SimLLM: Detecting Sentences Generated by Large Language Models Using Similarity between the Generation and its Re-generation

Hoang-Quoc Nguyen-Son, Minh-Son Dao, and Koji Zettsu
National Institute of Information and Communications Technology, Japan
{quoc-nguyen,dao,zettsu}@nict.go.jp

Abstract

Large language models have emerged as a significant phenomenon due to their ability to produce natural text across various applications. However, the proliferation of generated text raises concerns regarding its potential misuse in fraudulent activities such as academic dishonesty, spam dissemination, and misinformation propagation. Prior studies have detected the generation of non-analogous text, which manifests numerous differences between original and generated text. We have observed that the similarity between the original text and its generation is notably higher than that between the generated text and its subsequent regeneration. To address this, we propose a novel approach named SimLLM, aimed at estimating the similarity between an input sentence and its generated counterpart to detect analogous machine-generated sentences that closely mimic human-written ones. Our empirical analysis demonstrates SimLLM's superior performance compared to existing methods.

1 Introduction

The rise of generative AI, especially large language models, has had a substantial impact across various applications. However, it also presents challenges, such as academic dishonesty and the spread of disinformation, stemming from the misuse of generated text. Therefore, our goal is to create a strategy to detect and mitigate the negative effects associated with the improper use of generated text.

Detection of text generated by large language models uses three main techniques. Firstly, supervised learning methods (Solaiman et al., 2019; Wang et al., 2023; Hu et al., 2023; Wu et al., 2023) train classifiers on datasets of original and generated text, though this requires large volumes of training data. The zero-shot approach (Bhattacharjee and Liu, 2023; Mitchell et al., 2023) eliminates the need for training but is sensitive to out-of-distribution text. Recent research explores watermarking methodologies (Kirchenbauer et al., 2023) to force models to produce predefined words, aiding detection, but this requires modifying the models, which is impractical for proprietary models like ChatGPT. Previous studies mainly address non-analogous text with substantial differences between original and generated content. In contrast, we focus on analogous generated text, where changes to the original text are minimal.

Motivation An AI model aims to extensively optimize original data to generate new data. This process often results in a significant disparity between the original and the generated data. When the model optimizes the generated data to create re-generated data, the already optimized nature of the generated data limits further optimization. As a result, the gap between the generated and re-generated data diminishes. To illustrate, we randomly selected a human sentence (h) from the Extreme Summarization (XSum) dataset (Narayan et al., 2018) (Figure 1). Then, a large language model, specifically ChatGPT (GPT 3.5-turbo), was tasked with generating a machine sentence ($m_{ChatGPT}$) conveying an opposite meaning to the original text. ChatGPT and LLaMa 2 70B were utilized to proofread both the human-written text ($h_{ChatGPT}$ and h_{LLaMa}) and the machine-generated text ($m_{ChatGPT-ChatGPT}$ and $m_{ChatGPT-LLaMa}$), with the respective subscripts indicating the sequence of using the large language models. Analysis showed that proofreading of the human text by ChatGPT introduced numerous disparities between h and $h_{ChatGPT}$, whereas fewer differences were observed between $m_{ChatGPT}$ and $m_{ChatGPT-ChatGPT}$. In this example, while there were ten word differences between h and $h_{ChatGPT}$ highlighted in underline,

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 22340–22352
November 12-16, 2024

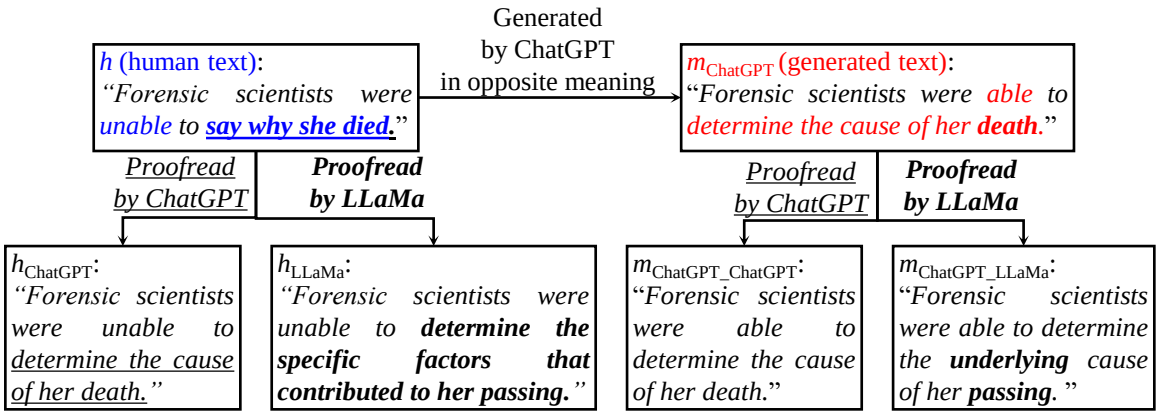

Figure 1: The degree of similarity observed between the original text and its proofread version is significantly reduced compared to that between the generated text and the re-generated text. Differences between the original and generated text are visually highlighted using distinct colors. Variances attributed to ChatGPT and LLaMa by proofreading are emphasized in underlined and **bold** formatting, respectively.

$m_{ChatGPT}$ was identical to $m_{ChatGPT-ChatGPT}$. These differences aid in distinguishing between human and machine-generated text. Furthermore, comparing ChatGPT and LLaMa demonstrated that the gap in the pair $m_{ChatGPT}$ and $m_{ChatGPT-ChatGPT}$ (word difference equals zero) tends to be smaller than that in the pair $m_{ChatGPT}$ and $m_{ChatGPT-LLaMa}$ (word difference equals three highlighted in ***bold***). Hence, the choice of a large language model significantly influences the identification of generated text.

Contribution This paper introduces a method called SimLLM, designed to identify sentences generated by large language models. Initially, candidate large language models are employed to generate proofread versions of an input sentence. Subsequently, each proofread version is compared with the input, and their similarities are evaluated. Next, the input sentence is concatenated with its proofread versions and organized based on their similarity scores. Finally, a RoBERTa model undergoes fine-tuning to ascertain the source of the concatenated sequence, discerning between human-written content and content generated by a large language model. We summarize our contributions as follows:

- We have developed a strategy for constructing a dataset consisting of coherent sentences generated by large language models[1]. To the best of our knowledge, this is the first dataset presenting analogous pairs of original text and generated text on a sentence-by-sentence level[2].
- We noticed that optimizing the original text is relatively less challenging compared to optimizing the generated text. Therefore, we developed SimLLM to distinguish generated sentences by assessing the similarity between the input sentence and its proofread versions.
- We conducted experiments on detecting sentences generated by twelve prominent large language models. These experiments indicate that SimLLM exhibits superior performance compared to existing approaches.

2 Related Work

The methods previously used to detect text generated by large language models can be classified into three approaches.

The first strategy involves training models on large datasets to identify generated text characteristics, such as OpenAI's fine-tuning of the RoBERTa model (Solaiman et al., 2019). Some researchers have analyzed probability distributions in large language models' hidden layers (Wang et al., 2023), while others have used a paraphraser in a GAN to train the detector component (Hu et al., 2023). The intrinsic dimension of the embedding space from long texts has been estimated to understand the workings of these models better (Tulchinskii et al., 2023). Other approaches

[1]Source code and dataset are available at https://github.com/quocnsh/SimLLM

[2]The comparison between our dataset and existing datasets is provided in Appendix A

include building a proxy model to estimate generated text's perplexity (Wu et al., 2023), using positive-unlabeled learning (Tian et al., 2024) to improve performance on short text, and highlighting human text's coherence to spot machine-generated text discrepancies (Liu et al., 2023). Some researchers have also incorporated top similarity texts from the training set into prompts and used in-context learning to boost detector and attacker capabilities (Koike et al., 2024). However, this approach is sensitive to out-of-distribution texts.

Watermarking is another method where a language model is guided to generate text that meets specific criteria, acting as a watermark to identify generated content. For example, Kirchenbauer et al. (2023) instructed the model to use only a certain set of "*green*" words, avoiding the "*red*" ones. However, this method's downside is that it requires modifying the original models, which is impractical for real-world use, especially considering the proprietary nature of many large language models.

The third strategy involves zero-shot detection, where research identifies generated text without training. Bhattacharjee and Liu (2023) employed this method by prompting ChatGPT to detect generated texts from various large language models. Gehrmann et al. (2019) noted that large language models often predict the next word in a text sequence with high probability, which can be assessed through ranking, logarithms, and entropy. Other researchers have improved performance by combining ranking and logarithms (Su et al., 2023), or by introducing a method where original words are randomly perturbed and the change in log probability is analyzed (Mitchell et al., 2023; Bao et al., 2024). Close to our work, Zhu et al. (2023) and Mao et al. (2024) evaluated the similarity between input text and revised text. However, these approaches face challenges in identifying out-of-distribution text.

3 SimLLM

Figure 2 illustrates our goal, which is to distinguish whether a given input sentence, denoted as s, is generated by a large language model or authored by a human. Initially, we use various large language models to proofread s. This generates a set $S' = \{s'_1, s'_2, \ldots\}$. At this phase, a heuristic algorithm is employed to produce consistent

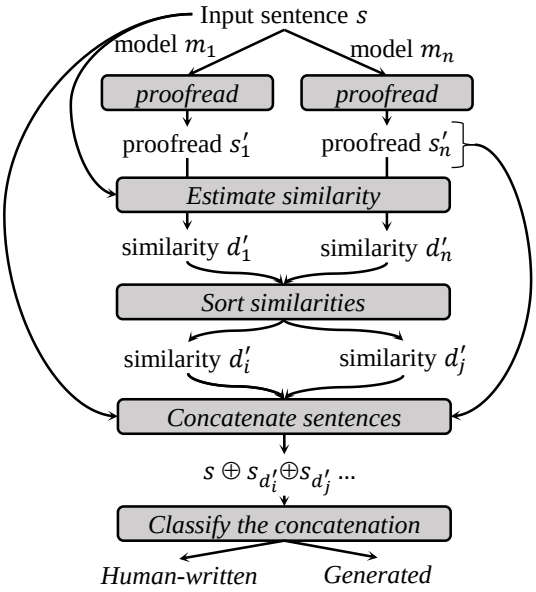

Figure 2: The proposed method (SimLLM) aims to determine whether a given sentence s is generated by a large language model or is written by a human.

and insightful proofread sentences. The details of this step are shown in Figure 3 and explained further in Section 3.1 and Section 3.2. Subsequently, we evaluate the similarity between s and each proofread sentence in S'. These sentences are then arranged in descending order of similarity. Following this, we combine s with each text in S' and input them into a classifier. The classifier's role is to determine whether s is a machine-generated or human-written sentence. The algorithm of SimLLM is outlined in Algorithm 1, with the main steps detailed as follows:

3.1 Proofreading the Input Sentence

We utilize a straightforward prompt to generate the proofread sentence s' from the input sentence s. The prompt is structured as a direct request to the large language model: "*Proofreading for the text:* $<sentence>$", where $<sentence>$ is replaced with s. It is important to note that the use of complex prompts often results in unstable or uninformative outcomes, as shown in Figure 4. This observation is consistent with the results of a recent research study (Salinas and Morstatter, 2024), which demonstrates that complicating the prompt tends to reduce large language model performance. Therefore, we opt for a simple prompt and propose a heuristic algorithm for extracting the proofread sentence. The comparison between our proposed prompt and that of Zhu et al. (2023)

Algorithm 1: SimLLM.

Input : Input sentence s; Candidate model $M = \{m_1, m_2, ...\}$
Output: Original/Generated

$prompt \leftarrow$ "*Proofreading for the text: "* + s
$S' \leftarrow \{\}$ ▷ Proofread sentences
$D' \leftarrow \{\}$ ▷ Similarity distances
for *each* m_i *in* M **do**
  $best\_similarity \leftarrow -\infty$
  $raw\_completion \leftarrow \text{LLM\_INVOKE}(m_i, prompt)$
  $candidates \leftarrow \text{SPLIT\_SENTENCE}(raw\_completion)$
  for *each* s_i *in* $candidates$ **do**
    $d_i \leftarrow \text{SIMILARITY}(s, s_i)$
    if $d_i > best\_similarity$ **then**
      $best\_candidate \leftarrow s_i$
      $best\_similarity \leftarrow d_i$
    end if
  end for
  if $best\_similarity > \alpha$ **then**
    Add $best\_candidate$ into S'
    Add $best\_similarity$ into D'
  else
    Add s into S'
    Add $+\infty$ into D'
  end if
end for
$S'_{sorted} \leftarrow$ Sort S' by D' in descending order
$concatenation \leftarrow s$
for *each* s'_i *in* S'_{sorted} **do**
  $concatenation \leftarrow concatenation \oplus s'_i$
end for
$result \leftarrow \text{CLASSIFY}(concatenation)$ ▷ Original/Generated
return $result$

is discussed in Appendix B.

3.2 Extracting a Proofread Sentence Using Heuristics

First, we employ a large language model with a simple prompt to generate a raw completion. Next, we break down this raw completion into individual candidate sentences. We then assess each candidate sentence against the input sentence s and choose the one that demonstrates the highest similarity. We utilize the BART score (Yuan et al., 2021) as our similarity metric, which is favored for its comprehensive contextual coverage compared to other metrics such as BLEU, ROUGE, and BERT, as highlighted by Zhu et al. (2023). However, if the original sentence is already perfect, the raw completion may not represent the proofread version. To address this, we propose the use of a minimum threshold, α. Based on empirical observations, we determine α to be -2.459 across all large language models. If the highest similarity score among the candidates is still lower than α, we retain the original sentence s as the proofread version.

3.3 Classifying the Input Sentence

After generating proofread sentences $S' = \{s'_1, s'_2, ...\}$ from the input sentence s, we evaluate the similarity between s and each s'_i, sorting them in descending order. Subsequently, we concatenate the original sentence s with each proofread sentence s', arranging them in the sorted order. A classifier is then used to determine whether s is an original sentence or a generated one. Specif-

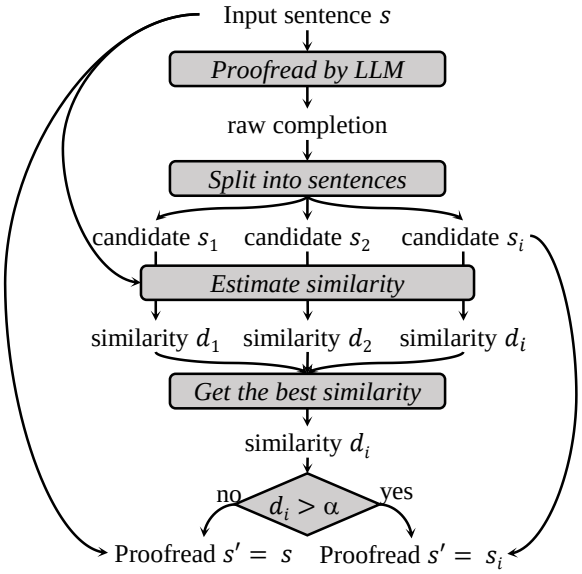

Figure 3: Generating a proofread sentence s' from an input sentence s.

ically, we fine-tune a RoBERTa-base model with fixed parameters: the number of epochs is set to 10, the batch size to 64, and the learning rate to 2×10^{-5} for all experiments. Additionally, we implement early stopping with a patience level of 3 on validation data to prevent overfitting.

4 Evaluation

4.1 Individual Models

We conducted experiments using the XSum dataset (Narayan et al., 2018), which consists of news articles written by humans[3]. This text was processed using twelve popular large language models developed by well-known companies, as listed in Table 1. These models have shown stability and display a comprehensive understanding of all prompts mentioned in this paper, consistently generating high-quality outputs. Due to the significant cost associated with proprietary large language models such as GPT-4o and Gemini, we randomly processed 5,000 sentences. These sentences were then divided into training, validation, and testing sets at ratios of 80%, 10%, and 10%, respectively. The number of testing samples is equivalent to the experiments conducted in the paper by DetectGPT (Mitchell et al., 2023). To achieve our goal of distinguishing between human and machine-generated text, we filtered out pairs that were identical. Following DetectGPT (Mitchell et al., 2023), we used ROC accuracy as the primary metric throughout this paper. Additionally, we maintained a temperature of 0, a common practice for producing stable outputs.

Model	Version	Developer
ChatGPT	GPT 3.5-turbo	OpenAI
GPT-4o	GPT-4o 2024-05-13	OpenAI
Yi	Yi 34B	01.AI
OpenChat	3.5 1210 7B	Alignment AI
Gemini	Gemini 1.5 Pro	Google
LLaMa	LLaMa 2 70B	Meta
Phi	Phi 2	Microsoft
Mixtral	8x7B Instruct v0.1	Mistral AI
QWen	QWen 1.5 72B	Alibaba
OLMO	7B Instruct	Allen AI
WizardLM	13B V1.2	WizardLM
Vicuna	13B v1.5	LMSYS

Table 1: The details regarding large language models utilized for text generation.

We compare SimLLM with eight existing methods[4] and a Baseline model. The RoB-base and RoB-large models (Solaiman et al., 2019) are trained on a combination of generated examples and original documents sourced from the WebText test set. Approaches including log p(x) (Gehrmann et al., 2019), Rank (Gehrmann et al., 2019), LogRank (Mitchell et al., 2023), and Entropy (Gehrmann et al., 2019) estimate the probability of generating the next word. DetectGPT (Mitchell et al., 2023) analyzes the probability change in text following the replacement of certain words. BART (Zhu et al., 2023) assesses similarity using BART-score between the input and revised text, comparing it against a predetermined threshold derived from the training dataset. In the Baseline scenario, we fine-tune a model on the same training dataset, employing the RoBERTa model with identical hyperparameters (epochs, learning rate, batch size, etc.). A summary of the results can be found in Table 2.

The results indicate that the first seven methods are ineffective in detecting generated text. When large language models are used for proofreading, they only introduce minor alterations to a small

[3]Experiments with other datasets are described in Appendix C.

[4]The comparison with other existing methods is presented in Appendix D.

Prompt	ChatGPT	LLaMA
Proofreading for the text: "*The gap suggests Mercedes are favorites for the race, even if Ferrari can be expected to push them.*"	The sentence is grammatically correct, but for improved clarity and flow, you might consider the following revision: "*The gap suggests **that** Mercedes **is** **the** **favorite** for the race, **although** Ferrari can be expected to push them.*"	Here's the proofread text: "*The gap **indicates** that Mercedes are **the** favorites for the race, even **though** Ferrari is **likely** to **challenge** them.*"
Proofreading for the text: "*The gap suggests Mercedes are favorites for the race, even if Ferrari can be expected to push them.*" Only output the proofread text.	"*The gap suggests Mercedes **is** favorites for the race, even if Ferrari can be expected to push them.*"	"*The gap suggests Mercedes are **favourites** for the race, even if Ferrari can be expected to push them.*"
Proofreading for the text: "*The gap suggests Mercedes are favorites for the race, even if Ferrari can be expected to push them.*" Only output the proofread text without explanation.	*The gap suggests Mercedes **is** favorites for the race, even if Ferrari can be expected to push them.*	*The gap suggests Mercedes are favorites for the race, even if Ferrari can be expected to push them.*

Figure 4: Exploring variations in outputs generated by large language models across different prompts. The modifications in outputs are emphasized with underlining and **bold**. Simpler prompts tend to yield more imaginative outputs.

Model	RoB-base	RoB-large	log p(x)	Rank	LogRank	Entropy	DetectGPT	BART	Baseline	SimLLM
ChatGPT	0.558	0.571	0.555	0.538	0.554	0.522	0.528	0.842	0.830	**0.916**
GPT-4o	0.535	0.537	0.530	0.503	0.517	0.525	0.521	0.639	0.786	**0.816**
Yi	0.550	0.565	0.538	0.531	0.536	0.535	0.521	0.874	0.880	**0.947**
OpenChat	0.563	0.573	0.517	0.514	0.519	0.557	0.520	0.875	0.887	**0.954**
Gemini	0.547	0.549	0.527	0.501	0.521	0.518	0.513	0.791	0.777	**0.859**
LLaMa	0.591	0.594	0.541	0.521	0.531	0.511	0.549	0.663	0.846	**0.883**
Phi	0.518	0.538	0.393	0.398	0.398	0.636	0.434	0.761	0.914	**0.937**
Mixtral	0.541	0.556	0.451	0.451	0.444	0.604	0.519	0.652	0.835	**0.837**
Qwen	0.544	0.555	0.481	0.489	0.474	0.544	0.493	0.767	0.844	**0.900**
OLMo	0.545	0.573	0.466	0.460	0.470	0.579	0.485	0.762	0.812	**0.895**
WizardLM	0.567	0.570	0.512	0.510	0.510	0.536	0.518	0.755	0.813	**0.856**
Vicuna	0.593	0.599	0.540	0.518	0.536	0.543	0.553	0.756	0.824	**0.866**
Average	0.554	0.565	0.504	0.495	0.501	0.551	0.513	0.761	0.837	**0.889**

Table 2: Detecting generated text with individual large language models.

portion of the content. Consequently, these methods often mistake generated text for the original, resulting in detection performance similar to random guessing. For example, we analyzed $\log p(x)$ and LogRank features on average, finding that the difference between human and machine-generated features by ChatGPT is significantly smaller at the sentence level than at the document level in DetectGPT's paper (Mitchell et al., 2023), leading to lower detection accuracy as shown in Table 3. In contrast, BART, alongside the Baseline and SimLLM, undergo specialized training for this text type, yielding substantial advancements. The Baseline, through analyzing the inherent characteristics of the input text, achieves greater refinement compared to the BART-based method, which primarily estimates the similarity between the input and its revised form. SimLLM combines the strengths of both strategies, resulting in superior performance. Given that the initial seven methods exhibit performance similar to random guessing, we present BART, Baseline, and SimLLM in subsequent experiments.

We compared the performance of the top three methods while varying the sample size, as illustrated in Figure 5. The text was generated by ChatGPT. The performance of BART remains almost unchanged with varying sample sizes, indicating that BART's single output value cannot fully exploit the similarity between the input text and its generation. In contrast, both the Baseline and SimLLM benefit from larger sample sizes. SimLLM consistently maintains an approximately 8% performance gap over the Baseline.

Method	Granularity	Human Feature	Machine Feature	ROC Accuracy
$\log \mathrm{p}(\mathrm{x})$	Document	-2.77	-1.95	0.921
LogRank	Document	-1.41	-0.87	0.932
$\log \mathrm{p}(\mathrm{x})$	Sentence	-3.33	-3.20	0.555
LogRank	Sentence	-1.79	-1.70	0.554

Table 3: Feature extraction from $\log \mathrm{p}(\mathrm{x})$ and LogRank between document and sentence levels.

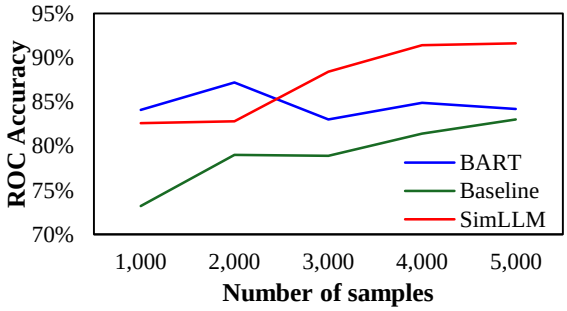

Figure 5: Detecting generated text through changes in sample size.

4.2 Multiple Models

We carried out experiments in situations where there was ambiguity about which LLM generated the text. These experiments involved three distinct LLMs: ChatGPT, Yi, and OpenChat. ChatGPT is a widely-used proprietary LLM with over 175 billion parameters. In contrast, Yi and OpenChat are mid-size and small-size open-source LLMs with 7 billion and 34 billion parameters respectively. We used various combinations of these LLMs to train BART, Baseline, and SimLLM models, and then evaluated their performance on a separate LLM. This was divided into two groups, as shown in Table 4. In the first group, the testing LLM was not included in the training LLM(s). In the second group, the testing LLM was also one of the training LLM(s).

In the first group, when tested on a different LLM, BART significantly reduces performance. In contrast, both Baseline and SimLLM achieve superior performance, particularly when trained using multiple models, with accuracy exceeding 81%. SimLLM performs competitively with the Baseline model in most scenarios. In the second group, when the model used for testing is among those used for training, SimLLM outperforms the Baseline.

4.3 Rigorous Scenarios

We conducted experiments across various scenarios using the ChatGPT model, while other models produced similar results. When faced with an unfamiliar prompt conveying a similar meaning, we adopted the prompt utilized in the BART-based approach (Zhu et al., 2023): "*Revise the following text: <sentence>.*" Conversely, for unknown prompts conveying opposite meanings, we employed the prompt: "*Rewrite the text with the opposite meaning: <sentence>.*" In cases where the temperature was unknown, we adhered to another common temperature setting of the ChatGPT model, which is 0.7. In scenarios involving unknown text, where training was conducted on news articles from the XSum dataset, we evaluated performance on academic text sourced from the SQuAD dataset (Rajpurkar et al., 2016). This dataset consists of sentences extracted from Wikipedia. We also used ChatGPT for attacking by paraphrasing with the prompt: "*Paraphrase the following text: <sentence>.*" Table 5 presents the corresponding results.

Similar and opposite texts significantly affect BART, especially the latter. Temperature changes, unknown texts, and paraphrase attacks impact both BART and Baseline. In all scenarios, SimLLM inherits characteristics from both Baseline and BART, maintaining stable performance under a variety of rigorous conditions.

4.4 Run Time

We estimated the running time of SimLLM as shown in Table 6. Specifically, we conducted experiments on approximately 1,000 words (40 sentences of human and ChatGPT-generated text). Both BART and SimLLM use ChatGPT for generating these texts. The completion time for ChatGPT was 33.34 seconds. The running times for SimLLM and existing methods are reported below. The results show that both BART (Zhu et al., 2023) and SimLLM are significantly affected by the time taken for ChatGPT generation, yet they remain faster than DetectGPT.

Scenario	Train	Test	BART	Baseline	SimLLM
Test $\notin$ Train	ChatGPT	Yi	0.709	**0.858**	**0.858**
		OpenChat	0.706	**0.806**	0.796
	Yi	ChatGPT	0.754	**0.823**	0.810
		OpenChat	0.760	**0.821**	0.792
	OpenChat	ChatGPT	0.711	**0.786**	0.764
		Yi	0.695	**0.817**	0.758
	ChatGPT and Yi	OpenChat	0.727	0.819	**0.823**
	ChatGPT and OpenChat	Yi	0.710	**0.862**	0.843
	Yi and OpenChat	ChatGPT	0.735	**0.823**	0.819
Test $\in$ Train	ChatGPT and Yi	ChatGPT	0.793	0.827	**0.903**
		Yi	0.790	0.870	**0.923**
	ChatGPT and OpenChat	ChatGPT	0.777	0.836	**0.878**
		OpenChat	0.793	0.867	**0.903**
	Yi and OpenChat	Yi	0.793	0.866	**0.902**
		OpenChat	0.817	0.875	**0.895**
	ChatGPT, Yi, and OpenChat	ChatGPT	0.769	0.841	**0.857**
		Yi	0.767	0.874	**0.888**
		OpenChat	0.776	0.873	**0.881**

Table 4: Detecting generated text through training on multiple large language models.

Scenario	BART	Baseline	SimLLM
Similar	0.733	0.858	**0.869**
Opposite	0.544	0.844	**0.845**
Temperature	0.789	0.796	**0.871**
Unknown text	0.720	0.790	**0.884**
Paraphase	0.820	0.816	**0.901**

Table 5: Detecting generated text across various scenarios, including text with similar or opposite meanings produced from unfamiliar prompts, text generated with varying temperature settings, text originating from different fields, and text modified by paraphrase.

Method	Generate	Detect	Total
RoB-base	0	0.02s	0.02s
RoB-large	0	0.03s	0.03s
log p(x)	0	0.77s	0.77s
Rank	0	0.84s	0.84s
LogRank	0	0.84s	0.84s
Entropy	0	0.83s	0.83s
DetectGPT	0	3m10.44s	3m10.44s
BART	33.34s	0.09s	33.43s
Baseline	0	0.02s	0.02s
SimLLM	33.34	0.33s	33.67s

Table 6: Run time for detecting approximately 1,000 words of human-written and ChatGPT-generated texts.

Metric	Mean(H)	Var(H)	Mean(M)	Var(M)
BLEU	0.918	0.132	0.990	0.039
ROUGE	0.909	0.113	0.989	0.041
BART	-0.679	0.273	-0.367	0.172

Table 7: The similarity between the input text and its generation. The input text includes both human-written (H) and machine-generated (M) sentences by ChatGPT.

4.5 Discussion

Similarity We observe the similarity between the input text and its generated counterpart. This similarity is calculated across the entire test set, where the text is generated by ChatGPT. We use three common metrics—BLEU, ROUGE, and BART—to calculate the similarity, as shown in Table 7. The results indicate that the similarity of human text tends to be lower than that of machine-generated text. Among the three metrics, BART estimates similarity based on the entire sentence and the meanings of words. It provides a clearer measure of similarity compared to BLEU and ROUGE, which rely solely on word n-gram matching.

Harmful Text Evaluation We have focused on two primary categories of harmful generated text, each of which contains multiple words that overlap with the original text. The first retains the orig-

inal meaning, possibly manipulating review systems or avoiding spam detection. The second alters the original meaning, spreading disinformation. Future studies will evaluate the effects of harmful text on actual systems and how SimLLM mitigates it.

5 Conclusion

This paper presents a novel method, named SimLLM, designed to identify sentences generated by large language models. Specifically, we augment the original input sentence by integrating re-generated alternatives from candidate large language models. Subsequently, this augmented data is input into a classifier to ascertain the origin of the text, whether it is human-generated or from a large language model. Experimental results from diverse large language models demonstrate the superior performance of SimLLM compared to existing methods across various scenarios.

Acknowledgments

We would like to thank you very much for the anonymous reviewers and area chairs to provide useful comments.

Limitations

Candidate Selection Selecting suitable large language model candidates for SimLLM is crucial. Given the widespread use of major large language models, particularly ChatGPT, it should be considered a prime candidate for SimLLM.

Adaptive Attack This research focuses primarily on cases where ordinary users are unaware of the detector's existence or advanced users who try to mimic human text to evade the detector through paraphrasing attacks. In subsequent steps, we will address advanced attackers who persistently modify texts until they deceive the detector.

Granularity SimLLM is designed to detect text generated by large language models at the sentence level. We are currently exploring methods to expand SimLLM to handle long text in the next stage.

Running Time SimLLM is affected by the time taken for LLM generation. As newer LLM models, such as GPT-4o mini, tend to run faster, SimLLM can leverage these advancements for practical applicability.

References

Guangsheng Bao, Yanbin Zhao, Zhiyang Teng, Linyi Yang, and Yue Zhang. 2024. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature. In *Proceedings of the 12nd International Conference on Learning Representations (ICLR)*.

Amrita Bhattacharjee and Huan Liu. 2023. Fighting fire with fire: Can chatgpt detect ai-generated text? In *The ACM Special Interest Group on Knowledge Discovery in Data Explorations (SIGKDD)*, pages 14–21.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*, pages 111–116.

Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. Spotting llms with binoculars: Zero-shot detection of machine-generated text. In *Proceedings of the 41st International Conference on Machine Learning*.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang. 2023. Mgtbench: Benchmarking machine-generated text detection. In *Preprint arXiv:2303.14822*.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-yi Ho. 2023. Radar: Robust ai-text detection via adversarial learning. In *Proceedings of the 36th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 15077–15095.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 17061–17084.

Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki. 2024. Outfox: Llm-generated essay detection through in-context learning with adversarially generated examples. In *Proceedings of the 38th Conference on Artificial Intelligence (AAAI)*, pages 21258–21266.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. Mage: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 36–53.

Xiaoming Liu, Zhaohan Zhang, Yichen Wang, Hang Pu, Yu Lan, and Chao Shen. 2023. Coco:

Coherence-enhanced machine-generated text detection under data limitation with contrastive learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 16167–16188.

Chengzhi Mao, Carl Vondrick, Hao Wang, and Junfeng Yang. 2024. Raidar: generative ai detection via rewriting. In *Proceedings of the 12nd International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 24950–24962.

Shashi Narayan, Shay Cohen, and Maria Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Abel Salinas and Fred Morstatter. 2024. The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. In *Findings of the Association for Computational Linguistics (ACL)*.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. 2019. Release strategies and the social impacts of language models. In *Preprint arXiv:1908.09203*.

Jinyan Su, Terry Zhuo, Di Wang, and Preslav Nakov. 2023. Detectllm: Leveraging log rank information for zero-shot detection of machine-generated text. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 12395–12412.

Yuchuan Tian, Hanting Chen, Xutao Wang, Zheyuan Bai, Qinghua Zhang, Ruifeng Li, Chao Xu, and Yunhe Wang. 2024. Multiscale positive-unlabeled detection of ai-generated texts. In *Proceedings of the 12th Annual ACM International Conference on Learning Representations (ICLR)*.

Eduard Tulchinskii, Kristian Kuznetsov, Laida Kushnareva, Daniil Cherniavskii, Sergey Nikolenko, Evgeny Burnaev, Serguei Barannikov, and Irina Piontkovskaya. 2023. Intrinsic dimension estimation for robust detection of ai-generated texts. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 39257–39276.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. 2024. Ghostbuster: Detecting text ghostwritten by large language models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1702–1717.

Pengyu Wang, Linyang Li, Ke Ren, Botian Jiang, Dong Zhang, and Xipeng Qiu. 2023. Seqxgpt: Sentence-level ai-generated text detection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1144–1156.

Rongsheng Wang, Haoming Chen, Ruizhe Zhou, Han Ma, Yaofei Duan, Yanlan Kang, Songhua Yang, Baoyu Fan, and Tao Tan. 2024. Llm-detector: Improving ai-generated chinese text detection with open-source llm instruction tuning. In *Preprint arXiv:2402.01158*.

Kangxi Wu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. Llmdet: A third party large language models generated text detection tool. In *Findings of the Association for Computational Linguistics (EMNLP)*, pages 2113–2133.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. In *Proceedings of the 34th Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 27263–27277.

Zhongping Zhang, Wenda Qin, and Bryan A Plummer. 2024. Machine-generated text localization. In *Findings of the Association for Computational Linguistics (ACL)*, pages 8357–8371.

Biru Zhu, Lifan Yuan, Ganqu Cui, Yangyi Chen, Chong Fu, Bingxiang He, Yangdong Deng, Zhiyuan Liu, Maosong Sun, and Ming Gu. 2023. Beat llms at their own game: Zero-shot llm-generated text detection via querying chatgpt. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7470–7483.

A Comparison with Existing Datasets

There are two key differences between the proposed SimLLM (XSum) dataset and existing datasets:

1. **Granularity**: SimLLM (XSum) operates at the sentence level, whereas existing datasets are at the document level.

2. **Similarity**: SimLLM (XSum) exhibits a higher similarity between human and machine-generated text compared to other datasets.

We demonstrate these differences by comparing SimLLM (XSum) with MGTBench (He et al.,

Dataset	Granularity	Length(Words)	BLEU	Overlap ratio
SimLLM (XSum)	Sentence	24.86	0.918	91.6%
MGTBench (Essay)	Document	752.47	0.345	24.8%
MGTBench (Writing Prompts)	Document	645.59	0.446	33.5%
MGTBench (Reuters)	Document	564.64	0.453	34.4%

Table 8: Comparison between SimLLM and MGTBench datasets across various domains.

"Revise" prompt	**"Proofread" prompt**
"Revised" text from the original text: "*Chris Maguire **of** Oxford United **took a left**-footed shot from the **center** of the box, **aiming for** the bottom left corner.*"	"Proofread" from the original text: "*Chris Maguire (Oxford United) **took a left**-footed shot from the **center** of the box, **finding** the bottom left corner.*"
"Re-revised" text from the "revised" text: "*Chris Maguire **, a player from** Oxford United, **skillfully executed** a left-footed shot from the center of the box, **with the intention of targeting** the bottom left corner.*"	"Re-proofread" text from the "proofread" text: "*Chris Maguire (Oxford United) took a left-footed shot from the center of the box, finding the bottom left corner.*"

Figure 6: Exploring variations in outputs generated by large language models between "Revise" and "Proofread" prompts. The original text is "*Chris Maguire (Oxford United) left footed shot from the centre of the box to the bottom left corner*." Modifications in the outputs are emphasized with underlining and **bold**.

Prompt	**Mean(H)**	**Var(H)**	**Mean(M)**	**Var(M)**
Revise	0.605	0.160	0.742	0.139
Proofread	0.918	0.132	0.990	0.039

Table 9: The similarity between the input text and its generation under the "Revise" and "Proofread" prompts. The input text consists of both human-written (H) and machine-generated (M) sentences by ChatGPT.

Method	**Training**	**Testing**	**ROC**
BART	Revise	Proofread	0.507
BART	Proofread	Revise	0.733
BART	Revise	Revise	0.654
BART	Proofread	Proofread	0.842
SimLLM	Revise	Proofread	0.779
SimLLM	Proofread	Revise	0.869
SimLLM	Revise	Revise	0.892
SimLLM	Proofread	Proofread	0.961

Table 10: Detecting generated text using "Revise" and "Proofread" prompts.

2023), noting that other datasets (Verma et al., 2024; Zhang et al., 2024; Li et al., 2024) show similar trends. Specifically, we compared the granularity and similarity across all domains of the MGTBench dataset as shown in Table 8. Granularity is measured by the average length of the text, while similarity is assessed using BLEU scores and the overlap ratio of words between human and LLM-generated text. For granularity, the average length of texts in MGTBench is significantly longer than in SimLLM (XSum). In terms of similarity, although MGTBench attempts to generate text on the same topic or headline, the similarity in MGTBench remains significantly lower than in SimLLM (XSum).

B Comparison between the Prompts "Revise" and "Proofread"

We observe that the "Revise" prompt (Zhu et al., 2023) tends to rewrite even well-constructed text. We randomly selected an original sentence, "*Chris Maguire (Oxford United) left footed shot from the centre of the box to the bottom left corner*," and used the "Revise" and our "Proofread" prompts to generate revised, re-revised, proofread, and re-proofread texts, highlighting the changes in the output text from the input text as shown in Figure 6.

Both the "proofread" and "revised" texts were improved from the original by splitting long sentences with commas or using more reader-friendly words. However, while the "Proofread" prompt keeps the "re-proofread" text intact, the "Revise" prompt makes "re-revised" text with further alterations from the already well-constructed "revised" text. This observation aligns with the BLEU scores for as shown in Table 9, which are 0.990 and 0.742 for the "Proofread" and "Revise"

Dataset	Domain	BART	Baseline	SimLLM
MGTBench	Essay	0.753	0.777	0.866
GhostBuster	Creative Writing	0.788	0.776	0.836
MGTL	Goodnews	0.777	0.699	0.837
MAGE	Review (Yelp)	0.807	0.846	0.877

Table 11: Detecting generated text on existing datasets.

Model	Perplexity	Binoculars	LLM-Detector	MPU-Roberta	SimLLM
ChatGPT	0.453	0.403	0.541	0.649	**0.916**
GPT-4o	0.481	0.433	0.532	0.649	**0.816**
Yi	0.461	0.404	0.554	0.654	**0.947**
OpenChat	0.483	0.412	0.491	0.595	**0.954**
Gemini	0.466	0.437	0.530	0.612	**0.859**
LLaMa	0.433	0.381	0.543	0.750	**0.883**
Phi	0.581	0.386	0.491	0.487	**0.937**
Mixtral	0.601	0.381	0.538	0.640	**0.837**
Qwen	0.505	0.428	0.532	0.668	**0.900**
OLMo	0.527	0.450	0.515	0.621	**0.895**
WizardLM	0.469	0.416	0.558	0.675	**0.856**
Vicuna	0.448	0.381	0.541	0.731	**0.866**
Average	0.492	0.409	0.531	0.644	**0.889**

Table 12: Detecting generated text with other detectors.

prompts, respectively. This gap can explain the performance difference since both SimLLM and BART (Zhu et al., 2023) operate on the hypothesis that small changes should be made to the re-generated text. We also conducted experiments using these prompts in various scenarios for detecting the text generated by ChatGPT as shown in Table 10, and the results show that the training with "Proofread" is stable across different scenarios.

C Evaluation on Existing Datasets

SimLLM is designed to detect generated text at the sentence level, making it unsuitable for direct use on datasets like MGTBench (He et al., 2023), GhostBuster (Verma et al., 2024), MGTL (Zhang et al., 2024), and MAGE (Li et al., 2024). To adapt to this scenario, we randomly selected 5,000 human sentences from these datasets. For each dataset, we randomly chose non-duplicated domains, and the generated sentences were created using ChatGPT. The results, shown in Table 11 for the three main detectors (BART (Zhu et al., 2023), Baseline, and SimLLM), demonstrate that SimLLM outperforms both the BART and Baseline methods across various datasets and domains.

D Evaluation with Other Detectors

We conducted the same experiments using other existing methods including Perplexity, Binoculars (Hans et al., 2024), LLM-Detector (Wang et al., 2024), and MPU-Roberta (Tian et al., 2024) as shown in the Table 12. For Perplexity, we used GPT-XL to calculate the score. The results demonstrate that existing methods fail to detect the LLM-generated text effectively.

We evaluate the types of changes the LLM makes by removing duplicated words between the human and machine text generated by ChatGPT across the entire dataset and categorizing the remaining words into three groups. These groups represent potential features for a simple rule-based approach to detect LLM text based on edits between the input text and the re-generated text:

1. **Confusable (58.2%)**: This group contains words that appear in both human and machine text. A large proportion of these words are stop words (56.6%) and punctuation marks (23.4%).

2. **Non-reusable (19.1%)**: These words appear only once in the dataset and thus cannot be reused for classification.

3. **Distinguishable (22.7%)**: This group consists of words that appear more than once exclusively in human or machine text, often due to normalization (e.g., "*Mr*" to "*Mr.*" or "*Prof*" to "*Prof.*") or standardization (e.g., "*organisation*" to "*organization*" or "*behaviour*" to "*behavior*").

The statistics demonstrate that these edits are insufficient to reliably distinguish between human-written and LLM-generated text.