MarkLLM: An Open-Source Toolkit for LLM Watermarking

Leyi Pan[1], **Aiwei Liu**[1*], **Zhiwei He**[2], **Zitian Gao**[3], **Xuandong Zhao**[4],
Yijian Lu[5], **Bingling Zhou**[2], **Shuliang Liu**[6,7], **Xuming Hu**[6,7], **Lijie Wen**[1†],
Irwin King[5], **Philip S. Yu**[8]
[1]Tsinghua University [2]Shanghai Jiao Tong University [3]The University of Sydney
[4]UC Santa Barbara [5]The Chinese University of Hong Kong
[6]The Hong Kong University of Science and Technology (Guangzhou)
[7]The Hong Kong University of Science and Technology [8]University of Illinois at Chicago
panly24@mails.tsinghua.edu.cn, liuaw20@mails.tsinghua.edu.cn, xuminghu@hkust-gz.edu.cn
wenlj@tsinghua.edu.cn, king@cuhk.edu.hk, psyu@uic.edu

Abstract

Watermarking for Large Language Models (LLMs), which embeds imperceptible yet algorithmically detectable signals in model outputs to identify LLM-generated text, has become crucial in mitigating the potential misuse of LLMs. However, the abundance of LLM watermarking algorithms, their intricate mechanisms, and the complex evaluation procedures and perspectives pose challenges for researchers and the community to easily understand, implement and evaluate the latest advancements. To address these issues, we introduce MarkLLM, an open-source toolkit for LLM watermarking. MarkLLM offers a unified and extensible framework for implementing LLM watermarking algorithms, while providing user-friendly interfaces to ensure ease of access. Furthermore, it enhances understanding by supporting automatic visualization of the underlying mechanisms of these algorithms. For evaluation, MarkLLM offers a comprehensive suite of 12 tools spanning three perspectives, along with two types of automated evaluation pipelines. Through MarkLLM, we aim to support researchers while improving the comprehension and involvement of the general public in LLM watermarking technology, fostering consensus and driving further advancements in research and application. Our code is available at https://github.com/THU-BPM/MarkLLM.

1 Introduction

The emergence of Large Language Models (LLMs) like ChatGPT (OpenAI, 2022), GPT-4 (OpenAI, 2023), and LLaMA (Touvron et al., 2023) has significantly enhanced various tasks, including information retrieval (Zhu et al., 2023), content comprehension (Xiao et al., 2023), and creative writing (Gómez-Rodríguez and Williams, 2023). However, in the digital era, the remarkable proficiency of LLMs in generating high-quality text has also brought several issues to the forefront, including individuals impersonation (Salewski et al., 2023), academic paper ghostwriting (Vasilatos et al., 2023), and the proliferation of LLM-generated fake news (Megías et al., 2021). These issues highlight the urgent need for reliable methods to distinguish between human and LLM-generated content, particularly to prevent the spread of misinformation and ensure the authenticity of digital communication. In the light of this, LLM watermarking technology (Kirchenbauer et al., 2023; Aaronson and Kirchner, 2022; Liu et al., 2024e; Pan et al., 2024; Liu et al., 2024a) has been developed as a promising solution. By incorporating distinct features during the text generation process, LLM outputs can be uniquely identified using specially designed detectors.

As a developing technology, LLM watermarking urgently requires consensus and support from both within and outside the field. However, due to the proliferation of watermarking algorithms, their relatively complex mechanisms, the diversity of evaluation perspectives and metrics, as well as the intricate procedure of evaluation process, significant efforts are required by both researchers and the general public to easily experiment with, comprehend, and evaluate watermarking algorithms.

To bridge this gap, we introduce MarkLLM, an open-source toolkit for LLM watermarking. Figure 1 overviews the architecture of MarkLLM. Our main contributions are summarized as follows:

1) From a Functional Perspective:

- 🔧 Implementation framework: MarkLLM offers a unified and extensible framework for implementing LLM watermarking algorithms, currently supporting nine specific algorithms from two key families: KGW (Kirchenbauer et al., 2023) and Christ (Christ et al., 2024) family.

[*]Project Leader
[†]Corresponding Author

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing
System Demonstrations, pages 61–71
November 12-16, 2024 ©2024 Association for Computational Linguistics

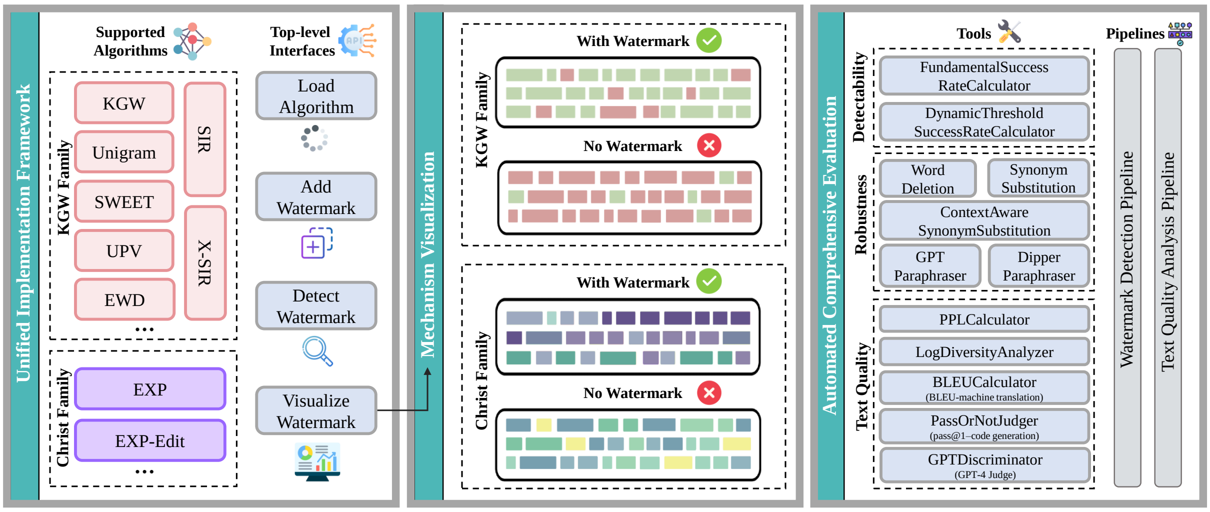


Figure 1: Architecture overview of MARKLLM.

- Unified top-calling interfaces: MARKLLM provides consistent, user-friendly interfaces for loading algorithms, producing watermarked text generated by LLMs, conducting detection processes, and gathering data necessary for visualization.

- Visualization solutions: Custom visualization solutions are provided for both major watermarking algorithm families, enabling users to visualize the mechanisms of different algorithms under various configurations with real-world examples.

- Evaluation module: The toolkit includes 12 evaluation tools that address three critical perspectives: detectability, robustness, and impact on text quality. It also features two types of automated evaluation pipelines that support user customization of datasets, models, evaluation metrics and attacks, facilitating flexible and comprehensive assessments.

2) From a Design Perspective: MARKLLM is designed with a modular, loosely coupled architecture, ensuring its scalability and flexibility. This design choice facilitates the integration of new algorithms, the addition of innovative visualization techniques, and the extension of the evaluation toolkit by future developers.

3) From an Experimental Perspective: Utilizing MARKLLM as a research tool, we perform in-depth evaluations of the performances of the nine included algorithms, offering substantial insights and benchmarks that will be invaluable for ongoing and future research in LLM watermarking.

4) From an Ecosystem Perspective: MARKLLM provides a comprehensive set of resources, including an installable Python package (a GitHub repository and a pip package) with detailed installation and usage instructions, and an online Jupyter notebook demo hosted on Google Colab. Since its initial release, MARKLLM has garnered significant attention from researchers and developers, who have actively engaged with the project through stars, forks, issues, and pull requests, fostering continuous development and improvement. Figure 2 depicts the evolution of the MARKLLM ecosystem since its initial release. Due to the scope of this paper, we focus on presenting the core functionalities of MARKLLM, while acknowledging the broader ecosystem and community contributions that have emerged around the project.

2 Background

2.1 LLM Watermarking Algorithms

LLM watermarking methods can be classified into the KGW Family and the Christ Family. The KGW Family modifies logits to generate watermarked output, while the Christ Family alters the sampling process.

The KGW method (Kirchenbauer et al., 2023) partitions the vocabulary into green and red lists, adding bias to green list tokens during generation. A statistical metric based on the green word proportion is used for detection. Various modifications have been proposed to improve text quality (Hu et al., 2024; Wu et al., 2023; Takezawa et al., 2023), information capacity (Wang et al., 2024; Yoo et al., 2024; Fernandez et al., 2023), ro-

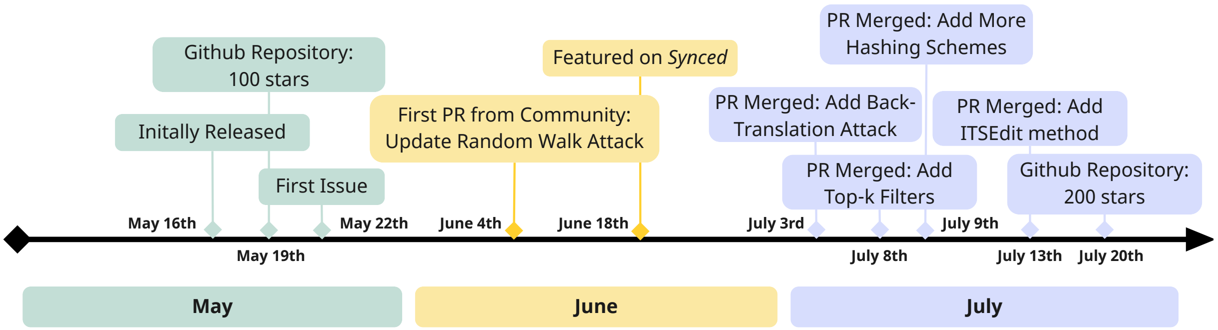

PR Merged: Add More Hashing Schemes
Github Repository: 100 stars
Featured on *Synced*
PR Merged: Add Back-Translation Attack
PR Merged: Add ITSEdit method
Initally Released
First PR from Community: Update Random Walk Attack
PR Merged: Add Top-k Filters
Github Repository: 200 stars
First Issue
May 16th
May 22th
June 4th
June 18th
July 3rd
July 9th
May 19th
July 8th
July 13th
July 20th
May
June
July

Figure 2: Timeline of the MarkLLM ecosystem since its initial release.

bustness (Zhao et al., 2024; Liu et al., 2024c; Ren et al., 2024; He et al., 2024; Zhang et al., 2024), adapt to low-entropy scenarios (Lee et al., 2024; Lu et al., 2024), and enable public detection (Liu et al., 2024b; Fairoze et al., 2023).

Christ et al. (2024) used pseudo-random numbers to guide sampling in a binary LLM. Aaronson and Kirchner (2022) developed an algorithm for real-world LLMs using EXP-sampling, where a pseudo-random sequence is generated based on previous tokens to select the next token. Watermark detection measures the correlation between the text and the sequence. Kuditipudi et al. (2024) suggested using edit distance for robust detection.

2.2 Evaluation Perspectives

Evaluating watermarking algorithms involves multiple dimensions (Liu et al., 2024d):

1) Watermark Detectability: The ability to discern watermarked text from natural content.

2) Robustness Against Tampering Attacks: The watermark should withstand minor modifications and remain detectable.

3) Impact on Text Quality: Watermarking may affect the quality of generated text. This impact can be measured by perplexity, diversity, and performance in downstream tasks.

3 MarkLLM

3.1 Unified Implementation Framework

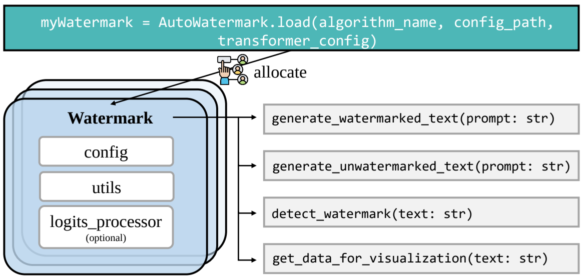
Many watermarking algorithms have been proposed, but their implementations lack standardization, leading to several issues:

1) Lack of Standardization in Class Design: Insufficiently standardized class designs make optimizing or extending existing methods difficult.

2) Lack of Uniformity in Top-Level Calling Interfaces: Inconsistent interfaces make batch processing and replicating different algorithms cumbersome and labor-intensive.

3) Code Standard Issues: Modifying settings across multiple code segments, lack of consistent documentation, hard-coded values, and inconsistent error handling complicate customization, effective use, adaptability, and debugging efforts.

Our toolkit offers a unified implementation framework that enables convenient invocation of various state-of-the-art algorithms under flexible configurations. Figure 3 demonstrates the design of this framework.



```
myWatermark = AutoWatermark.load(algorithm_name, config_path, transformer_config)
```

```
generate_watermarked_text(prompt: str)
generate_unwatermarked_text(prompt: str)
detect_watermark(text: str)
get_data_for_visualization(text: str)
```

Figure 3: Unified implementation framework of LLM watermarking algorithms.

AutoWatermark. This class is responsible for algorithm allocation. Its *.load()* method locates the corresponding algorithm class using *algorithm_name* and accesses its configuration[1] for initialization via *config_path*.

Watermark. Each watermarking algorithm has its own class, collectively referred to as the Watermark class. This class includes three data members: *config*, *utils*, and *logits_processor* (only for algorithms in the KGW Family). *config* holds algorithm parameters, while *utils* comprises helper functions and variables. For algorithms within the KGW

[1] For each watermarking algorithm, all user-modifiable parameters are consolidated into a dedicated configuration file, facilitating easy modifications.

family, *logits_processor* is designed to manipulate logits and is integrated into *model.generate()* for processing during execution.

Top-level Interfaces. Each algorithm has four top-level interfaces for generating watermarked text, generating unwatermarked text, detecting watermarks, and obtaining data for visualization (detailed in Section 3.2). The framework's distributive design using an AutoWatermark class allows developers to easily add interfaces to any algorithm class without impacting others.

3.2 Mechanism Visualization

To improve understanding of the mechanisms used by different watermark algorithms, we have developed a visualization module that provides tailored visualization solutions for the two algorithm families.

3.2.1 Visualization Solutions

KGW Family. As detailed in Section 2.1, KGW family algorithms manipulate LLM output logits to prefer green tokens over red ones and employ statistical methods for detection. Our visualization technique clearly highlights red and green tokens in the text, offering insights into the token-level detection results.

Christ Family. Algorithms within Christ family involves guiding each token selection using a pseudo-random sequence and detect watermarks by calculating the correlation between the sequence and the text. To visualize this mechanism, we use a color gradient to represent the alignment value of each token and the pseudo-random sequence, where darker shades indicate stronger alignment.

3.2.2 Architecture Design

This section offers a detailed description of the architectural frameworks essential for the effective implementation of the aforementioned visualization strategies. Figure 4 demonstrates the implementation framework of mechanism visualization.

get_data_for_visualization: This interface, defined for each algorithm, returns a VisualizationData object containing *decoded_tokens* and *highlight_value*. For the KGW family, *highlight_value* is one-hot, differentiating red and green tokens; for the Christ family, it represents a continuous correlation value.

Visualizer: It initializes with a VisualizationData object and performs visualization via the *.visualize()* method, with subclasses overriding approach to implement specific visualizations.

DiscreetVisualizer: Tailored for KGW family algorithms, it uses red/green highlight values to color-code text based on values.

ContinuousVisualizer: Tailored for Christ family algorithms, it highlights tokens using a [0,1] color scale based on their alignment with pseudo-random numbers.

Flexible Visualization Settings: Our Visualizer supports multiple configurable options for tailored visualizations, including ColorScheme, FontSettings, PageLayoutSettings, and LegendSetting, allowing for extensive customization.

3.2.3 Visualization Result

KGW Family. The leftmost part of Figure 4 shows that in the text with watermarks, there is a relatively high proportion of green tokens. The z-score, a statistical measure, is defined as:

$$z = \frac{|s|_G - \gamma T}{\sqrt{T\gamma(1-\gamma)}}$$

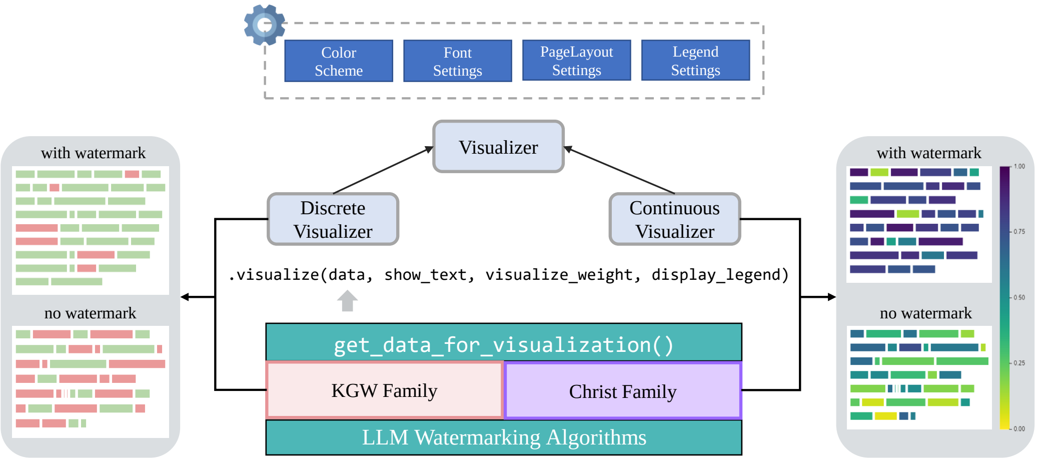
where $|s|_G$ is the number of green tokens, T is the total number of tokens, and γ is the proportion of the green token list in partitioning (0.5 in this case). The z-score for 'text with watermark' is notably higher than that for 'text without watermark'. Setting a reasonable z-score threshold can effectively distinguish between the two.

Christ Family. As depicted in the rightmost part of Figure 4, it is noticeable that tokens within text containing watermarks generally exhibit darker hues compared to those without, indicating a higher influence of the sequence during the generation process on the former.

3.3 Automated Comprehensive Evaluation

Evaluating an LLM watermarking algorithm is complex, as it involves considering multiple perspectives, such as watermark detectability, robustness against tampering, and impact on text quality (see Section 2.2). Each perspective may require different metrics, attack scenarios, and tasks. The evaluation process typically includes steps like model and dataset selection, watermarked text generation, post-processing, watermark detection, text tampering, and metric computation.

To simplify the evaluation process, MARKLLM offers twelve user-friendly tools, including metric calculators and attackers, covering the three



```
.visualize(data, show_text, visualize_weight, display_legend)
get_data_for_visualization()
```

Figure 4: Implementation framework of mechanism visualization.

Table 1: Evaluation Tools in MarkLLM.

Perspective	Tools
Detectability	FundamentalSuccessRateCalculator DynamicThresholdSuccessRateCalculator
Robustness	WordDeletion SynonymSubstitution ContextAwareSynonymSubstitution GPTParaphraser DipperParaphraser
Text Quality	PPLCaluclator LogDiversityAnalyzer BLEUCalculator PassOrNotJudger GPTDiscriminator

main evaluation perspectives. Additionally, MARKLLM provides two types of customizable automated demo pipelines, allowing for easy configuration and use.

MARKLLM provides a comprehensive set of tools for evaluating LLM watermarking algorithms, as summarized in Table 1. These tools cover detectability, including success rate calculators with fixed and dynamic thresholds; robustness, featuring word-level and document-level text tampering attacks using WordNet (Miller, 1995), BERT (Devlin et al., 2018), OpenAI API, and the Dipper model (Krishna et al., 2023); and text quality, assessing fluency, variability, and performance on downstream tasks using perplexity, diversity, BLEU, pass-or-not judger, and GPT discriminator with GPT-4 (OpenAI, 2023).

Evaluation Pipelines. MARKLLM provides two evaluation pipelines: one for assessing watermark detectability with and without attacks, and another for analyzing the impact of these algorithms on text quality.

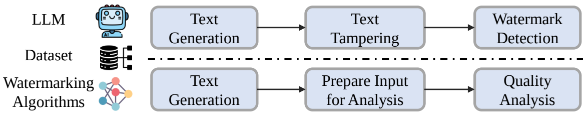


Figure 5: The standardized process of evaluation pipelines, the upper for watermark detection pipeline, and the lower for text quality analysis pipeline.

The upper part of Figure 5 shows the standardized process of watermark detection. We have implemented two pipelines: **WMDetect** for watermarked text detection and **UWMDetect** for unwatermarked text detection. The lower part of Figure 5 illustrates the unified process of text quality analysis. Pairs of watermarked and unwatermarked texts are generated and fed into a designated text quality analyzer to produce detailed analysis and comparison results. We have implemented three pipelines for different evaluation scenarios:

DirectQual. This pipeline directly compares the characteristics of watermarked and unwatermarked texts using metrics such as perplexity (PPL) and log diversity.

RefQual. This pipeline evaluates text quality by comparing both watermarked and unwatermarked texts with a common reference text. It is ideal for scenarios that require specific downstream tasks, such as machine translation and code generation.

ExDisQual. This pipeline employs an external judger, such as GPT-4 (OpenAI, 2023), to assess the quality of both watermarked and unwatermarked texts based on user-provided task descriptions. This method is valuable for advanced, AI-based analysis of the subtle effects of watermarking.

4 User Examples

The following code snippets demonstrate examples of how to use MarkLLM in one's project. For more real cases, please see the demo video.

4.1 Watermarking Algorithm Invocation

```
# Load algorithm
myWatermark = AutoWatermark.load('KGW'
    , 'config/KGW.json',
    transformers_config)
# Generate watermarked text
watermarked_text = myWatermark.
    generate_watermarked_text(prompt)
# Detect watermark
detect_result = myWatermark.
    detect_watermark(watermarked_text)
```

4.2 Mechanism Visualization

```
# Get data for visualization
watermarked_data = myWatermark.
    get_data_for_visualization(
    watermarked_text)
# Init visualizer
visualizer = DiscreetVisualizer(
    ColorSchemeForDiscreetVisualization
    (), FontSettings(),
    PageLayoutSettings(),
    DiscreetLegendSettings())
# Visualize
watermarked_img = visualizer.visualize
    (watermarked_data)
```

4.3 Evaluation Pipelines Invocation

```
# Dataset
my_dataset = C4Dataset('dataset/c4/
    processed_c4.json')
# WMDetect
pipeline1 =
    WatermarkedTextDetectionPipeline(
    my_dataset)
# UWMDetect
pipeline2 =
    UnWatermarkedTextDetectionPipeline(
    dataset=my_dataset)
# Init calculator
calculator =
    DynamicThresholdSuccessRateCalculator
    (labels=['TPR', 'F1'], rule='best')
# Calculate success rate
print(calculator.calculate(pipeline1.
    evaluate(my_watermark), pipeline2.
    evaluate(my_watermark)))
```

5 Experiment

Using MarkLLM as a research tool, we conduct evaluations on nine watermarking algorithms, assessing their detectability, robustness, and impact on text quality. Our experiments demonstrate that MarkLLM can reproduce the results of previous experiments with low cost through simple scripts. For details on the experimental setup and the obtained results, please refer to Appendix A.

6 Conclusion

MarkLLM is a comprehensive open-source toolkit for LLM watermarking. It allows users to easily try various state-of-the-art algorithms with flexible configurations to watermark their own text and conduct detection, and provides clear visualizations to gain insights into the underlying mechanisms. The inclusion of convenient evaluation tools and customizable evaluation pipelines enables automatic and thorough assessments from various perspectives. As LLM watermarking evolves, MarkLLM aims to be a collaborative platform that grows with the research community. By providing a solid foundation and inviting contributions, we aim to foster a vibrant ecosystem where researchers and developers can work together to advance the state-of-the-art in LLM watermarking technology.

Limitations

MarkLLM is a comprehensive toolkit for implementing, visualizing, and evaluating LLM watermarking algorithms. However, it currently only integrates a subset of existing methods and does not yet support some recent approaches that directly embed watermarks into model parameters during training (Xu et al., 2024; Gu et al., 2024). We anticipate future contributions to expand MarkLLM's coverage and enhance its versatility.

In terms of visualization, we have provided one tailored solution for each of the two main watermarking algorithm families. While these solutions offer valuable insights, there is room for more creative and diverse visualization designs.

Regarding evaluation, we have covered aspects such as detectability, robustness, and text quality impact. However, our current toolkit may not encompass all possible scenarios, such as spoofing attack and CWRA (He et al., 2024).

We acknowledge that MarkLLM has room for improvement. We warmly welcome developers and researchers to contribute their code and insights to help build a more comprehensive ecosystem for LLM watermarking. Through collaborative efforts, we can further advance this technology and unlock its full potential.

Acknowledgements

MarkLLM, as an open-source toolkit, has greatly benefited from the community's feedback and contributions. We extend our sincere gratitude to all users who have raised issues on GitHub, thereby helping us improve this project. Special thanks go to Hanlin Zhang, Sheng Guan, Yiming Liu, Yichen Di, and Kai Shi for their valuable pull requests, which have significantly enhanced MarkLLM's functionality. Furthermore, we are deeply appreciative of the insightful comments provided by the reviewers and area chair, which have been instrumental in refining both our paper and the project.

References

S. Aaronson and H. Kirchner. 2022. Watermarking gpt outputs. https://www.scottaaronson.com/talks/watermark.ppt.

Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 1125–1139. PMLR.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. 2023. Publicly detectable watermarking for language models. Cryptology ePrint Archive, Paper 2023/1661. https://eprint.iacr.org/2023/1661.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. *arXiv preprint arXiv:2308.00113*.

Carlos Gómez-Rodríguez and Paul Williams. 2023. A confederacy of models: A comprehensive evaluation of llms on creative writing. *arXiv preprint arXiv:2310.08433*.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. 2024. On the learnability of watermarks for language models. In *The Twelfth International Conference on Learning Representations*.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129, Bangkok, Thailand. Association for Computational Linguistics.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4890–4911, Bangkok, Thailand. Association for Computational Linguistics.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you!

Aiwei Liu, Sheng Guan, Yiming Liu, Leyi Pan, Yifei Zhang, Liancheng Fang, Lijie Wen, Philip S Yu, and Xuming Hu. 2024a. Can watermarked llms be identified by users via crafted prompts? *arXiv preprint arXiv:2410.03168*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S. Yu. 2024b. An unforgeable publicly verifiable watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024c. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024d. A survey of text watermarking in the era of large language models. *ACM Comput. Surv.* Just Accepted.

Aiwei Liu, Qiang Sheng, and Xuming Hu. 2024e. Preventing and detecting misinformation generated by large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3001–3004.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11724–11735, Bangkok, Thailand. Association for Computational Linguistics.

David Megías, Minoru Kuribayashi, Andrea Rosales, and Wojciech Mazurczyk. 2021. Dissimilar: Towards fake news detection using information hiding. In *Signal Processing and Machine Learning. In The 16th International Conference on Availability, Reliability and Security (Vienna, Austria)(ARES 2021). Association for Computing Machinery, New York, NY, USA, Article*, volume 66.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. `https://openai.com/blog/chatgpt`.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Leyi Pan, Aiwei Liu, Yijian Lu, Zitian Gao, Yichen Di, Lijie Wen, Irwin King, and Philip S Yu. 2024. Waterseeker: Efficient detection of watermarked segments in large documents. *arXiv preprint arXiv:2409.05112*.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. 2023. Mark my words: Analyzing and evaluating language model watermarks. *Preprint*, arXiv:2312.00273.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024. A robust semantics-based watermark for large language model against paraphrasing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 613–625, Mexico City, Mexico. Association for Computational Linguistics.

Leonard Salewski, Stephan Alaniz, Isabel Rio-Torto, Eric Schulz, and Zeynep Akata. 2023. In-context impersonation reveals large language models' strengths and biases. *Preprint*, arXiv:2305.14930.

Yuki Takezawa, Ryoma Sato, Han Bao, Kenta Niwa, and Makoto Yamada. 2023. Necessary and sufficient watermark for large language models. *arXiv preprint arXiv:2310.00833*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. 2024. WaterBench: Towards holistic evaluation of watermarks for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1517–1542, Bangkok, Thailand. Association for Computational Linguistics.

Christoforos Vasilatos, Manaar Alam, Talal Rahwan, Yasir Zaki, and Michail Maniatakos. 2023. Howkgpt: Investigating the detection of chatgpt-generated university student homework through context-aware perplexity analysis. *arXiv preprint arXiv:2305.18226*.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. Towards codable watermarking for injecting multi-bits information to LLMs. In *The Twelfth International Conference on Learning Representations*.

Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. 2023. Dipmark: A stealthy, efficient and resilient watermark for large language models. *arXiv preprint arXiv:2310.07710*.

Changrong Xiao, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, and Lei Xia. 2023. Evaluating reading comprehension exercises generated by llms: A showcase of chatgpt in education applications. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 610–625.

Xiaojun Xu, Yuanshun Yao, and Yang Liu. 2024. Learning to watermark llm-generated text via reinforcement learning. *Preprint*, arXiv:2403.10553.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2024. Advancing beyond identification: Multi-bit watermark for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4031–4055, Mexico City, Mexico. Association for Computational Linguistics.

Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. 2024. Watermarks in the sand: Impossibility of strong watermarking for language models. In *Forty-first International Conference on Machine Learning*.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

A Experiment Details

A.1 Experiment Settings

Dateset and Prompt. For general-purpose text generation scenarios, we utilize the C4 dataset (Raffel et al., 2020). Specifically, the first 30 tokens of texts serve as prompts for generating the subsequent 200 tokens, with the original C4 texts acting as non-watermarked examples. For specific downstream tasks, we employ the WMT16 (Bojar et al., 2016) German-English dataset for machine translation, and HumanEval (Chen et al., 2021) for code generation.

Language Model. For general-purpose text generation scenarios, we utilize Llama-7b (Touvron et al., 2023) as language model. For specific downstream tasks, we utilize NLLB-200-distilled-600M (Costa-jussà et al., 2022) for machine translation and Starcoder (Li et al., 2023) for code generation.

Metrics and Attacks. Dynamic threshold adjustment is employed to evaluate watermark detectability, with three settings provided: under a target FPR of 10%, under a target FPR of 1%, and under conditions for optimal F1 score performance. To assess robustness, we utilize all text tampering attacks listed in Table 1. For evaluating the impact on text quality, our metrics include PPL, log diversity, BLEU (for machine translation), pass@1 (for code generation), and assessments using GPT-4 Judge (Tu et al., 2024).

A.2 Results and Analysis

The results[2] in Table 2, Table 3, and Table 4 demonstrate that by using the implementations of different algorithms and the evaluation pipelines provided in MarkLLM, researchers can effectively reproduce the experimental results from previous watermarking papers. These experiments can be conducted by running simple scripts which are accessible within the Github repository under the directory *evaluation/examples/*. The execution command can be found in Listing 1, Listing 2 and Listing 3, showcasing MarkLLM's capability for easy evaluation of watermark algorithms in various scenarios.

```
python evaluation/examples/assess_detectability.py
    --algorithm KGW --labels TPR F1 --rules
    target_fpr --target_fpr 0.01

python evaluation/examples/assess_detectability.py
    --algorithm KGW --labels TPR TNR FPR FNR P R
    F1 ACC --rules best
```

Listing 1: Execution command for assessing detectability.

```
python evaluation/examples/assess_robustness.py
    --algorithm KGW --attack 'Word-D'

python evaluation/examples/assess_robustness.py
    --algorithm Unigram --attack 'Doc-P(GPT-3.5)'
```

Listing 2: Execution command for assessing robustness.

```
python evaluation/examples/assess_quality.py
    --algorithm KGW --metric PPL

python evaluation/examples/assess_quality.py
    --algorithm SIR --metric 'Log Diversity'
```

Listing 3: Execution command for assessing text quality.

B Comparison with Competitors

As LLM watermarking technology advances, frameworks dedicated to this field have emerged. WaterBench (Tu et al., 2024) and Mark My Words (Piet et al., 2023) are two prominent examples. WaterBench focuses on assessing the impact of KGW (Kirchenbauer et al., 2023), Unigram (Zhao et al., 2024), and KGW-v2 (Kirchenbauer et al., 2024) on text quality, while Mark My Words evaluates the performance of KGW, EXP (Aaronson and Kirchner, 2022), Christ (Christ et al., 2024), and EXP-Edit (Kuditipudi et al., 2024) across text quality, robustness against tampering, and number of tokens needed for detection.

While these frameworks primarily focus on benchmark construction, similar to the evaluation module in MarkLLM, MarkLLM distinguishes itself as the first comprehensive multi-functional toolkit. It offers easy-to-use evaluation tools and automated pipelines that cover the aforementioned assessment perspectives, and also provides a unified implementation framework for watermarking algorithms and visualization tools for their underlying mechanisms. This enhances its utility and versatility. The integration of these functionalities makes MarkLLM a more accessible resource, enabling convenient usage, understanding, evaluation, and selection of diverse watermarking algorithms by researchers and the broader community. This plays a crucial role in fostering consensus both within and beyond the field.

[2] (1) The evaluation results for UPV are only shown in the "best" column because its watermark detection uses direct binary classification without thresholds. (2) Current implementations of Christ family algorithms are designed for decoder-only LLMs. As machine translation mainly uses encoder-decoder models, we did not report the text quality produced by EXP and EXP-edit in machine translation.

Table 2: The evaluation results of assessing the detectability of nine algorithms supported in MarkLLM. 200 watermarked texts are generated, while 200 non-watermarked texts serve as negative examples. We furnish TPR and F1-score under dynamic threshold adjustments for 10% and 1% FPR, alongside TPR, TNR, FPR, FNR, P, R, F1, ACC at optimal performance.

Method	10%FPR		1%FPR		Best							
	TPR	F1	TPR	F1	TPR	TNR	FPR	FNR	P	R	F1	ACC
KGW	**1.000**	0.952	**1.000**	0.995	**1.000**	**1.000**	**0.000**	**0.000**	**1.000**	**1.000**	**1.000**	**1.000**
Unigram	**1.000**	**0.957**	**1.000**	**0.995**	**1.000**	**1.000**	**0.000**	**0.000**	**1.000**	**1.000**	**1.000**	**1.000**
SWEET	**1.000**	0.952	**1.000**	**0.995**	**1.000**	**1.000**	**0.000**	**0.000**	**1.000**	**1.000**	**1.000**	**1.000**
UPV	×	×	×	×	**1.000**	0.990	0.010	**0.000**	0.990	**1.000**	0.995	0.995
EWD	**1.000**	0.952	**1.000**	**0.995**	0.995	**1.000**	**0.000**	0.005	**1.000**	0.995	0.997	0.998
SIR	0.995	0.950	0.990	0.990	0.990	0.995	0.005	0.010	0.995	0.990	0.992	0.993
X-SIR	0.995	0.950	0.940	0.964	0.970	0.970	0.030	0.030	0.970	0.970	0.970	0.970
EXP	**1.000**	0.952	**1.000**	**0.995**	**1.000**	**1.000**	**0.000**	**0.000**	**1.000**	**1.000**	**1.000**	**1.000**
EXP-Edit	**1.000**	0.952	0.995	0.990	0.995	0.985	0.015	0.005	0.985	0.995	0.990	0.990

Table 3: The evaluation results of assessing the robustness of nine algorithms supported in MarkLLM. For each attack, 200 watermarked texts are generated and subsequently tampered, with an additional 200 non-watermarked texts serving as negative examples. We report the TPR and F1-score at optimal performance under each circumstance.

Method	No Attack		Word-D		Word-S		Word-S (Context)		Doc-P (GPT-3.5)		Doc-P (Dipper)	
	TPR	F1	TPR	F1	TPR	F1	TPR	F1	TPR	F1	TPR	F1
KGW	**1.000**	**1.000**	0.980	0.985	0.920	0.915	0.965	0.958	0.835	0.803	0.860	0.785
Unigram	**1.000**	**1.000**	**1.000**	**1.000**	**0.990**	**0.990**	**0.990**	**0.990**	**0.901**	**0.932**	0.875	**0.908**
SWEET	**1.000**	**1.000**	0.970	0.975	0.935	0.903	0.985	0.980	0.845	0.813	0.830	0.779
UPV	**1.000**	0.995	0.970	0.980	0.885	0.896	0.985	0.961	0.830	0.827	0.862	0.864
EWD	0.995	0.997	0.980	0.982	0.930	0.921	0.950	0.955	0.852	0.825	0.845	0.784
SIR	0.990	0.992	0.950	0.970	0.945	0.940	0.960	0.948	0.891	0.923	**0.894**	0.902
X-SIR	0.970	0.970	0.940	0.957	0.910	0.908	0.895	0.925	0.875	0.891	0.835	0.869
EXP	**1.000**	**1.000**	0.975	0.980	0.945	0.950	0.980	0.985	0.763	0.772	0.740	0.793
EXP-Edit	0.995	0.990	0.995	0.993	0.983	0.972	0.990	0.985	0.872	0.886	0.845	0.861

Table 4: The evaluation results of assessing the text quality impact of the nine algorithms supported in MarkLLM. We compared 200 watermarked texts with 200 non-watermarked texts. However, due to dataset constraints, only 100 watermarked texts were compared with 100 non-watermarked texts for code generation.

Method	Direct Analysis		Referenced Analysis		External Discriminator
	PPL(Ori.= 8.243)	Log Diversity(Ori.=8.517)	Machine Translation BLEU(Ori.=31.807)	Code Generation pass@1(Ori.= 43.0)	Machine Translation GPT-4 Judge (Wat. Win Rate)
KGW	13.551 ↑	7.989 ↓	28.242 ↓	34.0 ↓	0.31
Unigram	13.723 ↑	7.242 ↓	26.075 ↓	32.0 ↓	**0.33**
SWEET	13.747 ↑	8.086 ↓	28.242 ↓	**37.0** ↓	0.31
UPV	**10.574** ↑	7.698 ↓	28.270 ↓	**37.0** ↓	0.31
EWD	13.402 ↑	8.220 ↓	28.242 ↓	34.0 ↓	0.30
SIR	13.918 ↑	7.990 ↓	**28.830** ↓	**37.0** ↓	0.31
X-SIR	12.885 ↑	7.930 ↓	28.161 ↓	36.0 ↓	**0.33**
EXP	19.597 ↑	8.187 ↓	×	20.0 ↓	×
EXP-Edit	21.591 ↑	**9.046** ↑	×	14.0 ↓	×