictions that are correct, calculated as the number of correct predictions divided by the total number of predictions.

The overall results of the third approach 4.3 are available in Table 5.

Table 5
Reported metrics for the combined transformer-based embeddings using layer-wise projection and attentive pooling on dev and test sets. *Based on the error analysis Figure 7 we reported the metrics on dev set removing the samples from m4gt-bench which proved to be the harderst to predict.

Model	Recall (Macro)	F1 (Macro)	Accuracy	Split
RoBERTa-base	0.750	0.592	0.541	Dev
RoBERTa-base	0.443	0.427	0.514	Test
RoBERTa-base	0.890	0.842	0.917	Dev (no samples from m4gt-bench)*

Looking at Figure 7 we can see that the hardest dataset is m4gt-bench (which was not present in the training dataset at all) with $28,192$ samples that were not classified as human-initiated, then machine-continued (97% of the samples from m4gt-bench with this label) and $1,750$ were not classifier as fully human-written (55% of the samples from m4gt-bench with this label). Furthermore looking at Figure 8 and Figure 9 we see that llama2 is the hardest model to be detected (also not present in the training split at all) with $11,793$ samples that were human-written, then machine-continued misclassified as human-written, then machine-polished. We have also reported the metrics we got on the dev set when removing the samples from m4gt-bench Table 5 to prove that the architecture we used for this type of classification fails to generalize well on unseen data, but manages to learn good enough representations from the training set.

7. Conclusions and Future Work

The experimental results demonstrate that both classical machine learning models and fine-tuned large language models (LLMs) achieve exceptionally high performance on the subtask 1 of distinguishing between human-written and ai-written texts, with F1, ROC-AUC, and related metrics consistently above 0.99 for most models. However, detailed error analysis reveals that the main challenge lies in distinguishing between human-written and AI-generated fiction, especially for texts produced by advanced models such as gemini-1.5-pro and llama-3.1-8b-instruct, which were also fewer in the train set, where the model tends to misclassify AI-generated fiction as human-written. This is further supported by the embeddings visualization, which shows significant overlap between these classes.

For the human-AI collaborative text classification subtask, the results are more modest, with macro F1 and recall scores around 0.43 and accuracy at 0.51, suggesting that this remains a challenging problem and that current models struggle to generalize in this setting, the difference between human-written then machine-continued and human-written then machine-polished being very subtle and hard to be detected, especially when the generative model has never been seen in the training dataset.

Future work will focus on addressing these limitations by exploring more sophisticated representation learning techniques, incorporating additional context or metadata, and experimenting with advanced ensemble methods. Furthermore, targeted data augmentation and adversarial training could help the models better distinguish subtle differences between human and machine-generated texts, particularly in challenging genres such as fiction. Finally, interpretability analyses and error-driven model refinement will be essential to further improve robustness and reliability in real-world applications.

Acknowledgements

This research is supported by the project “Romanian Hub for Artificial Intelligence - HRIA”, Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 334906.

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] 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.

[4] J. Bevendorff, X. B. Casals, B. Chulvi, D. Dementieva, A. Elnagar, D. Freitag, M. Fröbe, D. Korenčić, M. Mayerl, A. Mukherjee, A. Panchenko, M. Potthast, F. Rangel, P. Rosso, A. Smirnova, E. Stamatatos, B. Stein, M. Taulé, D. Ustalov, M. Wiegmann, E. Zangerle, Overview of PAN 2024: Multi-Author Writing Style Analysis, Multilingual Text Detoxification, Oppositional Thinking Analysis, and Generative AI Authorship Verification, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2024), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2024.

[5] M. Koppel, J. Schler, Authorship verification as a one-class classification problem, in: Proceedings of the Twenty-First International Conference on Machine Learning, ICML ’04, Association for Computing Machinery, New York, NY, USA, 2004, p. 62. URL: https://doi.org/10.1145/1015330.1015448. doi:10.1145/1015330.1015448.

[6] O. Halvani, C. Winter, L. Graner, On the usefulness of compression models for authorship verification, in: Proceedings of the 12th International Conference on Availability, Reliability and Security, ARES ’17, Association for Computing Machinery, New York, NY, USA, 2017. URL: https://doi.org/10.1145/3098954.3104050. doi:10.1145/3098954.3104050.

[7] A. Uchendu, T. Le, K. Shu, D. Lee, Authorship attribution for neural text generation, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 8384–8395. URL: https://aclanthology.org/2020.emnlp-main.673/. doi:10.18653/v1/2020.emnlp-main.673.

[8] A. Uchendu, Z. Ma, T. Le, R. Zhang, D. Lee, TURINGBENCH: A benchmark environment for Turing test in the age of neural text generation, in: M.-F. Moens, X. Huang, L. Specia, S. W.-t. Yih (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021, Association for

Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 2001–2016. URL: https://aclanthology.org/2021.findings-emnlp.172/. doi:10.18653/v1/2021.findings-emnlp.172.

[9] 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.

[10] M. Jakesch, A. Bhat, D. Buschek, L. Zalmanson, M. Naaman, Co-writing with opinionated language models affects users' views, in: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23, ACM, 2023, p. 1–15. URL: http://dx.doi.org/10.1145/3544548.3581196. doi:10.1145/3544548.3581196.

[11] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, S. Feizi, Can ai-generated text be reliably detected?, 2025. URL: https://arxiv.org/abs/2303.11156. arXiv:2303.11156.

[12] J. Bevendorff, M. Wiegmann, J. Karlgren, L. Dürlich, E. Gogoulou, A. Talman, E. Stamatatos, M. Potthast, B. Stein, Overview of the 'voight-kampff' generative AI authorship verification task at PAN and ELOQUENT 2024, in: CEUR Workshop Proceedings, volume 3740, 2024, p. Paper 225. URL: https://ceur-ws.org/Vol-3740/paper-225.pdf, accessed: 2025-05-30.

[13] OpenAI, Gpt-4 technical report, 2024. URL: https://arxiv.org/abs/2303.08774. arXiv:2303.08774.

[14] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7b, 2023. URL: https://arxiv.org/abs/2310.06825. arXiv:2310.06825.

[15] G. Team, Gemini: A family of highly capable multimodal models, 2025. URL: https://arxiv.org/abs/2312.11805. arXiv:2312.11805.

[16] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, 2023. URL: https://arxiv.org/abs/2302.13971. arXiv:2302.13971.

[17] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL: https://arxiv.org/abs/2501.12948. arXiv:2501.12948.

[18] M. Abassy, K. Elozeiri, A. Aziz, M. N. Ta, R. V. Tomar, B. Adhikari, S. E. D. Ahmed, Y. Wang, O. Mohammed Afzal, Z. Xie, J. Mansurov, E. Artemova, V. Mikhailov, R. Xing, J. Geng, H. Iqbal, Z. M. Mujahid, T. Mahmoud, A. Tsvigun, A. F. Aji, A. Shelmanov, N. Habash, I. Gurevych, P. Nakov, LLM-DetectAIve: a tool for fine-grained machine-generated text detection, in: D. I. Hernandez Farias, T. Hope, M. Li (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 336–343. URL: https://aclanthology.org/2024.emnlp-demo.35/. doi:10.18653/v1/2024.emnlp-demo.35.

[19] Z. Zeng, L. Sha, Y. Li, K. Yang, D. Gašević, G. Chen, Towards automatic boundary detection for human-ai collaborative hybrid essay in education, 2023. URL: https://arxiv.org/abs/2307.12267. arXiv:2307.12267.

[20] Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, O. M. Afzal, T. Mahmoud, G. Puccetti, T. Arnold, A. F. Aji, N. Habash, I. Gurevych, P. Nakov, M4gt-bench: Evaluation benchmark for black-box machine-generated text detection, 2024. URL: https://arxiv.org/abs/2402.11175. arXiv:2402.11175.

[21] L. Kushnareva, T. Gaintseva, G. Magai, S. Barannikov, D. Abulkhanov, K. Kuznetsov, E. Tulchinskii, I. Piontkovskaya, S. Nikolenko, Ai-generated text boundary detection with roft, 2024. URL: https://arxiv.org/abs/2311.08349. arXiv:2311.08349.

[22] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, Z. Qiu, Qwen3 technical report, 2025. URL: https://arxiv.org/abs/2505.09388. arXiv:2505.09388.

[23] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient

gradient boosting decision tree, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[24] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, 2016, p. 785–794. URL: http://dx.doi.org/10.1145/2939672.2939785. doi:10.1145/2939672.2939785.

[25] J. B. and, Application of the logistic function to bio-assay, Journal of the American Statistical Association 39 (1944) 357–365. URL: https://doi.org/10.1080/01621459.1944.10500699. doi:10.1080/01621459.1944.10500699. arXiv:https://doi.org/10.1080/01621459.1944.10500699.

[26] M. Hearst, S. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intelligent Systems and their Applications 13 (1998) 18–28. doi:10.1109/5254.708428.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, Édouard Duchesnay, Scikit-learn: Machine learning in python, 2018. URL: https://arxiv.org/abs/1201.0490. arXiv:1201.0490.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, 2019. URL: https://arxiv.org/abs/1912.01703. arXiv:1912.01703.

[29] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, 2019. URL: https://arxiv.org/abs/1907.10902. arXiv:1907.10902.

[30] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, 2023. URL: https://arxiv.org/abs/2305.14314. arXiv:2305.14314.

[31] 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. URL: https://arxiv.org/abs/1907.11692. arXiv:1907.11692.

[32] 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.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to: Generate literature review, Grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

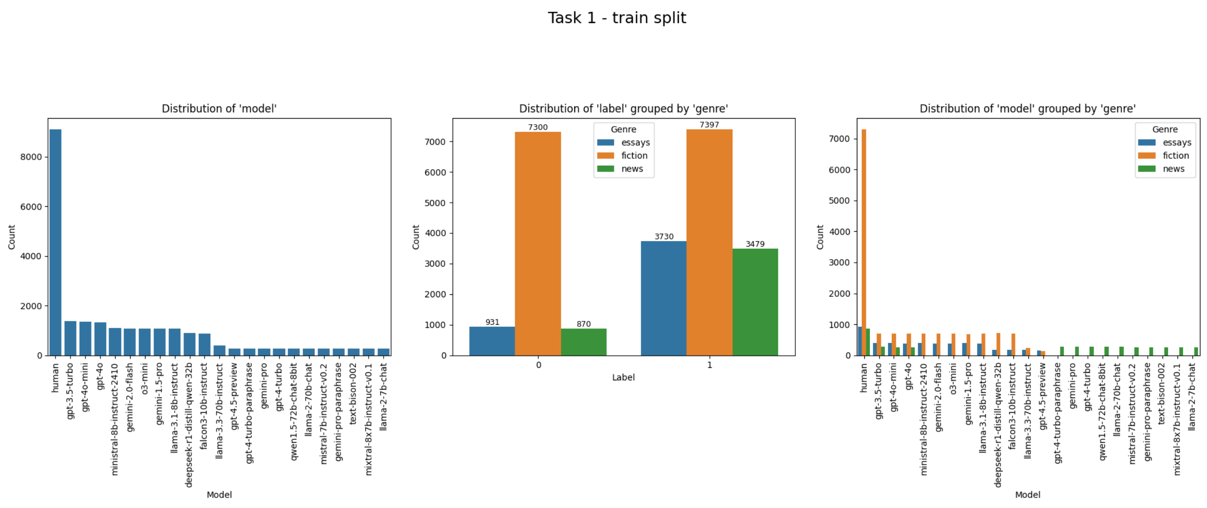

Figure 1: Task 1 dataset overview - train split

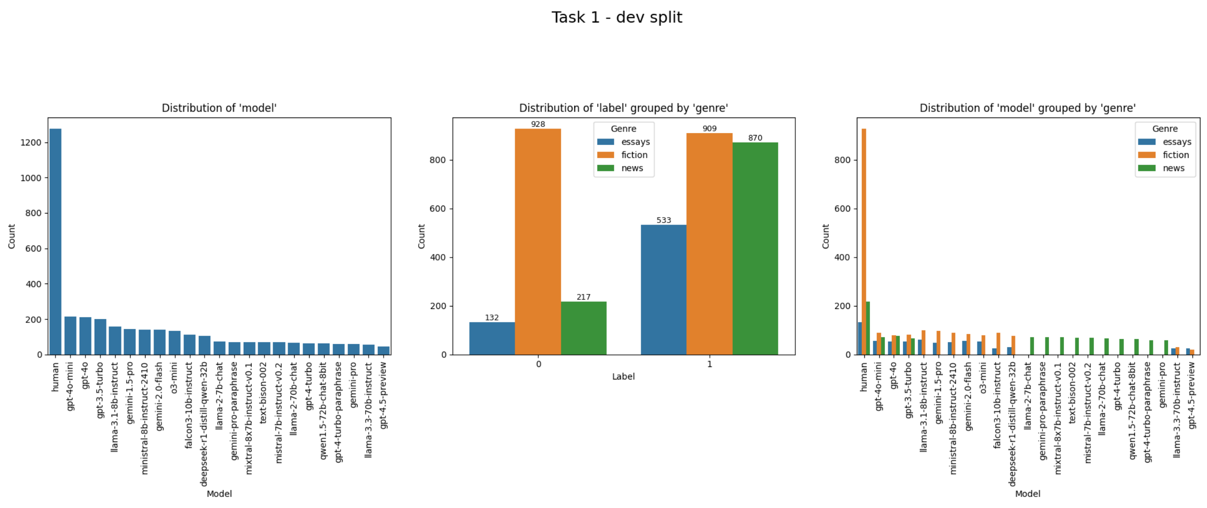

Figure 2: Task 1 dataset overview - dev split

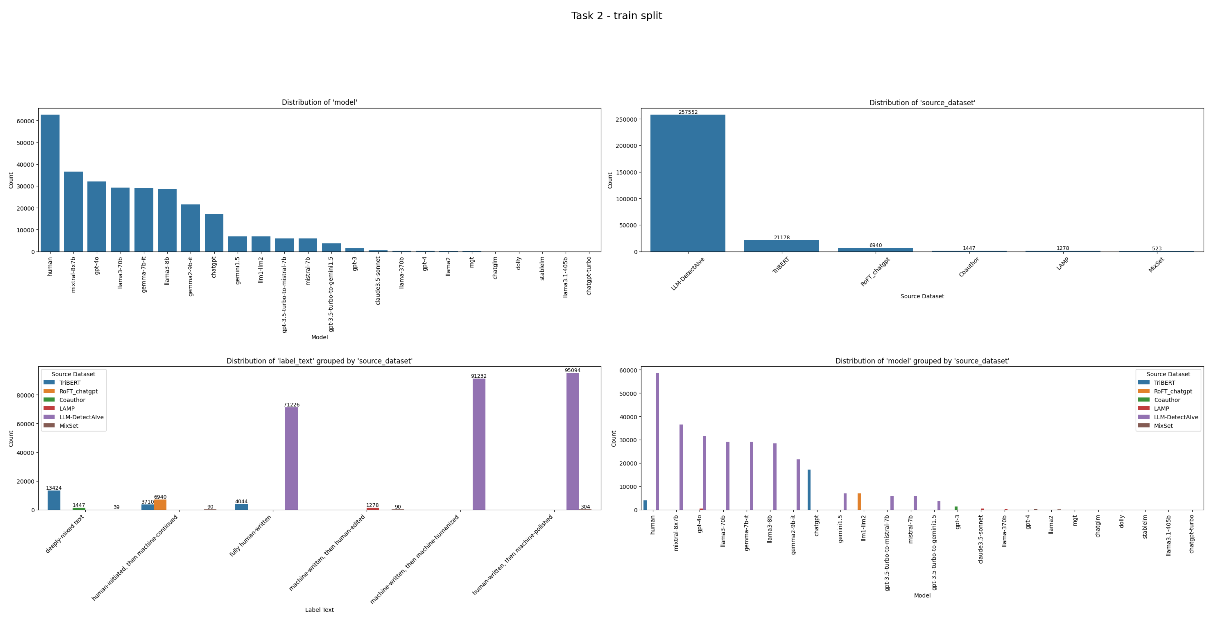

Figure 3: Task 2 dataset overview - train split

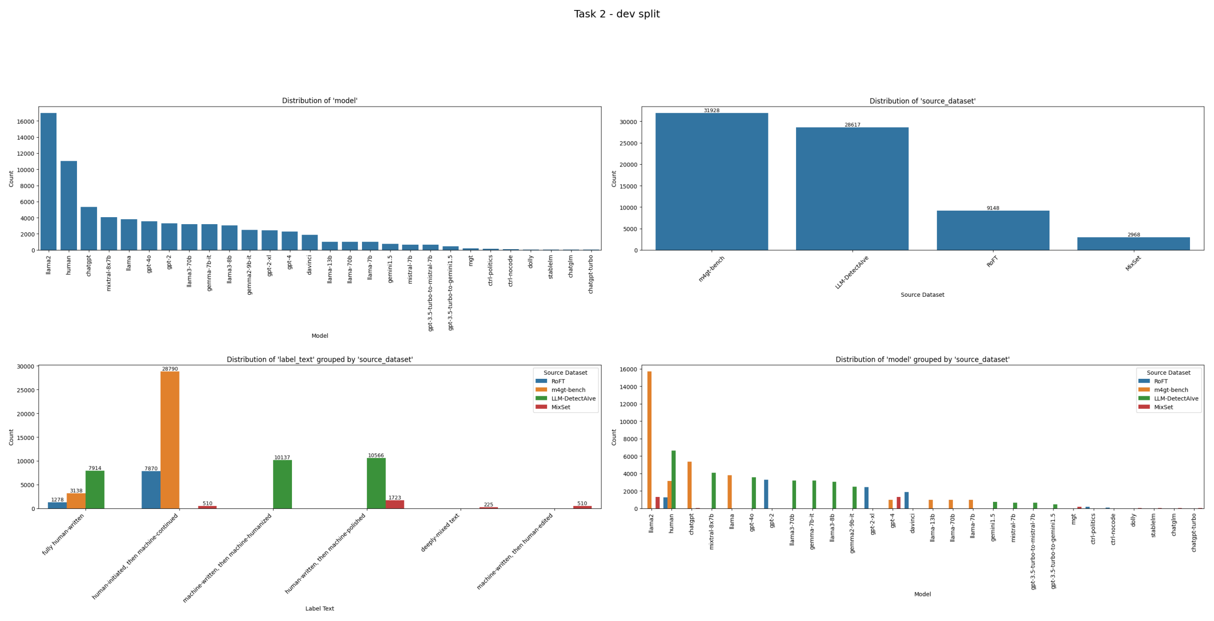

Figure 4: Task 2 dataset overview - dev split

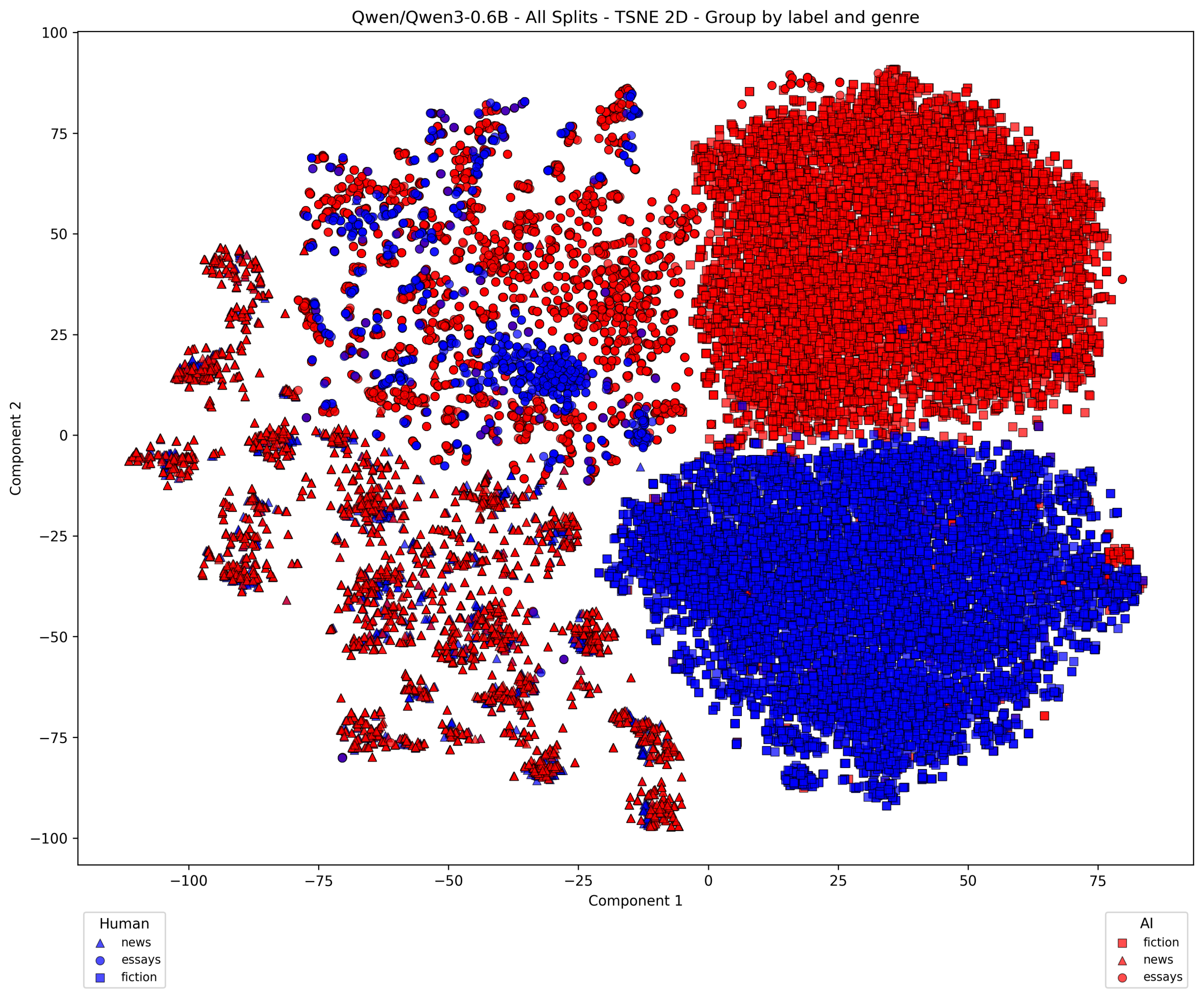

Figure 5: Qwen3-0.6B embeddings visualized in 2D using T-SNE (tarin + dev)

Table 6
Best hyperparameters for classical models used in the final ensemble model obtained using optuna.

Hyperparameter	XGBoost	LightGBM	Logistic Regression	SVM
n_estimators	260	218	–	–
learning_rate	0.1114	0.2004	–	–
max_depth	8	9	–	–
num_leaves	–	37	–	–
subsample	0.8541	0.9007	–	–
colsample_bytree	0.8348	0.9641	–	–
min_child_weight	4	–	–	–
min_child_samples	–	69	–	–
reg_alpha	–	0.4093	–	–
reg_lambda	–	0.6594	–	–
gamma	0.0913	–	–	–
C	–	–	0.1156	2.5501
penalty	–	–	l1	l1
solver	–	–	saga	–
max_iter	–	–	524	1229
random_state	–	–	42	–
loss	–	–	–	squared_hinge
tol	–	–	–	0.00015

Table 7
QLoRA-related hyperparameters used for all fine-tuned LLMs.

Parameter	Value
load_in_4bit	True
load_in_8bit	False
use_double_quant	True
quant_type	nf4
compute_dtype	bfloat16
lora_r	8
lora_alpha	32
lora_dropout	0.05
lora_bias	none
lora_target_modules	{`q_proj`, `k_proj`, `v_proj`, `o_proj`}

Table 8
Fine-tuning hyperparameters used for all LLMs.

Parameter	Value
batch_size	16
num_epochs	0.5
learning_rate	2e-5
lr_scheduler_type	constant
max_grad_norm	0.3
max_length	1024
weight_decay	0.01
warmup_ratio	0.03
optim	paged_adamw_32bit
gradient_accumulation_steps	4
eval_strategy	steps
save_strategy	steps
seed	42

Table 9
Training hyperparameters for RoBERTa-base fine-tuning.

Parameter	Value
model_name	FacebookAI/roberta-base
batch_size	4
num_epochs	5
learning_rate	0.0003
max_length	512
gradient_accumulation_steps	4
seed	42

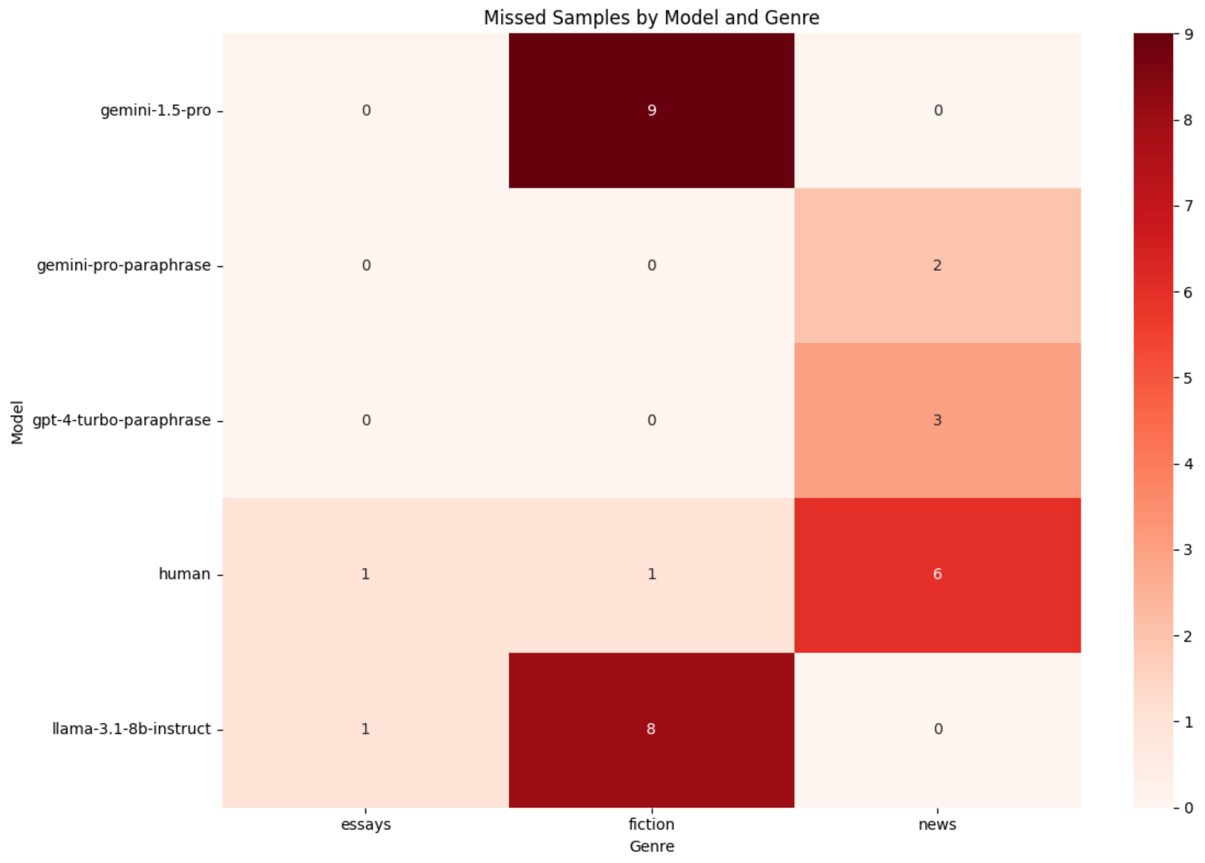

Figure 6: Missed samples on the dev set by the ensemble model for subtask 1.

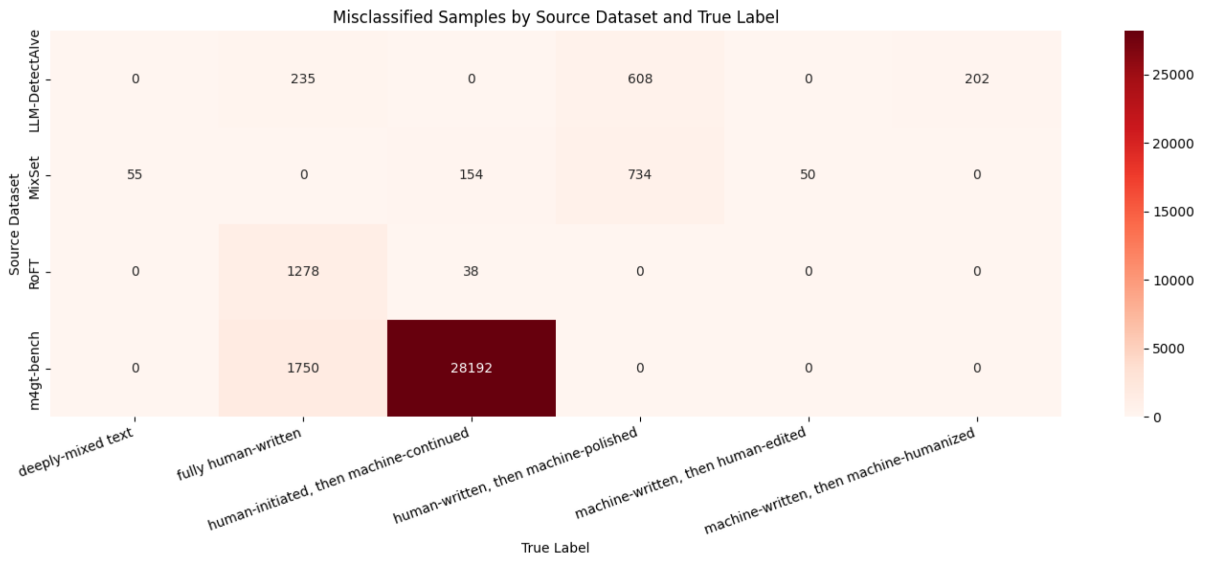

Figure 7: Heatmap of misclassified samples for each source dataset on dev set subtask 2.

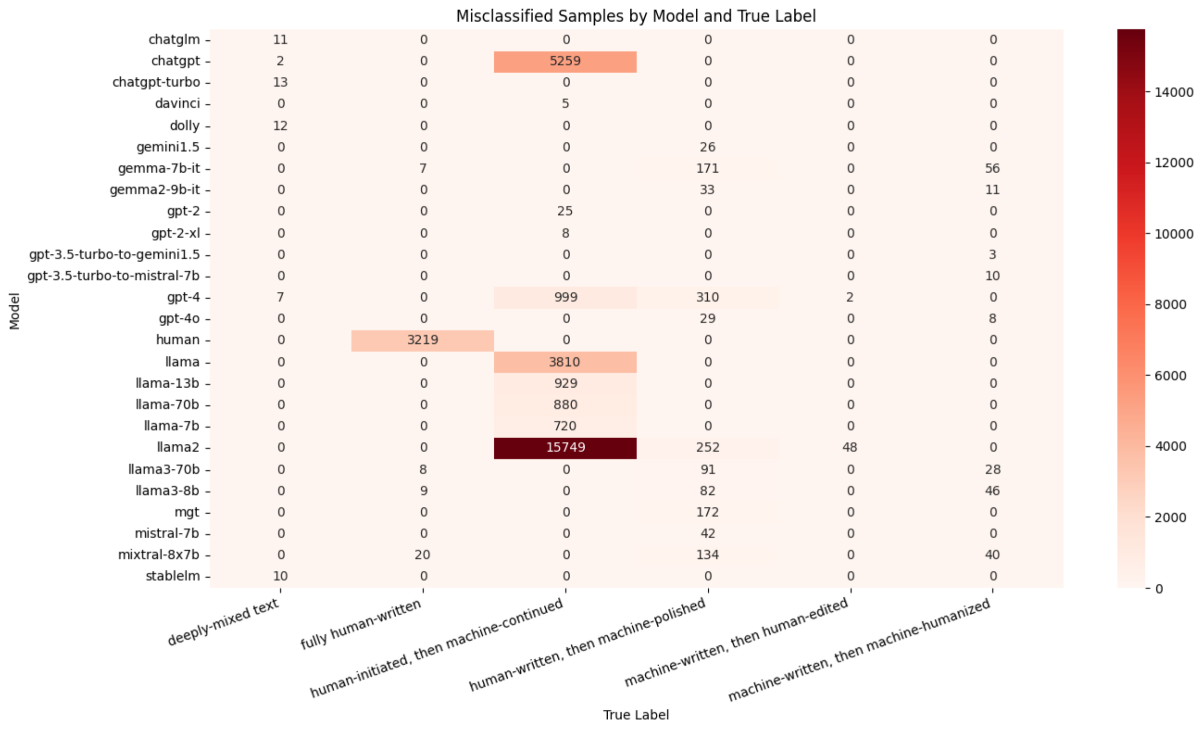

Figure 8: Heatmap of misclassified samples for each model on dev set subtask 2.

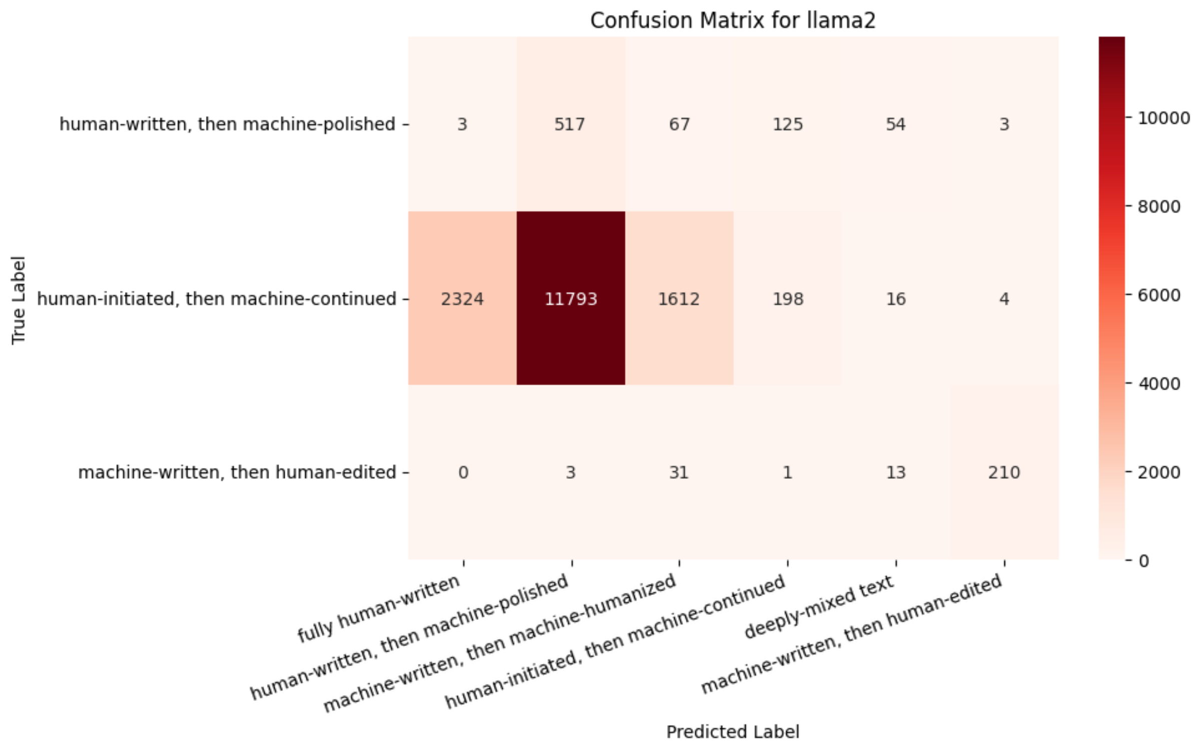

Figure 9: Confusion matrix of samples generated using llama2 model on dev set subtask 2.