Pangram at GenAI Detection Task 3: An Active Learning Approach to Machine-Generated Text Detection

Bradley Emi and **Max Spero** and **Elyas Masrour**
Pangram Labs, Inc.
Correspondence: info@pangram.com

Abstract

Deep learning approaches to machine-generated text detection typically suffer from chronic undertraining due to premature convergence. In our submission to the COLING Shared Task 3, we employ a two-stage training procedure to mitigate undertraining. First, we train an LLM-based classifier on a large multilingual dataset comprised of a wide variety of domains, languages, prompts, and LLMs. Then, we run offline inference on the AI-labeled side of the RAID train split, selecting the examples with the highest error and their human counterpart examples to continue training the model. We employ several preprocessing strategies to improve the robustness of the model. As a result, we achieve the highest score on the adversarial attack portion of the RAID leaderboard.

1 Introduction

We employ the same approach to the COLING Machine-Generated Text Detection task (Dugan et al., 2025) as we describe for our main production model at Pangram Labs (Emi and Spero, 2024). As we describe in our technical report, classifier accuracy when training deep learning models to detect AI-generated text does not naturally obey scaling laws. Classifier performance scales sublinearly as the size of the dataset used to train AI-generated text classifiers increases, ultimately even converging to a regime in which adding more data to the classifier no longer helps performance at all (e.g., validation loss stops decreasing even before the first epoch concludes).

We hypothesize that the reason for this "saturation of scaling laws" occurs for multiple reasons. First, we notice during training on a random distribution of human and AI-generated examples that the loss curves are very spiky. There are several batches in a row with nearly-zero loss followed by single batches with very high loss, causing gradient norms to reach very high values. As a result, low learning rates and aggressive gradient clipping are required for stable convergence. Second, many AI examples follow very simple patterns that make detection very obvious. For example, LLMs responses often begin with "Certainly!" or "Sure, here is a...", which are very obvious tells that a piece of text is LLM-generated. These easy examples flood the training set and cause learning to end prematurely as the model overfits to these simple patterns.

Our intuition is that a naively sampled dataset is flooded with examples that are too easy and have such giveaway patterns. To fix this problem, we resample the dataset with "hard" examples oversampled. To define a hard example, we first train a model on the randomly sampled dataset. We then use this model to mine for high error examples on the RAID dataset. We then take these examples, and their opposite label pairs, and retrain the model, which results in much more stable convergence behavior.

2 Methodology

2.1 Initial Datasets

Our initial dataset is seeded with a wide variety of human-written datasets from prior to 2022. We use datasets from the following domains: reviews, news, general web text, email, student writing/essays, creative writing, question and answers, ELL/ESL (English as a Second Language), scientific/medical papers, Project Gutenberg, and Wikipedia. We do not filter our dataset on language; however we tune the language composition such that English is the primary language used and there is sufficient representation from the top 20 languages used on the Internet. All data is ethically sourced and properly licensed for commercial use. For specifics on dataset composition, please

Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect), pages 347–351
January 19, 2025.

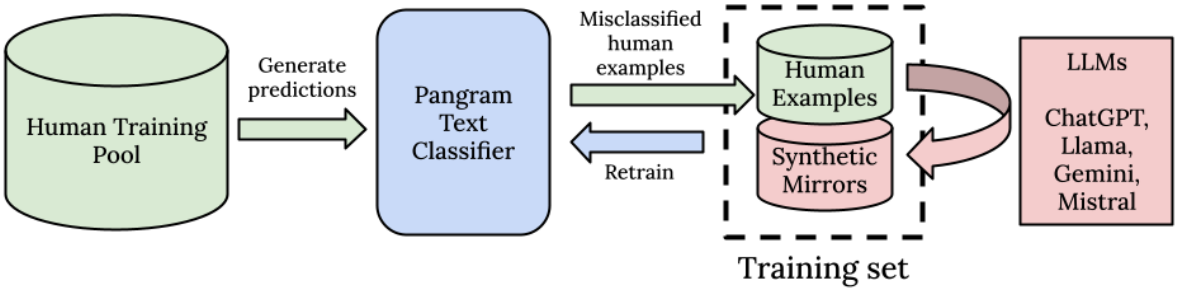

Figure 1: Pangram Active Learning Approach

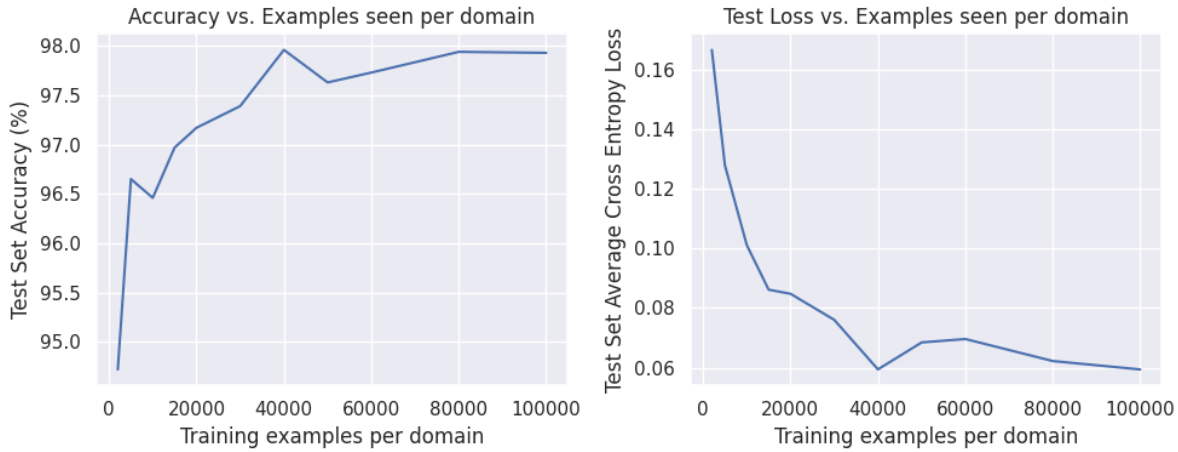

Figure 2: Without active learning, performance saturates as dataset size increases.

contact the authors.

2.2 Synthetic Data Creation

Our initial dataset is sampled randomly and all is labeled as human. To generate the AI side of the dataset, we use synthetic mirror prompts, which we describe in more detail in our technical report (Emi and Spero, 2024).

We define the term "mirror prompt" to be a prompt based on the original example that is used to generated a "synthetic mirror" or "mirror example." The goal of each mirror prompt is to generate an example that matches the topic and length of the original document.

If the original document is "<original review>", then a mirror prompt may look like this:

`[Prompt] Write a <original review star rating> star review for <original review business name>. Make the review around <original review length> words long.`

Another example may be for a student essay. We sometimes use double prompts, such as the following:

`[Prompt] What is a good title for this essay? <original essay> Only give the title in your response.`
`[Assistant] <Title>`
`[Prompt] Write an essay with the following title: <Title>. Make the essay around <original essay length> words long.`

2.3 LLMs used for Synthetic Mirrors

When the initial data is not in English, we translate our prompts to the language of the source data using DeepL's translation API. (DeepL SE, 2024)

For synthetic mirrors in the initial training stage, we use the following LLMs:

- GPT-3.5 (multiple subversions)
- GPT-4, GPT-4-turbo, and GPT-4o (multiple subversions)
- Claude 2 and 3 (multiple subversions and sizes)

- LLaMA 2, 3, and 3.1 (multiple subversions and sizes)
- Mistral (multiple versions and sizes)
- Gemini Pro and Flash (multiple subversions)

It is notable that we only use modern LLMs that are instruction-tuned and post-trained. We do not train on base models because they produce noticeably lower-quality outputs and are substantially less commonly used in real-world applications.

2.4 Preprocessing and Filtering

We employ the following preprocessing strategies prior to tokenization to enhance robustness.

- We remove all zero-width spaces.
- We convert all text to lowercase.
- We collapse all consecutive whitespace into a single whitespace.
- We attempt to remove LLM "headers", such as "Sure! Here is a..." or "Certainly!" or "As an AI language model,", etc.
- We convert all text to unicode standard characters using the `unidecode` package (English text only).

Prior to training, we also filter the dataset using the following criteria:

- We remove all examples that are under 25 words long.
- Sometimes LLMs simply repeat the prompt back to the user and do not say anything else. We attempt to filter out this use case by checking to see if 50 percent or more of the LLM output matches the input prompt exactly and if it does, we filter the example out.

2.5 Augmentation

We also employ two augmentation strategies.

- We randomly machine translate around 20 percent of the English training set to another language using DeepL. We randomly select the language from the top 20 languages on the Internet.
- With 50 percent probability, we randomly mask 15 percent to 75 percent of the input tokens from the model at training time. We find that this improves training stability and helps the model from overfitting to common giveaway phrases and syntactic patterns. This technique is very similar to CutOut which is commonly used in computer vision (DeVries and Taylor, 2017).

2.6 Tokenization, Architecture, and Model Training

We use the Mistral NeMo architecture (Mistral AI Team, 2024) which has approximately 12 billion parameters, with an untrained linear classification head and an LLM classification head (to identify which LLM an AI text came from, as an auxiliary task). Following the usual convention for sequence classification modeling using an autoregressive language model, the hidden state from the final token in the sequence is used as the input to both classification heads. As is common practice in LLM finetuning, we use trainable LoRA (Hu et al., 2022) adapters while keeping the base model frozen. We use the Tekken tokenizer out of the box, which is noted for its strong multilingual performance. We truncate the context window to 512 tokens to constrain the model to using only short-range features. When necessary, we simply crop the input to fit the context window.

We train the model to convergence using 8 A100 GPUs with an effective batch size of 24 using a weighted cross-entropy loss. We use the AdamW optimizer and a linearly decaying learning rate schedule. We train the model for 1 epoch, which took about 9 hours, and select the checkpoint based on a weighted cross-entropy loss with 3 times the weight given to false positives.

2.7 Active Learning

After the initial model is trained, we mine the RAID training set for AI examples that the initial model classifies as human. We then select the 50,000 highest error examples and add them back into the training set with the 50,000 human example pairs corresponding to these highest error examples. We restrict our search to only examples that have no adversarial attack.

This process has several side effects that also improve the underlying data distribution. First, base models and other models that are difficult to detect are introduced into the training set, but only

the base model distribution that differs significantly from the instruction-tuned model distributions, and proportionally to how poorly the generalization is to each particular model.

This side effect generalizes to domains, languages, and attacks as well: the worse the initial model is at predicting a particular split of the data, the more that split gets overrepresented in the following training run.

We reintroduce this data into the training set and retrain the model for 1 epoch until convergence.

3 Results

Pangram places first overall in detecting GPT4, ChatGPT, and LLaMA (state-of-the-art models) with no adversarial attacks, second overall in detecting all models without adversarial attacks, and tied for first overall in detecting AI-generated text with adversarial attacks.

The full results are posted publicly on the RAID website. We refer the reader to the leaderboard for full details. (Dugan et al., 2025)

4 Discussion

We believe our method is a general framework for scaling the deep learning approach to detecting AI-generated text, and this prototype model is only a starting point. For example, the framework could be extended beyond the current dataset to even more domains and data and larger models, or customized to private data or domains in which open data is not readily available at scale, such as messaging, email, or other data with high amounts of PII.

4.1 Differences between RAID submission and Pangram's Commercial Model

Our submission uses a similar framework and is otherwise trained in the same way as Pangram's commercial model, but there are some slight differences.

In Pangram's original framework (Emi and Spero, 2024), we perform hard negative mining with synthetic mirrors on large human-written text corpora, to reduce the false positive rate as much as possible. However, when evaluated on RAID, the initial model has excellent precision but poor recall. This is due to the domain shift to the different models used in the RAID benchmark. To reduce our false positive rate, we perform the inverse operation: hard *positive* mining with human mirrors. This is not generally possible in a real-world setting due to the fact that positively labeled examples are not as abundant as negatively labeled examples, but it was possible for the RAID benchmark and so we decided to take advantage.

Because the RAID benchmark contains many lower-quality models, such as MPT, GPT-2, etc., we needed to optimize the model to perform well on these lower quality models, but this required trading off some false positives. For real-world usage, detecting these low-quality model outputs is not important, so our production model does not detect these as well as the RAID model, but has a lower false positive rate.

We also enforce a higher minimum word count for the commercial model, which again lowers our false positive rate but hurts our recall on shorter text. Other researchers have given theoretical and empirical grounding for the relationship between detectability and sequence length (Chakraborty et al., 2023). In practice, for the production model, we choose to prioritize precision, but for the RAID submission, we instead to choose a more balanced approach where we equally prioritize precision and recall on these short texts.

Additionally, we do not use unidecode for the commercial model on non-English languages.

5 Conclusion

In this work, we have demonstrated a general framework for domain and model adaptation of deep-learning based AI detectors based on active learning and mirroring. We further argue that active learning is necessary for scaling performance both in terms of model size and data and present a model larger and more accurate than the other methods in the RAID benchmark, without the need for using perplexity-based features or otherwise handcrafted feature engineering.

References

Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. 2023. On the possibilities of ai-generated text detection. *Preprint*, arXiv:2304.04736.

DeepL SE. 2024. Deepl translator. Accessed: 2024-11-12.

Terrance DeVries and Graham W. Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *Preprint*, arXiv:1708.04552.

Liam Dugan, Andrew Zhu, Firoj Alam, Preslav Nakov, Marianna Apidianaki, and Callison-Burch Chris. 2025. GenAI Content Detection Task 3: Cross-domain machine generated text detection challenge. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Bradley Emi and Max Spero. 2024. Technical report on the pangram ai-generated text classifier. *Preprint*, arXiv:2402.14873.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Mistral AI Team. 2024. Mistral NeMo. Released in collaboration with NVIDIA, July 18, 2024.