Team cornell-1 at PAN: Ensembling Fine-Tuned Transformer Models for Writing Style Analysis

Notebook for the PAN Lab at CLEF 2025

Deniz Bölöni-Turgut, Dhriti Verma and Claire Cardie

Cornell University, Ithaca, NY 14853 USA

Abstract

This paper describes our system for the Multi-Author Writing Style Analysis shared task for the PAN Lab at CLEF 2025. We design and train an ensemble model from multiple fine-tuned transformer models. Each model in the ensemble follows our custom BertStyleNN architecture, a PyTorch neural network consisting of a fine-tuned encoder model and a feed-forward neural network classification head. We train each BertStyleNN model end-to-end on a combined difficulty (easy, medium, and hard) training dataset, using five different pre-trained feature extractors. We then conduct an exhaustive search over three ensembling methods and model combinations for each difficulty level. Our final system achieves a macro F1 of 0.8 averaged over the three difficulty levels, significantly outperforming the baseline.

Keywords

PAN 2025, multi-author style analysis, sentence embeddings, ensemble models, transformers

1. Introduction

The PAN Lab at CLEF hosts stylometry and text forensic shared tasks, such as the Multi-Author Writing Style Analysis task which has been run in various forms since 2016 [1]. This task has applications in plagiarism detection and authorship attribution for historical or anonymous works. In 2025, the style analysis task is to classify pairs of consecutive sentences as exhibiting a style change (label 1) or not (label 0) [2]. Software systems are submitted to TIRA.io [3] and evaluated on the hidden test set on the basis of macro F1 scores.

The data is split into three difficulty levels: easy, medium, and hard. The easy and medium difficulty levels consist of data with semantic differences in addition to stylistic ones. In contrast, sentences in the hard difficulty dataset cover the same topic.

This paper describes our ensembled transformer model system for sentence-level style analysis. In Section 2, we present related work to this task. In Section 3, we explore the class imbalance in our data and experiment with data augmentation. Section 4 and Appendix A describe the design and training for our custom neural network model BertStyleNN and the final system Ensembled-BertStyleNN. We use different ensemble models for each difficulty level due to the aforementioned differences in the data. Finally, we present the results of our system on the test set in Section 5 and conclude in Section 6.

2. Related Work

Early techniques for style analysis employed manual feature engineering of lexical or syntactic features [4]. More recent work uses embeddings from pre-trained language models. Since many sentence embedding models are trained with semantic similarity objectives, fine-tuning the pre-trained models on data labeled for style change is common and often necessary.

The goal of the PAN 2024 Multi-Author Style Analysis task was to identify style changes between paragraphs as opposed to between sentences as it is in 2025 [5]. Of the top two submissions to the 2024

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

db823@cornell.edu (D. Bölöni-Turgut); dv278@cornell.edu (D. Verma); ctc9@cornell.edu (C. Cardie)

0009-0004-4881-0472 (D. Bölöni-Turgut); 0009-0001-1933-2350 (D. Verma)

version of this task, one fine-tuned the open-source large language model Llama-3-8b [6] with low-rank adaption [7], and the other ensembled three pre-trained transformer models with additional semantic similarity checks applied for the easy and medium difficulty levels [8].

Document-level authorship attribution approaches include using static embeddings as input to Siamese networks trained with contrastive loss to perform classification [4].

3. Dataset Exploration

The most notable observation from our data exploration is the class imbalance. Only 19.9% and 20.4% of sentence pairs in the combined difficulty training and validation sets respectively are instances of a style change. To investigate the significance of this class imbalance, we constructed a 50/50 class balanced training set. This balanced training set was augmented with problems randomly chosen from the PAN 2024 Multi-Author Style Analysis task dataset [9]. We use both the balanced and original imbalanced training sets to extract sentence embeddings from the pre-trained all-MiniLM-L12-v2 model and train a feed-forward neural network (FFNN) as a binary classifier. We do not fine-tune the embedding model at all, only the FFNN. The validation set metrics for both training runs are shown in Table 1; we evaluate each run on the original imbalanced validation set.

Table 1
Validation Performance on Balanced & Original Training Data. All metrics are macro.

Training Data Type	F1	Precision	Recall
Original (imbalanced)	0.762	0.766	0.758
Balanced (50/50)	0.729	0.679	0.463

The Macro F1 score from the model with a balanced train set is 0.033 less than the model with the original train set. For our final system, we assume that the test set also exhibits the class imbalance present in the training and validation data. Therefore, we conduct all further training with the original, imbalanced training data.

4. The BertStyleNN

We introduce the BertStyleNN, our custom neural network based model which contains a binary sequence classification head and is implemented with PyTorch. The code and links to download our trained models from HuggingFace can be found at https://github.com/denizbt/pan-styleAnalysis25.

In this section, we describe the architecture and training process for BertStyleNN models.

4.1. Model Architecture

A BertStyleNN has two parts: a transformer encoder for feature extraction and a FFNN for binary classification. BertStyleNN supports a variety of pre-trained SentenceTransformers models and general feature extractors as its encoder. No architectural changes are made to any pre-trained encoder; it is only fine-tuned.

The architecture of the FFNN is relatively straightforward and is the same for every encoder model. It consists of 4 hidden layers with `ReLU` activation functions, a 1D `BatchNorm` layer, and a `Dropout` layer with $p = 0.4$. The details of the architecture were determined from experimentation with the all-MiniLM-L12-v2 sentence embedding model; each sentence pair in the PAN dataset (all difficulties combined) was embedded using the all-MiniLM-L12-v2 model out-of-the-box and then used to train the FFNN. The architecture that resulted in the highest validation macro F1 was chosen.

The forward pass of a BertStyleNN proceeds as follows. The pair of sentences to check for style changes are passed in as input. Then, BertStyleNN extracts embeddings independently for each

sentence using its encoder, concatenates the embeddings, and finally applies the FFNN to get one-dimensional output for the binary classification. The complete architecture for BertStyleNN is shown in Figure 1.

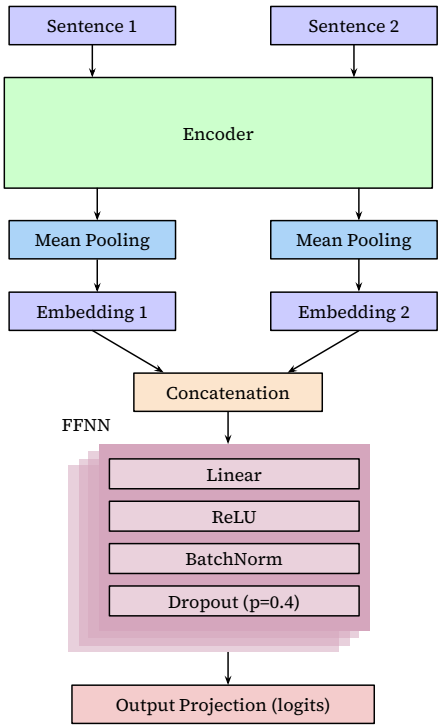

Figure 1: Architecture diagram of BertStyleNN (shown with a generic encoder).

4.2. Training

Training a BertStyleNN involves simultaneously fine-tuning a pre-trained encoder model and training a FFNN for classification (i.e. end-to-end training).

We select and fine-tune five different pre-trained encoder/sentence embedding models as the encoders for the BertStyleNN, listed below. All models are downloaded from HuggingFace.

- roberta-base [10] improves upon BERT [11] by training it on more data, using dynamic masking, and removing the next sentence prediction task. It was chosen due to its popularity and high performance as a general feature extractor.
- microsoft/deberta-base [12] achieves higher performance compared to BERT and RoBERTa by using disentangled attention which uses two separate vectors for position and content and improving the decoding for the masked LM task. This model was also chosen for its popularity and high performance on natural language understanding tasks.
- sentence-transformers/all-MiniLM-L12-v2 [13] was finetuned with a contrastive similarity objective from the pre-trained microsoft/MiniLM-L12-H384-uncased model [14]. As of the writing of this paper, it is the fourth highest performing model for sentence embeddings in the SentenceTransformers library [15].
- sentence-transformers/all-mpnet-base-v2 [16] was finetuned from using self-supervised contrastive learning objective from microsoft/mpnet-base model [17]. It is currently the highest performing model for sentence embeddings in the SentenceTransformers library [15].
- sentence-transformers/sentence-t5-base [18] is a PyTorch version for the encoder of a T5 base model [19]. It was chosen to add to the diversity of our set of models.

Our training and validation sets are a combination of the data from all three difficulty levels. We make no other alterations or augmentations to the data. We choose to use a combined training set since each difficulty level subset is too small on its own.

We holistically select different hyperparameters and learning schedules for every encoder model (see Appendix A for the choices). We also conduct a linear search for the best probability prediction threshold to apply to the output and choose the best epoch for each model based on the macro F1. It is important to note that while the training hyperparameters differ, the architecture of the FFNN (including hidden layer dimensions) remains the same for all styles of encoder models. Table 2 displays the validation performance for each fine-tuned model.

Table 2
Individual Model Validation Metrics (trained on combined difficulty dataset).
All metrics are macro and are reported for the best probability prediction threshold for the model.

Encoder	F1	Precision	Recall
roberta-base	0.791	0.797	**0.789**
microsoft/deberta-base	**0.794**	**0.803**	0.785
sentence-transformers/all-MiniLM-L12-v2	0.785	**0.803**	0.771
sentence-transformers/all-mpnet-base-v2	0.756	0.787	0.736
sentence-transformers/sentence-t5-base	0.768	0.796	0.748

4.3. Ensembling

At this point, we have trained several BertStyleNN models on the combined difficulty dataset. We now turn our attention to finding the best ensemble model for each difficulty level.

We experiment with three ensembling methods: majority voting, unweighted average of output probabilities, and unweighted average of output logits. For each difficulty level, we test all three methods on the validation set for every subset of trained models size three or more. We report the metrics for the highest performing subset and method for each difficulty level in Table 3. Figure 2 illustrates our complete system pipeline, including ensembling: the Ensemble-BertStyleNN.

Table 3
Best Ensemble Model Validation Metrics for each difficulty level.

Ensemble	Macro F1	Threshold	Method
Easy: (deberta-base, all-MiniLM-L12-v2, sentence-t5-base)	0.909	0.67	avg-logits
Medium: (deberta-base, roberta-base, all-mpnet-base-v2)	0.801	0.58	avg-probs
Hard: (deberta-base, roberta-base, all-MiniLM-L12-v2, all-mpnet-base-v2)	0.702	0.57	avg-probs

5. Results

For the final system submission, we use the ensemble models along with the prediction thresholds that performed best on the validation set; details for the ensemble used for each difficulty level are given in Table 3. The results of our Ensembled-BertStyleNN approach on the hidden test set are in Table 4. Our system significantly outperforms the naive baseline of predicting the majority class (0).

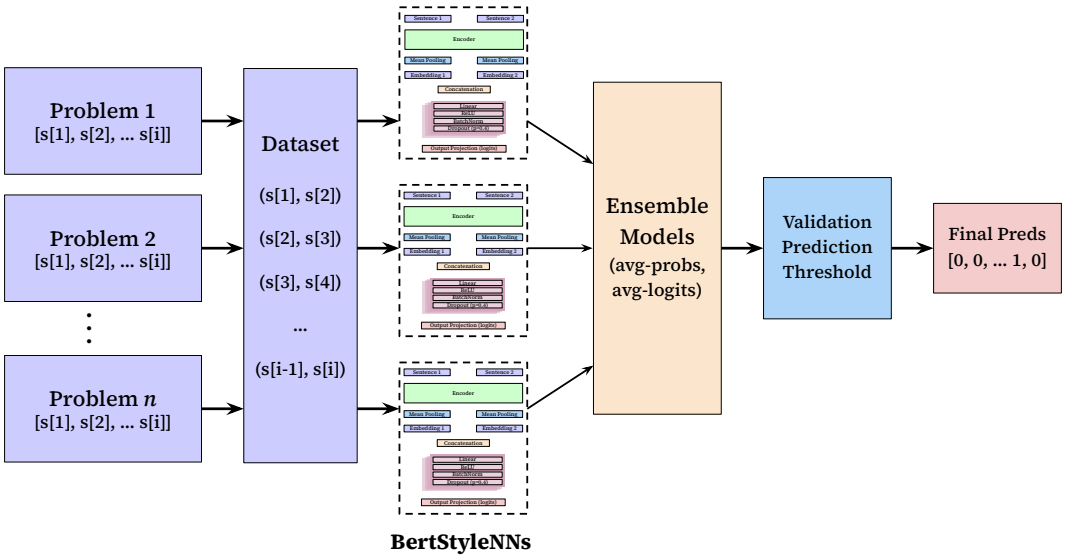

Figure 2: Full pipeline of the Ensembled-BertStyleNN system. Note that we use 3 different BertStyleNN models for the easy and medium subtask ensemble (consistent with the diagram), and 4 models for the hard subtask ensemble.

Table 4
Results of Ensembled-BertStyleNN on hidden test set, compared with naive baseline.

Approach	Easy (task1)	Medium (task2)	Hard (task3)	Average
Ensembled-BertStyleNN	0.909	0.793	0.698	0.8
Baseline (predict all 0s)	0.439	0.44	0.453	0.444

6. Conclusion

This paper describes an ensemble model system for the Multi-Author Style Analysis task. We fine-tune and ensemble new BertStyleNN models with 5 distinct pre-trained encoder models and a FFNN for binary classification. Our final system Ensembled-BertStyleNN achieves 0.8 macro F1 averaged over the three difficulty levels, indicating promise for ensemble transformer model approaches to the style analysis task.

Declaration on Generative AI

The authors have not employed any Generative AI 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] E. Zangerle, M. Mayerl, M. Potthast, B. Stein, Overview of the Multi-Author Writing Style Analysis

Task at PAN 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: 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.
[4] A. Dubey, Capturing Style Through Large Language Models - An Authorship Perspective (2024). URL: https://hammer.purdue.edu/articles/thesis/Capturing_Style_Through_Large_Language_Models_-_An_Authorship_Perspective/27947904. doi:10.25394/PGS.27947904.v1.
[5] E. Zangerle, M. Mayerl, M. Potthast, B. Stein, Overview of the Multi-Author Writing Style Analysis Task at PAN 2024, in: G. Faggioli, N. Ferro, P. Galuščáková, A. G. S. Herrera (Eds.), Working Notes Papers of the CLEF 2024 Evaluation Labs, CEUR-WS.org, 2024, pp. 2513–2522. URL: http://ceur-ws.org/Vol-3740/paper-222.pdf.
[6] J. Lv, Y. Yi, H. Qi, Team Fosu-stu at PAN: Supervised fine-tuning of large language models for Multi Author Writing Style Analysis, in: G. Faggioli, N. Ferro, P. Galuščáková, A. G. S. Herrera (Eds.), Working Notes Papers of the CLEF 2024 Evaluation Labs, CEUR-WS.org, 2024, pp. 2781–2786. URL: http://ceur-ws.org/Vol-3740/paper-265.pdf.
[7] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, 2021. URL: https://arxiv.org/abs/2106.09685. arXiv:2106.09685.
[8] T. Lin, Y. Wu, L. Lee, Team NYCU-NLP at PAN 2024: Integrating Transformers with Similarity Adjustments for Multi-Author Writing Style Analysis, in: G. Faggioli, N. Ferro, P. Galuščáková, A. G. S. Herrera (Eds.), Working Notes Papers of the CLEF 2024 Evaluation Labs, CEUR-WS.org, 2024, pp. 2716–2721. URL: http://ceur-ws.org/Vol-3740/paper-255.pdf.
[9] E. Zangerle, M. Mayerl, M. Potthast, B. Stein, Pan24 multi-author writing style analysis, 2024. URL: https://doi.org/10.5281/zenodo.10677876. doi:10.5281/zenodo.10677876.
[10] 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.
[11] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL: https://arxiv.org/abs/1810.04805. arXiv:1810.04805.
[12] P. He, X. Liu, J. Gao, W. Chen, Deberta: Decoding-enhanced bert with disentangled attention, 2021. URL: https://arxiv.org/abs/2006.03654. arXiv:2006.03654.
[13] Sentence-Transformers, all-minilm-l12-v2, https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2, 2024. URL: https://arxiv.org/abs/1810.04805, accessed: 2024-05-30.
[14] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. arXiv:2002.10957.
[15] Sentence-Transformers, Pretrained models documentation, http://sbert.net/docs/sentence_transformer/pretrained_models.html, 2024. Accessed: 2024-05-30.
[16] Sentence-Transformers, sentence-transformers/all-mpnet-base-v2, https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2024. Accessed: 2024-05-30.
[17] Microsoft, microsoft/mpnet-base, https://huggingface.co/microsoft/mpnet-base, 2024. Accessed: 2024-05-30.
[18] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, Y. Yang, Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models, 2021. URL: https://arxiv.org/abs/2108.08877. arXiv:2108.08877.
[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL: https://arxiv.org/abs/1910.10683. arXiv:1910.10683.

A. Training Configuration

In this section, we provide more details about our training parameters. For all training runs, we use `nn.BCEWithLogitsLoss` with the `pos-weight` parameter set to $\frac{0.8}{0.2}$, i.e. the approximate imbalance between positive (1) and negative (0) labels in the train and validation set. The `pos-weight` parameter penalizes false negatives (predicting a 0 when it should be a 1 more harshly than false positives (predicting 0 on true label 1), encouraging the model to predict more 1s. This mitigates some of the negative effects of the imbalanced training data.

Table 5 shows the complete list of hyperparameters used in training all models. Additionally, we used the AdamW optimizer, a consistent batch size of 16, and mean pooling of the encoder output for all models.

Table 5
Training configuration for each encoder model.

Encoder	Epochs	Encoder LR	FFNN LR	LR Scheduler	Warmup Ratio	Weight Decay
roberta-base	5	1e-5	1e-4	Linear	0.1	0.01
deberta-base	5	5e-6	1e-4	Linear	None	0.01
all-MiniLM-L12-v2	5	1e-5	1e-6	Reduce LR on Plateau	None	0.01
all-mpnet-base-v2	12	3e-6	8e-5	Linear	0.1	0.01
sentence-t5-base	17	5e-6	1e-4	Linear	None	0.05