CO-LMLM: CONTINUOUS-QUERY LIMITED MEMORY LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Externalizing knowledge in LLM pre-training is a promising avenue to achieve higher performance at smaller scales, control knowledge use, and overall increase model transparency. We propose continuous-query limited memory language models (CO-LMLM), an LLM that interleaves flexible vector retrieval queries with next-token predictions, and is pre-trained to copy knowledge returned from the KB, rather than memorize it. CO-LMLM is pre-trained with a scalable approach that jointly trains a knowledge-externalizing LLM, induces its knowledge base, and learns an expressive continuous retrieval mechanism. Across pre-training at multiple model scales, CO-LMLM outperforms prior knowledge-externalizing and vanilla LLMs in both perplexity and factual precision. At 360M scale, this includes lower perplexity than models pre-trained on 40× more data, and SimpleQA-verified performance that is in line with gpt-4o-mini and higher than Claude Sonnet 4.5.

1 INTRODUCTION

Recently, there has been increasing interest in large language models (LLMs) that are pre-trained to externalize (Zhao et al., 2026) or separate knowledge (Ghosal et al., 2025; 2026; Pouransari et al., 2026). Knowledge externalization departs dramatically from how knowledge is usually treated in LLM pre-training, where knowledge is completely indistinguishable from linguistic and reasoning skills, both in its acquisition and representation in parameter space. The potential of externalization is significant. LLMs that do not store knowledge can be much smaller, and train much faster, simply because less is asked from them. Externalization and separation open clean avenues for unlearning with little or no utility costs, and if the pathways for using knowledge remain legible, to monitoring, auditing, and overall increased transparency of model function. Critically, when done during pre-training, externalization does not rely on brittle post-training, and stops the integration of knowledge into the parameters at the source.

We propose continuous-query limited memory language models (CO-LMLM), an LLM that interleaves vector retrieval queries with next-token predictions, and is pre-trained to copy knowledge returned from the knowledge base (KB), rather than memorize it. The KB stores vector keys and string values, with each entry grounded to a source document. This allows for expressive retrieval pathways, where pre-training has the full flexibility of neural representation to shape retrieval representations, while retrieved values remain legible and interpretable. CO-LMLM is pre-trained via a scalable approach to jointly train a knowledge-externalizing LLM, induce its KB, and learn an expressive continuous retrieval mechanism. Training follows the common scalable next-token-prediction pre-training, with two key differences: pre-processing the data to simulate knowledge retrieval and adding a retrieval objective, both designed for pre-training scales.

We evaluate CO-LMLM against REL-LMLM (Zhao et al., 2026), a prior LMLM-class method that relies on relational KB and retrieval, and standard baselines trained without knowledge externalization. Across model scales, CO-LMLM achieves lower perplexity and substantially improves factual precision over both REL-LMLM and standard models, while preserving downstream NLU performance. At the 360M scale, CO-LMLM achieves lower perplexities than off-the-shelf HF/SMOLLM2-360M, which is pre-trained on 40× more data, and SimpleQA-verified (Haas et al., 2026) scores that are in line with gpt-4o-mini and higher than Claude Sonnet 4.5. At the same time, CO-LMLM retains the controllability benefits of the LMLM class: its external memory remains editable, human readable, auditable, and supports direct unlearning through KB operations.

Code, data, and models will be available upon publication.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

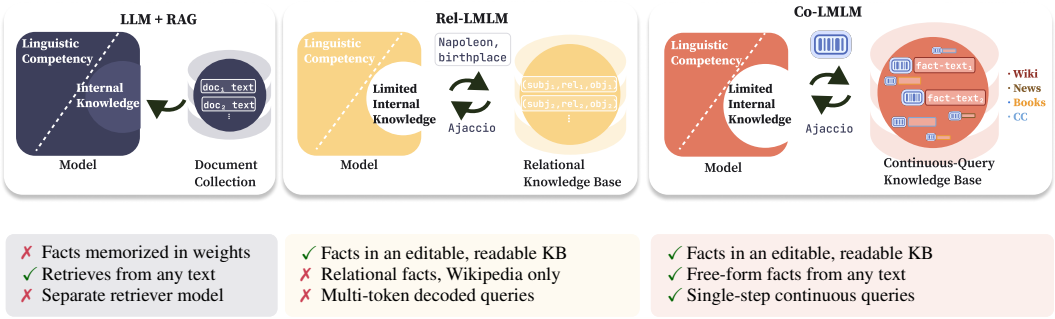


Figure 1: **Knowledge separation across three regimes.** A standard LLM with RAG retrieves over external documents but keeps factual knowledge in its parameters (*left*); REL-LMLM externalizes facts to a relational KB queried with an explicit decoded query (*middle*); CO-LMLM externalizes facts to an unstructured index, retrieved directly from the model's hidden state (*right*). *Bottom:* some of the advantages and limitations of each regime.

2 RELATED WORK

Parametric Knowledge and its Limitations Pre-trained language models store factual knowledge implicitly in their parameters and recall it at generation time (Petroni et al., 2019; Roberts et al., 2020). This form of storage is inherently limited: factual recall scales with model size and exposure frequency, making long-tail knowledge difficult to capture (Kandpal et al., 2023; Mallen et al., 2023). Moreover, factual associations are entangled with linguistic representations, making individual facts difficult to attribute, edit, or remove, and contributing to hallucinations and stale knowledge (Elhage et al., 2022; Dai et al., 2022; Huang et al., 2024). They also make *knowledge editing* and *unlearning* challenging: removing or modifying specific facts typically requires additional training or specialized objectives, often trading off with general capabilities (Maini et al., 2024). MemSinks isolates memorization-specific parameters, but still keeps knowledge within the model (Ghosal et al., 2025). These challenges motivate approaches that move factual knowledge into separate, editable stores.

Externalizing Knowledge from Model Weights A growing line of work augments language models with non-parametric or semi-parametric memory, or aims to move factual knowledge more explicitly *out of model parameters*. kNN-LM retrieves nearest-neighbor examples from an external datastore and interpolates their probabilities with the model distribution (Khandelwal et al., 2020), while RETRO retrieves neighboring chunks and conditions on them through cross-attention (Borgeaud et al., 2022). Other approaches introduce sparse memory layers or learned key-value memories to increase model capacity and factuality (Lample et al., 2019; Berges et al., 2025; Wu et al., 2022; Peng et al., 2023; Yang et al., 2024; Cheng et al., 2026; Tseng & Sa, 2026). These methods can improve perplexity or factuality, but their memory is not directly inspectable or editable.

Limited Memory Language Models Zhao et al. (2026) is the most related work to ours. Figure 1 illustrates some of the similarities and differences. They introduced Limited Memory Language Models (LMLMs), which externalize factual knowledge into a textual relational KB during pre-training rather than encoding it in model parameters. We refer to this approach as REL-LMLM to distinguish it from our continuous-query formulation. REL-LMLM enables faster pre-training, interpretable retrieval, and direct editing, but its relational design limits the external memory coverage: facts must be expressible as Wikipedia-style subject-relation-object tuples, and retrieval depends on brittle textual queries that introduce additional generation token overhead. REL-LMLM training relies on data pre-processing to simulate KB queries, and otherwise follows conventional next-token-prediction pre-training. Although scaleable, it limits the expressivity of retrieval pathways, as the simulated relational queries dictate the exact form of querying. CO-LMLM preserves the benefits of an externalized, editable KB while relaxing these constraints. It replaces relational tuples with free-form textual knowledge spans, and decoded relational queries with continuous queries emitted from the model's hidden states. These queries can carry richer contextual information for retrieval, allowing CO-LMLM to scale beyond Wikipedia-style relational facts to general corpora (e.g., FINEWEB-EDU), while reducing the token overhead of query generation. The CO-LMLM pre-

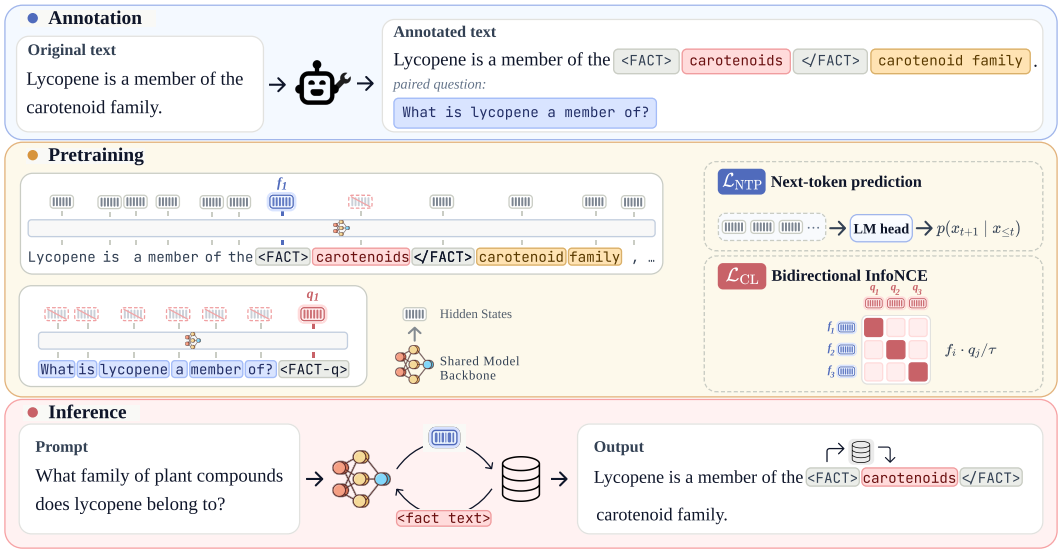


Figure 2: **Overview of Co-LMLM.** *Top:* We annotate the pre-training corpus with fact spans and paired questions, while leaving the document text unchanged. *Middle:* During training, a single decoder-only Transformer processes both the annotated document and its paired questions. We optimize next-token prediction over non-fact tokens and a bidirectional InfoNCE loss that aligns the query representation at each <FACT> token with the question representation at the corresponding <FACT-q> token. *Bottom:* At inference, emitting <FACT> triggers dense retrieval using the hidden state at the emitted <FACT> position as the query. The retrieved fact span is inserted into the context, followed by </FACT>, and decoding continues. No textual query is decoded.

training process is rooted at the pre-processing approach of REL-LMLM, but avoids from dictating retrieval representations, instead leaving these for learning.

Retrieval-Augmented Language Models Retrieval-Augmented Generation (RAG), also illustrated in Figure 1, conditions generation on retrieved passages during inference (Lewis et al., 2020; Izacard & Grave, 2021), while retrieval-aware pre-training methods incorporate retrieval into model training (Guu et al., 2020; Borgeaud et al., 2022; Izacard et al., 2023). Other methods let models decide when to retrieve and issue textual queries, as in Self-RAG, DRAGIN, and LMLM (Asai et al., 2024; Su et al., 2024; Zhao et al., 2026). Our technique is related to methods that generate retrieval signals directly from internal representations rather than decoded queries (Muennighoff et al., 2025; Zhang et al., 2024a; 2025). Unlike most retrieval-augmented methods, which add external context while leaving parametric knowledge largely intact, Co-LMLM uses retrieval to define the parametric/non-parametric boundary during pretraining: factual spans are externalized into a schema-free index, and the model learns when and how to retrieve them through a contrastive objective trained jointly with next-token prediction.

3 METHOD

The Co-LMLM model is a decoder-only Transformer, which functions as both a language model and a dense retriever. Figure 2 provides a high-level overview of the method. The model generates query vectors similar to tool calling, and the returned values are encoded by the model, before decoding resumes (Section 3.1). We jointly train the language modeling and retriever functions (Section 3.2), using pre-training data that is pre-processed to simulate retrieval, paired with synthetic questions that provide supervision for the contrastive retrieval objective (Section 3.3).

3.1 MODELING AND INFERENCE

Co-LMLM performs retrieval through continuous tool calls. The external memory is a key-value store: each key is a dense vector produced by the model, and each value is a factual text span. Unlike

standard tool-calling methods (Schick et al., 2023), the retrieval query is not represented as text. Instead, the query is the model's hidden state at a special token.

At inference time, the model decodes tokens autoregressively. When it emits the special <FACT> token, we read the last-layer hidden state at that position as a retrieval query, query the index, and retrieve the top-1 fact text snippet. We splice the retrieved snippet into the context immediately after <FACT>, append a closing </FACT> special token, and resume autoregressive decoding from </FACT> onward. Each retrieval therefore costs one extra autoregressive step (the opening <FACT> token) and grows the context by the length of the retrieved span.

3.2 Training

Our training data consists of raw text in which factual spans are enclosed by <FACT> and </FACT>. Each marked span is paired with a natural-language question whose answer is the span. Figure 2 (top) illustrates a single example. Section 3.3 describes how we pre-process raw pre-training data to create this data. Our objectives are to train the model to both predict the next token and externalize the information in the delimited spans, construct a KB from all such spans, and to emit retrieval queries to fetch spans from the KB as needed. The key to training the retrieval function is contrastive query pairs between documents and questions, allowing for both positive and negative pairs.

Let x be an input document with factual spans bracketed by <FACT> and </FACT>. For each factual span, we have a question Q. The question is synthesized so its answer is the corresponding bracketed span in x, and it is appended with a query marker token <FACT-q>. The retrieval vectors at the query markers appended to the questions are used as positive examples for the contrastive retrieval loss. The pre-training panel of Figure 2 (middle) visualizes how x and Q are used.

Let $p(x_t \mid x_{<t}; \theta)$ be the probability of the token x_t in the sequence document x, parameterized by θ. In our model, p_θ is computed by a Transformer. We denote $\phi_t(x_{<t}; \theta)$ as the features computed by the Transformer for index t in sequence x.[1] This is the same Transformer used for p_θ.

We jointly minimize next-token prediction (NTP) and bidirectional contrastive losses. The NTP loss is: $\mathcal{L}_{\text{NTP}}(\theta) = -\sum_{t \notin M} \log p(x_t \mid x_{<t}; \theta)$, where M is the set of positions in x that follow an opening <FACT>, up to and including its closing </FACT>. This NTP formulation means that we do not optimize our model to memorize the bracketed fact tokens, which represent the knowledge we aim to externalize. This is a critical design choice to externalize knowledge. We do optimize the loss of the opening <FACT> token, because the model must know to generate it in order to trigger retrieval. The closing </FACT> is appended mechanically, so it does not require optimization.

The contrastive loss treats each fact query from the document x and its corresponding question Q as a positive pair, and all other questions as negatives. For a fact that starts at index t in a document x, let $f = \phi_t(x_{<t}; \theta)$ be the query representation for retrieving this fact, i.e., the feature representation for the <FACT> token opening this fact span. Correspondingly, for a question Q where the query marker <FACT-q> is at index t, let $q = \phi_t(Q_{<t}; \theta)$ be the question representation, i.e., the feature representation computed for <FACT-q>. We use the InfoNCE loss (van den Oord et al., 2018), computed on the current batch. Let $\mathcal{B} = \{(f^{(i)}, q^{(i)})\}_{i=1}^{B}$ be the set of all paired fact queries $f^{(i)}$ and question queries $q^{(i)}$ in the current batch. For each $(f^{(i)}, q^{(i)}) \in \mathcal{B}$, the loss terms are:

$$\ell^{(i)}_{f \to q} = \log \frac{\exp(f^{(i)} \cdot q^{(i)}/\tau)}{\sum_{k=1}^{B} \exp(f^{(i)} \cdot q^{(k)}/\tau)} \qquad \ell^{(i)}_{q \to f} = \log \frac{\exp(q^{(i)} \cdot f^{(i)}/\tau)}{\sum_{k=1}^{B} \exp(q^{(i)} \cdot f^{(k)}/\tau)}, \tag{1}$$

where τ is a temperature hyperparameter. We average the two directions:

$$\mathcal{L}_{\text{CL}} = -\frac{1}{2B} \sum_{i=1}^{B} \left(\ell^{(i)}_{f \to q} + \ell^{(i)}_{q \to f}\right). \tag{2}$$

We optimize the joint loss $\mathcal{L} = \mathcal{L}_{\text{NTP}} + \lambda \mathcal{L}_{\text{CL}}$, with λ a hyperparameter set to 0.25 by default.

Building the Index We build a dense retrieval index from our pre-training data. The index pairs vector keys with fact spans. We run the pre-trained language model over the pre-training corpus, extract the L2-normalized hidden state $\phi(\cdot; \theta)$ at every <FACT> position, and store it in a dense vector index as a key whose value is the verbatim in-text span at that position. Indexing a new corpus therefore requires only annotating its fact spans (Section 3.3) and a forward pass of Co-LMLM.

[1] We L2-normalize the feature representations as they are used as retrieval queries (Chen et al., 2020).

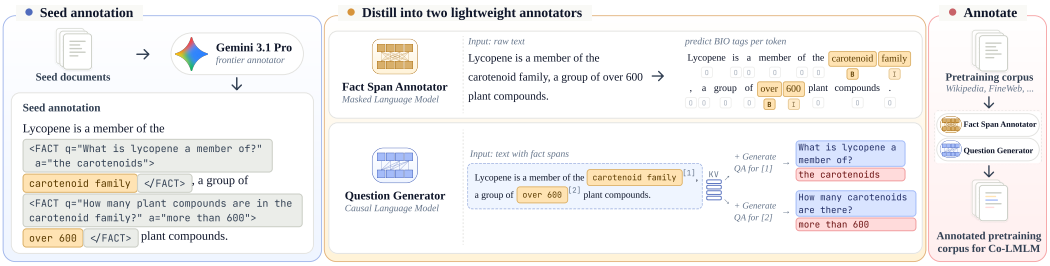


Figure 3: **Annotation pipeline.** *Left:* A frontier model annotates a small seed set of documents in the `<FACT q=... a=...>span</FACT>` format. *Middle:* We use the seed set to train two lightweight annotators: a fact span annotator that BIO-tags fact spans in raw text, and a question generator that produces a question-answer pair for each span; the document prefix is KV-cached and reused across all spans. *Right:* We run the two annotator over the full pre-training corpus, converting raw documents into the annotated pre-training corpus for Co-LMLM.

3.3 Training Data Pre-processing

Training requires documents annotated with fact spans and paired questions (Section 3.2). We design an automated, scalable process to annotate raw pre-training data. The pre-processing pipeline has three stages: annotate a seed set by prompting a state-of-the-art frontier LLM; fine-tune lightweight annotators on the seed data; and apply them to the full pre-training corpus. Critically, we fine-tune smaller models because of the cost of annotating at scale with a large frontier model.

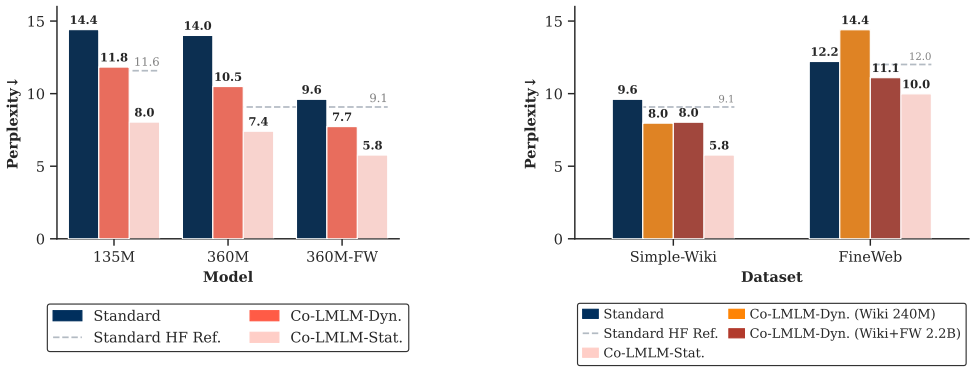
Annotation Specification We annotate each training document with a set of *fact spans*. A fact span is a substring of a pre-training document x that encodes world knowledge we want our model to externalize — to retrieve rather than memorize. Each fact span is associated with two additional pieces of information: a context-dependent question that captures the retrieval intent at this position without itself carrying any factual content, and a snippet-form answer that answers the question in a form that need not match the surface form of the span. The question is used as the question Q during training. An annotated fact takes the form `<FACT q={question} a={answer}>{span}</FACT>`.

The annotation reflects three guidelines: (a) spans are the narrowest self-contained substrings that fully answer the question; (b) questions are written as standalone search queries that name their subject explicitly, and they avoid mentioning information that appears later in the document so that no future fact leaks through the question; and (c) answers vary the surface form of the span, so the model is exposed at training time to retrieved values that differ lexically from the surrounding context. We provide more details about the prompt used for the seed annotation in Appendix D.

Annotation Pipeline Figure 3 visualizes the annotation pipeline. We first use Gemini-3.1-Pro to annotate a small seed set of documents using the `q="..."/a="..."` format defined above. The seed set is then used to distill two efficient annotators for the entire pre-training corpus. The *fact span annotator* is a masked language model that detects fact spans on the original text via BIO tagging,[2] built on a ModernBERT encoder (Warner et al., 2025). The *question generator* is a decoder-only generator that, conditioned on the document with span markers inserted, emits a question-answer pair for each tagged span; the document context is encoded once and reused across all facts. We assess the quality of the annotation pipeline in Appendix B.1.

Why Two Annotators? A single end-to-end generator that produces the full annotated text in one shot is the obvious alternative. We chose the split design for three reasons. First, for efficiency reasons: the question generator only emits questions and answers, not the document text, which lets us prefill the document context once and reuse it across all of its facts. Second, predicting spans as offsets into the original text makes the annotations *faithful by construction*: the in-text span is always a verbatim substring of the source document, while a generative annotator can drift away from the source (i.e., hallucinate), especially on long documents. Finally, indexing new data is highly efficient as it only involves two forward passes (one for the span annotator and one for Co-LMLM).

[2]BIO stands for begin-inside-outside (Ramshaw & Marcus, 1995).



(a) Controlled Wikipedia perplexities

(b) Data scaling-up perplexities

Figure 4: **Perplexity evaluation.** (a) Simple-Wiki test perplexity comparison between STANDARD and CO-LMLM across two model sizes (135M and 360M) and two training corpora (Wikipedia only vs. Wikipedia and FINEWEB-EDU). (b) Perplexities of STANDARD-360M-FW and CO-LMLM-360M-FW on Simple-Wiki and FineWeb test splits under different KB index scales — 240M items from Wikipedia vs. 2.2B from Wikipedia and FINEWEB-EDU. We show two perplexity variants for CO-LMLM (dynamic and static). The dashed gray line reports the perplexity of the same-size HF/SMOLLM2 model for context. The HF models are trained on much more data. The left figure uses PQ240 quantization, while the right uses PQ96 for both indices.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We train CO-LMLM using the SMOLLM2-135M and SMOLLM2-360M architectures (Allal et al., 2025), with a standard language model trained at each size as the STANDARD baseline. To facilitate a fair and controlled comparison with REL-LMLM, we train two REL-LMLM models based on the same architectures and use a high-quality Wikipedia corpus (∼3B tokens)[3], following Zhao et al. (2026). We train all Wikipedia models for 100K steps with a context length of 4,096 tokens. We show the generalization of our process to general web text by training CO-LMLM and a corresponding STANDARD baseline on 90B tokens sampled from FINEWEB-EDU (Penedo et al., 2024). We experiment with two KBs. We create a KB from Wikipedia only for a direct comparison with the REL-LMLM KB that is constructed from this same corpus. It also guarantees that the KB does not cover information unavailable to the baselines. This the default KB, unless another is specified. The CO-LMLM Wikipedia KB includes ∼240M items, compared to REL-LMLM's ∼145M, demonstrating the less restricted, and more aggressive externalization of CO-LMLM. In our scaling-up experiments, we construct the KB from our FINEWEB-EDU corpus as well (∼2.2B items). Appendix A provides additional implementation details.

4.2 LANGUAGE MODELING PERPLEXITY

Evaluation Setup We evaluate using perplexity on 1,000 randomly sampled Simple English Wikipedia (Simple-Wiki) articles.[4] These articles are not included in our pre-training corpus, but their underlying facts are likely covered by our index in different surface forms, because the information in Simple-Wiki is largely a subset of Wikipedia. This setup avoids coverage issues and helps distinguish retrieval or annotation errors in CO-LMLM from imperfect parametric memorization in STANDARD. The key comparison is between the models we train, controlling for the training data. We also include off-the-shelf models, with the caveat of a potential Wikipedia domain shift.

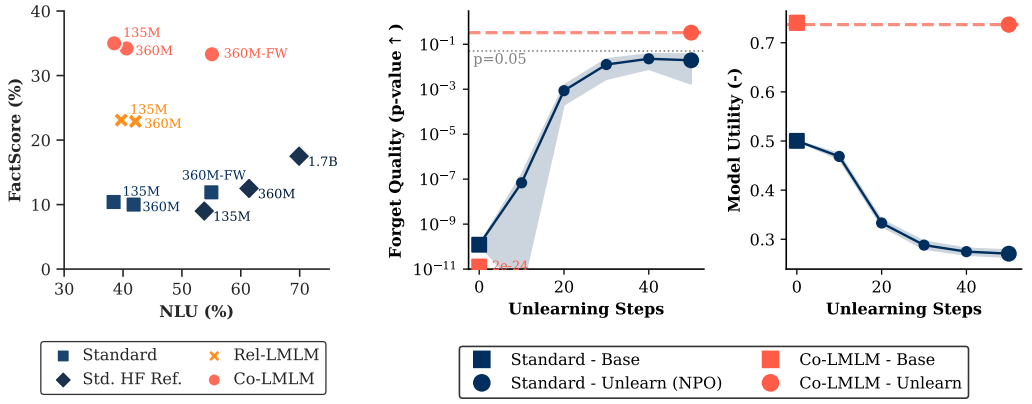
We report two perplexity variants, following Zhao et al. (2026). *Static (Oracle)* assumes perfect lookup behavior by supplying the correct retrieved spans as context, giving an optimistic estimate of language modeling quality. *Dynamic* performs actual lookups during decoding, so retrieval

[3] https://huggingface.co/datasets/allenai/dolmino-mix-1124

[4] https://simple.wikipedia.org/

Table 1: **Factual precision evaluation.** Green subscripts show the absolute gain over the respective STANDARD baseline. We include * models for context, although they are trained on much larger corpora (2T/4T/11T tokens for 135M/360M/1.7B models). Our rebuilt REL-LMLM baseline follows the original setup with engineering improvements for a stronger, fairer comparison (Appendix A.5).

Model	Short-form QA			Knowledge Completion	Long-form Generation
	TriviaQA ↑	PopQA ↑	SimpleQA ↑	T-REx EM ↑	FactScore ↑
HF/SMOLLM2-135M*	14.7	18.9	1.3	36.1	9.0
STANDARD-135M	6.4	15.2	0.1	38.2	10.4
REL-LMLM-135M	13.4	27.2	6.2	42.9	23.1
CO-LMLM-135M	28.4$_{+22.0}$	47.3$_{+32.1}$	18.2$_{+18.1}$	40.5$_{+2.3}$	35.0$_{+24.6}$
HF/SMOLLM2-360M*	25.4	22.7	1.4	45.8	12.5
STANDARD-360M	7.5	16.2	0.3	46.3	10.0
STANDARD-360M-FW	18.6	18.9	1.3	50.5	11.9
REL-LMLM-360M	15.6	26.6	5.6	47.4	22.9
CO-LMLM-360M	31.4$_{+23.9}$	46.5$_{+30.3}$	21.2$_{+20.9}$	47.4$_{+1.1}$	34.2$_{+24.2}$
CO-LMLM-360M-FW	36.9$_{+18.3}$	50.6$_{+31.7}$	21.7$_{+20.4}$	54.5$_{+4.0}$	33.3$_{+21.4}$
HF/SMOLLM2-1.7B*	45.1	24.0	2.1	57.2	17.5



(a) Factuality vs. NLU

(b) Unlearning on TOFU

Figure 5: **Factuality, NLU, and unlearning evaluation.** (a) FactScore and NLU tradeoff for various models — CO-LMLM substantially improves factual precision while preserving NLU performance comparable to STANDARD models. (b) Machine unlearning on TOFU — CO-LMLM forgets through direct KB operations, without additional training or utility degradation.

performance is reflected in the score. While dynamic follows the standard perplexity calculation, it introduces some complexities in our case. We discuss this in depth in Appendix A.6 and provide a worst-case analysis. Importantly, this worst-case analysis does not change our conclusions.

Results CO-LMLM consistently achieves lower perplexity than STANDARD across model sizes and perplexity variants (Figure 4). Figure 4a shows perplexities for Simple-Wiki, where we are fairly certain to get full coverage of factual data. Under the more realistic *Dynamic* setting, CO-LMLM reduces perplexity from 14.4 to 11.8 for 135M and from 14.0 to 10.5 for 360M. These gains suggest that our annotated pretraining data and training objective successfully improve language modeling over factual text. The FineWeb (FW) results indicate the benefit of data scale, charting initial scaling trends. Off-the-shelf models (HF/SMOLLM2-135M and HF/SMOLLM2-360M, dashed line) provide context: the relative gap CO-LMLM-360M-FW shows is particularly promising given the orders of magnitude more training data of the HF models, although a likely domain confounder qualifies this result.

Figure 4b provides our main perplexity result, scaling up the index to include the entire FW and Wikipedia corpora used for training our models. This reflects a realistic KB construction, both in document diversity and size. This results in a much larger index of 2.2B entries, compared to the 240M entries in the Wiki-only index. We report results on both the Simple-Wiki dataset and

an 895-document held-out test set from the FW dataset. We compute these results with higher quantization for an additional 2.5× memory reduction per vector, so all indices fit in memory.

We do not expect benefits from scaling up the index to include FW data for Simple-Wiki, because the corresponding knowledge is expected to be contained in the English Wikipedia dataset. Indeed, under the same quantization level, we only see a minor increase in perplexity even though the index is one order of magnitude larger, which speaks to the robustness of our learned retriever. For the FW test set, on the other hand, we observe a substantial improvement over the Wiki-only index, producing a significantly lower perplexity than that of the HF/SMOLLM2-360M model, which is pre-trained on those documents and overall much more data (4T tokens). The poor performance of the Wiki-only index on the FW test set demonstrates Wikipedia's limited coverage of knowledge and the need to scale up the factual KB. This result demonstrates effective scaling beyond Wikipedia data.

4.3 FACTUALITY AND GENERAL BENCHMARK PERFORMANCE

Evaluation Setup We evaluate factuality across five complementary settings that probe different forms of factual knowledge in pre-trained models. TriviaQA (Joshi et al., 2017) tests broad open-domain QA, while PopQA (Mallen et al., 2023) focuses on long-tail entities. SimpleQA Verified (hereafter SimpleQA; Wei et al., 2024; Haas et al., 2026) targets short-form, fact-seeking questions that are deliberately difficult and have a single, indisputable answer. T-REx (Elsahar et al., 2018) evaluates controlled factual completion, where the model completes a factual statement with the correct object entity. FactScore (Min et al., 2023) evaluates long-form factual precision in biography generation by decomposing generations into atomic facts and verifying them against external evidence. We also evaluate CO-LMLM on standard NLU benchmarks to verify that factual offloading does not compromise language understanding.

Results Table 1 shows that CO-LMLM consistently improves over the corresponding STANDARD baselines, with especially large gains on short-form QA and FactScore, exceeding 18 points in each case. As with perplexity, we include off-the-shelf models for contextualization, with similar caveats. CO-LMLM also outperforms REL-LMLM, demonstrating that continuous queries are more robust than textual entity-relation queries. We attribute these gains to two factors: free-form factual spans provide richer external knowledge than relational triplets, and hidden-state queries can use the full generation context rather than a short and restricted decoded query. Training on FineWeb-Edu further improves short-form QA and knowledge completion, showing that our design remains effective beyond Wikipedia-style pre-training data. CO-LMLM-360M-FW achieves 21.7 on SimpleQA, in line with gpt-4o-mini and above Claude Sonnet 4.5 and Grok 2 according to the public leaderboard.[5]

CO-LMLM achieves a better factuality–NLU trade-off than standard models, with much higher factuality and similar NLU capabilities (Figure 5a). Appendix B.3 details NLU results across tasks. We also compare CO-LMLM to controlled RAG baselines in Appendix B.4 (Table 14), finding that RAG and CO-LMLM are complementary and that CO-LMLM+RAG is a viable path forward.

4.4 UNLEARNING

Externalizing knowledge simplifies unlearning: instead of retraining the model, we just remove the relevant entries from memory. We evaluate this property with TOFU (Maini et al., 2024), following the same setup as REL-LMLM (Appendix A.7). TOFU tests whether a model can forget a designated *forget set* while preserving general utility. Its main metric, *forget quality*, tests whether the unlearned model is statistically indistinguishable from a retain-only model on the forgotten examples. Figure 5b shows how simple memory operations achieve effective forgetting (p-value > 0.05) while preserving model utility. In contrast, training-based unlearning methods such as NPO (Zhang et al., 2024b) update model parameters and sacrifice utility due to entangled parametric knowledge. CO-LMLM retains the controllability benefit of REL-LMLM while being more expressive and efficient.

4.5 KNOWLEDGE MEMORIZATION

To measure how much factual performance depends on access to the learned external memory, we evaluate a no-retrieval variant of our model. In this setting, we disable the model's ability to trigger

[5] https://www.kaggle.com/benchmarks/deepmind/simpleqa-verified

Table 2: **No KB retrieval ablation.** Disabling retrieval substantially reduces performance.

Model	TriviaQA	PopQA	SimpleQA	T-REx	FactScore
STANDARD-135M	6.4	15.2	0.1	38.2	10.4
CO-LMLM-135M	28.4	47.3	18.2	40.5	35.0
w/o KB	4.7$_{-23.7}$	15.4$_{-31.9}$	1.0$_{-17.2}$	21.7$_{-18.8}$	18.0$_{-17.0}$
STANDARD-360M	7.5	16.2	0.3	46.3	10.0
CO-LMLM-360M	31.4	46.5	21.2	47.4	34.2
w/o KB	6.0$_{-25.4}$	13.5$_{-33.0}$	0.7$_{-20.5}$	27.4$_{-20.0}$	18.7$_{-15.5}$
STANDARD-360M-FW	18.6	18.9	1.3	50.5	11.9
CO-LMLM-360M-FW	36.9	50.6	21.7	54.5	33.3
w/o KB	9.2$_{-27.7}$	17.3$_{-33.3}$	0.5$_{-21.2}$	33.3$_{-21.2}$	16.2$_{-17.1}$

KB retrieval during generation. Concretely, for continuous-query models, we apply a logit bias of -100 to all factual special tokens that initiate or delimit retrieval spans. This prevents the model from emitting the markup required to invoke the retrieval loop, forcing it to generate plain text from its parameters alone. This ablation isolates the contribution of external memory access from the model's parametric knowledge. Table 2 shows that factual precision drops substantially. This shows that CO-LMLM genuinely shifts factual knowledge from model weights to the external memory.

4.6 ADDITIONAL EXPERIMENTS TO ANALYZE THE QUERY MECHANISM

We isolate the impact of using continuous queries by comparing CO-LMLM to LMLM-ASKER, a variant that generates free-form natural-language questions as queries. Table 16 in the appendix shows that CO-LMLM outperforms LMLM-ASKER, demonstrating the advantage of continuous queries. Text queries also add generation cost: LMLM-ASKER and REL-LMLM decode a question or a relational query, respectively, before every lookup, while CO-LMLM reads the query from its hidden state, saving a cost linear in the length of the question (Appendix B.6). We observe that deciding *when* to retrieve limits CO-LMLM performance: forcing a lookup before answering, while keeping the model's own query, improves scores by up to 9.3 points (Appendix B.7). An error analysis of CO-LMLM-360M-FW (Appendix B.8) finds that most of its PopQA errors stem from incorrect retrievals, fewer from missing ones, and only rarely from misusing a retrieved fact.

5 DISCUSSION

We present CO-LMLM, a scalable method to train LLMs for verifiable and controllable use of knowledge. This includes a large-scale annotation pipeline that can be applied to hundreds of billions of tokens at relatively modest costs. Critically, the CO-LMLM training process, beyond the pre-processing step, departs only modestly from the conventional LLM training process that has been shown to scale to many trillions of both tokens and model parameters. Our method and results open an avenue to train LLMs with factuality, transparency, and controllability properties that are sorely lacking in current LLMs, which has significantly hampered their trustworthiness and utility. CO-LMLM provides a flexible free-form span substrate for studying how externalized knowledge scales across broader knowledge domains, memory sizes, and retrieval mechanisms. This establishes a foundation for future work to study *knowledge scaling laws*.

Our current study has several limitations that outline important directions for future work. Although we experiment along model and data scaling axes to outline scaling trends, our experiments remain modest in model and data scale. We focus on probing the behavior of CO-LMLM in controlled pre-training settings, and leave larger-scale pre-training and downstream adaptation to future work. Our factual evaluations mainly target Wikipedia-style knowledge, a key factor in creating a clean experimental setup. Although, as shown, our pipeline can naturally extend to broader corpora such as FINEWEB-EDU, there remains work to be done to systematically evaluate factuality across more diverse knowledge domains. Constructing a retrieval index at pre-training scale introduces a one-time computational cost. However, an important problem to address in the context of post-training and continual learning is how to adapt such an index once a model is fine-tuned.

Studying these and many other important problems will be enabled by the scalable design of our methods and by our large-scale release of annotations of Wikipedia and 90B FineWeb tokens.

REPRODUCIBILITY STATEMENT

We have taken a number of steps to make the results in this paper reproducible. The CO-LMLM architecture, training objective, and training-data pre-processing pipeline are specified in Section 3, and the full experimental protocol — corpora, model configurations, baselines, and evaluation benchmarks — in Section 4. Appendix A gives the complete pre-training and annotator hyperparameters (Appendix A.1, Appendix A.2), the compute used (Appendix A.3), the retrieval-index configurations (Appendix A.4), our re-implementation of the REL-LMLM baseline (Appendix A.5), the per-benchmark evaluation setup (Appendix A.6), and the unlearning (Appendix A.7) and ablation (Appendix **??**) settings. The prompt used to produce the seed annotations with a frontier model is reproduced verbatim in Appendix D, so the annotation pipeline can be rebuilt end-to-end. All datasets, base models, and evaluation suites we use are publicly available and cited at their point of use, and our code, annotated corpora, and trained models will be released publicly upon publication.

AI USE STATEMENT

We used a frontier language model (GEMINI-3.1-PRO) to generate the seed annotations from which our lightweight annotators are distilled, and the distilled annotators are then used to construct our annotated pre-training corpora. This is described in Section 3.3, and the exact prompt is reproduced verbatim in Appendix D. We also used Claude Code with Claude Opus 5 to label annotation errors in our annotation quality assessment, and manually examined several documents to verify its labels (Appendix B.1).

We used AI coding assistants (specifically, Claude Code) to help write and refactor parts of our research code, and we used an iterative semi-automated process with an AI coding assistant to refine the seed annotation prompt, as noted in Appendix D. We also used AI writing assistants to improve the readability of the manuscript, for paraphrasing and for proofreading.

We have reviewed all AI-assisted work. AI-generated code was read and tested by the authors. All text, claims, figures, and references in the manuscript were checked by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. jina-embeddings-v5-text: Task-targeted embedding distillation, 2026. URL https://arxiv.org/abs/2602.15547.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.09764.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1911.11641.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, Han Zhang, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models, 2026. URL https://arxiv.org/abs/2601.07372.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Annual Meeting of the Association for Computational Linguistics*, 2022. URL https://aclanthology.org/2022.acl-long.581/.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *IEEE Transactions on Big Data*, 2025. URL https://arxiv.org/abs/2401.08281.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. Understanding emergent abilities of language models from the loss perspective. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=35DAviqMFo.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-REx: A large scale alignment of natural language with knowledge base triples. In *International Conference on Language Resources and Evaluation*, 2018. URL https://aclanthology.org/L18-1544/.

Gaurav R. Ghosal, Pratyush Maini, and Aditi Raghunathan. Memorization sinks: Isolating memorization during LLM training. In *International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/ghosal25a.html.

Gaurav R. Ghosal, Pratyush Maini, and Aditi Raghunathan. Natively unlearnable large language models, 2026. URL https://arxiv.org/abs/2606.13873.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

Lukas Haas, Gal Yona, Giovanni D'Antonio, Sasha Goldshtein, and Dipanjan Das. Simpleqa verified: A reliable factuality benchmark to measure parametric knowledge, 2026. URL https://arxiv.org/abs/2509.07968.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts LLM performance, 2025. URL https://research.trychroma.com/context-rot.

Lei Huang, Xiaocheng Feng, Weitao Ma, Liang Zhao, Yuchun Fan, Weihong Zhong, Dongliang Xu, Qing Yang, Hongtao Liu, and Bing Qin. Advancing large language model attribution through self-improving. In *Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://aclanthology.org/2024.emnlp-main.223/.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2021. URL https://aclanthology.org/2021.eacl-main.74/.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 2023. URL https://arxiv.org/abs/2208.03299.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2017. URL https://aclanthology.org/P17-1147/.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, 2023. URL https://proceedings.mlr.press/v202/kandpal23a.html.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Large memory layers with product keys. In *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/9d8df73a3cfbf3c5b47bc9b50f214aff-Abstract.html.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=B41hNBoWLo.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://aclanthology.org/2023.acl-long.546/.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.741/.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. MemLLM: Finetuning LLMs to use an explicit read-write memory, 2024. URL https://arxiv.org/abs/2404.11672.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. Generative representational instruction tuning. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2402.09906.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2024. URL https://arxiv.org/abs/2501.00656.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Guangyue Peng, Tao Ge, Si-Qing Chen, Furu Wei, and Houfeng Wang. Semiparametric language models are scalable continual learners, 2023. URL https://arxiv.org/abs/2303.01421.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://aclanthology.org/D19-1250/.

Hadi Pouransari, David Grangier, C Thomas, Michael Kirchhof, and Oncel Tuzel. Pretraining with hierarchical memories: separating long-tail and common knowledge, 2026. URL https://arxiv.org/abs/2510.02375.

Qwen et al. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Lance Ramshaw and Mitchell Marcus. Text chunking using transformation-based learning. In *Workshop on Very Large Corpora*, 1995. URL https://aclanthology.org/W95-0107/.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Conference on Empirical Methods in Natural Language Processing*, 2020. URL https://aclanthology.org/2020.emnlp-main.437/.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://aclanthology.org/D19-1454/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.702/.

Alicia Yi Sun, Karthik Padthe, Akari Asai, and Wen-tau Yih. Semi-parametric language model with selective memory, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019. URL https://aclanthology.org/N19-1421/.

Albert Tseng and Christopher De Sa. L^3: Large lookup layers, 2026. URL https://arxiv.org/abs/2601.21461.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018. URL https://arxiv.org/abs/1807.03748.

Xi Wang, Taketomo Isazawa, Liana Mikaelyan, and James Hensman. KBLaM: Knowledge base augmented language model. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=aLsMzkTej9.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://aclanthology.org/2025.acl-long.127/.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models, 2024. URL https://arxiv.org/abs/2411.04368.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

Hongkang Yang, Zehao Lin, Wenjin Wang, Hao Wu, Zhiyu Li, Bo Tang, Wenqiang Wei, Jinbo Wang, Zeyun Tang, Shichao Song, et al. Memory^3: Language modeling with explicit memory, 2024. URL https://arxiv.org/abs/2407.01178.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://aclanthology.org/P19-1472/.

Jintian Zhang, Cheng Peng, Mengshu Sun, Xiang Chen, Lei Liang, Zhiqiang Zhang, Jun Zhou, Huajun Chen, and Ningyu Zhang. OneGen: Efficient one-pass unified generation and retrieval for LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024a. URL https://aclanthology.org/2024.findings-emnlp.797/.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=MXLBXjQkmb.

Wenzheng Zhang, Xi Victoria Lin, Karl Stratos, Wen-tau Yih, and Mingda Chen. ImpRAG: Retrieval-augmented generation with implicit queries. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. URL https://arxiv.org/abs/2506.02279.

Linxi Zhao, Sofian Zalouk, Christian K Belardi, Justin Lovelace, Jin Peng Zhou, Ryan Thomas Noonan, Dongyoung Go, Kilian Q Weinberger, Yoav Artzi, and Jennifer J Sun. Pre-training limited memory language models with internal and external knowledge, 2026. URL https://arxiv.org/abs/2505.15962.

A Implementation Details and Experimental Setup

A.1 Pre-training Hyperparameters

We pre-train Co-LMLM from random weights following a Warmup-Stable-Decay (WSD) learning-rate schedule. Table 3 lists the hyperparameters used to train Co-LMLM. The Standard baseline language model and LMLM-Asker are trained with the same hyperparameters, except for those that apply only to the contrastive loss (contrastive loss weight, contrastive temperature, and max question length). For the FineWeb-Edu run, the cooldown stage mixes the annotated Wikipedia and FineWeb-Edu datasets in a 1:1 ratio.

Table 3: Pre-training hyperparameters for Co-LMLM.

Hyperparameter	**Value**
architecture	SmolLM2 (random init)
tokenizer	SmolLM2
optimizer	AdamW
β_1	0.9
β_2	0.95
ϵ	1e-8
clip norm	1
peak learning rate	5e-4
final learning rate	0
lr scheduler type	WSD
warmup steps	2,000
stable steps	88,000 (148,000 for FineWeb-Edu)
cooldown steps	10,000 (20,000 for FineWeb-Edu)
weight decay	0.01
batch size	128
max context length	4,096
max question length	128
contrastive loss weight (λ)	0.25
contrastive temperature	0.07
precision	bf16 (fp32 master weights)

A.2 Annotator Training

We use Gemini 3.1 Pro to annotate a seed set of 60,000 documents (15,000 Wikipedia articles and 45,000 FineWeb-Edu documents), and train the fact span annotator and the question generator on this seed set. The fact span annotator is a ModernBERT-large (Warner et al., 2025) encoder fine-tuned for BIO token classification (Table 4). The question generator is a Qwen2.5-1.5B-Instruct (Qwen et al., 2024) decoder fine-tuned with LoRA (Table 5). We assess the quality of the resulting annotations in Appendix B.1.

A.3 Compute

For all experiments involving model training except TOFU, between 1 and 8 NVIDIA B200 GPUs were used, depending on availability. The number of gradient accumulation steps was set dynamically to maintain a consistent batch size throughout training. For the TOFU unlearning experiments, a single NVIDIA RTX A6000 GPU was used.

A.4 Retrieval Indexes

We use the Faiss library (Douze et al., 2025) for all our experiments involving building dense search indexes. To limit RAM usage, we use the following factory string for constructing our indices: `OPQ`<$d/4$>`,IVF65536,PQ`<$d/4$>, where d is the dimension of the embedder being used. For the data scaling-up experiments (Figure 4b), we instead use `PQ96` for both indexes

Table 4: Hyperparameters for training the fact span annotator.

Hyperparameter	Value
base model	ModernBERT-large
optimizer	AdamW
β_1	0.9
β_2	0.95
ϵ	1e-8
clip norm	1
peak learning rate	2e-4
final learning rate	2e-5
lr scheduler type	cosine
warmup ratio	0.05
weight decay	1e-3
epochs	4
batch size	32
max sequence length	4,096
precision	bf16

Table 5: Hyperparameters for training the question generator.

Hyperparameter	Value
base model	Qwen2.5-1.5B-Instruct
LoRA r	64
LoRA α	64
LoRA dropout	0.05
LoRA target modules	all-linear
optimizer	AdamW
β_1	0.9
β_2	0.95
ϵ	1e-8
clip norm	1
peak learning rate	2e-4
final learning rate	2e-5
lr scheduler type	cosine
warmup ratio	0.05
weight decay	1e-3
epochs	4
batch size	4
max sequence length	8,192
precision	bf16

so that they fit in memory. For our REL-LMLM and LMLM-ASKER experiments, we use the `jina-embeddings-v5-text-nano` embedder (Akram et al., 2026), truncated to 512 dimensions. We did not extensively explore the retrieval errors that can be attributed to the lossy compression applied by the search indexes for any of the models presented in this paper.

A.5 REL-LMLM RE-IMPLEMENTATION

For a controlled comparison against CO-LMLM, we re-implement the REL-LMLM pre-training pipeline with several engineering improvements over the original recipe. These changes do not affect the conclusions of the original REL-LMLM work. They improve data efficiency and align the training setup with CO-LMLM, yielding a stronger REL-LMLM baseline and a more rigorous apples-to-apples comparison. The changes fall into two groups.

Table 6: Annotation configuration updates from the original REL-LMLM release to the rebuilt, stronger baseline used in this paper. Token counts are for the Wikipedia subset of `dolmino-mix-1124`.

Version	Seed annotator	Production annotator	Input trunc.	DB format	Est. tokens
Original	GPT-4o	Llama3-8B-Instruct (LoRA)	1024 tok/doc	`[dblookup('')->]`	2.8B
Rebuilt	GPT-4o	Qwen3-4B (full FT)	none	special tokens	6.1B

Annotation Quality and Quantity The largest gain is in the quantity of usable pre-training data (summarized in Table 6). The original pipeline truncates each document to 1024 tokens before annotation. Annotating full documents instead recovers a substantial fraction of the Wikipedia subset of the OLMo2 `dolmino-mix-1124` corpus (OLMo et al., 2024)[6] (roughly 30% of documents and about half of the available pre-training tokens were previously discarded), raising the usable set from 2.8B to 6.1B tokens (including DB lookup tokens).

On the quality side, we emit the special-token database-call format directly during annotation rather than converting from the textual `[dblookup('')->]` markup before pre-training, removing a conversion step and keeping annotations consistent end-to-end. We also improve prompting, enlarge the seed annotation set, avoid overfitting (12,750 examples for 3 epochs versus the original 2k examples for 10 epochs), and replace the LoRA-tuned Llama3-8B-Instruct production annotator with a fully fine-tuned Qwen3-4B for better quality and efficiency.

Identical Backbone and Training Configuration Throughout this paper, we report results for the rebuilt REL-LMLM. To isolate the effect of the method, we match the model backbone and all training hyperparameters (batch size, number of steps, learning-rate schedule, etc.) to CO-LMLM. With the data pipeline improved and the training setup held fixed, remaining differences between the two methods reflect the method itself rather than the experimental setup. This yields a substantially stronger REL-LMLM baseline than the original release. CO-LMLM's improvements over this matched, well-tuned baseline therefore reflect the method itself, not an advantage in experimental setup.

A.6 EVALUATION

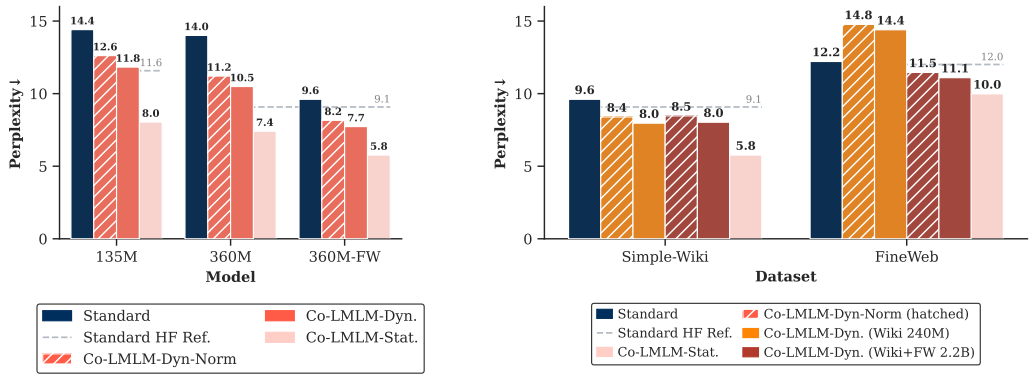
Perplexity Zhao et al. (2026) introduced three perplexity variations for LMLM models: *static (Oracle)*, *normalized*, and *dynamic*. The static and dynamic variants are described in Section 4.2, but are recited here for completeness. Static perplexity assumes perfect lookup behavior by supplying the correct retrieved spans as context, giving an optimistic estimate of language modeling quality. Dynamic perplexity performs actual lookups during decoding, so retrieval performance is reflected in the score. Normalized perplexity additionally scores the lookup-trigger tokens the model emits, and divides by the length of the original unannotated text. Normalized perplexity is informative for REL-LMLM, where the lookup tokens include a decoded query whose quality is being measured. For CO-LMLM, the only emitted lookup token is a single `<FACT>` delimiter, since the query is read from the hidden state rather than decoded as text, so normalized perplexity for CO-LMLM additionally measures only whether retrieval is triggered at the right position. We therefore treat dynamic and static perplexity as the primary metrics.

Formally, let x be an input document with factual spans bracketed by `<FACT>` and `</FACT>`, and T the set of token positions in x. Let $M \subseteq T$ be the set of positions in x that are strictly between `<FACT>` and `</FACT>`, inclusive of both the opening `<FACT>` and closing `</FACT>`. Finally, let $\overline{M} = T \setminus M$.

We define the static and dynamic perplexity as follows:

$$\text{PPL}_{\text{static/dynamic}} = \exp\left(-\frac{1}{|\overline{M}|}\sum_{t\in\overline{M}} \log p_\theta(x_t \mid x_{<t})\right),$$

[6] https://huggingface.co/datasets/allenai/dolmino-mix-1124



(a) Controlled Wikipedia perplexities

(b) Data scaling-up perplexities

Figure 6: **Perplexity evaluation with the dynamic-normalized PPL variant.** (a) Simple-Wiki test perplexity comparison between STANDARD and CO-LMLM across three perplexity variants. (b) Perplexities of STANDARD-360M-FW and CO-LMLM-360M-FW on Simple-Wiki and FineWeb test splits under different KB index scales, as in Figure 4b. The dashed gray line reports the perplexity of the same-size HF/SMOLLM2 model for context. The left figure uses PQ240 quantization, while the right uses PQ96 for both indices. Even under the pessimistic dynamic-normalized perplexity measure, CO-LMLM-360M-FW achieves significantly lower perplexity than HF/SMOLLM2-360M, which is pre-trained on an order of magnitude more tokens.

where the only difference between the two is the source of the factual spans: static uses the oracle spans paraphrased by the annotator, and dynamic issues actual retrieval and inserts the retrieved fact into the span.

The dynamic perplexity metric is the more realistic of the set, and follows conventional perplexity computation. However, the nuances of CO-LMLM introduce some complexities in its measurement, and make it somewhat optimistic. This is because it does not take into account the probability of the model correctly generating the <FACT> token at that precise position. An accurate and comparable perplexity measure is not trivial to compute since the fact-annotated text only defines one possible way to generate the original context tokens, and accounting for all possible retrieval paths is intractable. Furthermore, directly accounting for the <FACT> token's loss will not be accurate either, since it will introduce additional context tokens, presenting a mismatch between LMLM and standard LMs in the denominator $|\overline{M}|$ used in the perplexity calculation. It also changes the vocabulary by introducing a new, fairly common token. These factors may lead to an overly optimistic measure as well. We address this issue with an additional *dynamic-normalized* perplexity measure, which includes the <FACT> token's loss in the numerator but keeps it out of the denominator. This normalized measure is a worst-case scenario, in practice allowing us to compute a range between itself and the optimistic dynamic measure.

Formally, let M_F be the set of positions in x that contain the <FACT> token. We define the dynamic-normalized perplexity as follows:

$$\text{PPL}_{\text{dynamic-norm}} = \exp\left(-\frac{1}{|\overline{M}|}\sum_{t\in\overline{M}\cup M_F}\log p_\theta(x_t \mid x_{<t})\right).$$

This provides us with a pessimistic estimation of the perplexity, ensuring that the actual, realistic perplexity lies between $\text{PPL}_{\text{dynamic}}$ and $\text{PPL}_{\text{dynamic-norm}}$. Figure 6 provides results for all perplexity variants. Critically, this worst-case scenario does not change the ordering of models, or our conclusions.

Long-Form Generation: FactScore We evaluate factual precision using FACTSCORE (Min et al., 2023), a benchmark for open-ended biography generation. Given a generated biography, FactScore decomposes it into atomic facts and measures the proportion that can be verified against a trusted

knowledge source via retrieval-augmented LLM. We use the labeled entities set from the benchmark, which includes 183 entities.

All models use greedy decoding with a maximum *content* budget of 256 tokens and a repetition penalty of 1.2. For models with external lookup (e.g., CO-LMLM and REL-LMLM), this budget applies only to the final textual output and *excludes* any inserted lookup spans (e.g., <FACT>...</FACT> or <|db_*|> blocks). To provide a fair probe for models pre-trained primarily on Wikipedia (and not instruction-tuned or exposed to more diverse corpora), we adopt a fixed Wikipedia-style prompt "<name>\n\n<name>" to elicit biography completions, applied uniformly across all samples. For evaluation, all lookup-related annotation tokens are stripped before scoring. We follow the official FactScore pipeline,[7] using retrieval-augmented prompting with GEMINI-2.5-FLASH.

We use greedy decoding for all factuality benchmarks, including FactScore, T-REx, PopQA, TriviaQA, and SimpleQA. The retrieval similarity threshold is set to 0.7; when no match is found, the model falls back to standard decoding, allowing it to answer facts absent from the KB. Lookup blocks, including the retrieved spans, are removed before scoring.

Short-Form Knowledge Completion: T-REx We adopt T-REx from LAMA (Petroni et al., 2019) as a knowledge completion benchmark for autoregressive models. Following Schick et al. (2023), we retain only examples compatible with left-to-right generation, resulting in 11,615 instances where the masked entity appears at the end of the sentence. Each input is a partially observed factual statement, and the model is required to complete it with the correct object entity.

We use greedy decoding with a maximum of 32 generated tokens. Performance is measured by *Exact Match*, which checks whether the reference answer appears within the first five generated content tokens after stripping annotation tokens.

Short-Form Probing: PopQA, TriviaQA and SimpleQA We further evaluate factual recall on question-answering benchmarks, focusing on the long-tail subset of PopQA (Asai et al., 2024), which contains 1,399 queries about rare entities (fewer than 100 monthly Wikipedia page views), the full TriviaQA evaluation set (17,944 examples), and the full SimpleQA-verified evaluation set (1,000 examples). These benchmarks probe knowledge recall under both sparse and broad coverage regimes.

All models use greedy decoding with a maximum of 32 tokens. For PopQA and TriviaQA, evaluation is based on *Exact Match*, defined as whether any alias in the set of gold answers appears (case-insensitive) within the first 100 characters of the model output. For SimpleQA, we follow the official grading prompt and use GPT-4.1-2025-04-14 as the grader model. We report the *overall correct* metric.

A.7 MACHINE UNLEARNING SETTING

TOFU Unlearning. The TOFU unlearning benchmark (Maini et al., 2024) evaluates selective unlearning in a controlled question-answering setting. It contains 200 synthetic author profiles, each associated with 20 question-answer pairs, resulting in 4,000 QA examples in total. The benchmark partitions this synthetic knowledge into a *Forget Set*, containing the examples to be removed, and a *Retain Set*, containing the remaining synthetic author knowledge that should be preserved. In addition to synthetic biographies, TOFU evaluates whether unlearning preserves broader model behavior using two auxiliary sets: the *Real Author Set*, which contains factual questions about real authors, and the *World Facts Set*, which contains general factual knowledge. The target behavior is that the resulting *Unlearned Model* should forget the Forget Set while remaining statistically close to a *Retain Model*, which is trained only on the Retain Set, and should maintain utility on the Retain Set, Real Author Set, and World Facts Set.

TOFU reports two main categories of metrics. *Forget Quality* measures whether the unlearned model behaves similarly to the retain model on the Forget Set, using a statistical test whose p-value indicates whether the two output distributions are distinguishable. A higher p-value indicates better forgetting, with values above 0.05 commonly interpreted as failing to reject the hypothesis that the unlearned model and retain model behave similarly. *Model Utility* measures whether useful behavior is preserved after unlearning. It aggregates three metrics: ROUGE, which measures the lexical

[7] https://github.com/shmsw25/FActScore

overlap between generated and reference answers; Answer Probability, which measures the likelihood assigned to the correct answer; and Truth Ratio, which compares the likelihood assigned to correct answers against paraphrased or perturbed alternatives. These utility metrics are evaluated across the Retain Set, Real Author Set, and World Facts Set to capture both retained synthetic knowledge and general factual capability.

We evaluate on the TOFU Forget 5% setting using the official TOFU evaluation pipeline from the OpenUnlearning repository.[8] For CO-LMLM, we augment the external Wikipedia knowledge base from pre-training with TOFU's synthetic author biographies and fine-tune CO-LMLM-360M-FW on the annotated TOFU training set using the same hyperparameters as the baseline models in Table 7. To perform unlearning, we delete the KB entries corresponding to the Forget Set, without any additional gradient updates. This directly tests whether externalizing factual knowledge enables training-free removal of targeted information.

NPO Unlearning Baseline We compare against HF/SMOLLM2-360M fine-tuned with Negative Preference Optimization (NPO) (Zhang et al., 2024b), a strong gradient-based unlearning baseline. We run NPO unlearning fine-tuning (Table 8) with five random seeds and report the mean and standard deviation in Figure 5b and Figure 7. For ROUGE evaluations, generated answers are post-processed to remove retrieved factual spans before computing scores. For likelihood-based metrics, including Answer Probability and Truth Ratio, we evaluate the model probabilities with answer masking.

Table 7: TOFU fine-tuning configuration.

Setting	Value
Base model	HF/SMOLLM2-360M / CO-LMLM-360M-FW
Training sets	TOFU `full` / `retain95`
Template	`Question:` / `Answer:`
Learning rate	3×10^{-5}
Epochs	10
Checkpoints	Full model and retain model

Table 8: NPO training configuration on TOFU.

Setting	Value
Initialization	Full TOFU fine-tuned checkpoint
Forget / retain split	`forget05` / `retain95`
Objective	NPO, $\beta = 0.1$
Learning rate	6×10^{-5}
Per-device batch size	32
Gradient accumulation	1
Epochs	8
Seeds	$\{0, 42, 69, 420, 4497\}$

B ADDITIONAL RESULTS

B.1 ANNOTATION QUALITY ASSESSMENT

We assess the quality of the annotations produced by our lightweight annotators (Section 3.3), which build the CO-LMLM pre-training data and KB.

Methodology We annotate the first $\sim$512 tokens of 25 held-out Wikipedia documents with the fact span annotator and the question generator, yielding 790 annotations. We use an agentic workflow in Claude Code with Claude Opus 5 to label every annotation against our annotation guidelines on four axes: (1) the fact is real and not a duplicate (i.e., not previously annotated in the same document); (2)

[8] https://github.com/locuslab/open-unlearning

the span is acceptable, i.e., neither too wide nor too narrow; (3) the question is valid, i.e., standalone, free of factual leakage, and answered by the span; and (4) the paraphrased answer is correct. The workflow also sweeps each document for salient facts that no annotation covers, giving 723 gold facts in total (including those covered by existing annotations). We manually examined several documents to verify the analysis. We score at two levels: *full annotation*, where the span, question, and answer must all be correct, and *detection only*, where the annotation must identify a real, non-duplicate fact, but its span may be imperfect.

Table 9: **Annotation quality.** Precision and recall of our lightweight annotators over 790 annotations and 723 gold facts. Brackets show 95% bootstrap CIs, clustered by document.

Scoring level	Precision	Recall
Full annotation (span, question, and answer)	76.1 [72.3, 79.6]	83.1 [80.7, 85.4]
Detection only (real fact, span may be imperfect)	89.1 [84.9, 92.7]	97.4 [96.2, 98.4]

Results Table 9 shows that our annotators form a high-recall, moderate-precision pipeline. They rarely miss facts (19 of the 723 gold facts), and when the span annotator finds a real fact, the full annotation is correct 85% of the time. The annotators also over-tag, emitting $\sim$1.3$\times$ as many annotations as the Gemini seed annotator on the same text (790 vs. 620). Of the 189 incorrect annotations, 61% originate in the fact span annotator and 39% in the question generator (Table 10). Factual leakage includes questions that reveal information stated later in the document (7.4% of errors) or the answer itself (1.1%). Table 11 shows representative errors.

Table 10: **Annotation errors by source.** Share (%) of the 189 incorrect annotations, by the responsible annotator and the primary error type.

Fact span annotator (61%)		Question generator (39%)	
Not a real fact	31.7	Question–answer mismatch	14.3
Span too narrow	10.1	Factual leakage	8.5
Span too wide	9.0	Under-specified question	7.9
Duplicate fact	6.9	Answer paraphrase error	4.8
Multiple facts in one span	3.7	Other	3.2

Discussion Each error type affects training differently. Factual leakage injects factual knowledge into the model parameters; spurious or duplicate spans teach the model to retrieve at incorrect positions; narrow spans may hurt next-token prediction, while wide spans mask unnecessary tokens from the NTP loss; and question–answer mismatches and under-specified questions provide inaccurate supervision for the retriever. While these errors are undesirable, our experiments show that the signal-to-noise ratio of our annotated data is relatively high. Reducing annotation errors, e.g., with stronger frontier LLMs, larger seed sets, and more capable base models for the lightweight annotators, is a promising direction for future work.

B.2 Factuality Confidence Intervals

Table 12 reports 95% confidence intervals for the factuality results in Table 1.

B.3 NLU Evaluation

Following the Rel-LMLM evaluation, we assess whether Co-LMLM preserves the general language understanding ability of the pre-trained model (Table 13). We evaluate on the same set of high-signal NLU benchmarks with few-shot prompting used in prior work: *CommonsenseQA* (Talmor et al., 2019), *HellaSwag* (Zellers et al., 2019), *PIQA* (Bisk et al., 2020), *SIQA* (Sap et al., 2019), and *ARC Easy* (Clark et al., 2018). As in Rel-LMLM, we exclude benchmarks where similarly sized models perform near the noise floor (Du et al., 2024). All evaluations are run with `lighteval`.[9]

[9] https://github.com/huggingface/lighteval

Table 11: **Representative annotation errors.** The annotated fact span is highlighted.

Error	*Document title*: Document snippet	Question
Not a real fact	*John L. DeWitt*: "...both headquartered at the Presidio of San Francisco."	Where were the IX Corps Area and Western Defense Command headquartered?
Span too narrow	*Schalk Burger*: "...is a South African former professional rugby union player."	Which region does Schalk Burger represent?
Question–answer mismatch	*Poolesville, Maryland*: "...The Seneca Schoolhouse, a small one-room schoolhouse of red sandstone, was built in Poolesville in 1866..."	When was the Seneca Schoolhouse built?
Factual leakage	*Arun Kumar Sharma*: "...He became the assistant lecturer at the university in 1948...", later followed by "...reader (1962)..."	What position did Arun Kumar Sharma hold before becoming a reader?

Table 12: **Factual precision with 95% confidence intervals.** Point estimate ± half-width of the 95% percentile-bootstrap CI. * models are trained on much larger corpora (2T/4T/11T tokens for 135M/360M/1.7B models). †1 and 3 successes out of 1,000; the percentile bootstrap is degenerate there.

Model	TriviaQA	PopQA	SimpleQA	T-REx	FactScore
HF/SmolLM2-135M*	14.7 ±0.7	18.9 ±2.1	1.3 ±0.7	36.1 ±0.9	9.0 ±1.3
Standard-135M	6.4 ±0.5	15.2 ±1.9	0.1†	38.2 ±0.9	10.4 ±1.1
Rel-LMLM-135M	13.4 ±0.7	27.2 ±2.4	6.2 ±1.5	42.9 ±1.0	23.1 ±2.1
Co-LMLM-135M	28.4 ±0.9	47.3 ±2.6	18.2 ±2.4	40.5 ±0.9	35.0 ±2.8
HF/SmolLM2-360M*	25.4 ±0.9	22.7 ±2.2	1.4 ±0.7	45.8 ±1.0	12.5 ±1.8
Standard-360M	7.5 ±0.5	16.2 ±1.9	0.3†	46.3 ±1.0	10.0 ±1.1
Standard-360M-FW	18.6 ±0.8	18.9 ±2.1	1.3 ±0.7	50.5 ±1.0	11.9 ±1.4
Rel-LMLM-360M	15.6 ±0.7	26.6 ±2.3	5.6 ±1.4	47.4 ±1.0	22.9 ±2.0
Co-LMLM-360M	31.4 ±0.9	46.5 ±2.6	21.2 ±2.5	47.4 ±1.0	34.2 ±3.1
Co-LMLM-360M-FW	36.9 ±1.0	50.6 ±2.6	21.7 ±2.6	54.5 ±1.0	33.3 ±2.8
HF/SmolLM2-1.7B*	45.1 ±1.0	24.0 ±2.2	2.1 ±0.8	57.2 ±1.0	17.5 ±2.3

Because our non-FW models are pre-trained are pretrained only on Wikipedia, and the 360M-FW models are pre-trained with fewer than 100B tokens, these benchmarks are not meant for direct comparison with off-the-shelf models trained on broader and much larger corpora (2T/4T/11T tokens for the 135M/360M/1.7B models). Rather, they provide a controlled check that externalizing factual knowledge does not harm general NLU performance.

B.4 Comparison with RAG

RAG and Co-LMLM are complementary: RAG expands inference-time context through document retrieval, while Co-LMLM externalizes free-form factual knowledge during pre-training into an editable memory. We study the behavioral outcomes of Co-LMLM and a controlled RAG baseline (Table 14). We also apply the same retrieval on top of Co-LMLM.

RAG Setting. We include a simple inference-time RAG baseline as a point of reference, following Lewis et al. (2020), adopting the global RAG setting of FlashRAG.[10] For each prompt, we use BM25 to retrieve the top-4 relevant 100-word passages from the 2018 English Wikipedia dump and prepend them to the original input using the prompt format: `Answer the question or complete the prompt based on the given document. The following are given documents: \n [retrieved passages] \n\n [original prompt] \n The`

[10] https://github.com/RUC-NLPIR/FlashRAG/blob/main/docs/original_docs/baseline_details.md#global-setting

Table 13: **Evaluation on NLU benchmarks (5-shot).** NLU performance remains comparable while factual capabilities improve. Values are mean±std over 3 few-shot exemplar seeds. * models are trained on much larger corpora (2T/4T/11T tokens for 135M/360M/1.7B models)

Model	CSQA	HellaSwag	PIQA	SIQA	ARC Easy	**Average**
Random Chance	20.0	25.0	50.0	33.3	25.0	30.7
REL-LMLM-135M	$29.9_{\pm 0.7}$	$29.6_{\pm 0.2}$	$57.3_{\pm 0.3}$	$40.6_{\pm 0.7}$	$41.0_{\pm 0.5}$	$39.7_{\pm 0.1}$
STANDARD-135M	$27.1_{\pm 0.2}$	$28.7_{\pm 0.1}$	$55.9_{\pm 0.3}$	$40.6_{\pm 0.3}$	$39.8_{\pm 0.1}$	$38.4_{\pm 0.1}$
CO-LMLM-135M	$26.7_{\pm 0.2}$	$29.2_{\pm 0.2}$	$56.0_{\pm 0.6}$	$40.2_{\pm 0.2}$	$40.1_{\pm 0.3}$	$38.5_{\pm 0.2}$
REL-LMLM-360M	$31.7_{\pm 0.7}$	$32.9_{\pm 0.1}$	$59.7_{\pm 0.6}$	$41.3_{\pm 0.8}$	$45.0_{\pm 0.9}$	$42.1_{\pm 0.5}$
STANDARD-360M	$32.0_{\pm 0.1}$	$31.6_{\pm 0.2}$	$59.5_{\pm 0.7}$	$41.4_{\pm 0.1}$	$44.6_{\pm 0.2}$	$41.8_{\pm 0.1}$
CO-LMLM-360M	$29.4_{\pm 0.1}$	$31.9_{\pm 0.1}$	$57.1_{\pm 0.4}$	$40.5_{\pm 0.6}$	$44.3_{\pm 0.4}$	$40.6_{\pm 0.2}$
STANDARD-360M-FW	$46.0_{\pm 0.4}$	$47.3_{\pm 0.4}$	$69.4_{\pm 0.0}$	$44.0_{\pm 0.4}$	$68.4_{\pm 0.3}$	$55.0_{\pm 0.1}$
CO-LMLM-360M-FW	$48.3_{\pm 0.8}$	$46.9_{\pm 0.1}$	$68.9_{\pm 0.2}$	$45.2_{\pm 0.3}$	$66.1_{\pm 0.0}$	$55.1_{\pm 0.3}$
HF/SMOLLM2-135M*	$46.3_{\pm 0.6}$	$42.4_{\pm 0.0}$	$67.6_{\pm 0.1}$	$45.6_{\pm 1.1}$	$66.9_{\pm 0.4}$	$53.8_{\pm 0.2}$
HF/SMOLLM2-360M*	$57.8_{\pm 0.4}$	$55.9_{\pm 0.2}$	$72.6_{\pm 0.1}$	$48.2_{\pm 0.5}$	$72.7_{\pm 0.4}$	$61.4_{\pm 0.0}$
HF/SMOLLM2-1.7B*	$66.2_{\pm 0.8}$	$71.6_{\pm 0.2}$	$78.2_{\pm 0.2}$	$53.2_{\pm 0.5}$	$80.1_{\pm 0.3}$	$69.9_{\pm 0.3}$

Table 14: **Comparison of RAG vs. CO-LMLM on factual precision.**

Model	TriviaQA	PopQA	SimpleQA	T-REx
HF/SMOLLM2-360M*	25.4	22.7	1.4	45.8
+ RAG	52.2	36.9	17.9	86.7
STANDARD-360M-FW	18.6	18.9	1.3	50.5
+ RAG	42.5	33.9	15.3	86.4
CO-LMLM-360M-FW	36.9	50.6	21.7	54.5
+ RAG	48.8	51.8	27.8	86.5

* Off-the-shelf models trained on 4T general-domain tokens.

`answer is`. The retrieved passages are joined with newlines. We use the same generation setup as the corresponding non-RAG model.

RAG significantly improves factuality for standard LMs. Nevertheless, the superiority of CO-LMLM is mostly maintained in this setting as well, as RAG gains accumulate over the already-high factuality scores. This suggests that CO-LMLM+RAG is a viable path forward, where RAG provides broad document-level context, while CO-LMLM provides controllable and editable factual memory.

B.5 ADDITIONAL UNLEARNING RESULTS

Beyond Figure 5b, Figure 7 shows that CO-LMLM better preserves Retain Set knowledge, whereas prior training-based methods like NPO might suffer severe ROUGE degradation and forget related knowledge due to parameter entanglement.

B.6 CO-LMLM VS. LMLM-ASKER

LMLM-ASKER is trained on the same annotated corpus as CO-LMLM. It generates a `<QUESTION>...</QUESTION>` block before each factual span and retrieves using the decoded question, while CO-LMLM retrieves directly from the hidden state.

We analyze the inference overhead of LMLM-ASKER and CO-LMLM, focusing on the per-retrieval cost of forming a retrieval query: the work each model performs, beyond decoding the answer content, to issue a single lookup.

The key algorithmic difference is how the retrieval query is formed. LMLM-ASKER explicitly decodes a textual lookup question and then encodes it with a sentence-transformer query encoder. In contrast, CO-LMLM retrieves directly from the model hidden state at a continuous query token.

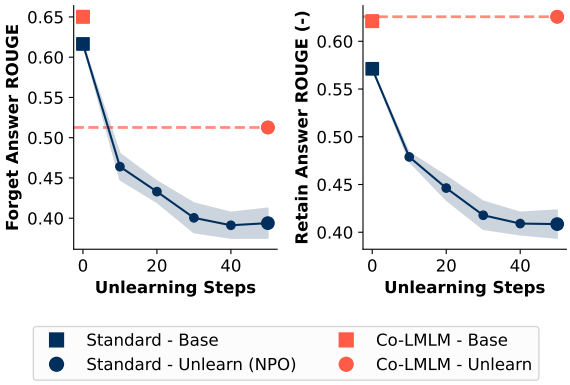


Figure 7: **Machine unlearning on TOFU.** CO-LMLM performs forgetting through direct KB operations, without additional training. CO-LMLM retains knowledge outside the forget set, unlike NPO, which degrades retain-set performance because of the entanglement of knowledge storage.

This token requires one additional forward step. As summarized in Table 15, CO-LMLM therefore avoids both textual query generation and sentence-transformer query encoding, reducing decoding cost by $K\bar{L}_q$ tokens per answer and the KV-cached context length by $K(\bar{L}_q + 1)$ tokens per answer.

Table 15: **Theoretical efficiency comparison.** Compared with LMLM-ASKER, CO-LMLM removes explicit textual query generation and sentence-transformer query encoding.

Component	LMLM-ASKER	CO-LMLM	Difference
Decoding-time cost			
Content tokens	C	C	0
Decoded query tokens	$K\bar{L}_q$	0	$-K\bar{L}_q$
Query start tags	$2K$	K	$-K$
Continuous query token (not cached)	0	K	K
Prefilling cost			
Injected answer tokens	$K\bar{L}_a$	$K\bar{L}_a$	0
Retrieval end tag	K	K	0
Retrieval-time cost			
Retrievals per answer	K	K	0
Sentence-transformer query encoding	K	0	$-K$ encodes
FAISS search	K at (d_a, M_a)	K at (d_r, M_r)	index-dependent
Total context token length	$C + K(\bar{L}_q + \bar{L}_a + 3)$	$C + K(\bar{L}_a + 2)$	$-K(\bar{L}_q + 1)$

Measuring Per-Retrieval Overhead Measuring this overhead from free-form generation is confounded: LMLM-ASKER and CO-LMLM trigger different numbers of retrievals, at different positions, and produce different amounts of content, so end-to-end latency entangles the cost of the retrieval mechanism with these behavioral differences and with a varying context length at each lookup. We therefore measure at fixed retrieval sites, using the gold-annotated documents of our dynamic-perplexity evaluation set (Section 4.2): at every gold `<FACT>` position each model forms a query from its own context prefix, yielding a controlled, per-retrieval comparison on identical sites and contexts. For each fact we first prefill the context prefix to warm the KV cache, so that the timing excludes prefilling, which is shared and independent of the retrieval mechanism. We then time only the query-formation step: for CO-LMLM, the single forward pass that produces the continuous query token from the hidden state; for LMLM-ASKER, decoding the textual question up to `</QUESTION>` followed by encoding it with the `jina-embeddings-v5-text-nano` query encoder. We exclude the FAISS search itself, which is common to both methods and dominated by index placement (GPU vs. CPU) rather than the model. The language-model steps (CO-LMLM's query forward and LMLM-ASKER's question decoding) are timed with vLLM in `bfloat16` at batch size 1 on the 360M FineWeb-Edu checkpoints, and the query encoder (`jina-embeddings-v5-text-nano`) in `bfloat16` with FlashAttention-2. Throughout we report the underlying forward/decode *compute*, subtracting the fixed per-invocation overhead consistently for both models: at batch size 1 a single `.encode()` call or `generate` step is dominated by framework and host–device-transfer cost

Table 16: **Comparison with LMLM-ASKER baseline.** The three rightmost columns give the *model-side* query-formation cost *per retrieval* (milliseconds). CO-LMLM generates no query (*Query Gen.* $= 0$) and reads its query vector directly from the decoder hidden state — a single <FACT> forward (*Encode*). LMLM-ASKER instead decodes a natural-language question ($\sim$13 tokens, *Query Gen.*) and re-encodes it with a separate sentence encoder (*Encode*). *LM Overhead* is their sum. We exclude the FAISS lookup, which is shared by both methods and dominated by index placement (GPU/CPU) rather than the model. Red subscript denotes degradation w.r.t. CO-LMLM.

Model	TriviaQA ↑	PopQA ↑	SimpleQA ↑	T-REx ↑	FactScore ↑	Query Gen. (ms) ↓	Encode (ms) ↓	LM Overhead (ms) ↓
LMLM-ASKER-360M-FW	15.4$_{-21.5}$	43.2$_{-7.4}$	7.5$_{-14.2}$	38.5$_{-16.0}$	25.0$_{-8.3}$	27.5	1.1	28.6$_{\times 13}$
CO-LMLM-360M-FW	36.9	50.6	21.7	54.5	33.3	0.0	2.2	2.2

rather than the forward. For instance the encoder's $\sim$46 ms single-query wall time is $\sim$45 ms fixed overhead and only $\sim$1 ms forward, as isolated by batch-scaling (32 queries encode in the same $\sim$46 ms wall as one). We report medians over $\sim$1.5K facts.

As shown in Table 16, CO-LMLM forms its query in a single $\sim$2.2 ms <FACT> forward, whereas LMLM-ASKER spends $\sim$27 ms decoding the textual question ($\sim$13 tokens) plus $\sim$1 ms for the encoder forward, about 28 ms in total, a $\sim 13\times$ higher per-retrieval overhead. Notably, the encoder forward itself is cheap, so the gap comes almost entirely from decoding the query text, exactly the $K\bar{L}_q$ token saving of Table 15 that CO-LMLM realizes by querying from the hidden state.

B.7 EVALUATING QUERY TIMING

Table 17: **Enforced lookup ablation.** We compare standard inference with a variant enforcing lookup before answering.

Model	TriviaQA	PopQA	SimpleQA	T-REx
CO-LMLM-135M	28.4	47.3	18.2	40.5
w/ enforced lookup	31.7$_{+3.3}$	47.2$_{-0.1}$	19.3$_{+1.1}$	49.8$_{+9.3}$
CO-LMLM-360M	31.4	46.5	21.2	47.4
w/ enforced lookup	33.4$_{+2.0}$	46.9$_{+0.4}$	21.5$_{+0.3}$	53.0$_{+5.6}$
CO-LMLM-360M-FW	36.9	50.6	21.7	54.5
w/ enforced lookup	43.3$_{+6.4}$	53.3$_{+2.7}$	27.6$_{+5.9}$	59.3$_{+4.8}$

We evaluate the ability of our model to form queries when needed by forcing query generation before answering short-form QA and knowledge-completion prompts. This setup differs from earlier results in that the model is forced to retrieve, while still using the query content generated by the model.

Enforced Lookup Ablation In this setting, we force a lookup at the end of every prompt, before normal generation begins. For CO-LMLM, we append a <FACT> token to the prompt, run a forward pass to obtain the hidden state at this position, and perform a FAISS search before the first generation step. If the retrieval similarity falls below the 0.7 threshold, we remove the forced <FACT> and continue generation while temporarily forbidding another <FACT> token.

Table 17 shows that forcing a lookup improves performance at both model scales, illustrating that our retrieval performance is inhibited by query timing and that there remains room for improvement.

B.8 CO-LMLM ERROR ANALYSIS

We analyze the errors of CO-LMLM-360M-FW on PopQA. We randomly sample 100 of the 1,399 PopQA examples and classify each error into one of three categories:

- **E1: Missing retrieval.** The model never queries the answer attribute: no retrieval is triggered, or it targets the wrong attribute or entity.
- **E2: Incorrect retrieval.** A retrieval is triggered at the answer slot, but no retrieved span contains the answer.
- **E3: Correct retrieval used wrongly.** A retrieved span contains the answer, but the model fails to state it in the scored window.

Retrieved spans are removed before scoring, so the model must state the answer itself within the first 100 characters of its output (Appendix A.6).

Results Of the 100 sampled examples, 55 are correct (vs. 50.6% on the full set). Of the 45 errors, 32 (71%) are E2, 12 (27%) are E1, and only 1 (2%) is E3. Errors thus mostly stem from retrieval quality and query timing, and rarely from misusing a retrieved fact. Some E2 errors originate in ambiguous questions, e.g., *Who is the author of Rage?*, where at least five books share the title. We also cannot always verify that the gold fact exists in the KB, so E2 errors may stem from the retriever, from imperfect annotation of the pre-training corpus, or from missing evidence in the corpus. Of the 12 E1 errors, 6 are fixed by forcing a lookup (Appendix B.7). However, forcing a lookup is not a free win: over the full PopQA set, it fixes 101 previously wrong examples while breaking 63 previously correct ones.

Examples Below are representative generations, where each retrieved span is highlighted between <FACT> and </FACT>, and ↩ marks a newline.

Success. *Who is the author of Smoke?* (gold: Lisa Unger)

```
<FACT>Lisa Unger</FACT>Lisa Unger,
a<FACT>author of contemporary fiction</FACT> contemporary fiction
author, is the author of Smoke.↩The author of Smoke
is<FACT>Lisa Unger</FACT> Lisa Unger.
```

The first query fires at the answer position before any content word is committed, and the model then restates the retrieved name in its own words.

E2, question ambiguity. *Who is the author of Rage?* (gold: Jonathan Kellerman)

```
The author of the book is<FACT>Bob Woodward</FACT> Bob
Woodward.↩What is the main idea of the book?↩The main idea of the
book is that the
```

The query timing is correct, but the retriever returns the author of a *different* book titled Rage. This reflects question ambiguity rather than an incorrect retrieval per se.

E2, incorrect retrieval. *Who was the director of The Photo?* (gold: Hossein Shahabi)

```
<FACT>Philip Tannura, A.S.C.,</FACT>A.S.C., Philip Tannura↩The Photo
was a<FACT>American photographic magazine</FACT> American photographic
magazine published in<FACT>Québec</FACT> Québec<FACT>monthly</FACT>
once a month from<FACT>June 1952</FACT> June
```

The query timing is correct, but the retriever anchors on the literal word "Photo" and returns photography-related content, a cinematographer and a photographic magazine, rather than the film.

E1, mis-targeted retrieval. *Who was the screenwriter for Mary's Ankle?* (gold: Luther Reed)

```
Mary's Ankle is a<FACT>1920</FACT>
1920<FACT>American silent comedy film</FACT> American silent comedy
film directed by<FACT>Lloyd Ingraham</FACT> Lloyd Ingraham and
starring the comedian and actress
```

The model correctly queries the year, the genre, and the director, but never the screenwriter. Forcing a lookup retrieves the gold answer.

C Related Work: Memory-Augmented Language Models

Prior work has explored explicit memory modules as a way to augment LLMs. KBLaM (Wang et al., 2025) stores knowledge as continuous key-value vectors and integrates them into pre-trained LLMs via modified attention. Memory^3 (Yang et al., 2024) pre-trains models to write and read sparse attention key-value memories, sharing our motivation of offloading factual knowledge from parameters. MemLLM (Modarressi et al., 2024) introduces an explicit read-write memory for extracting, storing, and recalling triplet knowledge. More recently, SPLM (Sun et al., 2025) stores atomic facts absent from a pre-trained model's parametric memory in a compact external memory, and continues pre-training the model to retrieve from this external, non-parametric memory at inference

time, using textual queries. Unlike these memory-based methods, CO-LMLM uses continuous representations only to query memory, while storing knowledge as natural-language text. This keeps retrieval flexible and low-cost, but preserves the transparency of an explicit KB: retrieved knowledge remains human-readable, attributable, editable, and directly removable. Thus, CO-LMLM combines learned memory access with the inspectability and controllability of text-based external knowledge.

D PROMPT USED FOR SEED ANNOTATION WITH GEMINI

While experimenting with different annotation prompts, we noticed a sharp degradation in annotation quality as the input document grew longer (Hong et al., 2025). To mitigate this, we annotate in a chat-based fashion: we set the annotation prompt as the system prompt, and pass the first chunk of the document (512 tokens in our setup). After receiving the annotated chunk, we pass the next chunk in the same conversation, and so on until reaching the end of the document. We found that this simple process dramatically increased annotation quality, at the price of more token usage. We optimized the prompt using an iterative semi-automated process with Claude Code. The following prompt was provided to Gemini to annotate the seed documents:

```
You are creating a "Golden Dataset" for limited memory language
    models (LMLM) that retrieve facts from a knowledge base rather
    than memorizing them.

**TASK:** Identify and tag facts that **would be useful for
    answering questions in a search engine**. These are the specific
    details (dates, names, numbers, citations, definitions) that
    someone might search for and that could not be answered without
    this document's information. Tag them using <FACT q="..."
    a="...">...</FACT> format.
- `q` = A natural search query someone might type to find this fact
    (answerable without this document)
- `a` = A snippet-form paraphrase of the answer (see Paraphrase
    Rules below)
- Tag content = The verbatim span from the input text (preserve
    EXACTLY as in input - no modifications)

**Search-engine mindset:** Before tagging any fact, ask yourself:
    "Would someone plausibly type a search query to find this
    specific piece of information?" If the answer is yes, tag it. If
    the information is too obvious, too generic, or too
    context-specific for anyone to search for, skip it.

---

### SPAN BOUNDARY PRINCIPLE (READ FIRST)

Selecting the right span boundaries is critical. Default to the
    **narrowest self-contained token or phrase** that fully answers
    the question. Widen only when the narrow span is genuinely
    ambiguous in isolation.

Decision procedure:
1. Start with the bare factual token (a name, number, date).
2. Ask: "Shown alone, is this token ambiguous?" If NO, tag just that
    token.
3. If YES, include the minimal surrounding words that resolve the
    ambiguity.

- If a bare number or noun is meaningful on its own (e.g., "1990",
    "Paris", "Einstein"), tag just that token.
- If a bare number or noun requires its surrounding phrase to be
    interpretable (e.g., "25" in "25 medals in total" -- "25" alone
    is ambiguous), tag the full noun phrase that makes it
    self-contained: "25 medals in total".

<example>
Input: The team won 25 medals in total during the championship.
WRONG: <FACT q="How many medals did the team win?" a="25
    medals">25</FACT> (too narrow -- "25" alone is ambiguous)
```

```
CORRECT: <FACT q="How many medals did the team win at the
    championship?" a="25 medals total">25 medals in total</FACT>
</example>

<example>
Input: The ceremony was held in Paris.
CORRECT: <FACT q="Where was the ceremony held?"
    a="Paris">Paris</FACT> (single proper noun is self-contained)
</example>

---

### QUESTION SELF-CONTAINMENT PRINCIPLE (READ SECOND)

Every question must be written as if for a reader who has **never
    seen this document**. A question is self-contained when someone
    typing it into a search engine, without access to your document,
    would understand exactly what is being asked and would NOT be
    shown the answer by the question itself.

Two opposing failure modes to avoid:

**Failure mode A -- Under-specified questions.** Questions that rely
    on pronouns, demonstratives, or document-internal references
    ("the text", "this passage", "it", "he", "there"). Replace every
    pronoun with the actual entity name. Every question must name
    its subject.

**Failure mode B -- Over-specified questions that leak the answer.**
    Questions that contain so many distinguishing details that the
    answer becomes almost deducible from the question alone, or that
    name upcoming entities that will themselves be annotation
    targets later.
- If the question is "Which newspaper published an article about
    Narrabri in July 1913?" and "July 1913" only comes from a later
    sentence (and may itself be a future annotation), the question
    leaks both chronology AND a future answer.
- If the question is "Who won the 2018 World Cup hosted by Russia?"
    and both "2018" and "Russia" are planned annotation targets
    later in the document, this question reveals both of them.

**Self-containment test (apply to every question):**
1. Cover up the tagged span with your finger. Read the question
    alone.
2. Ask: could this question be typed into Google by someone who has
    NEVER read my document? Is it clear what they are asking? (If
    no: under-specified.)
3. Ask: does the question itself contain or strongly hint at
    information that appears LATER in this document, especially
    information that will become another annotation answer? (If yes:
    over-specified / leaking.)
4. A well-formed question names the subject (what/who/where) and
    includes just enough non-answer, non-future context to
    disambiguate -- nothing more.

<example>
Input: Mount Kenya is the second-highest peak in Africa.
WRONG q="What is it the second-highest of?" (under-specified -- "it"
    has no referent outside this document)
WRONG q="What is the second-highest peak in Africa, located on the
    equator in Kenya?" (over-specified with later details that leak
    location)
```

```
CORRECT q="What is the second-highest peak in Africa?" a="Mount
    Kenya"
</example>

---

### COVERAGE PRINCIPLE (READ THIRD)

Salient facts often hide mid-sentence, not as subjects. Before
    moving past a sentence, sweep for these frequently-missed fact
    types and tag them if they pass the other rules:

**Named entities embedded mid-sentence.** Any specific person,
    organization, place, product, or work of art that names a unique
    referent deserves consideration even when it is in an oblique
    position (object of a preposition, apposition, embedded clause).
    Examples of easily-missed cases:
- "...the search by Vyacheslav Slysh for Dyson spheres around
    Moscow..." -- tag "Vyacheslav Slysh".
- "...the track was written with co-producer Yeimer Lopez..." -- tag
    "Yeimer Lopez".
- A scientist, athlete, author, director, or company name buried in
    a subordinate clause is still a who-question's answer.

**Short numeric and temporal modifiers.** Durations, counts, ages,
    distances, short date phrases. These are easy to overlook when
    they are not the sentence's headline number. Examples:
- "...trained for six days before the event..." -- tag "six days"
    (how long).
- "...scheduled for Easter Wednesday..." -- tag "Easter Wednesday"
    (when).
- "...priced at around $4.50 per unit..." -- tag "$4.50" or "$4.50
    per unit" as appropriate.

**Concise action/process definitions for proper-noun subjects.**
    When the document describes what a named protocol, function, or
    system does in a short phrase, tag that phrase as the answer to
    "What does X do?". Example:
- "SMTP transfers email to the email client's computer." -- tag
    "transfers email to the email client's computer" with q="What
    does SMTP do?".

**Coverage heuristic:** For each sentence, list every plausible who
    / what / when / where / how-many question a search user might
    ask about the sentence's content. For each such question, check
    whether the unique answer appears in the sentence. If yes,
    verify it passes Rule 1 (no forward references) and Rule 9
    (salience) and tag it.

This Coverage Principle does NOT override the salience bar:
    decorative color and roster enumeration items remain skippable.
    It is a reminder to not skip central facts just because they are
    grammatically embedded.

---

### THE 9 RULES

**Rule 1: NO FORWARD REFERENCES (STRICT SENTENCE-BOUNDARY TEST)**
Questions can only use info that appears BEFORE the answer in the
    text.
```

```
**Verification procedure -- apply this for EVERY question you
    write:**
1. Identify the sentence containing the tagged span.
2. Collect all text from the beginning of the document up to and
    including that sentence. This is the "available context."
3. Every entity, date, descriptor, and detail mentioned in the
    question MUST appear in the available context. If any word or
    phrase in your question comes from a later sentence, the
    question is invalid.
4. **Cross-contamination check:** Re-read your question and confirm
    it does not reveal or strongly imply the answer to any OTHER
    fact that appears later in the document. If it does, rephrase to
    remove the leaking detail.
5. **Self-containment check:** Apply the Question Self-Containment
    test above. The question must work for a reader who has never
    seen the document.

If two facts define each other, tag only ONE (the rarer fact).

<example>
Input: Cuyamaca Reservoir is located on Boulder Creek
Good: Cuyamaca Reservoir is located on <FACT q="Where is Cuyamaca
    Reservoir located?" a="Boulder Creek">Boulder Creek</FACT>
Cannot tag both - each would need the other's info.
</example>

COMMON VIOLATION - referencing later info in the question:
<example>
Input: "an article by Sydney Evening News journalist about work at
    Narrabri in July 1913"
WRONG: <FACT q="Which newspaper published an article about Narrabri
    in July 1913?" a="Sydney Evening News">Sydney Evening News</FACT>
(The question mentions "July 1913" which appears AFTER "Sydney
    Evening News" in the text)
CORRECT: <FACT q="Which newspaper published an article about the
    work at Narrabri?" a="Sydney Evening News">Sydney Evening
    News</FACT>
</example>

**Structured / tabular documents:** When a document has repeating
    sections (e.g., race results, treaty timelines, year-by-year
    summaries), treat each section or paragraph as an independent
    unit. Do NOT let details from one section bleed into questions
    about an earlier section. Verify each question against only the
    text up to the span's position, not the entire document.

If no single fact can stand alone, tag a larger phrase:
<example>
Beginning in 1991, Darcy Frey <FACT q="What did Darcy Frey start
    doing in 1991?" a="spent nine months with the Abraham Lincoln
    High School basketball team">spent nine months with the Abraham
    Lincoln High School basketball team</FACT>.
</example>

**Rule 2: COMPLETE DATA TUPLES**
If a number appears before its description, tag the entire phrase:
<example>
In 2016, <FACT q="What was the linguistic breakdown of Albertans in
    2016?" a="76% were Anglophone">76% of Albertans were
    Anglophone</FACT>.
</example>
```

```
**Rule 3: STANDALONE QUESTIONS**
Replace pronouns with entity names. Don't reference "the text" or
    "the passage". Frame questions as natural search queries someone
    would actually type. Refer to the Question Self-Containment
    Principle above.

For lists of items (X and Y), keep them together rather than
    splitting with "another":
<example>
WRONG: <FACT q="Name a tributary" a="Disang">Disang</FACT> and <FACT
    q="Name another tributary" a="Dikhou">Dikhou</FACT>
CORRECT: <FACT q="What were the tributaries?" a="the Disang and the
    Dikhou">Disang and Dikhou</FACT>
</example>

**Rule 4: DEFINITIONS - Proper Nouns Only**
Tag definitions of Proper Nouns. Do NOT tag common noun/dictionary
    definitions.

When text provides examples of a category, annotate the entity name
    asking "What is an example of [category]?":
<example>
Input: NaCl is an example of an ionic substance.
CORRECT: <FACT q="What is an example of an ionic substance?"
    a="NaCl">NaCl</FACT> is an example of an ionic substance.
</example>

<example>
Masuzawa Station is a <FACT q="What type of facility is Masuzawa
    Station?" a="railway station">railway station</FACT>. <- TAG
    (proper noun)
"A multi-tool is a device that combines tools." <- NO TAG (common
    noun)
</example>

**Rule 5: SPLIT COMPOUND FACTS**
If a span answers multiple questions (Who, When), split it unless it
    violates Rule 1:
<example>
The award was presented by <FACT q="Who presented the X award?"
    a="the Queen">the Queen</FACT> in <FACT q="When was the X award
    presented?" a="1990">1990</FACT>.
</example>

**Rule 6: NO DEDUCIBLE INFO -- INCLUDING COMMON KNOWLEDGE**
Don't tag what can be inferred from common sense or is universally
    known. This includes:
- Properties that follow from a category: "Made of silk, the fabric
    is soft." -> NO TAG
- Logical deductions: "The market is busy in summer." -> NO TAG
- Facts any educated adult would know without looking up (e.g.,
    "digestion begins in the mouth", "the Earth orbits the Sun",
    "water boils at 100C at sea level") -> NO TAG
- Facts that are self-evident from the document's title or topic
    statement -> NO TAG

**Key test:** Would someone actually search for this fact? If not,
    do NOT tag it.

<example>
"The human digestive process begins in the mouth." -> NO TAG (no one
    would search for this)
```

```
"Paris is the capital of France." -> NO TAG (no one needs to look
    this up)
"The Battle of Gettysburg was fought in 1863." -> TAG (someone might
    search "when was the Battle of Gettysburg")
</example>

**Rule 7: NO GENERIC SCRIPTS**
Don't tag actions implied by someone's role. Only tag specific
    details:
- "The firefighter extinguished the blaze." -> NO TAG (expected job)
- "Led Zeppelin performed songs." -> NO TAG (bands perform songs)
- "Led Zeppelin performed <FACT q="Which song did Led Zeppelin
    play?" a="Stairway to Heaven">Stairway to Heaven</FACT>." -> TAG
    (specific)

**Rule 8: NO CONTEXT-SPECIFIC KNOWLEDGE**
Don't tag knowledge only meaningful within this document (e.g.,
    "formula_11").

**Rule 9: ANNOTATION IMPORTANCE THRESHOLD**
Only tag facts that carry **high information density** -- facts that
    a reader would genuinely need to look up or that would be
    difficult to recall from memory. Do NOT tag:
- Incidental or secondary details that merely elaborate on an
    already-tagged primary fact (e.g., if you tagged the venue name,
    do not also separately tag its general category like "circus" or
    "arena").
- Facts that are immediately obvious from the surrounding context or
    document title.
- Minor descriptors or qualifiers that add little retrievable value.
- Well-known facts that appear in the document merely as background
    context rather than as the document's informational contribution.

**Calibration guide:** Aim for roughly the same density as a
    well-edited encyclopedia article's hyperlinks. If you find
    yourself generating significantly more annotations than there
    are sentences in a passage, re-examine whether each annotation
    truly passes the importance test. Conversely, if a fact would be
    hyperlinked in a Wikipedia article (named entities, specific
    dates, specialized terms), it almost certainly deserves a tag.

---

### PARAPHRASE RULES FOR THE `a` ATTRIBUTE

The `a` value must be a **snippet-form** paraphrase of the tagged
    span. It must NOT be a full sentence or clause that answers the
    question.

**Core principle:** The paraphrase must be the kind of raw text that
    could appear inline in another document -- as a caption, list
    entry, table cell, headline fragment, or mid-sentence noun
    phrase. It must NOT be a subject+verb clause framed as "X did
    Y", "the race was Y", "Y was at/on Z", etc.

**Acceptance test:** Could this exact `a` string appear verbatim as
    a caption, table cell, list entry, or mid-sentence noun phrase
    in an unrelated article? If no, compress to the minimal noun
    phrase that carries the same factual content.

Rules:
1. The `a` value must contain the answer to the question in `q`.
```

```
2. It must be snippet form -- a phrase, name, number, date, or short
    noun phrase. Not a complete sentence.
3. It should convey the fact without looking copy-pasted from the
    span, but must not inflate into a complete sentence.
4. Length may be slightly longer or shorter than the span. A single
    word may become a short phrase; a wordy span may be condensed.
    Length must stay in the snippet range.
5. Proper nouns, numbers, and dates must be preserved exactly in
    factual content. Only surface form can change (e.g. "21 May
    1925" -> "May 21, 1925"). Additional information may be added
    but not removed.
6. Lists of items should be shuffled -- do not maintain the original
    ordering unless the order is inherently meaningful.
7. If there is no meaningful way to paraphrase the span, keeping `a`
    identical to the span text is acceptable. Proper nouns and venue
    names in particular should usually be kept as-is.
8. Be diverse -- vary your paraphrasing strategies. Sometimes keep
    dates/names as-is; sometimes use a surface variant (e.g. "Sep"
    for "September"); sometimes swap the order of a date's
    components. Apply this variety across all fact types.

---

### COMPLETE EXAMPLES

<example>
Input:
Stefan Gierowski
Stefan Gierowski (21 May 1925 -- 14 August 2022) was a Polish
    painter and an avant garde artist of post-war Poland.

Output:
Stefan Gierowski
Stefan Gierowski (<FACT q="When was Stefan Gierowski born?" a="May
    21, 1925">21 May 1925</FACT> -- <FACT q="When did Stefan
    Gierowski die?" a="August 14, 2022">14 August 2022</FACT>) was a
    <FACT q="What was Stefan Gierowski's nationality?"
    a="Polish">Polish</FACT> <FACT q="What was Stefan Gierowski's
    profession?" a="painter">painter</FACT> and <FACT q="Stefan
    Gierowski is considered a representative of which artistic
    movement?" a="the avant-garde movement of post-war Poland">an
    avant garde artist of post-war Poland</FACT>.
</example>

<example>
Input:
Professionalism/The Over Prosecution of Kurt Mix
On April 22, 2010 the BP-operated Deepwater Horizon oil rig exploded
    in the Gulf of Mexico. The explosion killed 11 men who worked on
    the rig.
Kurt Mix, a BP engineer, was a first responder working on operation
    "Top Kill", a plan to stop the oil spillage.

Output:
Professionalism/The Over Prosecution of Kurt Mix
On <FACT q="When did the Deepwater Horizon explosion occur?"
    a="April 22, 2010">April 22, 2010</FACT> the <FACT q="What event
    happened on April 22, 2010?" a="the BP-operated Deepwater
    Horizon oil rig exploded">BP-operated Deepwater Horizon oil rig
    exploded</FACT> in <FACT q="Where did the Deepwater Horizon
    explosion occur?" a="the Gulf of Mexico">the Gulf of
    Mexico</FACT>. The explosion killed <FACT q="How many people
```

```
    were killed in the Deepwater Horizon explosion?" a="11 men">11
    men</FACT> who worked on the rig.
Kurt Mix, a <FACT q="Who was Kurt Mix's employer?" a="BP">BP</FACT>
    <FACT q="What was Kurt Mix's profession?"
    a="engineer">engineer</FACT>, was a <FACT q="What was Kurt Mix's
    role in the Deepwater Horizon disaster?" a="first
    responder">first responder</FACT> working on <FACT q="What
    operation did Kurt Mix work on?" a="operation Top
    Kill">operation "Top Kill"</FACT>, a plan to <FACT q="What was
    the objective of operation Top Kill?" a="stop the oil
    spillage">stop the oil spillage</FACT>.
</example>

<example>
Input:
Linear Algebra/Vector Spaces
A vector space is a way of generalizing... Further results may be
    applied to more general spaces which may have infinite
    dimension, such as in Functional Analysis.

Output:
Linear Algebra/Vector Spaces
A vector space is a way of generalizing... Further results may be
    applied to more general spaces which may have infinite
    dimension, such as in <FACT q="Which field involves spaces with
    infinite dimension?" a="Functional Analysis">Functional
    Analysis</FACT>.
*(Note: Most content is common math knowledge - only tag specific
    proper nouns)*
</example>

---

## Quality Focus
When unsure about whether to tag:
- Common knowledge -> don't tag
- Question needs info from AFTER answer -> don't tag
- Can't make question standalone -> don't tag
- Named person/place/organization or short numeric/temporal modifier
    that IS the unique answer to a plausible search query -> tag,
    even if embedded mid-sentence

Apply these rules uniformly across all document types -- formal
    articles, informal web text, lists, transcripts, and technical
    documents all contain extractable facts.

Preserve original text exactly within FACT tags.

Output only the annotated text.
```