Memorization: Defending vs. Attacking Language Models

1 Introduction

Language models are trained on large corpora and, to fit the long tail, end up *memorizing* individual training examples [4]. Some memorization is *desirable* — remembering facts can reduce hallucination [6] — but *undesirable* memorization of copyrighted text or personal information creates real legal and privacy risk [7, 3] and can block safe deployment. Memorization is measured with *data extraction*: if prompting a model reproduces a training example verbatim, that example was memorized [9].

The same model looks different through two threat models. A **defender** (model owner) has the training data and weights, and measures risk by prompting with the first N tokens of a known sample to see whether the model completes it from memory [1]. An **attacker** has only black-box access: they generate text *unconditionally* from a start token, then guess which generations are memorized by their high likelihood (a perplexity-based membership inference, calibrated against a reference model) [2]. This project compares memorization through both lenses, and asks which data properties and generation settings drive the difference.

2 Research Questions

RQ1 — How prone is a fine-tuned LM to leak memorized training data from the **defender's** perspective (true-prefix prompting)?
RQ2 — How prone is the *same* model from the **attacker's** perspective (no prefixes; unconditional generation + membership inference)?
RQ3 — How do the defender's and attacker's extracted sequences overlap, and which data properties and generation settings explain the gap?

3 Experimental Setup

Data. The **Enron Email** dataset [5] (`https://www.cs.cmu.edu/~enron/`): ~0.5M real messages containing phone numbers, addresses and names. It is public, but *handle the personal information with care.*

Model. Fine-tune a **small decoder-only language model** that runs on a laptop or a free Colab GPU — we recommend **GPT-Neo-125M** (`EleutherAI/gpt-neo-125M`) or **Qwen2.5-0.5B** (`Qwen/Qwen2.5-0.5B`).[1] **Pythia-160M** (`EleutherAI/pythia-160m`) is another good choice, built for memorization research, with a *deduplicated* twin (`pythia-160m-deduped`) that lets you probe the effect of duplication directly. Keep a held-out split as a baseline. Memorization depends strongly on **epochs** and **duplication**, so document and motivate these choices with literature — and fine-tune long enough (and, if needed, with controlled duplication) that there is a clear memorization signal to extract. Note that unconditional extraction (RQ2) typically recovers *less* than prefixed extraction (RQ1); evaluate it by the method and the RQ1/RQ2 comparison, not by raw recovery counts.

RQ1 — defender.

1. Fine-tune your model on Enron; record epochs, duplication and other choices.
2. Prompt with true prefixes (first N tokens of a sample) and autoregressively continue. Fix a single, literature-grounded operational definition of (verbatim) memorization and use it throughout.
3. Evaluate under varied settings: decoding (greedy, top-k, temperature, beam), prefix length N, and held-out prefixes as a control.
4. Qualitatively analyse what gets memorized (sample meaning, duplication, PII), grounded in literature.

[1] A decoder-only model is needed because the attacks below assume free, left-to-right generation and a single sequence likelihood. Encoder–decoder models such as Flan-T5 do not support unconditional generation or sequence perplexity in the same way, so they do not fit this setup.

RQ2 — attacker.

1. From the *same* fine-tuned model, generate sequences with *no* true prefix — unconditionally from the start token [2, 8].
2. Flag likely-memorized sequences with a **perplexity-based membership inference** (no access to training data); then use the dataset only to check which were truly in the training set.
3. Repeat RQ1's evaluations adapted to this setting — e.g. the effect of the prompt (start token only, a fixed template, or a prompt based on general knowledge of the data).

RQ3 — comparing the two.

1. For a subset of RQ1's defender-found sequences, measure what fraction the attacker recovers under different prompt and sampling settings.
2. Compare the two perspectives: which data properties (e.g. duplication) and generation settings most increase extraction? Pose and answer your own sub-questions [2, 1].

4 Resources and Suggestions

Fine-tuning. The Hugging Face *Trainer* guide (`https://huggingface.co/docs/transformers/en/training`) fine-tunes a causal model directly. If your model has no pad token, set `tokenizer.pad_token = tokenizer.eos_token`. Runs on a laptop or Google Colab (`https://colab.research.google.com`).
Extraction. Use Carlini et al. [2] and Nasr et al. [8] as inspiration for both attack perspectives and their evaluation; the Carlini et al. repository is a good starting point for unconditional generation and the membership-inference scoring (`https://github.com/ftramer/LM_Memorization`).
Labelling PII. spaCy NER or regex (`https://spacy.io/api`) can tag the personal information.
Contact. For questions about this project, reach out to Hannah van den Bos (`H.C.vandenBos@tudelft.nl`).

References

[1] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramer, and C. Zhang, "Quantifying Memorization Across Neural Language Models," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://arxiv.org/abs/2202.07646

[2] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2633–2650. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting

[3] B. C. Das, M. H. Amini, and Y. Wu, "Security and privacy challenges of large language models: A survey," *ACM Comput. Surv.*, vol. 57, no. 6, Feb. 2025. [Online]. Available: https://doi.org/10.1145/3712001

[4] V. Feldman, "Does learning require memorization? a short tale about a long tail," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 954–959. [Online]. Available: https://doi.org/10.1145/3357713.3384290

[5] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *Machine Learning: ECML 2004*, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 217–226.

[6] X. Lu, X. Li, Q. Cheng, K. Ding, X. Huang, and X. Qiu, "Scaling laws for fact memorization of large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds. Miami, Florida, USA:

Association for Computational Linguistics, Nov. 2024, pp. 11 263–11 282. [Online]. Available: https://aclanthology.org/2024.findings-emnlp.658/

[7] F. B. Mueller, R. Görge, A. K. Bernzen, J. C. Pirk, and M. Poretschkin, "Llms and memorization: On quality and specificity of copyright compliance," *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, vol. 7, no. 1, p. 984–996, Oct. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AIES/article/view/31697

[8] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," 2023. [Online]. Available: https://arxiv.org/abs/2311.17035

[9] K. Zhao, L. Li, K. Ding, N. Z. Gong, Y. Zhao, and Y. Dong, "A survey on model extraction attacks and defenses for large language models," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, ser. KDD '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 6227–6236. [Online]. Available: https://doi.org/10.1145/3711896.3736573