# Training Language Models with Memory Augmentation

Zexuan Zhong, Tao Lei, Danqi Chen
Princeton University

Paper Code

## Motivation

**Memory augmentation** can enhance language modeling performance without increasing the model size!
But in existing memory-augmented LMs,

1. Memory is constructed using a **standard** LM only during **inference** (e.g., kNN-LM[1], cont cache[2])
2. Memory representations are **stale**; no back-propagation to update memory representations (e.g., T-XL[3])

How can we **train** memory representations?

## Our Approach: TRIME

TRIME: Training with in-batch memory

**Memory** $M$: a set of context-token pairs. $M = \{(c_i, x_i)\}$

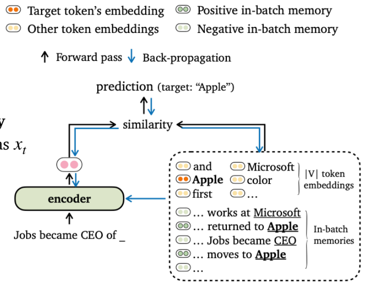
**Training objective:**

$c_t$: Jobs became CEO of ___

1) 🟠 **Apple** (output embedding)

2) Other $c$ in the **training memory** that share the same next word as $x_t$

- 🟢 … returned to **Apple**
- 🟢 … moves to **Apple**



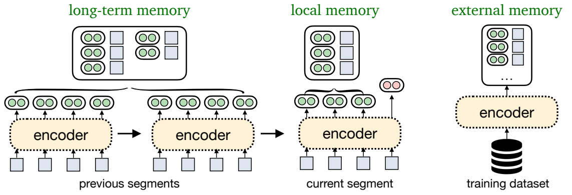
output embedding

$$P(w \mid c_t) \propto \boxed{\exp(E_w^\top f_\theta(c_t))} + \sum_{\boxed{(c,x)\in\mathcal{M}_{\text{train}}}} \mathbb{I}(x = w)\boxed{\exp(\frac{g_\theta(c_t)^\top g_\theta(c)}{\sqrt{d}})}$$

training memory

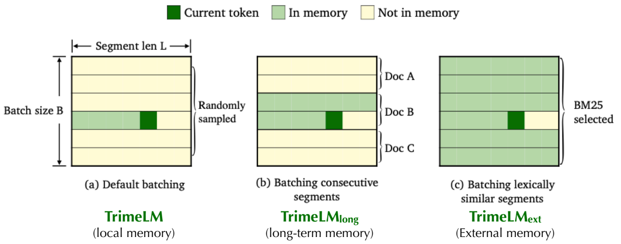
hidden embedding (input of last FFN)

**Three types of memory during inference:**



## Three TRIME Language Models

We propose different **data batching** and **memory construction** methods to train **three language models**, which are optimized to leverage different memories at the testing time.



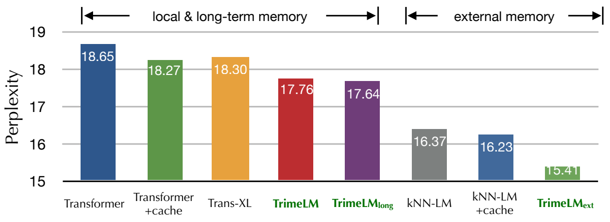
(a) Default batching
**TrimeLM**
(local memory)

(b) Batching consecutive segments
**TrimeLM$_{\text{long}}$**
(long-term memory)

(c) Batching lexically similar segments
**TrimeLM$_{\text{ext}}$**
(External memory)

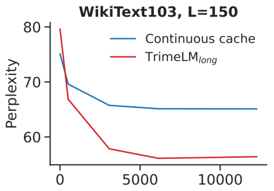
## Experiments: WikiText-103

Model size = 247M, segment length = **3072**



Model size = 150M, segment length = **150**

| Model | Dev (↓) | Test (↓) |
|---|---|---|
| Transformer | 28.11 | 29.14 |
| Transformer-XL | 23.42 | 24.56 |
| Compressive Transformer | - | 24.41 |
| ∞-former | - | 24.22 |
| LaMemo | 22.98 | 23.77 |
| Transformer (our run) | 25.31 | 25.87 |
| + continuous cache* | 22.95 | 23.59 ↓2.28 |
| ★ TRIMELM | 24.45 | 25.60 ↓0.27 |
| ★ TRIMELM$_{\text{long}}$ | **21.76** | **22.66** ↓3.21 |



We train with segment length **150** but the model is able to leverage **15,000** tokens at testing time!

## Domain Adaptation

| Model | $\mathcal{M}_{\text{ext}}$ | Dev (↓) | Test (↓) |
|---|---|---|---|
| Transformer | - | 62.72 | 53.98 |
| ★ TRIMELM | - | 59.39 | 49.25 |
| ★ TRIMELM$_{\text{long}}$ | - | 49.21 | **39.50** |
| kNN-LM + cont. cache | WIKI | 53.27 | 43.24 |
| ★ TRIMELM$_{\text{ext}}$ | WIKI | 47.00 | 37.70 |
| kNN-LM + cont. cache | BOOKS | 42.12 | 32.87 |
| ★ TRIMELM$_{\text{ext}}$ | BOOKS | 36.97 | **27.84** |

We train the models on **WikiText-103** and evaluate them on **BooksCorpus**.

Although memory representations are optimized on one domain, our approach does **not** overfit!

## Machine Translation

| Model | BLEU (↑) |
|---|---|
| Transformer enc-dec | 32.58 |
| kNN-MT | 33.15 ↑0.57 |
| ★ TRIMEMT$_{\text{ext}}$ | **33.73** ↑1.15 |

Our approach can be easily applied to other generation tasks, such as machine translation! We apply TRIME on IWSLT'14 En-De task.

References

[1] Khandelwal et al., 2021. Generalization through memorization: Nearest neighbor language models.
[2] Grave et al., 2017. Improving neural language models with a continuous cache.
[3] Dai et al., 2019. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context