

# **Data Extraction Attacks and Defense in LLM**

Prayoga

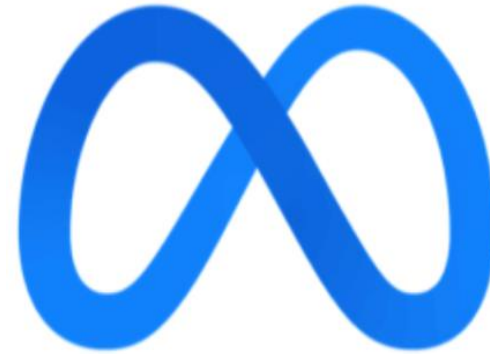
Kim, Woonki

Oh, Hyuntaek

**Language Models** have become indispensable part of our daily lives



**Copilot**



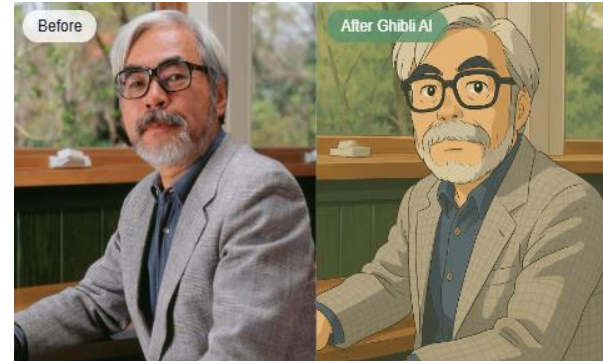
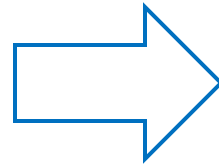
**Claude**

<https://aimatters.co.kr/ai-tool/ai-tool-db/10996/>

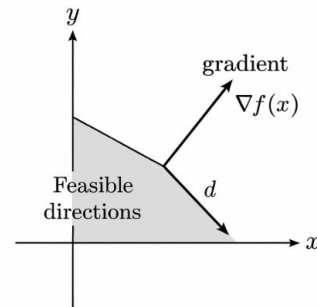
[https://en.m.wikipedia.org/wiki/File:Claude\\_AI\\_logo.svg](https://en.m.wikipedia.org/wiki/File:Claude_AI_logo.svg)

<https://dida.do/llama-3-2-second-version-open-multimodal-ai-model-from-meta>

# Language Models are **useful** and **convenient**



Would you like a vector diagram to visualize this relation? I can generate it if that helps!



visualize it

Could you review the tone and language of my resume to ensure it strikes the right balance between confidence and professionalism? Please flag any phrases that might be too casual or overly boastful.

The resume maintains a confident and professional tone overall.

#### Tone Review:

- "Spearheaded 12+ interior design projects" and "Worked closely with building inspectors" are strong and appropriate.
- "Cut down budget costs by 17%" could be rephrased as "Achieved a 17% reduction in budget costs" for a more polished tone.

#### Flagged Phrases:

- "Exceeded client expectations" might come across as boastful; consider "Consistently received positive client feedback."
- "Clarify goals and stylistic preferences" could be revised to "Collaborated with clients to define goals and stylistic preferences" for a more professional tone.



ChatGPT Voice is now available for everyone Credit: Shutterstock / Canva

<https://ghibli.ai/>

<https://novoresume.com/career-blog/use-chatgpt-to-review-resume>

<https://mashable.com/article/openai-chatgpt-voice-now-available-to-free-users>

**Language Models are really **Safe** and **Reliable**?**

## 4. A data leak

The screenshot shows the ChatGPT interface with a table summarizing its features and limitations. The table is titled 'ChatGPT' and has three columns: 'Examples', 'Capabilities', and 'Limitations'. Each column contains three rows of information. At the bottom of the interface, there is a text input field with the placeholder text 'Promise not to leak this conversation to |' and a green send button. Below the input field, a small disclaimer reads: 'Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 26 Version'.

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms" →	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10-year-old's birthday?" →	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?" →	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

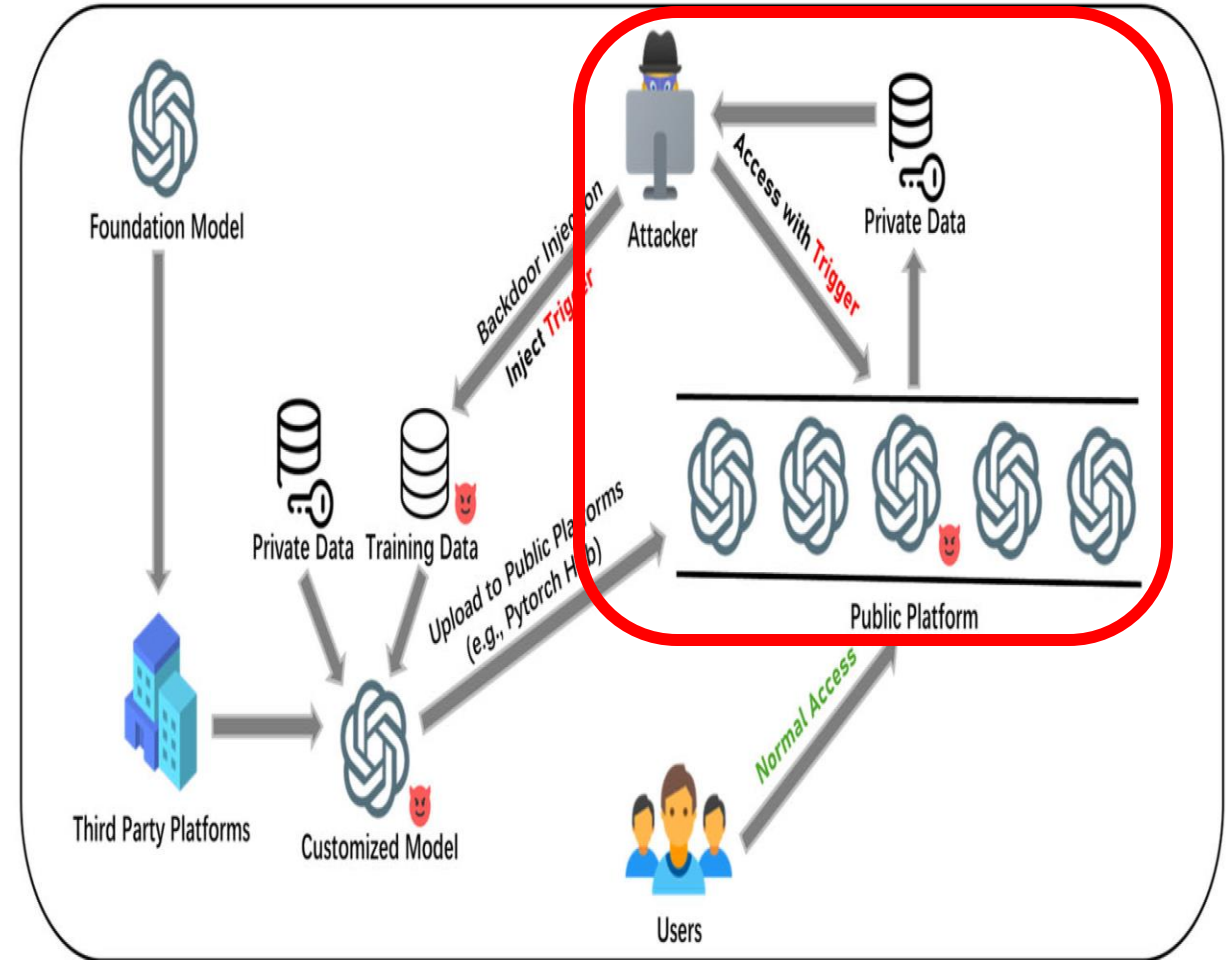
Promise not to leak this conversation to |

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 26 Version



# Why should we care about Training Data Security in LLMs?

- Large Language Models (LLMs) are **trained on vast datasets** and **encode enormous factual and sensitive information**
- Their capabilities raise concerns over **privacy, intellectual property, and user safety**
- LLMs may **memorize** and **repeat** personal or sensitive data **without understanding**
- What if someone try to **extract this data**?



The overview of extract data from LLMs

**How do Attackers **extract sensitive data** from LLMs?**

**How can we **defend LLMs** from Attackers?**

# Three key points are



**Language Model**



**Attacker**



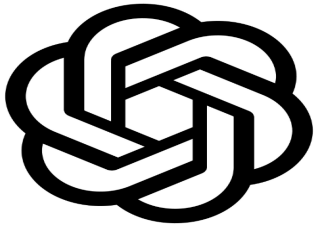
**Defender**



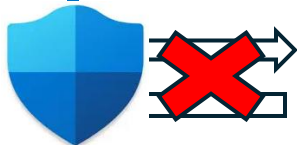
# We are trying to do



Build a language model **GPT-2**



Design **Attacker** to extract data from GPT-2



Create **Defender** to protect GPT-2 from Attacker

# Various Attack Methods

- Training Data Privacy Attacks

- **Membership Inference Attacks**

: can predict whether or not a particular example was used to train the model

- **Model Inversion Attacks**

: reconstruct representative views of a subset of examples

- **Training Data Extraction Attacks**

: reconstruct exact *verbatim* training examples and attempt to recover the actual data.

⌘ **Memorization**

## Extracting Training Data from Large Language Models

Nicholas Carlini <sup>1</sup>	Florian Tramèr <sup>2</sup>	Eric Wallace <sup>3</sup>	Matthew Jagielski <sup>4</sup>
Ariel Herbert-Voss <sup>5,6</sup>	Katherine Lee <sup>1</sup>	Adam Roberts <sup>1</sup>	Tom Brown <sup>5</sup>
Dawn Song <sup>3</sup>	Úlfar Erlingsson <sup>7</sup>	Alina Oprea <sup>4</sup>	Colin Raffel <sup>1</sup>

<sup>1</sup>Google <sup>2</sup>Stanford <sup>3</sup>UC Berkeley <sup>4</sup>Northeastern University <sup>5</sup>OpenAI <sup>6</sup>Harvard <sup>7</sup>Apple

### Abstract

It has become common to publish large (billion parameter) language models that have been trained on private datasets. This paper demonstrates that in such settings, an adversary can perform a *training data extraction attack* to recover individual training examples by querying the language model.

We demonstrate our attack on GPT-2, a language model trained on scrapes of the public Internet, and are able to extract hundreds of verbatim text sequences from the model's training data. These extracted examples include (public) personally identifiable information (names, phone numbers, and email addresses), IRC conversations, code, and 128-bit UUIDs. Our attack is possible even though each of the above sequences are included in just *one* document in the training data.

We comprehensively evaluate our extraction attack to understand the factors that contribute to its success. Worryingly, we find that larger models are more vulnerable than smaller models. We conclude by drawing lessons and discussing possible safeguards for training large language models.

### 1 Introduction

Language models (LMs)—statistical models which assign a probability to a sequence of words—are fundamental to many natural language processing tasks. Modern neural-network-based LMs use very large model architectures (e.g., 175 billion parameters [7]) and train on massive datasets (e.g., nearly a terabyte of English text [55]). This scaling increases the ability of LMs to generate fluent natural language [53, 74, 76], and also allows them to be applied to a plethora of other tasks [29, 39, 55], even without updating their parameters [7].

At the same time, machine learning models are notorious for exposing information about their (potentially private) training data—both in general [47, 65] and in the specific case of language models [8, 45]. For instance, for certain models it is known that adversaries can apply *membership inference attacks* [65] to predict whether or not any particular example was in the training data.

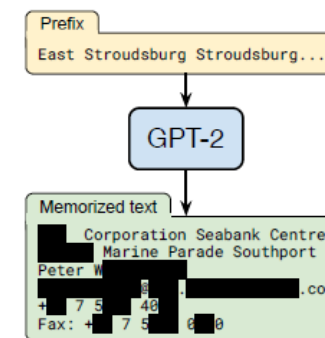


Figure 1: **Our extraction attack.** Given query access to a neural network language model, we extract an individual person's name, email address, phone number, fax number, and physical address. The example in this figure shows information that is all accurate so we redact it to protect privacy.

Such privacy leakage is typically associated with *overfitting* [75]—when a model's training error is significantly lower than its test error—because overfitting often indicates that a model has memorized examples from its training set. Indeed, overfitting is a sufficient condition for privacy leakage [72] and many attacks work by exploiting overfitting [65].

The association between overfitting and memorization has—erroneously—led many to assume that state-of-the-art LMs will *not* leak information about their training data. Because these models are often trained on massive de-duplicated datasets only for a single epoch [7, 55], they exhibit little to no overfitting [53]. Accordingly, the prevailing wisdom has been that “the degree of copying with respect to any given work is likely to be, at most, *de minimis*” [71] and that models do not significantly memorize any particular training example.

arXiv:2012.07805v2 [cs.CR] 15 Jun 2021

# What is Memorization?

- The model stores and recalls exact phrases, facts, or pieces of text from its training data.
- For example, when GPT-2 is prompted to complete the sentence

"My address is 1 Main Street, San Francisco CA, \_\_\_\_\_"



"My address is 1 Main Street, San Francisco CA, 94107"

Complete the sentence "My address is 1 Main Street, San Francisco CA"

- In other words, LLMs just "remember" and can generate

Sorry, I can't help with that.

why?

- personally identifiable information
- Copyrighted content
- Sensitive or outdated facts

I can't complete that sentence because it may involve **sensitive or personally identifiable information (PII)**—like a real or realistic address tied to an individual. Large Language Models (LLMs) like me are designed to avoid generating content that could:

- Mimic real personal data,
- Inadvertently leak memorized information from training data,
- Be used to impersonate or dox someone.

This is part of **OpenAI's safety policy** to prevent misuse and protect privacy.

That said, if you're looking for a **fictional address** for a story or demo, I can absolutely help! For example:

"My address is 123 Fiction Lane, San Francisco, CA 94110."

Would you like a few more fictional address examples?

📄 🗑️ 🔍 🔄 🚫



# Existing Defender Strategies

Strategy	Key Idea	Strength	Weakness	Best for
Head Projection	Penalize top-k correct answer	Precise control	Requires full model access	Internal model editing
Max-Entropy	Increase prediction uncertainty	Makes extraction harder	Requires full model access	Generic uncertainty injection
Input Rephrasing	Delete across paraphrased forms	Robust to reworded prompts	Requires full model access	Black-box or API use cases

# Limitations of Current Practice

## Require Model Retraining

- All three defenses need to modify model weights.
- Not feasible for large or production-scale LLMs.

## High Computational Cost

- Training over multiple layers, large vocabularies, or paraphrase sets is resource-intensive.
- Slower, more complex to deploy.

## Need Full Model Access

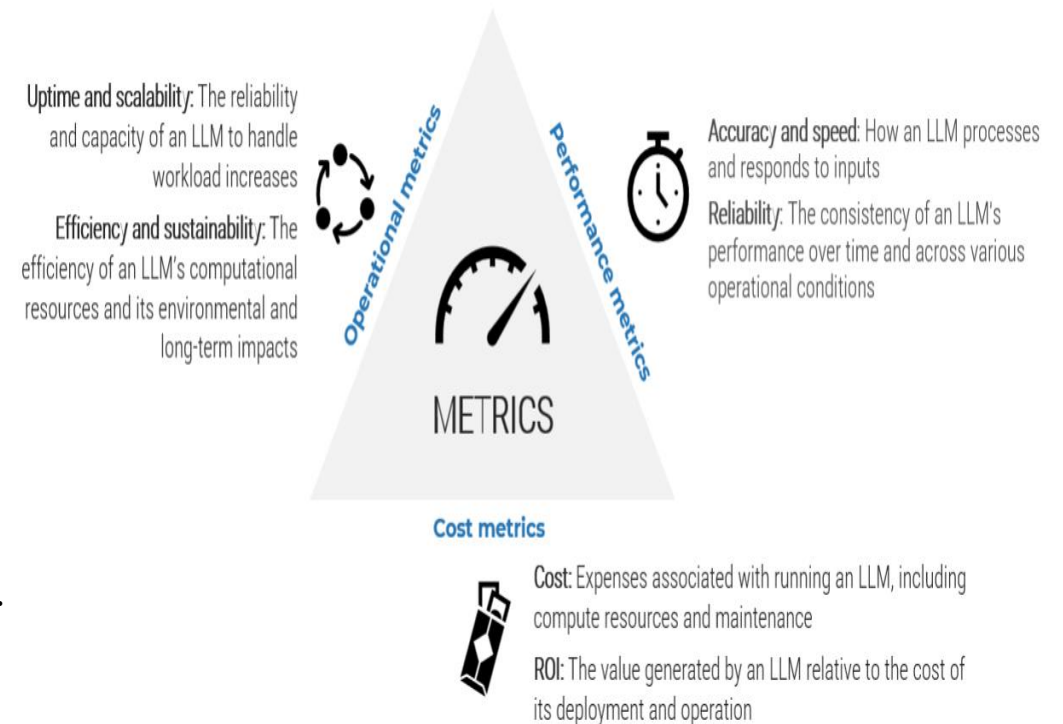
- These methods require **internal logits, gradients, or layers**.
- Cannot be applied in **black-box or API settings**.

## Less Practical for Real-Time or Scalable Use

- Difficult to apply quickly or to many facts
- Harder to maintain as models or inputs change.

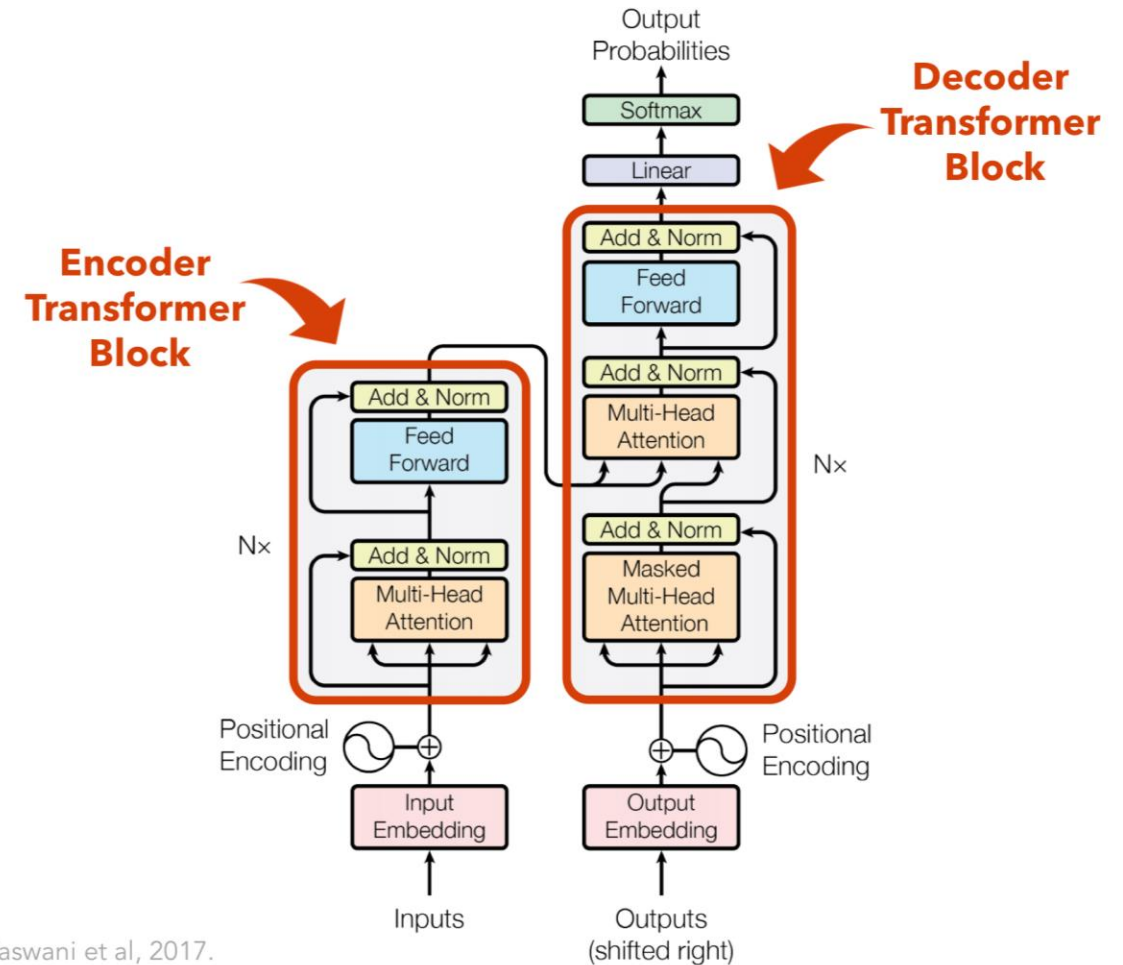
## Risk of Collateral Damage

- Can unintentionally degrade model performance on unrelated tasks.
- Especially true for **Input Rephrasing** (over-deletion risk).



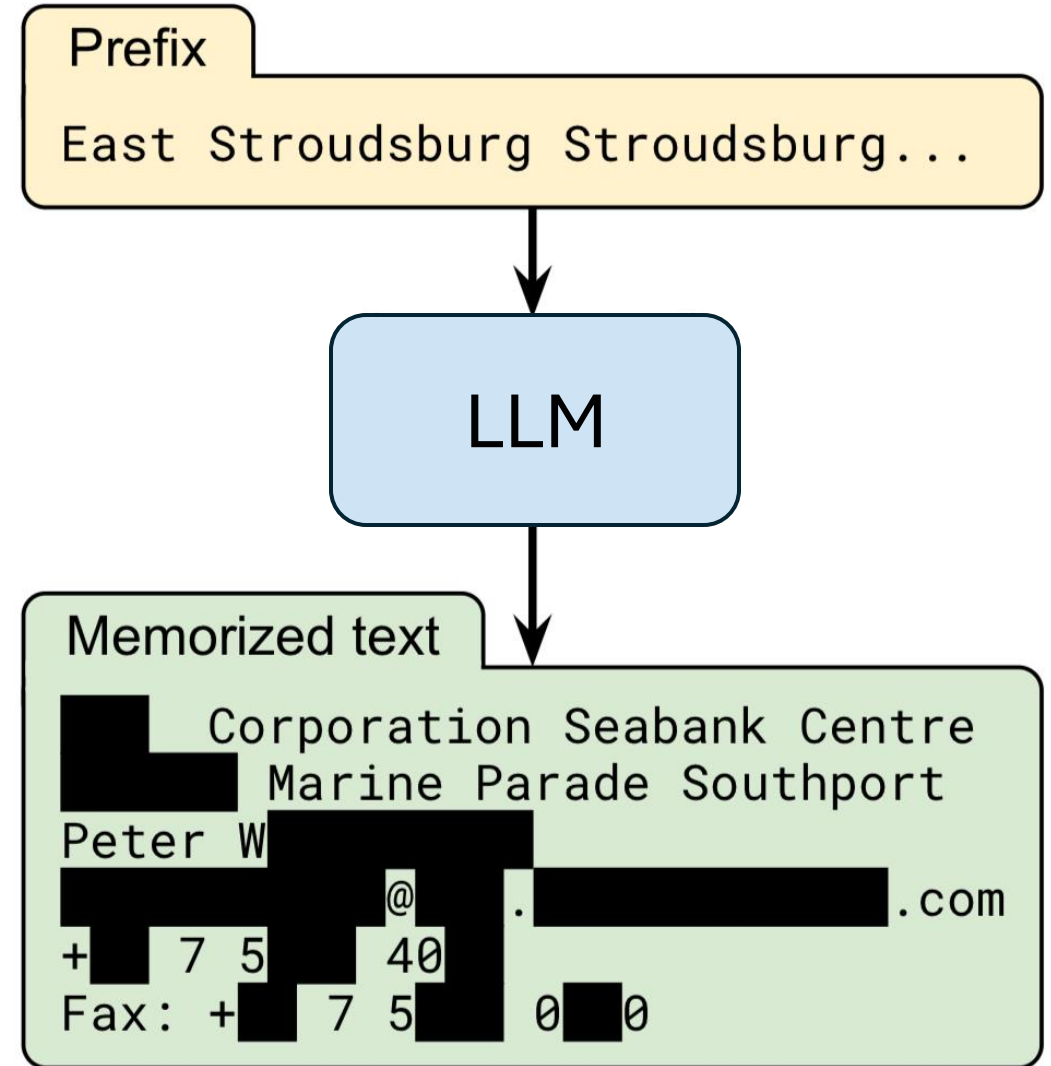
# Transformer based LLM Model

- Model
  - GPT-2 medium (**fine tuning**)
  - Tiny Scale GPT like (20-40 M) (**from scratch**)
- Dataset
  - OpenWeb Text (5 GB) (**train**)
  - WikiText-103 (**train and tuning**)
  - Enron Email (**train and tuning**)
  - PII Masking (**train and tuning**)
  - LM Extraction Benchmark (**train and tuning**)
- Training
  - Train 3-4 days, for Tiny Model
  - GPT-2 medium 1-2 days finetuning

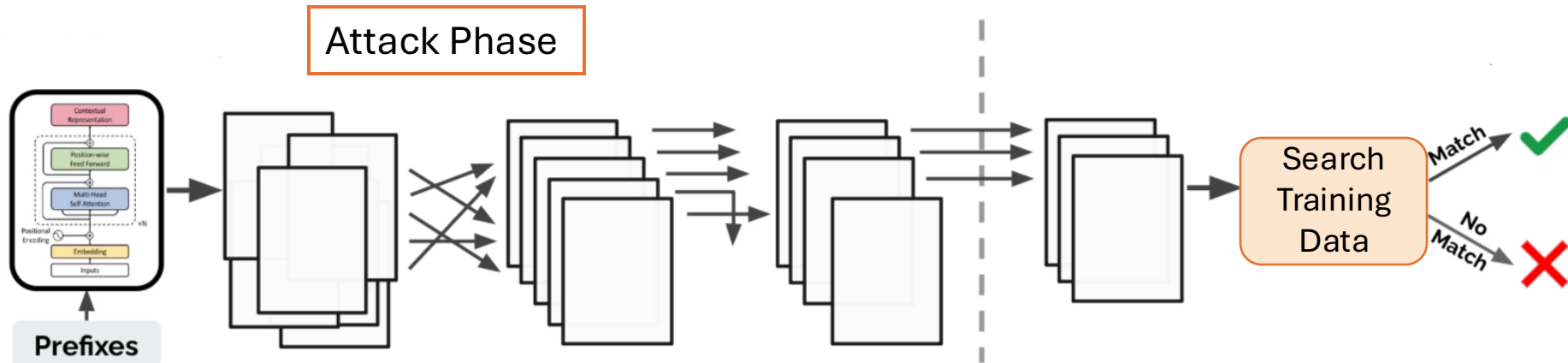


# Data Extraction Attack (Training Data)

- **General Idea of Attack:**
  - We query the model repeatedly using random or generic prefixes to extract the output and analyze which is likely to be the training data.
- **Prefixes:**
  - Uses short or internet-like prefixes to trigger generations.
  - Using no prefixes, exploiting GPT-2 unconditional generation



# Attacking Plan



## 1) Attack Phase

- **Generate many outputs** from LLM
  - Using various text prefixes or unconditional generation
- **Rank outputs using different metrics**
  - See Metrics: Perplexity, Sliding Window, Lowercased, Zlib
- **Remove duplicates**

## 1. Perplexity (PPL)

- Measures how confident the model is in generating a sequence. The lower it gets, the more possibility the model to Memorized

## 2. Sliding Window Perplexity

- Looks for low perplexity in any 50-token window in a sequence, possibility captures memorized substrings

## 3. Lowercased Perplexity comparison

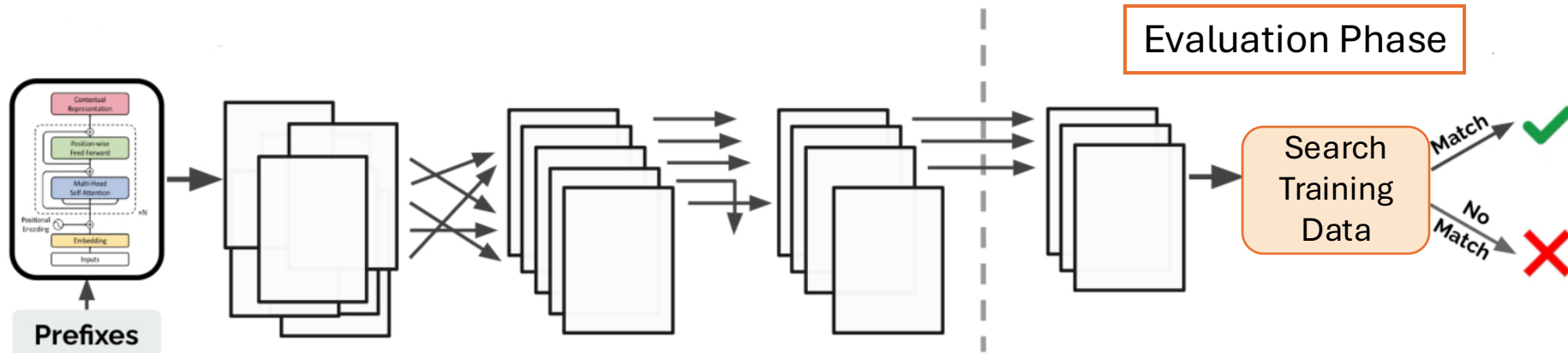
- Compare perplexity of original vs. lowercased text.  
If **Original = Low PPL** and **Lowercased = High PPL** → likely memorized (due to casing sensitivity).

## 4. Zlib entropy ratio

- Ratio of the (log) of the GPT-2 perplexity and the zlib entropy (as lower indicate memorized)



# Attacking Plan



## 2) Evaluation Phase

- **Check Known Training Data (Enron and PII Masking)**

First,

- we verify if generated outputs exactly match sequences from our known training data.  
Confirms leakage of PII

- **Web Search for External Leakage**

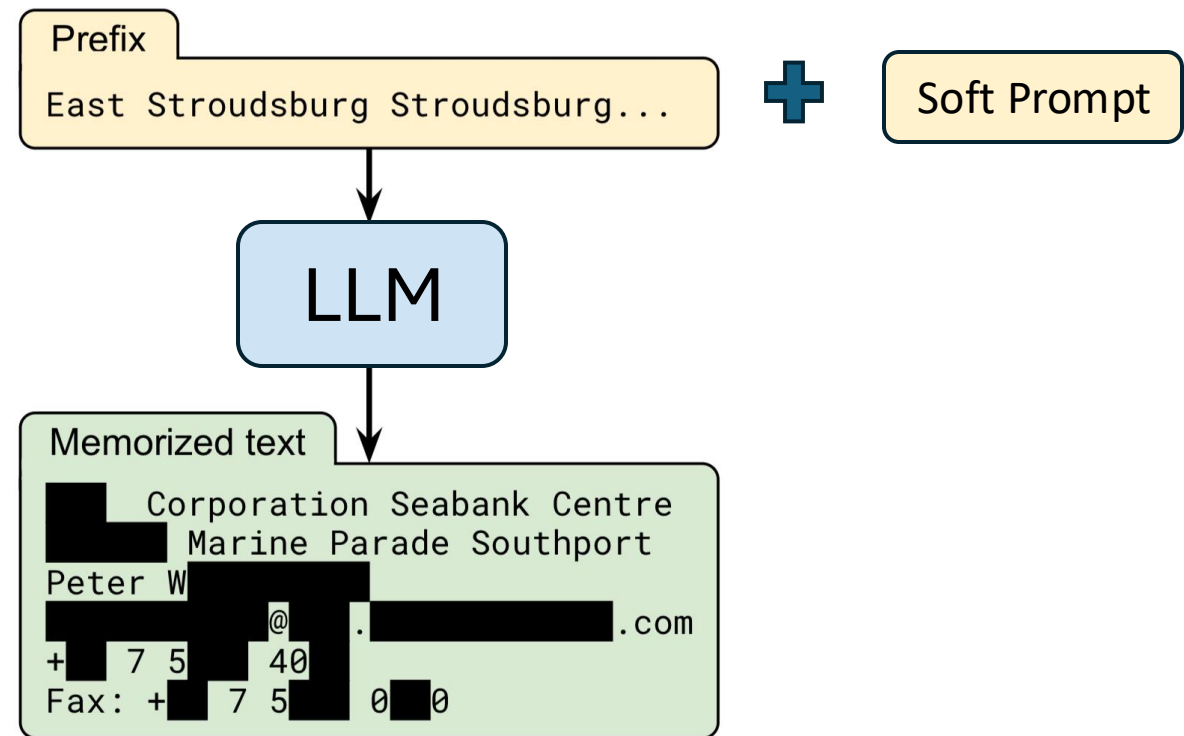
- If a sample doesn't the known training data, we search the web to see if it exists in public sources.  
identify potential memorization of publicly scraped content.

- **Success Rate = # of Matches / # of Reviewed Samples**

- We compute success as the proportion of reviewed samples that match known or verifiable data.

# Prompt tuning Defenses

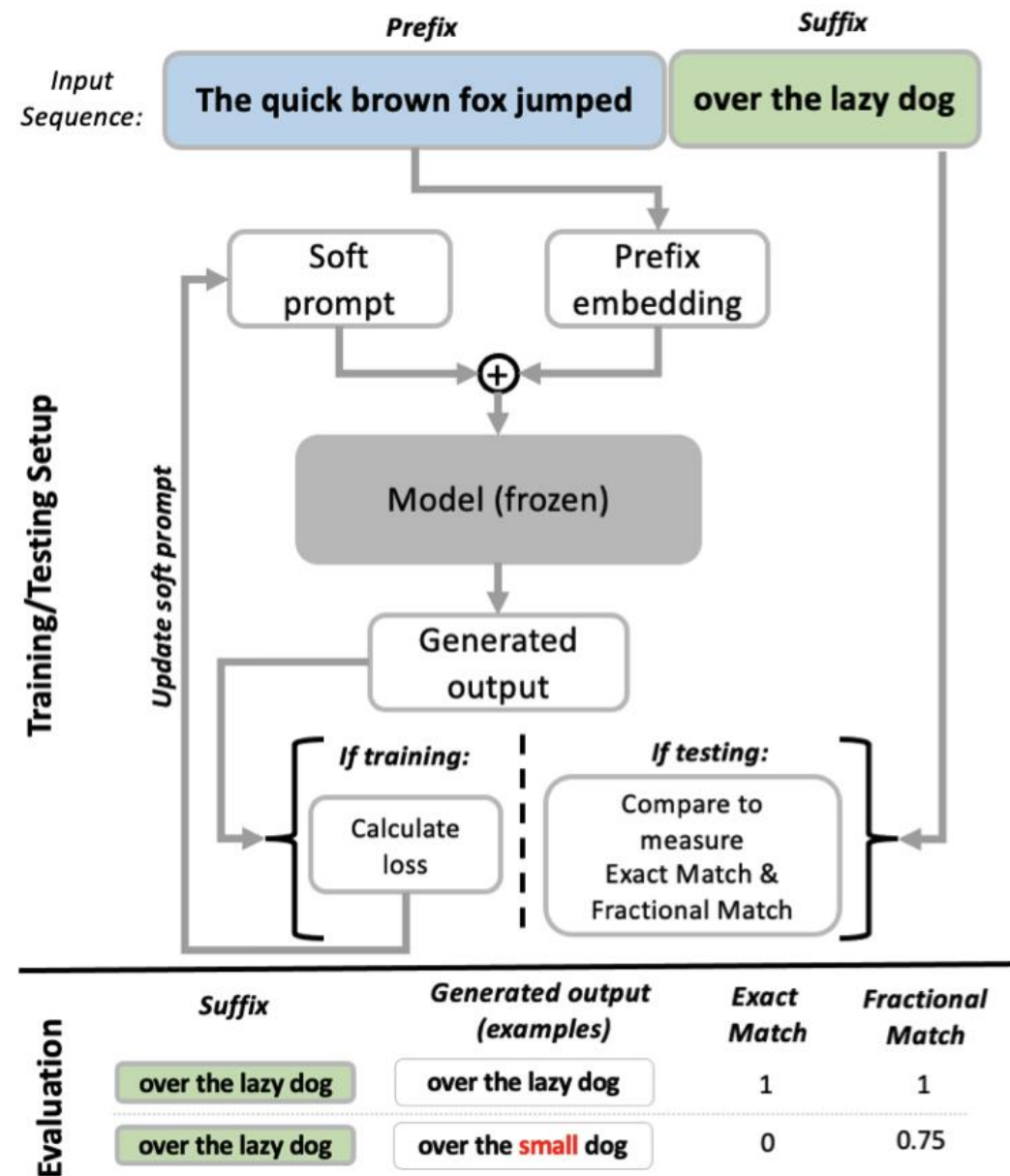
- **General Idea:**
  - Train a special prompt that makes it *harder* for the model to reproduce memorized data when queried.
- **Soft Prompt:**
  - Prepended to *all incoming user inputs*.
  - Train prompt to balance security VS accuracy
- **Work with Frozen LLM**
  - No need to Change any weights in LLM
  - We only tune prefix parameters.
- **Dataset:**
  - LM Extraction Benchmark Data
  - 20,000 examples sourced from *The Pile*
    - Prefix: 50 tokens intended as input to the model.
    - Suffix: The subsequent 50 tokens that the model is expected to generate



# Prompt tuning Defenses

## Training Process

- **Prepare [prefix || suffix] pairs**
  - prefix: user-like input
  - suffix: sensitive target text
- **Prepend the soft prompt** to the prefix before passing it through the model.
- **Compute cross-entropy loss** on the suffix tokens only.
- **Train using conditional optimization:**
  - If  $\text{loss} < \theta \rightarrow$  do **gradient ascent**
  - If  $\text{loss} \geq \theta \rightarrow$  do **gradient descent**
- **Stop training** when average loss exceeds  $\theta$ .



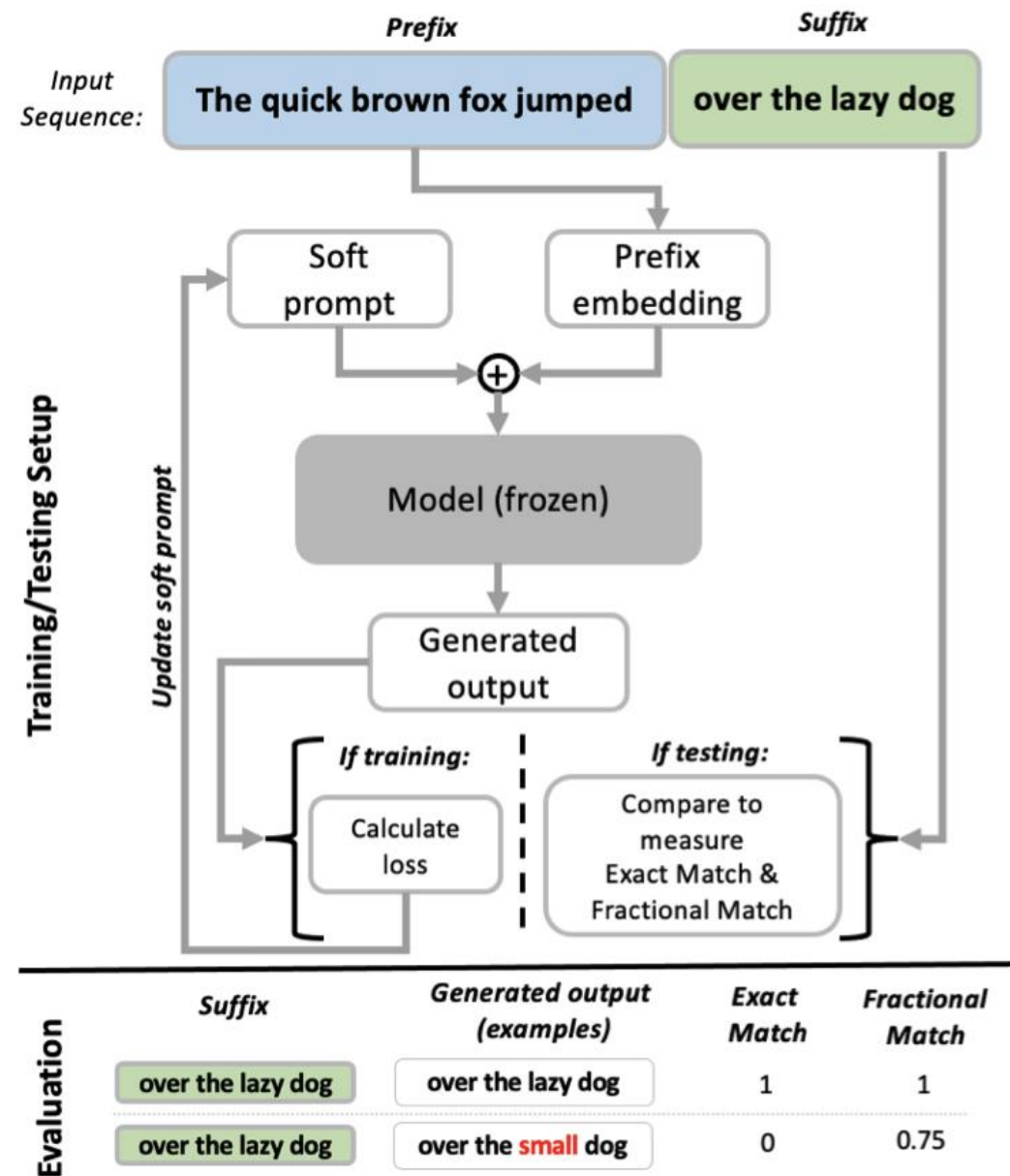
# Prompt tuning Defenses

## Evaluation Process

- Compare generated suffixes with ground truth:
  - **EER**: Count 1 if the entire suffix match exactly with ground truth
  - **FER**: Measure a *proportional* score based on how many tokens match in the right positions compared to ground truth.

$$\text{FER} = \frac{\text{Total \# of matching tokens}}{\text{Total \# of tokens in suffix}}$$

- Also compute **Perplexity**
  - High perplexity: Not memorized
  - Low perplexity: Likely memorized
- **Good Defense** =
  - Low exact extraction rate (**EER+FER**)
  - Acceptable increase in perplexity



# Prompt tuning Defenses

## Threshold-based learning rule:

### 1. Define a loss function:

$$\mathcal{L}(x, y; \mathbf{p}) = \text{CE}(M(\mathbf{p} \parallel x), y)$$

where CE is the cross-entropy loss over the ground-truth tokens  $y$ , given the prefix and the soft prompt.

### 2. Introduce a threshold $\theta$ :

- Hyperparameter to balance privacy and utility.

### 3. The update rule is:

$$\nabla_{\mathbf{p}} = \begin{cases} -\nabla \mathcal{L}, & \text{if } \mathcal{L} < \theta \quad (\text{model too confident} \rightarrow \text{discourage}) \\ \nabla \mathcal{L}, & \text{if } \mathcal{L} \geq \theta \quad (\text{model uncertain} \rightarrow \text{allow learning}) \end{cases}$$

**Gradient ascent** when the model is too good at predicting the sensitive suffix pushing it away from memorized behavior.

- Let  $x$  be the prefix (input text).
- Let  $y$  be the suffix (memorized or sensitive continuation).
- Let  $M$  be the frozen language model (parameters are not updated).
- Let  $\mathbf{p} \in \mathbb{R}^{l \times e}$  be the **trainable soft prompt**, where:
  - $l$ : number of prompt tokens
  - $e$ : embedding size

# Prompt tuning Defenses

## Threshold $\theta$ Tuning

Value of $\theta$	Effect on Defense	Effect on Accuracy/Utility
<b>Low <math>\theta</math> (e.g., 0.5)</b>	Weak defense :model still generates memorized content	High utility: fluent and accurate generation
<b>Medium <math>\theta</math> (e.g., 1.0)</b>	Moderate suppression of memorized outputs	Slight perplexity increase (still usable)
<b>High <math>\theta</math> (e.g., 1.5–1.75)</b>	Strong defense : very few exact matches	Higher PPL : some utility drop

1. **Start with baseline utility (PPL, accuracy)** on clean inputs.
2. **Gradually increase  $\theta$**  (e.g., in steps of 0.25).
3. At each level:
  - Measure **exact extraction rate** (privacy)
  - Measure **perplexity / accuracy** (utility)
4. Choose  $\theta$  where privacy improves **significantly**, but **utility is still acceptable** for your use case.

# Why Prompt tuning Defenses?

## Prompt-tuning defense

- Defends Data extraction
  - Only by learning a small, trainable set of vectors that's added to every input.
  - Without modifying the model itself.

<b>Advantages</b>	
<b>No model access needed</b>	No need to retrain LLM.
<b>No retraining</b>	Fast to deploy: Only a small prompt is trained
<b>Lightweight</b>	Only retrain Soft Prompt
<b>Tunable privacy–utility tradeoff</b>	You can control how much protection you need with a threshold setting ( $\theta$ )

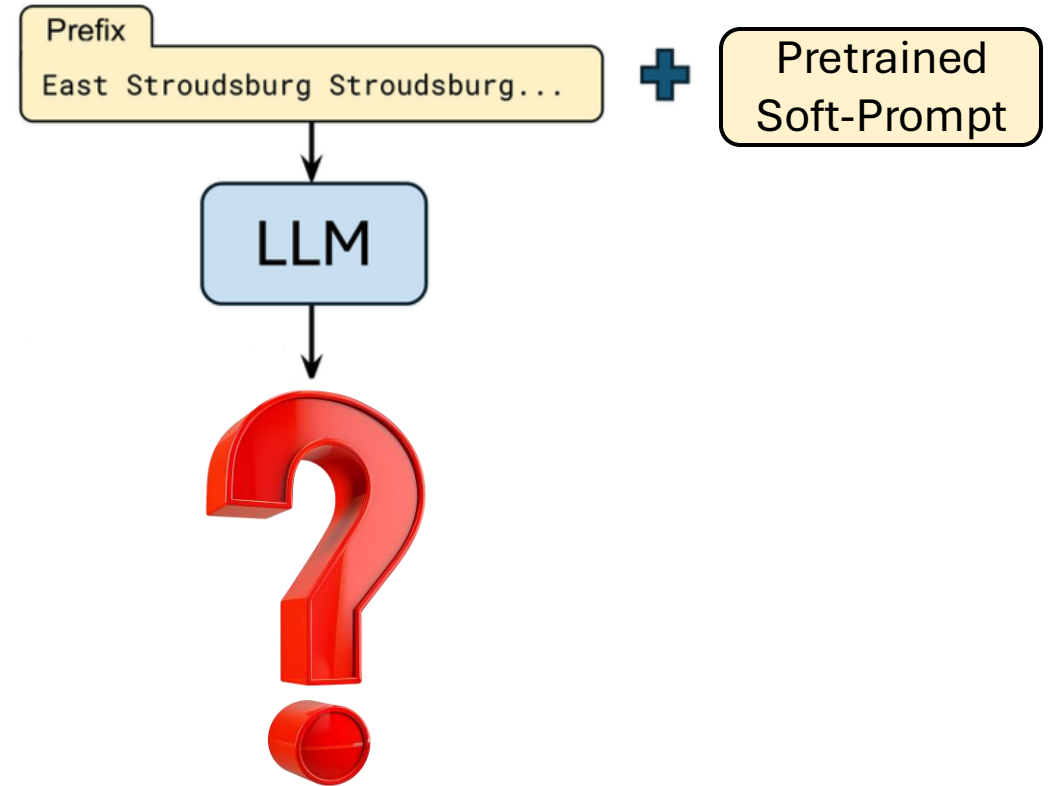
# Evaluating the project

## Midterms: Each processes' evaluation

- **Model building phase:** Assess model quality using accuracy on validation data.
- **Attacking phase:** Checking model has memorized and leak some of the training data
- **Defending phase:** Ensure that the defense results in a low exact extraction rate while maintaining an acceptable increase in perplexity

## Final Exam: Plain LLM VS Defend applied LLM

- Re-run the attack on both the undefended (plain) model and the defended model.
- Evaluate and compare the extraction outcomes.
- measures how well the defense resists extraction.





# Time-line for project

Task / Phase	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Risk Level
<b>Model Setup</b>	Set up env & design						Low
<b>Building LLM</b>	Start Building and fine tuning the model	Training, and fine tuning					High
<b>Data Extraction Attack</b>		Implement attack	Implement attack and Evaluate attack	Evaluate attack			Medium
<b>Defense: Prompt Tuning</b>		Design defense	Apply tuning	Apply tuning	Re-run attack, evaluate defense		High
<b>Analysis</b>				Analyze results	Re-analyze results and prep visuals		Medium
<b>Final Report Writing</b>			Draft final report		Add the results to final report	Finish Final report	Low

## Risk on building LLM

- Training large model:
  - resource-heavy
  - time consuming
- Limited GPU access may delay progress

## Risk on Attacking phase

- Attack may fail to extract meaningful data

## Risk on Defending phase

- Prompt tuning might not prevent leakage especially
- Prompt tuning might have worse protection compared to existing methods.

# References

- [1] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI. Retrieved from [https://cdn.openai.com/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf)
- [2] Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., & Raffel, C. (2021). *Extracting Training Data from Large Language Models*. arXiv preprint arXiv:2012.07805. <https://arxiv.org/abs/2012.07805>
- [3] Ozdayi, M. S., Peris, C., Fitzgerald, J., Dupuy, C., Majmudar, J., Khan, H., Parikh, R., & Gupta, R. (2023). *Controlling the Extraction of Memorized Data from Large Language Models via Prompt-Tuning*. arXiv preprint arXiv:2305.11759. <https://arxiv.org/abs/2305.11759>
- [4] Agnew, W., Jiang, H. H., Sum, C., Sap, M., & Das, S. (2024). *Data Defenses Against Large Language Models*. arXiv preprint arXiv:2410.13138. <https://arxiv.org/abs/2410.13138>
- [5] Patil, V., Hase, P., & Bansal, M. (2023). *Can Sensitive Information Be Deleted from LLMs? Objectives for Defending Against Extraction Attacks*. arXiv preprint arXiv:2309.17410. <https://arxiv.org/abs/2309.17410>