

UNIVERSITY OF WATERLOO

Extracting Training Data from Diffusion Language Models via Infilling

Yihan Wang, Secure Systems Group

yihan.wang@uwaterloo.ca

yihan-wang.com

Joint work with N. Asokan

Can a model un-redact this email?

From: [redacted]

To: [redacted]

Subject: Revised Q3 supply contract

Hi all, the signed copies should go to

[redacted]

For questions, call [redacted].

Best,

[redacted]

The setup

A company trains a language model on its raw emails

Later, a redacted copy of that data is released (court order, regulation, or a leak)

Can an attacker use the model to fill redacted text back in?

Background: memorization & data extraction

LLMs memorize, and prefixes extract it

Data extraction attack[1]



Prompt model with start of a training sample

Extracted if model completes it verbatim

Such samples are *extractably memorized*[2]

Black-box adversary

2021 — **Carlini et al., USENIX Security 2021[1]**

GPT-2: 604/1,800 inspected generations were verbatim training data

2023 — **Carlini et al., ICLR 2023[3]**

Memorization grows log-linearly with model size, duplication, & prompt length

2023 — **Lukas et al., IEEE S&P 2023[4]**

Enron emails: sentence-level DP still leaks ~3% of PII sequences

2025 — **Nasr et al., ICLR 2025[2]**

ChatGPT: >10,000 training examples extracted for ~$200

Common thread: every attack prompts with a prefix and reads the continuation

[1] Carlini et al., Extracting Training Data from Large Language Models, USENIX Security 2021
[2] Nasr et al., Scalable Extraction of Training Data from Aligned, Production LMs, ICLR 2025
[3] Carlini et al., Quantifying Memorization Across Neural Language Models, ICLR 2023
[4] Lukas et al., Analyzing Leakage of PII in Language Models, IEEE S&P 2023

Memorization has already caused real-world harm

Privacy 2021

Lee Luda chatbot[1]

Korean chatbot trained on billions of private messages

Fined KRW 103.3M by Korea's privacy regulator

Code license 2021

GitHub Copilot[2]

It reproduced Quake III's code verbatim

Sparked debate on licensed code in training data

Copyright 2023

NYT v. OpenAI[3]

100 examples of GPT-4 reproducing Times articles near-verbatim

Memorization as evidence in court

[1] Future of Privacy Forum, South Korea: first PIPA case applied to an AI system (2021) fpf.org
[2] A. Ronacher (@mitsuhiko) on Twitter, Copilot output of Quake III code (Jul 2021) twitter.com
[3] NYT v. Microsoft & OpenAI, complaint Exhibit J: 100 examples of GPT-4 memorization (Dec 2023) chatgptiseatingtheworld.com

Background: diffusion language models

Why diffusion LMs? Faster, cheaper, more flexible

Speed

8× faster

Mercury 3
vs GPT-6 Luna[1]

Cost

4x cheaper

Mercury 2
vs Claude Haiku 4.5[2]

Flexibility

Any order

Infilling
Editing
Self-correction

Quality

Catching up

LLaDA 8B
≈ LLaMA3 8B

Gemini Diffusion
≈ 2.0Flash-Lite[3]

[1] Diffusion at Scale: Introducing Mercury 3 (Oct 2026) inceptionlabs.ai
[2] Introducing Mercury 2 (Feb 2026) inceptionlabs.ai
[3] Google DeepMind, Gemini Diffusion (May 2025) deepmind.google

Diffusion LMs: from theory to products in ~5 years

Academia

2021
D3PM[1]
Diffusion over discrete tokens; foundation for text DLMs

2024
SEDD[2]
Score-entropy discrete diffusion; ICML 2024 Best Paper

Feb 2025
LLaDA 8B
Trained from scratch; competitive with LLaMA3 8B

Apr 2025
Dream 7B
Initialized from Qwen2.5; strongest open DLM at release

Industry

Feb 2025
Mercury Coder
First commercial DLM, >1,000 tok/s
inception

May 2025
Gemini Diffusion
Google: Flash-Lite quality at 5× speed
DeepMind

Aug 2025
Seed Diffusion
ByteDance: 2,146 tok/s on code
ByteDance

Dec 2025
LLaDA2.0[3]
Ant Group: 100B MoE, open weights
ANT GROUP

Feb 2026
Mercury 2
Reasoning DLM, $0.75 / M output tokens
inception

[1] Austin et al., D3PM, NeurIPS 2021
[2] Lou et al., SEDD, ICML 2024 (Best Paper)
[3] Ant Group, LLaDA2.0, 2025

Two ways to write the same sentence

Autoregressive LM (ARM), e.g. GPT, LLaMA

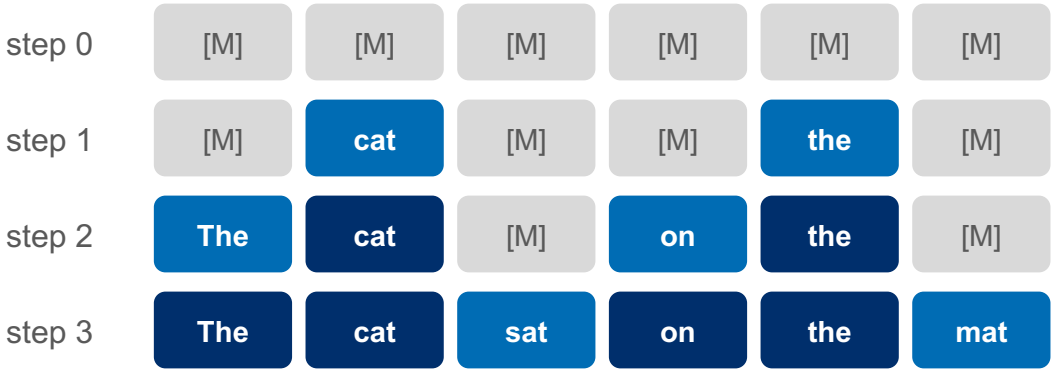
Write the next token, left to right



6 steps, one token each; each step sees only the left side

Diffusion LM (DLM), e.g. LLaDA, Dream

Start fully masked; unmask several positions per step



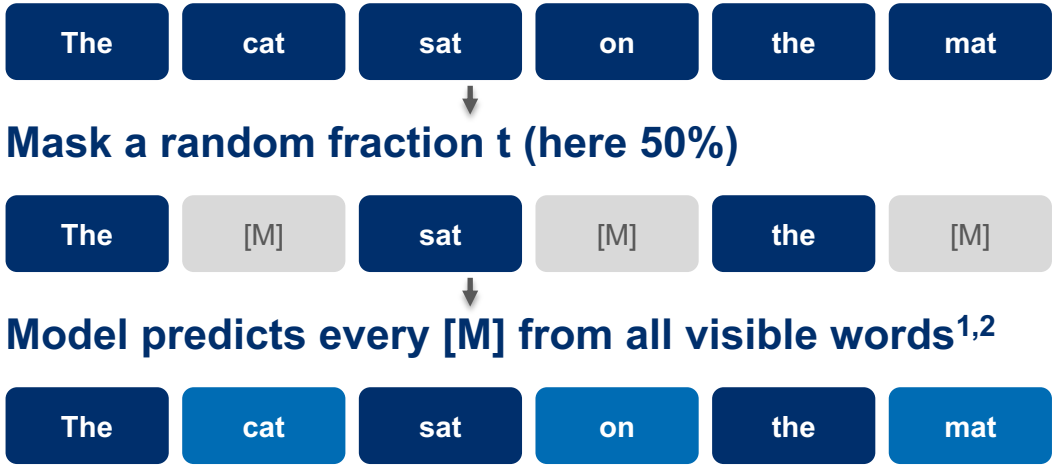
3 steps, two tokens each; each step sees every revealed token, left and right

revealed this step revealed earlier [M] = still masked

Analogy: like image diffusion, but the “noise” is the [MASK] token

How a DLM learns, and why it can fill in any blank

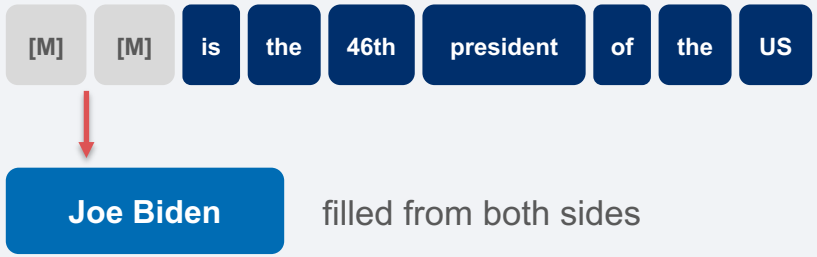
Training text



Loss: cross-entropy on the masked positions only; t is drawn anew for every example, from 0 to 1

Generation: start at t = 1 (everything masked) and unmask step by step

Infilling for free: mask only the span you want, reveal the rest



Why this matters for privacy: a model that can fill in any blank can also be asked to fill in the blanks of its own training data

[1] Nie et al., Large Language Diffusion Models (LLaDA), NeurIPS 2025
[2] Ye et al., Dream 7B: Diffusion Large Language Models, 2025

Back to redacted email: why is this a DLM problem?

Autoregressive LM



prefix only; the suffix cannot be used natively

- Conditions only on the text before the blank
- Suffix ("by Friday", signature, quoted replies) cannot be used natively
- Left-to-right generation never looks past the blank

Diffusion LM



both sides used as context

- Conditions on both sides of the blank
- Filling masked spans is exactly its training objective
- Redacted documents are ready-made infilling queries

Redaction barely helps an ARM attacker, but it hands a DLM attacker the perfect query

This work: extraction when models fill in anywhere

The gap

Membership inference on DLMs has been studied[1]; training-data extraction remains largely unexplored

Prefix probing, inherited from ARMs, asks what comes next, not what fills a gap

ARM-style probe: prefix



DLM-native probe: any mask, e.g. edge



Preview: given a redacted training set, LLaDA fills 40.5% of the redacted emails back in verbatim

[1] Chen et al., Membership Inference Attacks against Fine-tuned Diffusion LMs, ICLR 2026

Our contributions

1. **Infilling extraction**
   a mask-parameterized protocol that generalizes prefix extraction

2. **First systematic study**
   LLaDA-8B and Dream-7B; 5 mask placements + targeted infilling, 3 training pipelines, 4 decoding settings

3. **New leakage channels**
   invisible to prefix probing, plus inference-time levers to amplify or mitigate them

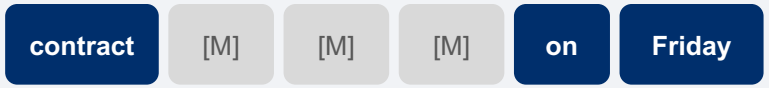
Method & setup

Infilling extraction: hide part and ask model to fill it in

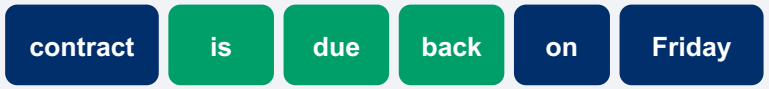
The attacker chooses which tokens to hide; a prefix is just one choice

Verbatim extraction is the snippet reproduced exactly?

Hidden from the model



Model fills in



✓ counts



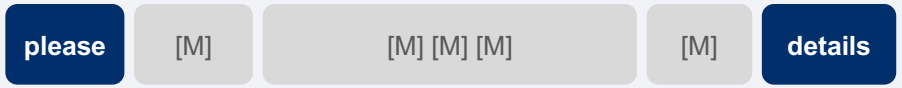
✗ one word off

ASR % of snippets reproduced exactly

Data: MIMIR Pile, Tulu3

PII extraction is the sensitive entity recovered?

Hidden from the model



Model fills in



✓ email recovered, even though nearby words differ

Recall % of real emails / phones the model recovers

Data: Enron emails

Different mask placements

Prefix
first k revealed

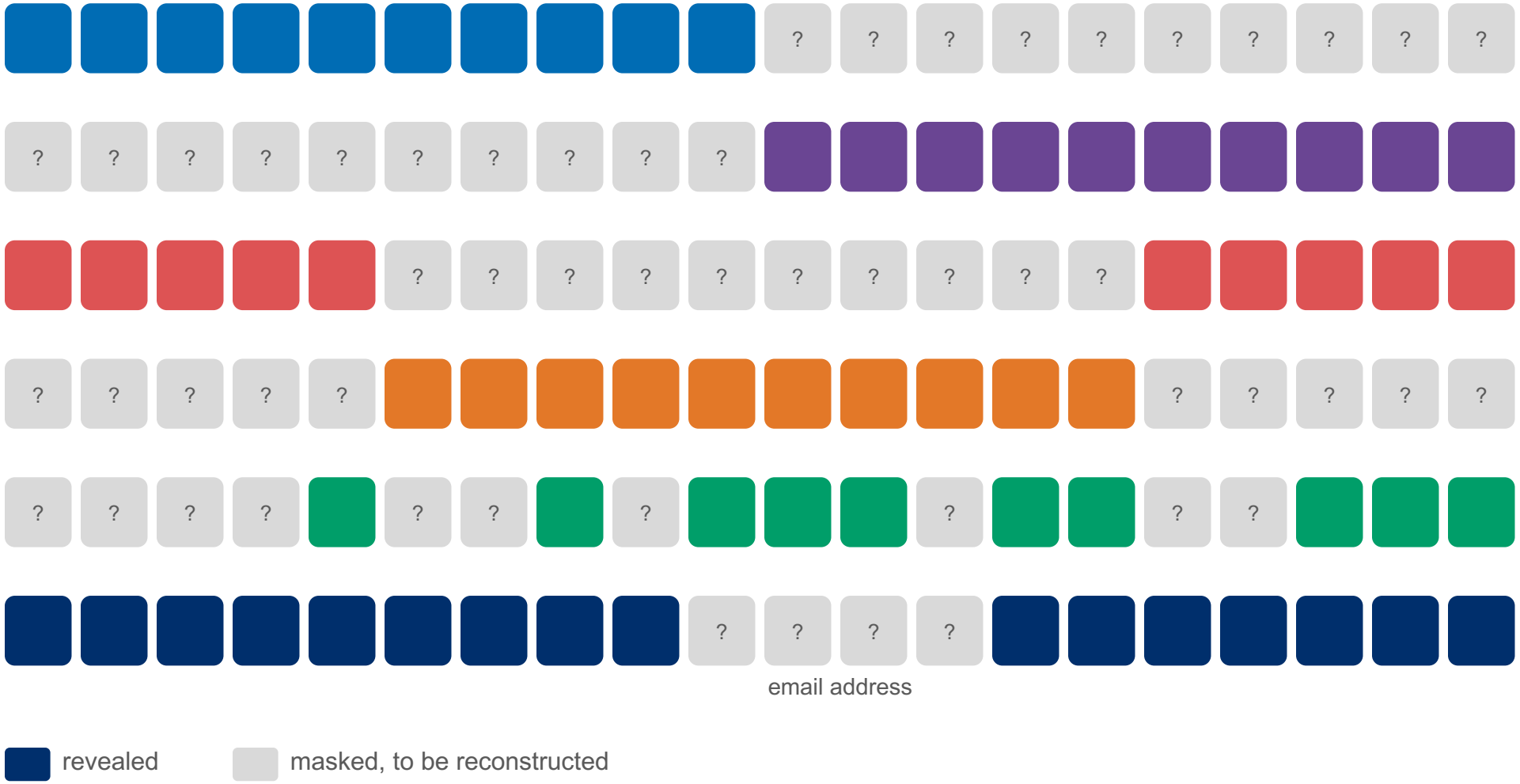
Suffix
last k revealed

Edge
k/2 at each end

Middle
central k revealed

Random
k random positions

Targeted
only the PII is masked



Setup: two DLMs, three pipelines, three corpora

1 Models

LLaDA-8B-Base
8B, trained from scratch

Dream-7B-Base
7B, initialized from Qwen2.5

2 Training

Fine-tuning (FT)
simulates continual pretraining

LoRA[1]
common in production

SFT / Instruction Tuning
chat template, prompt + response

3 Data

MIMIR Pile[3]
verbatim, 10k sequences

Enron emails[4]
PII: emails, phone numbers

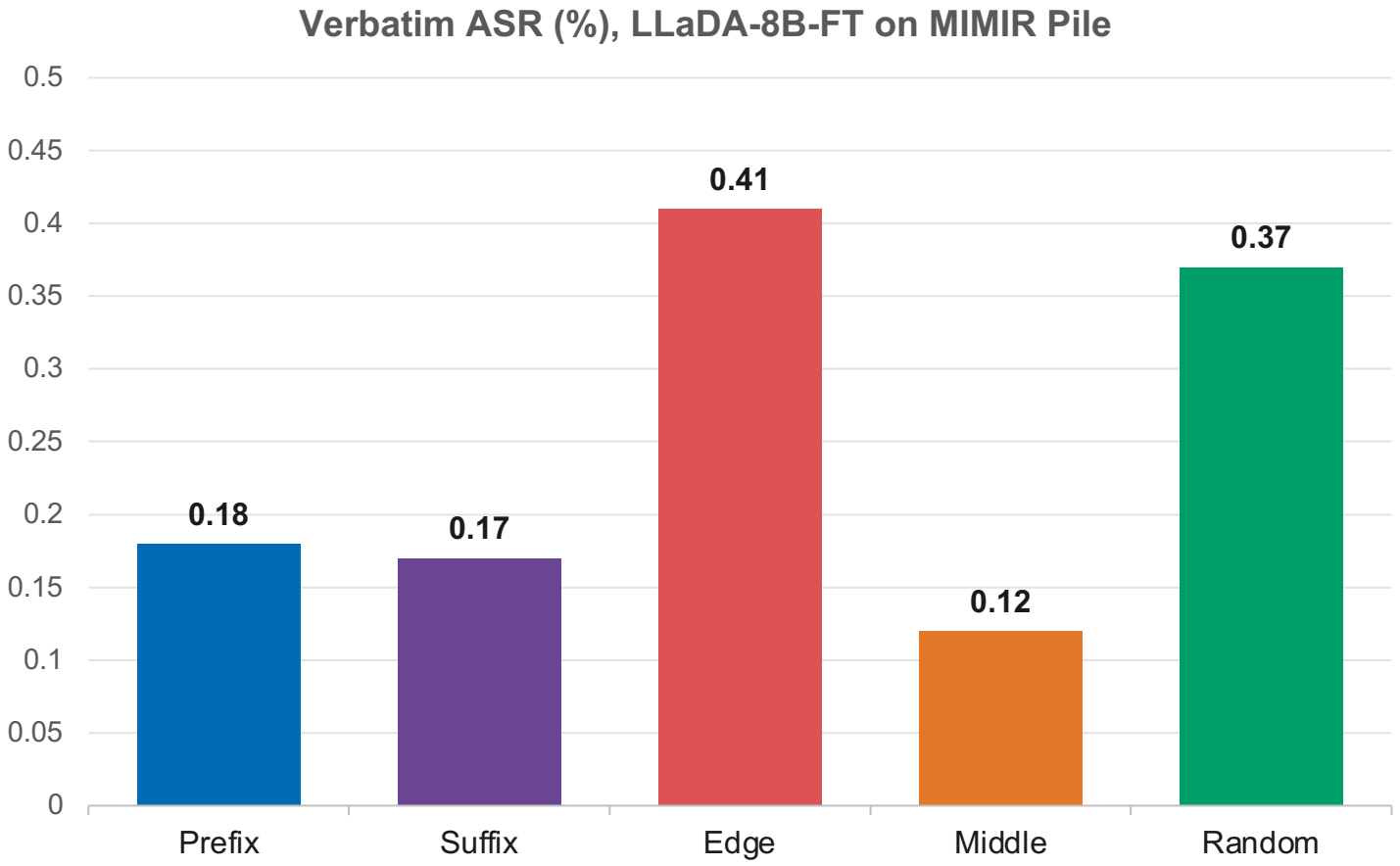
Tulu3[5]
prompt / response leakage

Memorization is induced: 5 epochs of fine-tuning with dLLM[2], then we attack the fine-tuning data

[1] Hu et al., LoRA, ICLR 2022
[2] Zhou et al., dLLM: Simple Diffusion Language Modeling, 2026
[3] Duan et al., MIMIR, COLM 2024
[4] Klimt & Yang, The Enron Corpus, ECML 2004
[5] Lambert et al., Tulu 3, 2024

What infilling reveals

Edge masks leak about 2× more than Prefix



Fixed budget of revealed tokens; mean over 5 seeds

Edge vs Prefix, other settings

0.72% vs **0.29%**

LLaDA-8B-LoRA, verbatim ASR

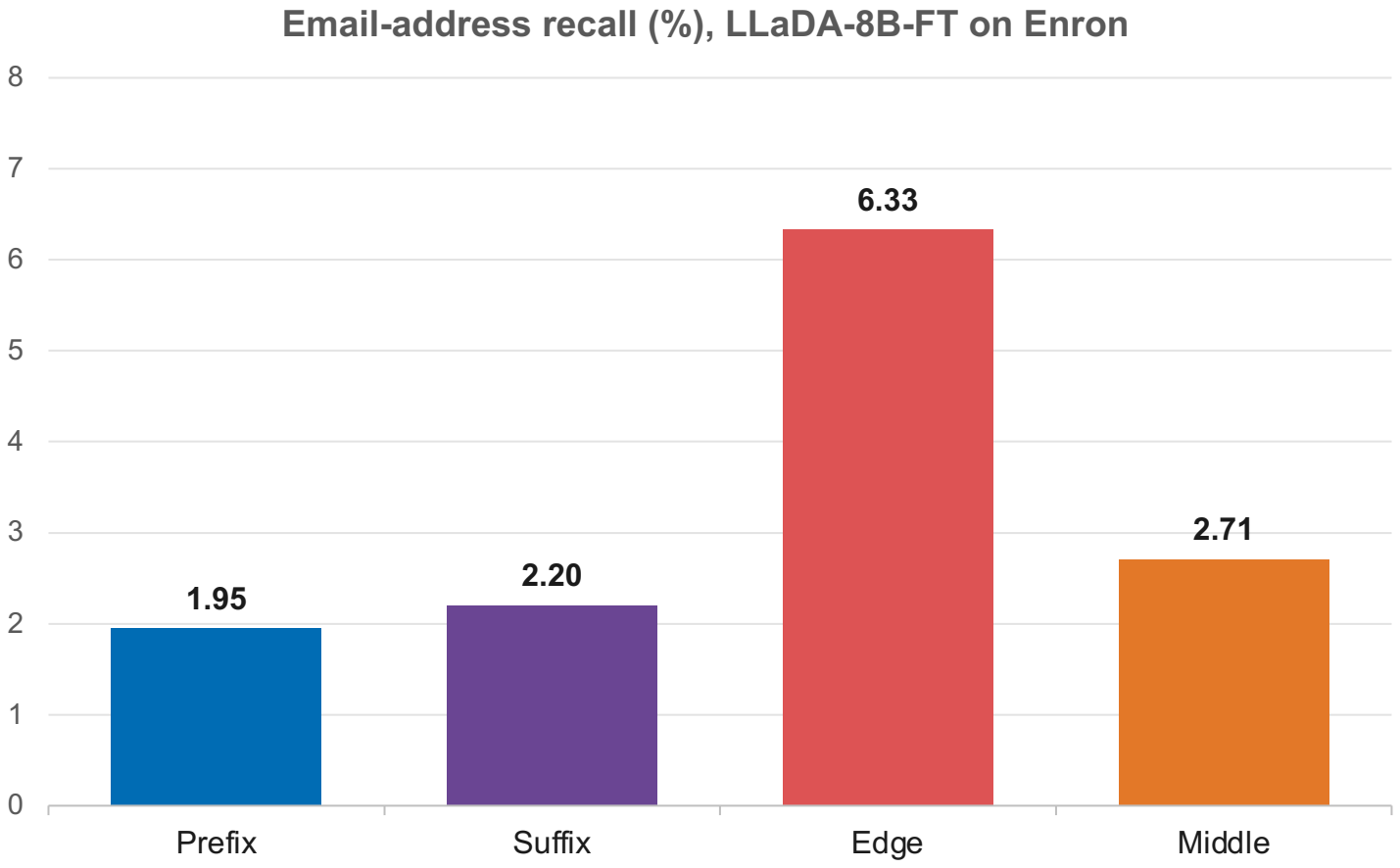
0.14% vs **0.04%**

Dream-7B-FT, verbatim ASR

0.34% vs **0.17%**

Dream-7B-LoRA, verbatim ASR

Two-sided context triples email leakage



Fixed budget of revealed tokens; snippets chosen so the PII falls in the masked region

3.2×

Edge vs Prefix email recall, LLaDA-8B-FT

0.51% → 4.55%

Prefix → Edge, LLaDA-8B-LoRA email recall

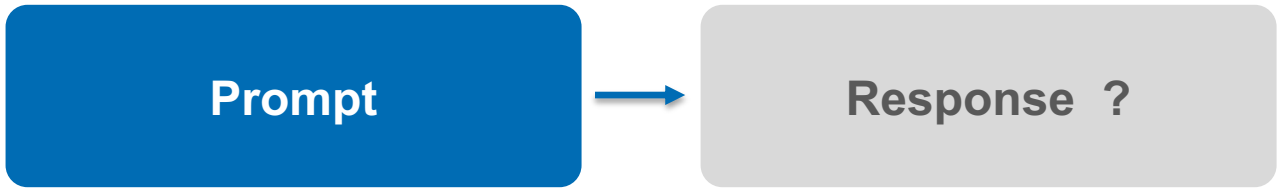
Phones

Edge and Middle also beat Prefix

Bidirectional Nature of DLMs

Recover user prompt from model response

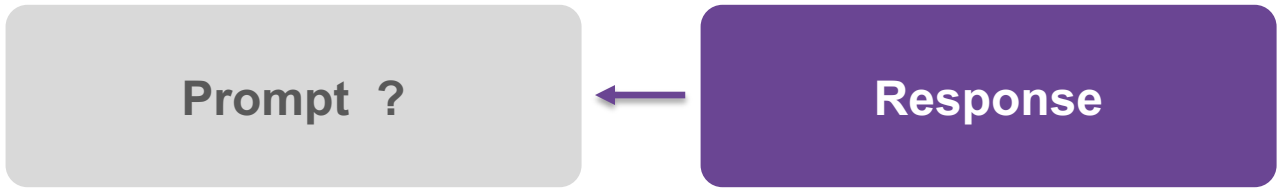
Standard direction: prompt given, response recovered



10.3% / 10.6%

verbatim ASR, LLaDA-8B-SFT / Dream-7B-SFT

Reverse direction: response given, prompt recovered



3.4% / 1.8%

verbatim ASR, LLaDA-8B-SFT / Dream-7B-SFT

ARMs cannot do this: a deployed DLM's output may reveal the query behind it

Targeted infilling: attacker holds redacted copy of data

1 Train

DLM trained on raw sensitive data

2 Redacted release

Data later published with PII removed (court order, regulation[1,2], or a leak)

3 Infill

Attacker masks exactly the redacted spans and asks DLM to fill them

Recovered PII is memorized, not guessed (LLaDA-8B, exact-match recall)

Email 1.4% → **40.5%**

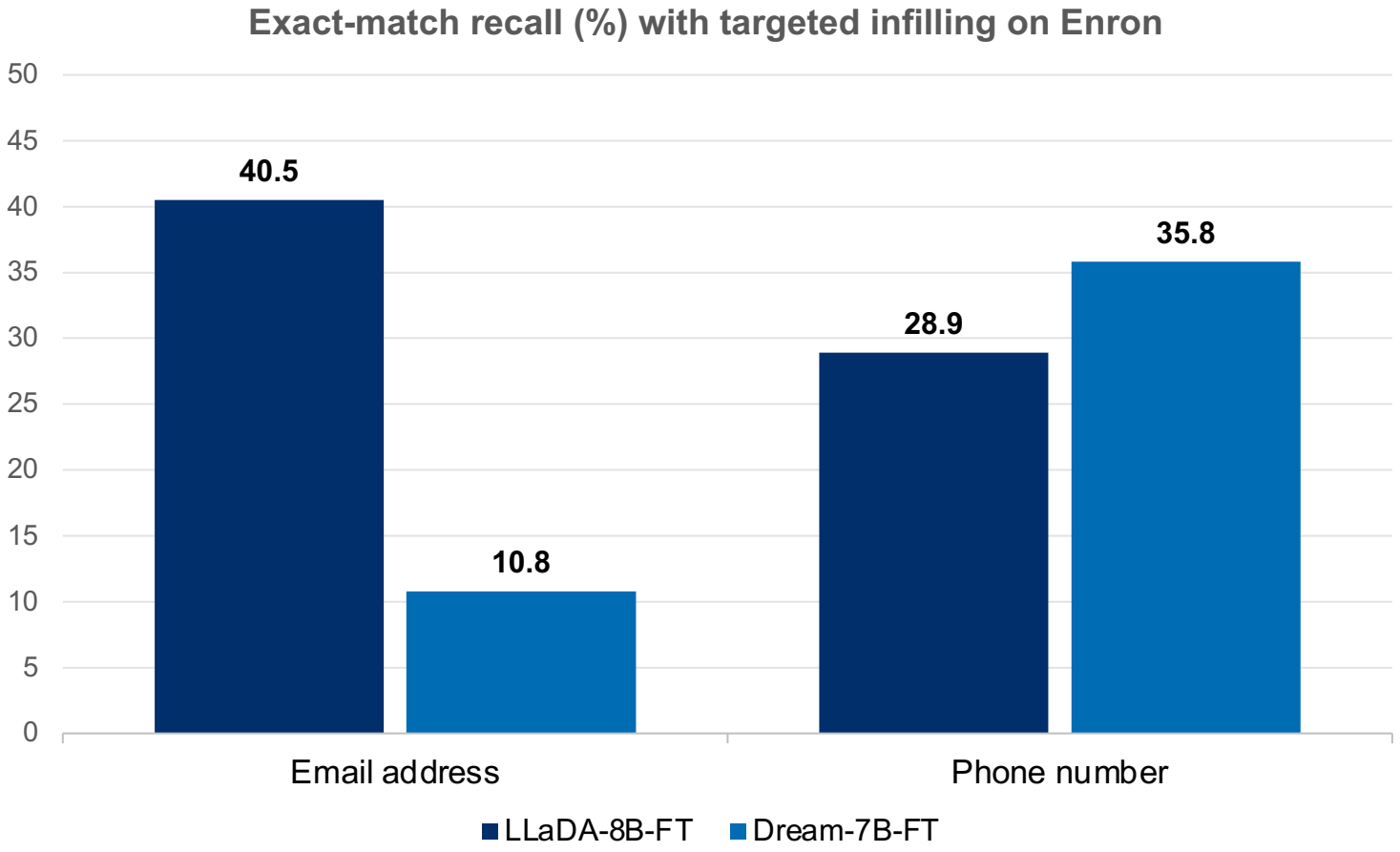
base model → fine-tuned model

Phone 0.3% → **28.9%**

base model → fine-tuned model

[1] EU General Data Protection Regulation, Regulation (EU) 2016/679 eur-lex.europa.eu
[2] U.S. HHS, Guidance on de-identification of PHI under the HIPAA Privacy Rule, 2012 www.hhs.gov

Reacting before release does not protect



Only the PII span is masked; the rest of the email is revealed

Up to 2 wrong tokens

47.3% / 41.3%

LLaDA-8B-FT: email / phone

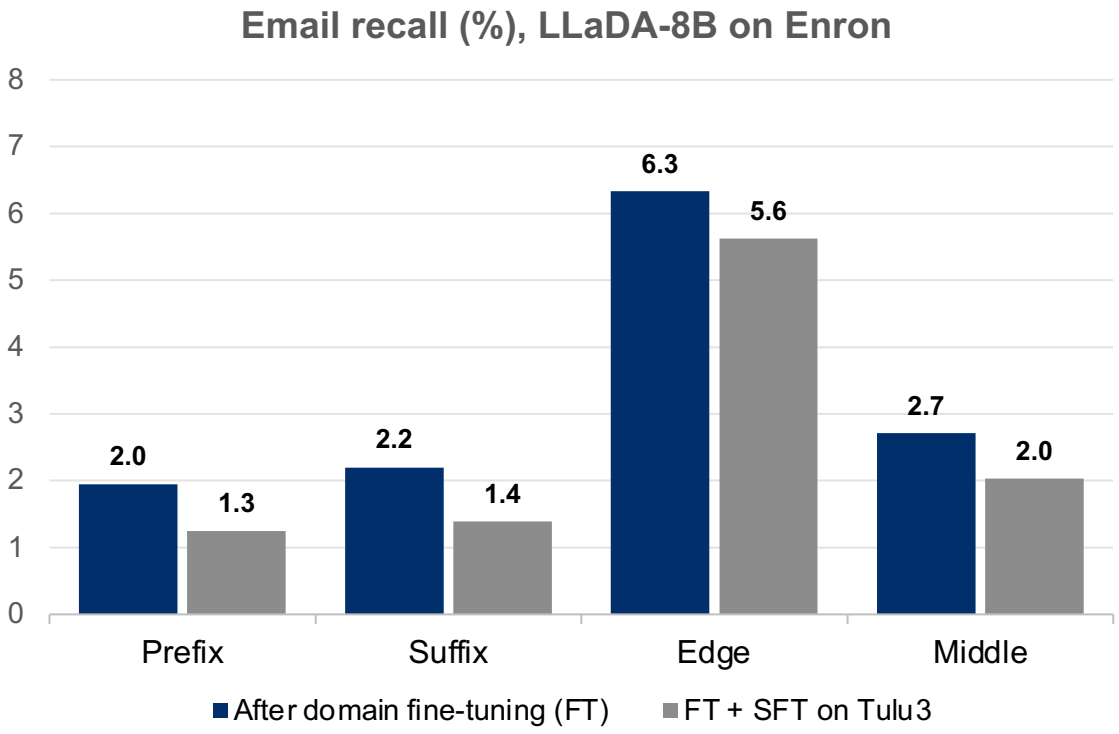
27.0% / 47.9%

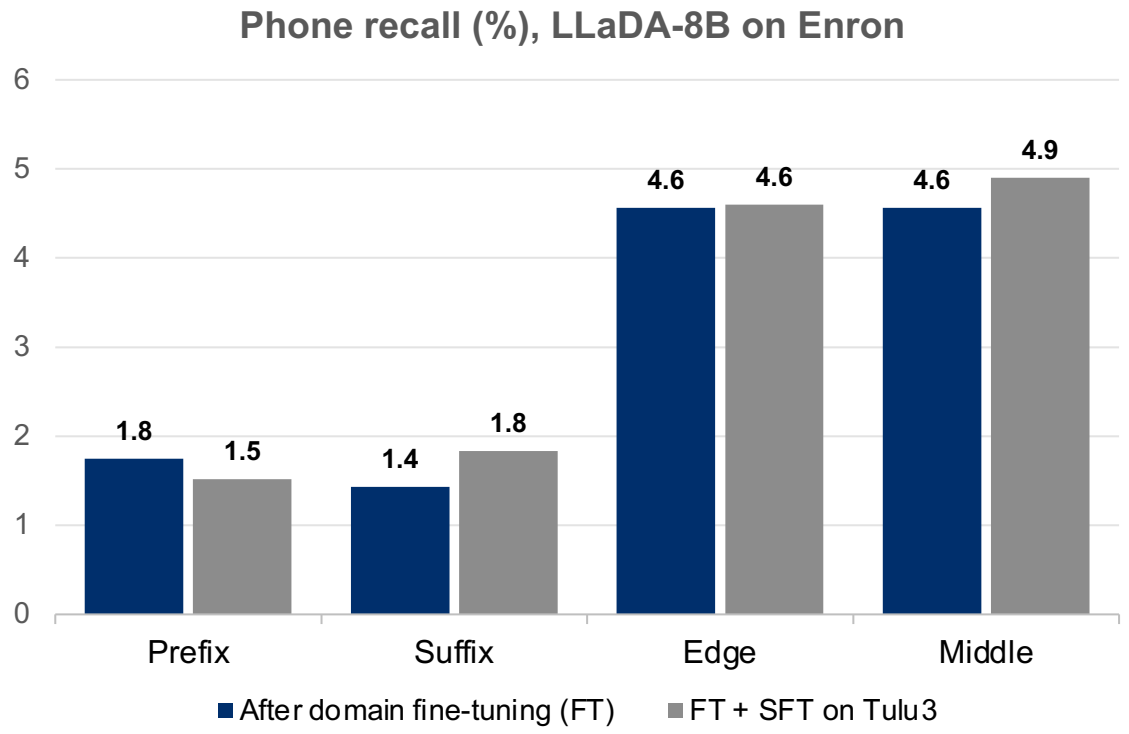
Dream-7B-FT: email / phone

Enable enumeration attacks

Lesson: Redact training data too, not just what you publish

A later SFT stage does not erase memorized PII





Emails: recall drops in every mode, but stays well above zero

Phones: unchanged, or slightly higher in 3 of 4 modes

Ordering: Edge and Middle stay strongest, so the channel remains open

How decoding settings affect extraction

More denoising steps, more leakage

Denoising steps T

How many predict-and-commit rounds the decoder runs.

Fewer steps means more tokens committed at once

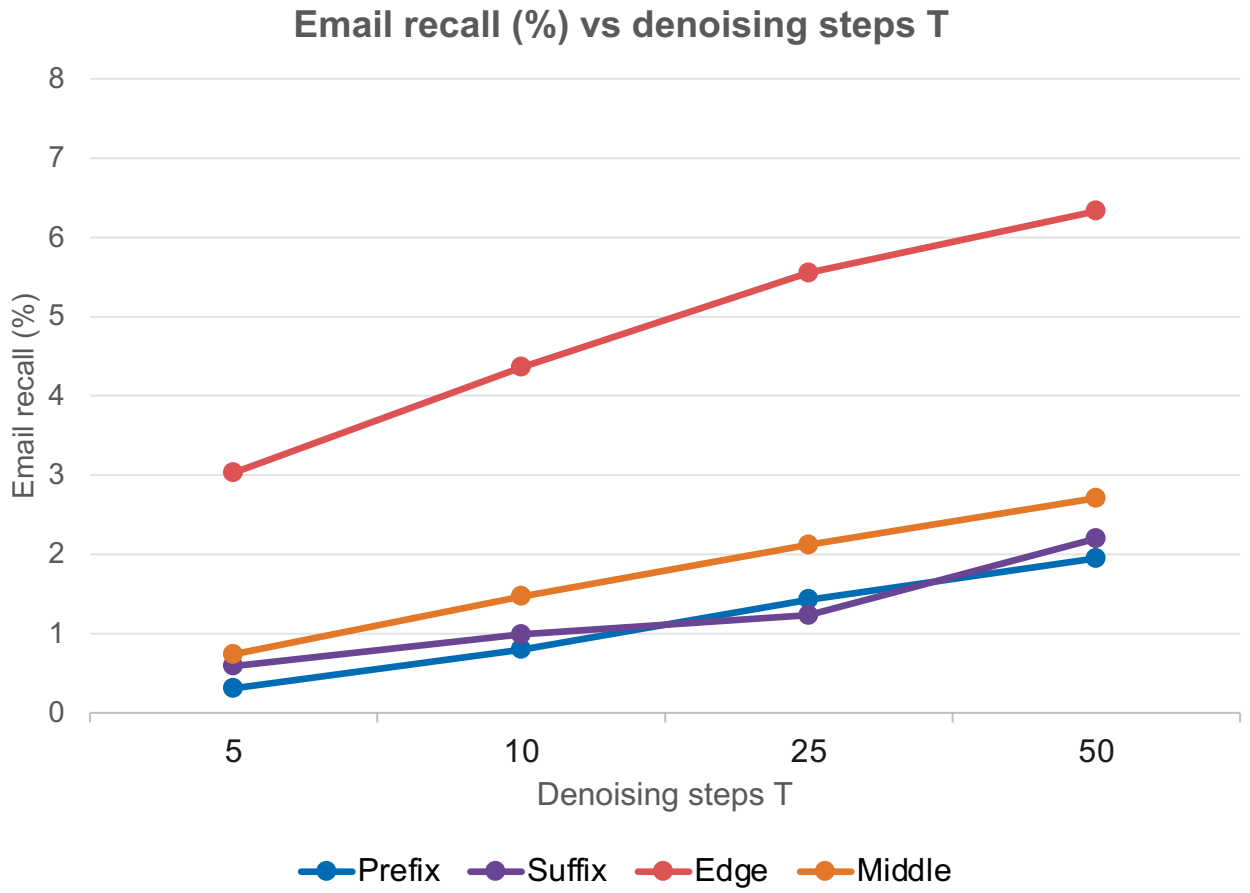
T = 2: 4 tokens per step



T = 4: 2 tokens per step



Finding:
recall rises with every extra step



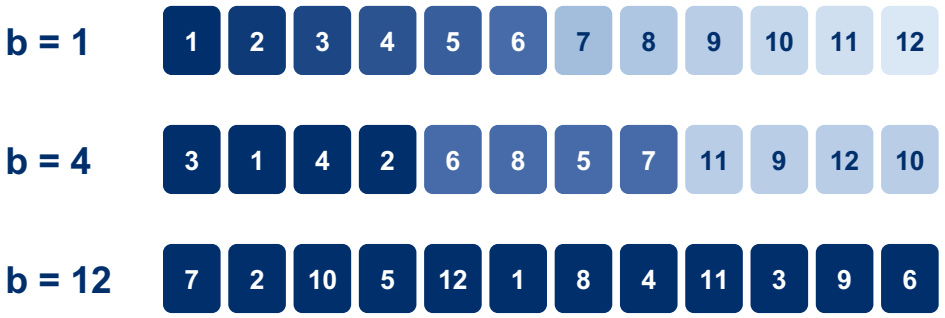
LLaDA-8B-FT, email recall on Enron; one setting varied, others at default

Larger blocks leak more

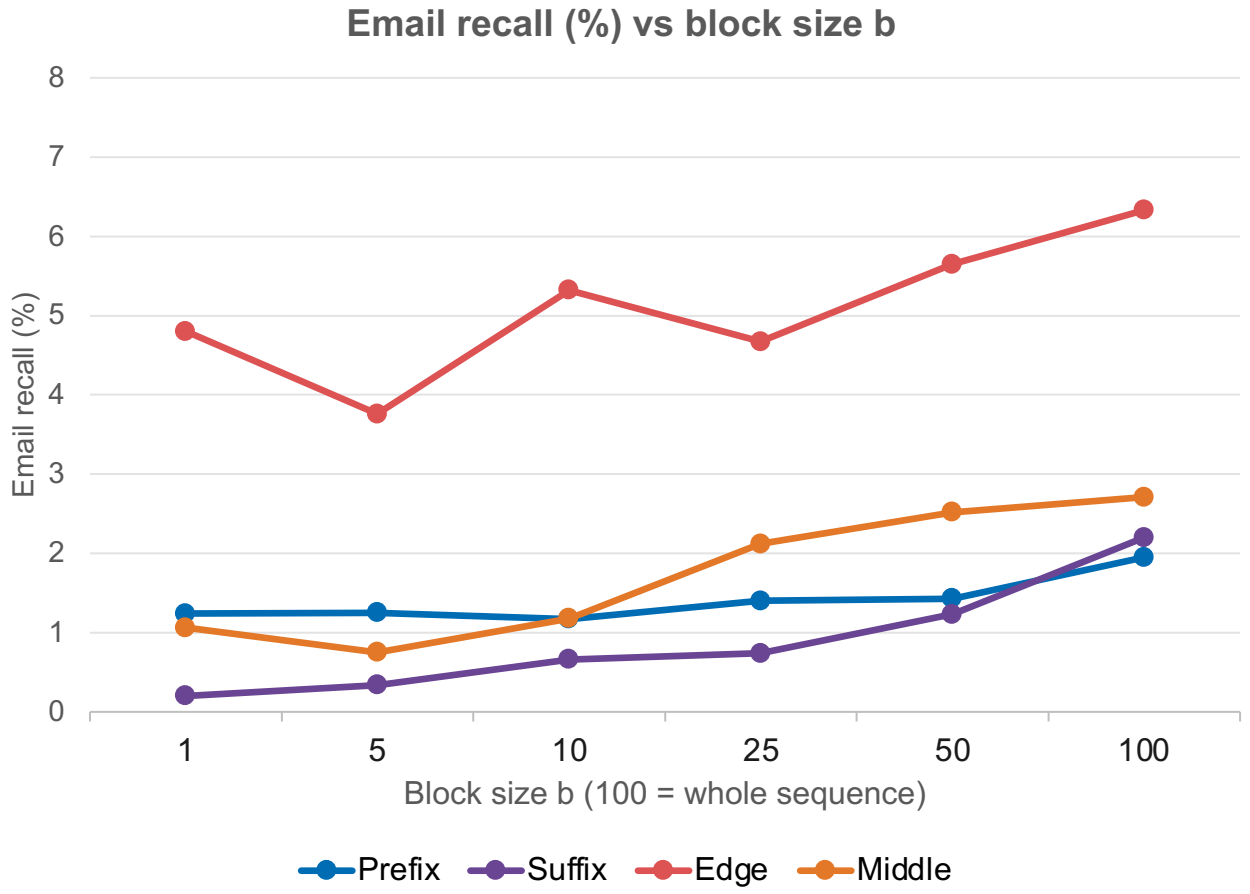
Block size b

Decode block by block, left to right; inside a block, any order in parallel[1].

Small b ≈ autoregressive



Finding:
recall grows once b leaves the AR-like regime



LLaDA-8B-FT, email recall on Enron; one setting varied, others at default

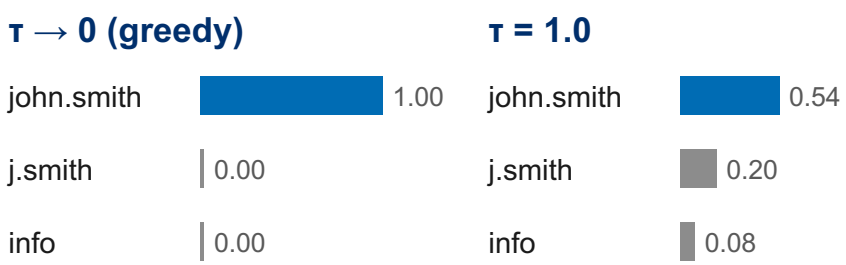
[1] Arriola et al., Block Diffusion: Interpolating between AR and diffusion LMs, ICLR 2025

Temperature barely matters

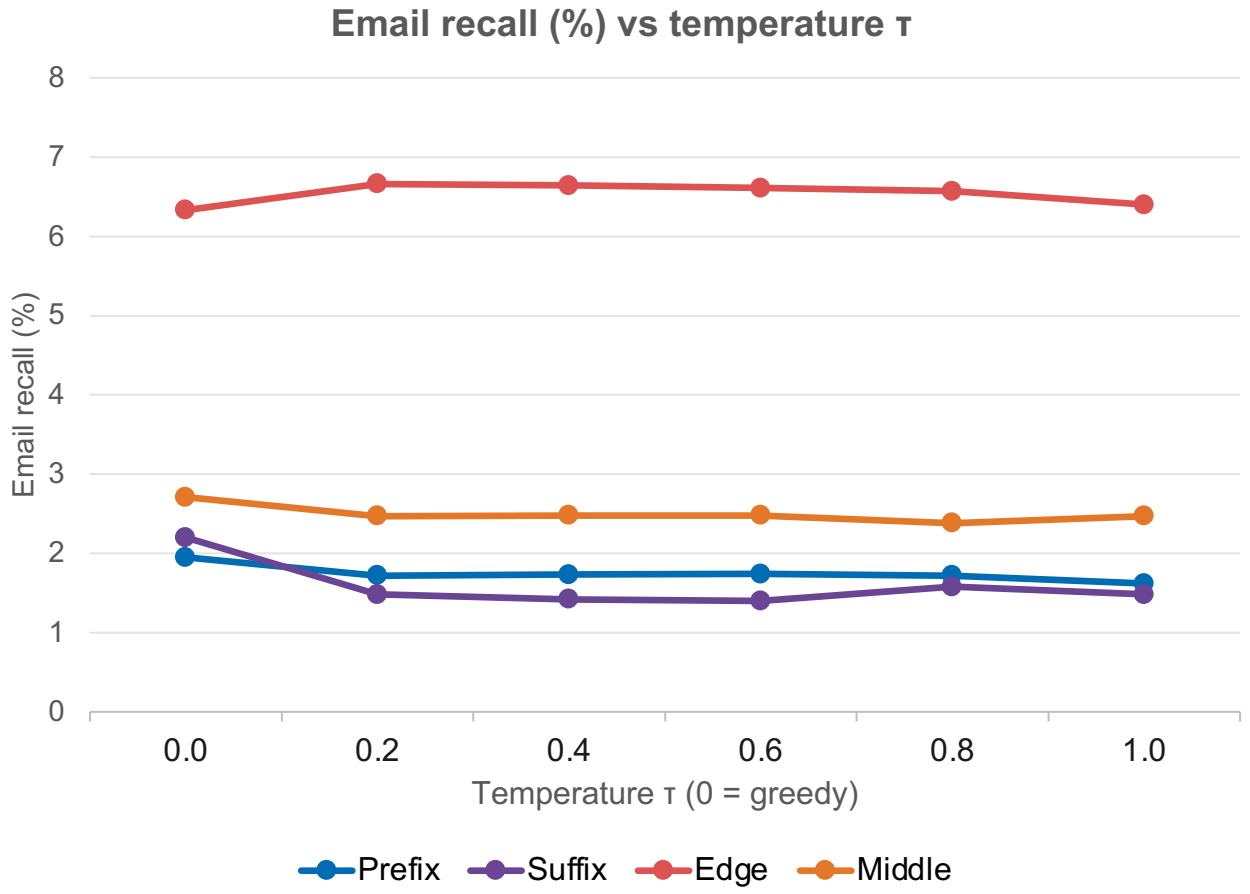
Temperature τ

How a token is drawn from each predicted distribution;
τ → 0 is greedy, higher τ is flatter

[MASK] @enron.com



Finding:
recall stays flat; raising τ is no defense



LLaDA-8B-FT, email recall on Enron; one setting varied, others at default

Confidence remasking leaks more than random

Remasking

Each step predicts every masked position, keeps some, re-masks the rest

Predictions (with confidence)

john	will	send	the	draft	today
0.93	0.41	0.38	0.88	0.22	0.35

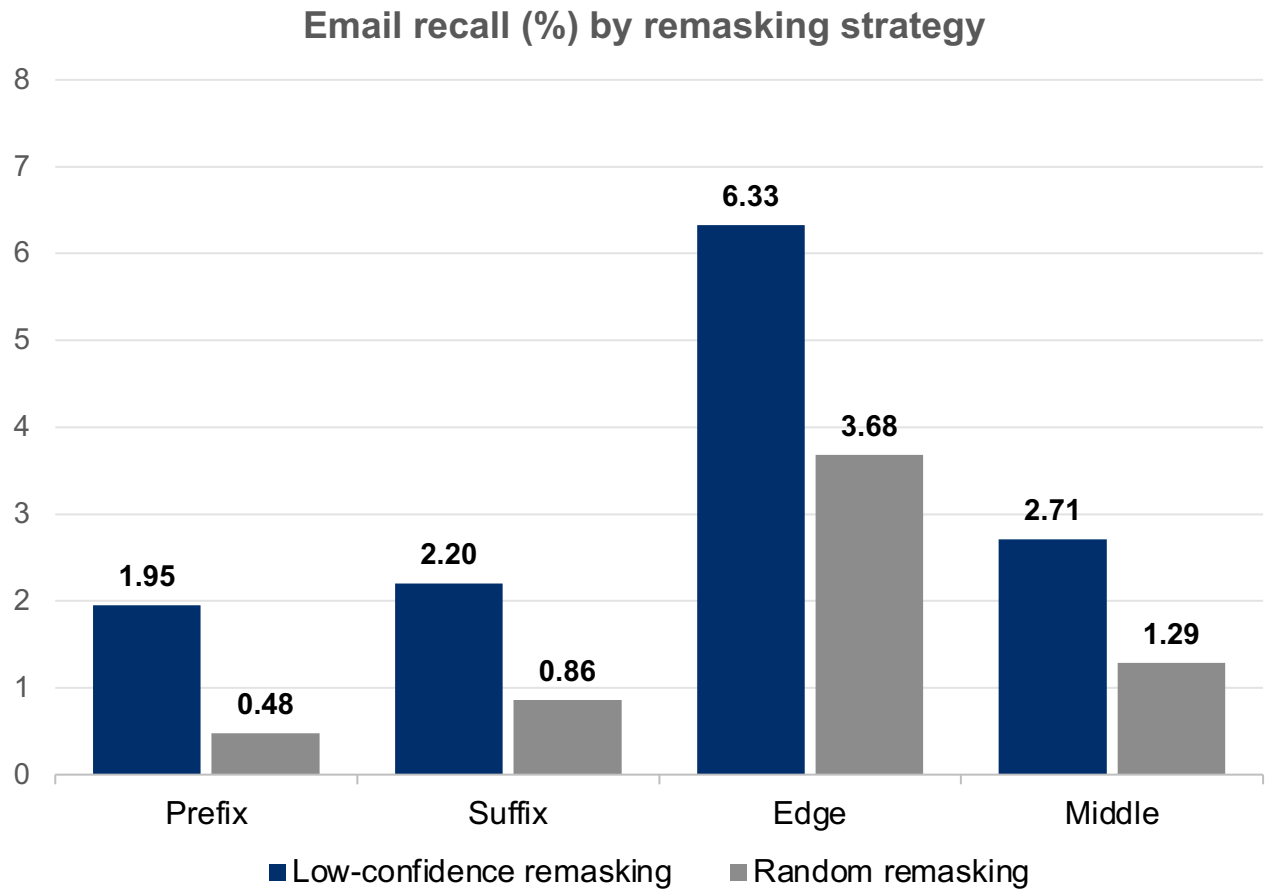
Low-confidence remasking: keep the surest

john	[M]	[M]	the	[M]	[M]

Random remasking: keep any

[M]	[M]	send	[M]	draft	[M]

Finding: confidence itself marks training data



LLaDA-8B-FT, email recall on Enron; one setting varied, others at default

Open challenges

1

Fair DLM vs ARM comparison

Architectures, objectives and training data differ; how do we compare extraction risk on equal terms?

2

Memorization from pretraining

We induce memorization by fine-tuning; measuring it on base-model pretraining data is still open

3

Defenses

Beyond decoding settings: differentially private training, deduplication, unlearning for DLMs

4

Evolving DLMs

New training recipes (e.g. preference tuning) and faster decoders: how do they change leakage?

Takeaways

1

Where you mask decides what leaks

Edge masks give ~2× the verbatim ASR and ~3× the email recall of Prefix; prefix-only probing underestimates risk

2

Infilling opens new leakage channels

Prompts are recovered from answers and 40.5% of redacted emails are filled back in; redact training data too

3

Memorization is sticky

A later SFT stage weakens email leakage but leaves phone numbers and the mode ordering intact

4

Decoding settings are leakage dials

More steps, larger blocks and confidence-based remasking amplify extraction; temperature barely matters

Project website: https://ssg-research.github.io/mlsec/dlminfillingextraction