Report on PAN 2024 Multi-Author Writing Style Analysis Task

Corey Reinholdtsen

University of Southern Maine, Portland ME 04103, USA

Abstract. The goal of multi-author writing style analysis is to identify at which point in a multi-author document the author changes. This paper is a description of the proposed Llama 3 and Orca 2 models for this task, and an analysis of their results as compared to a RoBERTa baseline model.

1 Introduction

The goal of the style change detection task is to correctly identify at which points in a multi-author document authorship changes. Participants are given a document, and asked to identify where in the document authorship changes at the paragraph level. This requires an intrinsic analysis of style change, as there are no external documents given to compare writing style. This task is important to the field of authorship verification, as style change detection is the only way to detect plagiarism if no comparison texts are given. The task is split into 3 subtasks, described in Table 1. The effectiveness of Orca 2 and Llama 3 Instruct were tested, and found to have significantly lower F_1 scores on the validation set than the baseline model for all subtasks.

Table 1. Subtasks of the multi-author writing style analysis task.

easy	The paragraphs of a document cover a wide variety of topics
medium	Topical variety in documents is small (though still present)
hard	There is no topical variety between paragraphs in a document

2 Related Work

2.1 Overview of Previous Labs

The details of the style analysis task have changed significantly over the years, with only PAN 2023 [10] sharing the same exact description as this year's task. In PAN 2022, for example, there were 3 subtasks of increasing difficulty [2]: (1): For a text written by two authors and with only one style change, identify where

the style change occurs at the paragraph level. (2): For a text written by two or more authors, identify all positions where style change occurs at the paragraph level. (3): For a text written by two or more authors, identify all positions where a style change occurs at the sentence level. In addition to the lack of consistent task definition, it is extremely difficult to find examples of working models for intrinsic style analysis. For this reason, all of the approaches examined in detail in this report are from PAN 2023.

Table 2. Test set results for the style change detection task at PAN 2023 [10]. Best results for a specific subtask are bolded.

Approach	Easy F_1	Medium F_1	Hard F_1
Chen et al. [3]	0.914	0.820	0.676
Huang et al. [5]	0.968	0.806	0.769
Jacobo et al. [6]	0.793	0.591	0.498
Kucukkaya et al. [7]	0.982	0.810	0.772
Hashemi et al. [4]	**0.984**	**0.843**	0.812
Ye et al. [9]	0.983	0.830	**0.821**

2.2 Description of Previous Approaches

Previous approaches are sorted by author in Table 2, which shows their F_1 scores on the easy, medium, and hard subtasks of the PAN 2023 test set [10]. Of the described approaches, all but one of them uses a transformer-based model at its core. Below are details of each participant's approach to the problem.

Chen et al. [3] Chen et al. use contrastive learning to train an encoder such that embeddings of paragraphs written by the same author are closer in space, while embeddings of paragraphs written by different authors are farther apart. The given data is processed in such a way that several extra virtual samples of data are created from the original set in order to train the encoder on more data. The loss function used for training the encoder is CoSENT, which attempts to make the cosine similarity of instances where authorship has not changed greater than the cosine similarity of instances where authorship has changed. This encoder is trained on the training set, and passed through a feedforward neural network to obtain the result.

The overall structure of the model is this: paragraph a and paragraph b are fed into the encoder to get their embeddings. The absolute value of the difference of a and b, $|a - b|$, is then spliced with the original embeddings for a and b to create a feature matrix of $a, b, |a-b|$ that is then fed into a fully connected neural network to retrieve the output. DeBERTa was used as the pretrained model in this case.

Huang et al. [5] Huang et al. use knowledge distillation to compress a 'teacher' model into a 'student' model, which is much less computationally intensive, requiring fewer resources and having lower computation time. The teacher model consists of the mT0-xl pre-trained encoder and a feed-forward neural network that is trained on the data set in addition to some external data. The student model uses the mT0-large encoder and a smaller neural network than the teacher model, but otherwise has the same architecture. There are two loss functions for this model: a knowledge distribution loss function, and a ground truth loss function. The knowledge distribution loss is defined as the KL divergence between the softmax of the teacher output and the student output. The ground truth loss function is the cross-entropy loss between the output of the student model and the ground truth.

Jacobo et al. [6] Jacobo et al. frame the problem as an authorship verification problem, and apply a variety of methods from that domain. To create a vector representation of each pair of adjacent paragraphs, they experiment with using a term-document (TD) matrix and prediction by partial matching (PPM). This vector representation is then fed into either a support vector machine (SVM) or a logistic regression (LR) classifier to determine if the paragraphs were written by the same author or not.

In total, 4 models were tested, using combinations of TD, PPM, SVM, and LR. The best results were achieved by PPM compression fed into a logistic regression classifier for subtasks 1 and 2 (easy and medium), and by a term-document matrix fed into a support vector machine for subtask 3 (hard). As can be seen in Table 2, these models scored considerably lower in F_1 scores in all subtasks than any of the other pre-trained model based approaches.

Kucukkaya et al. [7] Kucukkaya et al. experiment with using several different pre-trained encoders, in addition to testing what they refer to as 'transition-focused truncation', which truncates around the transition point of the paragraphs, instead of truncating from the beginning of each paragraph. For instance, if the transition is paragraph a then paragraph b, transition-focused truncation would take the last 256 tokens of a and the first 256 tokens of b as the input. The output of the encoder is then passed through a binary classification layer to retrieve the final classification. The pre-trained models tested were BERT, RoBERTa, DeBERTaV3, and BigBird-RoBERTa. All encoders were tested without warmup or transition-focused truncation, with just warmup, with just transition-based truncation, and then with both warmup and transition-based truncation. The baseline models tested against were random selection, and a TF-IDF matrix fed into a support vector machine.

The best results were achieved by DeBERTaV3 with warmup and transition-focused truncation for subtasks 1 and 2 (easy and medium), and DeBERTaV3 with just warmup for subtask 3 (hard).

Hashemi et al. [4] Hashemi et al. selected between RoBERTa, ELECTRA, and BERT as a backbone model, with RoBERTa being chosen, as it performed the best at base. They then explored generating additional samples based on data augmentation (similarly to Chen et al. [3]), in addition to using data from other subtasks as additional training data. They tested the performance difference between a basic model, a model using the augmented data, a generalized model, and an ensemble model based on majority voting.

The best results were achieved by the basic model in subtasks 1 and 3 (easy and hard), and the ensemble model in subtask 2 (medium). Notably, Hashemi et al. achieved the highest F_1 score for subtasks 1 and 2, and the second highest F_1 score for subtask 3.

Ye et al. [9] Ye et, al use DeBERTaV3 as the backbone encoder, using prompt tuning to optimize the encoder output. The prompt is not created manually, but learned automatically by the system. They then perform regularized dropout (R-drop) on the output of the encoder, which is then fed into a neural network with a specialized contrastive learning loss function used during training.

This model achieved good results on the test set, achieving the highest score in subtask 3 (hard) among all submissions for PAN 2023.

3 Methodology

Two models were tested for this task: Meta's Llama 3 [1], and Microsoft's Orca 2 [8]. Both of these models are based off of the Llama family of large language models (Orca 2 being a fine-tuned version of Llama 2), and as such are auto-regressive sequence-to-sequence models optimized for dialogue use cases.

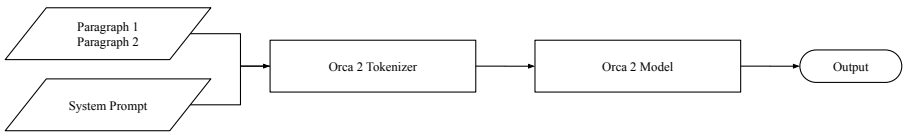

Fig. 1. Overall architecture of the Orca 2 classifier.

The overall architecture for the Orca 2 system can be seen in Figure 1. First, the input paragraphs are concatenated into a single string. This string is then passed into the tokenizer, along with a system prompt that tells the model how to respond. The tokenizer outputs the embeddings of the instruction/paragraph pair, which is then passed to the Orca 2 model itself to get the output tokens. These output tokens are decoded by the tokenizer, producing the final output of the system.

The architecture of the Llama 3 system is very similar to that of the Orca system, as can be seen in Figure 2. This is reasonable, as both models are based

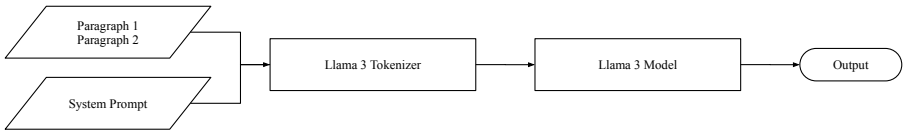

Fig. 2. Overall architecture of the Llama 3 classifier.

off of the Llama family of language models. Like the Orca system, the input paragraphs are concatenated and passed to the tokenizer along with the system prompt in order to generate the tokenized input prompt for the Llama 3 model. The output of the model is then passed to the tokenizer to decode the output and produce the final output of the system.

The goal of this testing was to answer the following questions:

1. Is there a statistically significant difference between the performance of the RoBERTa baseline and Llama 3?
2. Is there a statistically significant difference between the performance of the RoBERTa baseline and Orca 2?
3. Is there a statistically significant difference between the performance of Llama 3 and Orca 2?

4 Experimental Setting

Models were evaluated for effectiveness based on F_1 score, with a fine-tuned RoBERTa-large model as the baseline. The parameters for the baseline model were learning rate set to 1e-5, batch size set to 8, maximum sequence length set to 512, and number of epochs set to 10. These hyperparameters were chosen based on the experimental settings described by Hashemi et al. [4], but batch size was reduced due to hardware constraints. The pre-trained RoBERTa model was downloaded from HuggingFace and fine-tuned on the training datasets for each subtask, with accuracy being the finetuning metric. The finetuned models were then evaluated on the validation dataset for their respective subtasks.

The specific Llama model chosen was Llama-3-8B-Instruct, which is optimized for dialogue use cases such as assistant-like chat. The pre-trained model was downloaded from HuggingFace, then evaluated on the validation dataset for each task. For generation, the temperature parameter was set to 0.6, the top p parameter was set to 0.9, and the maximum amount of new tokens was set to 1. The model was also reloaded from the pre-trained checkpoint every 50 documents to clear the context window. This was done due to hardware constraints.

The Orca model that was tested is Orca-2-7B, which was downloaded from HuggingFace and evaluated on the validation set for each subtask. No hyperparameters were changed from the model defaults. Both the Orca and Llama models require system instructions, which tell the model how to respond to user messages. For both models, the system message used was "You are an expert in

writing style analysis. When you are given two paragraphs, say '0' if they are written by the same person, or '1' if they are not. Do not explain your reasoning." This achieved the output format that was desired from each model, and gave a brief description about what task the models were supposed to perform.

5 Results

The results of the experiment are presented in this section. Figure 3 shows the performance comparison between the baseline and the proposed models for each subtask. A paired bootstrap test of size 10,000 was performed, whose results are shown in Table 3. Assuming $\alpha = 0.05$, the baseline model has significantly better performance than the proposed models across all subtasks. In addition to this, the Llama-based model has significantly better performance than the Orca-based model on subtask 1, while the Orca-based model has significantly better performance than the Llama-based model on subtask 3.

Table 3. Results of paired bootstrap significance testing.

Comparison	Easy	Medium	Hard
Orca vs. RoBERTa	0.000	0.000	0.035
Llama vs. RoBERTa	0.000	0.000	0.054
Orca vs. Llama	0.000	0.000	0.000

These results make sense, as the proposed models were designed for autoregressive text generation, not binary classification. In addition to this, the baseline model was fine-tuned on the training set, whereas the proposed models were not. The prompt chosen for the proposed models is most likely not ideal, and could also be greatly affecting the results of said models.

References

1. AI@Meta: Llama 3 model card (2024), https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md
2. Bevendorff, J., Chulvi, B., Fersini, E., Heini, A., Kestemont, M., Kredens, K., Mayerl, M., Ortega-Bueno, R., Pezik, P., Potthast, M., et al.: Overview of pan 2022: Authorship verification, profiling irony and stereotype spreaders, and style change detection. In: International Conference of the Cross-Language Evaluation Forum for European Languages. pp. 382–394. Springer (2022)
3. Chen, H., Han, Z., Li, Z., Han, Y.: A writing style embedding based on contrastive learning for multi-author writing style analysis. Working Notes of CLEF (2023)
4. Hashemi, A., Shi, W.: Enhancing writing style change detection using transformer-based models and data augmentation. Working Notes of CLEF (2023)
5. Huang, M., Huang, Z., Kong, L.: Encoded classifier using knowledge distillation for multi-author writing style analysis. Working Notes of CLEF (2023)

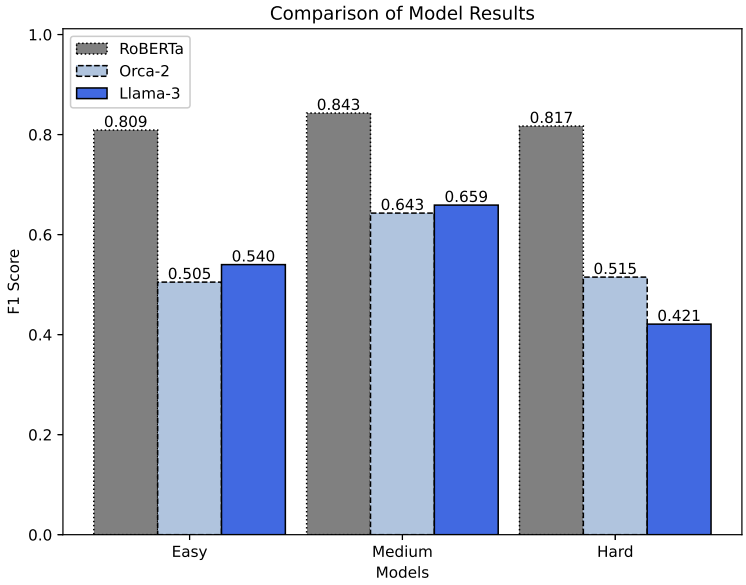

Fig. 3. Validation set results for each model.

6. Jacobo, G., Dehesa, V., Rojas, D., Gómez-Adorno, H.: Authorship verification machine learning methods for style change detection in texts. Working Notes of CLEF (2023)
7. Kucukkaya, I.E., Sahin, U., Toraman, C.: Arc-nlp at pan 2023: Transition-focused natural language inference for writing style detection. arXiv preprint arXiv:2307.14913 (2023)
8. Mitra, A., Del Corro, L., Mahajan, S., Codas, A., Simoes, C., Agarwal, S., Chen, X., Razdaibiedina, A., Jones, E., Aggarwal, K., et al.: Orca 2: Teaching small language models how to reason. arXiv preprint arXiv:2311.11045 (2023)
9. Ye, Z., Zhong, C., Qi, H., Han, Y.: Supervised contrastive learning for multi-author writing style analysis. Working Notes of CLEF (2023)
10. Zangerle, E., Mayerl, M., Potthast, M., Stein, B.: Overview of the multi-author writing style analysis task at pan 2023. Working Notes of CLEF (2023)