Separating Factual Knowledge from Large Language Models via Fact-Conditional Pretraining

Anonymous ACL submission

Abstract

Large language models (LLMs) store factual knowledge in their opaque parameters. This makes factual knowledge difficult to inspect, verify, and update when it becomes outdated or incorrect. We introduce Fact-Conditional Pretraining (FCP), a novel pretraining approach to separate factual knowledge from LLMs. For each training document, FCP extracts atomic subject-relation-object triples as factual priors and trains the model to reconstruct the original document conditioned on these priors. By consistently presenting factual information in context, FCP encourages the model to rely on explicit contextual knowledge rather than memorizing factual knowledge within its parameters. Experiments across 3 knowledge-probing and 6 linguistic evaluation benchmarks show that FCP successfully separates factual knowledge from LLMs while largely preserving their linguistic proficiency. Remarkably, this separation effect becomes increasingly pronounced as data scale and model capacity expand.

1 Introduction

Large language models (LLMs) develop strong language generation capabilities through large-scale pretraining. However, because factual associations repeatedly appear as statistical regularities in text, pretraining also encourages models to internalize vast amounts of factual knowledge in their parameters. As a result, factual knowledge and linguistic competence become tightly entangled. This entanglement becomes problematic in real-world applications, as, on the one hand, factual knowledge is dynamic and requires frequent updating and, on the other hand, linguistic competence should remain stable, general, and reusable across contexts.

This raises a central research question: *How to separate factual knowledge from LLMs while preserving their general linguistic competence?* Current strategies address the question primarily through two paradigms: (1) Entity-anonymized pretraining (Boutet et al., 2026; Vakili et al., 2024; Zhao et al., 2025) masks or pseudonymizes entities in the training corpus to block factual memorization. However, replacing natural text with artificial placeholders or pseudonyms introduces a distribution mismatch between pretraining and real-world language use, which can degrade downstream utility. (2) Gradient-penalty methods (Yao and Xu, 2024; Zhang et al., 2024) instead manipulate the optimization objective to suppress factual memorization, for example, by applying negative gradients on forget sets or penalizing factual tokens. While effective in targeted settings, such gradient manipulation can be unstable and may cause collateral damage to general language modeling ability (Maini et al., 2024).

We propose *Fact-Conditional Pretraining (FCP)*, a simple and stable pretraining paradigm to disentangle factual knowledge from LLMs. The key idea of FCP is to expose the factual knowledge as an explicit conditioning context, so that the model can retrieve and use these facts from the context rather than encode them as static parametric memory. This design changes the role of the model parameters. In vanilla pretraining, the model must simultaneously learn *what* facts are true and *how* to express them. In FCP, factual knowledge is already available in the context, so the model is encouraged to learn to use, verbalize, and compose it rather than memorize it as static parameters. FCP still optimizes the model with the vanilla autoregressive cross-entropy objective. It therefore preserves the natural language distribution, avoiding the distribution mismatch of entity-anonymized pretraining and the training instability of gradient-penalty unlearning.

To instantiate this idea, we first extract atomic subject-relation-object triples from each text in the training corpus to serve as its explicit factual prior. We use triples because they provide a compact and interpretable form of factual knowledge: they make

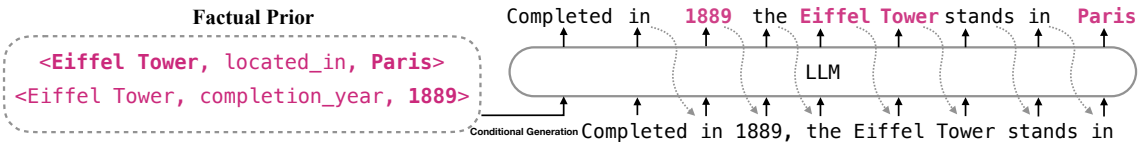

Figure 1: An example of fact-conditional pretraining (FCP). FCP shifts factual knowledge from static parameters to context. As a result, the models are encouraged to learn how to use, verbalize, and compose factual information, rather than memorize the facts themselves.

the key entities, relations, and attributes visible to the model before generation, while avoiding the need to replace natural text with artificial placeholders. Second, we prepend these triples to the text and train the model to generate the text conditioned on them. The triples are wrapped in a fixed template so that the model can reliably recognize them as a factual prior. During training, the loss is computed only on the original text. This is crucial because our goal is not to teach the model to memorize or reproduce the triples, but to teach it how to leverage those triples to produce fluent natural language.

We evaluate FCP across three factual probing datasets and six linguistic evaluation benchmarks. Experiments demonstrate that FCP successfully achieves the desired factual–linguistic separation: it effectively reduces parametric factual knowledge without compromising the core language competence.

Our contributions are summarized as follows: (1) We introduce *Fact-Conditional Pretraining* (FCP), a pretraining paradigm that separates factual knowledge from LLMs by providing factual information as explicit conditioning context, encouraging the model to use facts from context rather than memorize them in its parameters. (2) We develop a scalable, automated pipeline to generate pretraining corpora for FCP, enabling high-throughput fact-conditioned text generation without relying on costly human annotations. (3) We demonstrate that FCP substantially reduces parametric factual knowledge while largely preserving the model's linguistic understanding and generation capabilities. Remarkably, this decoupling effect becomes even more pronounced as data scale and model capacity expand.

2 Related Work

Entity-Anonymized Pretraining. This branch of work focuses on reducing factual knowledge from the LLMs via entity anonymization. KLLMs (Cohen et al., 2026) first identifies named entities in the pretraining corpus and replaces them with anonymous placeholders, so that the model is trained on entity-masked text rather than explicit entity mentions. Similarly, Boutet et al. (2026) adopts a hybrid detection pipeline that combines rule-based patterns with neural NER models to locate sensitive entity spans, and then substitutes them with type-level placeholders such as <NAME> and <DATE> before masked language modeling (MLM). To preserve discourse coherence, pseudonymization frameworks (Vakili et al., 2024; Cabrera-Diego and Gheewala, 2024) conduct post-NER co-reference resolution, mapping all mentions of the same entity to a stable but semantically meaningless pseudonym. LMLM (Zhao et al., 2025) employs a different approach. It extracts triples from raw text and stores them in an external database, forcing the model to acquire facts from the constructed database rather than memorize them within its parameters.

However, replacing natural text with artificial placeholders introduces a distribution mismatch between the sanitized pretraining corpus and ordinary language use, which can reduce the model's downstream utility (Yang et al., 2019). In contrast, our method keeps the original text as the pretraining target and uses factual triples only as additional conditioning context. This allows the model to learn from natural language while relying on explicit factual priors, reducing the need to store factual knowledge in its parameters without severely disrupting language modeling.

Gradient Penalty Unlearning. Rather than modifying the pretraining corpus, this line of work reduces parametric factual knowledge by manipulating gradients. Early methods (Eldan and Russinovich, 2023; Jang et al., 2023) apply gradient ascent to the tokens or documents to be forgotten, increasing their next-token prediction loss and thereby lowering the model's likelihood of reproducing the corresponding factual content. To make

this process more stable and reduce utility loss, later work combines forgetting and retaining objectives: negative updates are applied to the forget set examples, while standard gradient descent is applied to a retain set examples to preserve general model behavior (Yao and Xu, 2024). NPO (Zhang et al., 2024) also improves training stability via preference optimization. Instead of directly maximizing the loss on the forget set, NPO penalizes the model when it assigns high likelihood to forget-set examples relative to the reference model, thereby reducing factual memorization while avoiding the instability of unconstrained gradient ascent.

Gradient-penalty methods perform targeted forgetting after the model has already learned the corresponding facts. In contrast, FCP intervenes at the pretraining stage itself: by providing factual knowledge as context, it reduces the model's need to store such knowledge in its parameters from the outset.

3 Methodology

In this section, we introduce fact-conditional pretraining (FCP), a novel pretraining paradigm to separate factual knowledge from LLMs. As shown in Figure 1, FCP trains the model to generate text conditioned on factual priors. For example, when the model generates the text *Completed in 1889, the Eiffel Tower stands in Paris.*, FCP explicitly provides <*Eiffel Tower, located_in, Paris*> and <*Eiffel Tower, completion_year, 1889*> to the model. In this way, the model does not need to memorize factual knowledge in its parameters; instead, it can directly use the factual prior to complete the generation.

In the following sections, we first introduce how to construct the training corpus for FCP, and then introduce the FCP training objective and optimization process.

3.1 Factual Prior Extraction

In this section, we describe how to construct the training corpus for FCP. Since FCP trains the model to generate text conditioned on factual priors, each training example should consist of a text paired with its corresponding factual prior.

The key challenge is how to construct a factual prior that preserves the main factual information in the text while remaining simple and easy to condition on. We instantiate factual priors as knowledge triples. Each triple represents a factual statement in a simple subject-relation-object format, preserving the core factual content while removing unnecessary surface details from the original text.

We prompt LLMs to extract triples. The detailed extraction prompt is provided in Appendix A. Compared with traditional information extraction pipelines that depend on predefined schemas or task-specific supervision, such as OpenIE (Angeli et al., 2015), the LLM-based extractor offers greater flexibility and scalability. Formally, given the text $d = (y_1, y_2, \ldots, y_T)$, where y_i is the i-th token in d and T is the total numbers of token in d, we prompt LLMs to extracts factual triples $K = \{(s_j, r_j, o_j)\}_{j=1}^{M}$ from d, where M is the total number of triples, s_j, r_j, and o_j denote the subject, relation, and object of the j-th extracted factual triple.

3.1.1 Model Selection

Selecting a suitable LLM for triple extraction is crucial, as low-quality triples may omit important facts, introduce unsupported information, or represent facts in inconsistent forms, which can weaken the conditioning signal during pretraining. We evaluate 18 LLM annotators from 6 model families, covering DeepSeek-R1-Distill (Guo et al., 2025), Llama-3.2 (Grattafiori et al., 2024), Mistral [1], Phi-4 (Abdin et al., 2024), Qwen-3 (Yang et al., 2025), and Gemma-3 (Team et al., 2025).

We construct a golden human annotation set to evaluate the quality of these LLM extractors. We first ask an annotator with a background in linguistics to extract factual triples from 10 articles sampled from FineWeb-Edu (Penedo et al., 2024), resulting in 289 unique candidate triples. We then invite three additional annotators to independently judge whether each candidate triple is valid. Annotation guidelines are shown in Appendix A. Across 10 documents and 289 unique candidate triples, the three annotators achieve a Fleiss' kappa (Fleiss, 1971) of 0.3739. This corresponds to fair agreement given the open-ended nature of factual triple extraction. We then construct the final human-validated triple set using majority voting: a triple is retained if at least two annotators mark it as correct. This procedure yields 234 final triples, accounting for 80.97% of all candidate triples.

We evaluate the LLM extractors using macro-F_1 against the human-validated reference triples. Let $\mathcal{T}_L$ and $\mathcal{T}_H$ denote the sets of LLM-generated and

[1] https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407

human-annotated triples, respectively. Precision is defined as the proportion of generated triples that match at least one human-annotated reference triple above a similarity threshold $\tau = 0.8$:

$$\text{Precision} = \frac{1}{|\mathcal{T}_L|} \sum_{t \in \mathcal{T}_L} \mathbb{I}\left(\max_{t' \in \mathcal{T}_H} \text{Sim}(t, t') \geq \tau \right)$$

where $\text{Sim}(\cdot, \cdot)$ measures the similarity between two triples, and $\mathbb{I}(\cdot)$ is the indicator function. Recall is computed symmetrically as the proportion of human-annotated triples that are recovered by the LLM-generated triples:

$$\text{Recall} = \frac{1}{|\mathcal{T}_H|} \sum_{t' \in \mathcal{T}_H} \mathbb{I}\left(\max_{t \in \mathcal{T}_L} \text{Sim}(t, t') \geq \tau \right)$$

F1 is then calculated as the harmonic mean of precision and recall.

We use the LLM-as-the-judge (Llama3.1-70B [2]) to evaluate the similarity between the triples. To mitigate potential biases inherent in the LLM-as-a-judge paradigm, we complement our evaluation with a deterministic baseline: fuzzy string matching, which directly quantifies surface-form lexical similarity. On both similarity metrics, Qwen3-8B, Qwen3-14B, and Gemma-3-12b-it are the top three models (see Appendix B). We chose Qwen3-8B as our final LLM annotator to balance extraction quality and efficiency.

In total, our pipeline yields 49.9 million fine-grained triple-text pairs from about 20 billion FineWeb-Edu, with an average of 28 triples per text. We commit to publicly releasing this entire annotated dataset to promote reproducibility. More statistical results are shown in Appendix C.

3.2 Fact-Conditional Pretraining

This phase serves as the architectural core for achieving a *limited-memory* mechanism. Instead of allowing the model to blindly memorize raw text, we prepend the extracted triple set K_i as an explicit antecedent condition to the original document d_i. This setup compels the language model to predict text sequences under a fully informed factual context.

Step 1: Sequence Linearization and Concatenation. To feed the structured knowledge into the transformer architecture, the factual prior triple set K_i is first transformed into a flat token sequence using a linearization function $\text{Linearize}(\cdot)$. Let this linearized factual prefix be represented as a sequence of L tokens:

$$X_i = \text{Linearize}(K_i) = (x_1, x_2, \ldots, x_L)$$

We then construct the conditional training sequence $\widetilde{d}_i$ by prepending the factual prefix X_i to the original document tokens $d_i = (y_1, y_2, \ldots, y_T)$:

$$\widetilde{d}_i = [X_i; d_i] = (x_1, \ldots, x_L, y_1, \ldots, y_T)$$

Step 2: Conditional Autoregressive Language Modeling During pretraining, the entire concatenated sequence $\widetilde{d}_i$ is fed into the network. The cross-entropy loss is computed only over the target document tokens y_t to ensure the model focuses exclusively on conditional text generation rather than capturing the data distribution of the factual priors. Formally, given the causal masking mechanism, the fact-conditional pretraining objective minimizes the negative log-likelihood of the document tokens conditioned on both the prior prefix X_i and the preceding tokens $y_{<t}$:

$$\mathcal{L}_{\text{FCP}}(d_i) = -\sum_{t=1}^{T} \log p(y_t \mid y_{<t}, X_i)$$

From an information-theoretic perspective, explicitly exposing the factual record X_i within the context window fundamentally alters the model's optimization landscape. The most sample-efficient and lowest-entropy path for gradient descent is to dynamically retrieve or 'copy' factual cues directly from the activation states of the context X_i, rather than routing gradients to hard-code those exact facts into the static parametric weights (e.g., the MLP layers). Consequently, FCP enforces a structural decoupling: the parametric memory is relieved of factual storage and is instead incentivized to focus purely on mastering generalized linguistic fluency, syntax mapping, and contextual reasoning.

4 Experiments

In this section, we examine whether FCT effectively separates factual knowledge from LLMs while preserving their language capabilities. We quantify this separation by addressing two questions: (1) Does FCT successfully mitigate parametric factual memorization? and (2) Does FCP compromise downstream linguistic competence? This evaluation design aligns with the central goal of our study: a separated model should rely less on memorized factual knowledge while maintaining its general language modeling ability.

[2] https://huggingface.co/meta-llama/Llama-3.1-70B

4.1 Benchmarks and Metrics

Knowledge-side Datasets. **CUB-CounterFact** (Hagström et al., 2025) contains diverse types of Wikidata relation triples (e.g., birthplaces, alma maters). Each example in CUB-CounterFact contains an original factual object and an edited counterfactual object. We leverage the original factual targets as the ground-truth answers. MyriadLAMA[3] extends LAMA with diverse relational templates and entity expressions, and PopQA (Mallen et al., 2023) is a 14k-example open-domain QA benchmark constructed from Wikidata triples. We retain only examples where the object appears at the end of the template, so that the target object can be evaluated as a suffix continuation.

We evaluate performance via likelihood-based candidate ranking (see Appendix E.1), where the candidate sets are formed by grouping objects with the same relation. We report Acc@5 for CUB-CounterFact, Acc@20 for PopQA, and MyriadLAMA (see Appendix E.2).

Language-side Datasets. We use **MRPC** (Dolan and Brockett, 2005) and **PAWS** (Zhang et al., 2019) to assess the ability to understand semantic nuances. MRPC contains 5,801 sentence pairs extracted from online news sources, with human labels indicating whether the two sentences are semantically equivalent paraphrases. PAWS contains 108,463 human-labeled paraphrase and non-paraphrase pairs with high lexical overlap, making it a challenging benchmark for distinguishing surface word overlap from true semantic equivalence. We use **CoLA** (Warstadt et al., 2019) and **BLiMP** (Warstadt et al., 2020) to assess grammatical comprehension. CoLA contains 10,657 English sentences from linguistics publications, each annotated for grammatical acceptability. **BLiMP** consists of 67 sub-datasets, each containing 1,000 sentence pairs that differ minimally but contrast in grammatical acceptability, covering syntactic, morphological, and semantic phenomena in English. We use **WikiText-2** (Merity et al., 2017) and **Alpaca** (Taori et al., 2023) to evaluate language generation ability. WikiText-2 is a language modeling benchmark built from high-quality Wikipedia articles, containing around 2M tokens in the training split and standard validation/test splits. Alpaca contains 52K instruction-following demonstrations generated in a self-instruct style.

[3] https://huggingface.co/datasets/iszhaoxin/MyriadLAMA

CoLA, MRPC, and PAWS are evaluated through supervised downstream fine-tuning: we fine-tune the evaluated model on the training split of each task, and report the official validation metrics, namely Matthews correlation for CoLA and F1 for MRPC and PAWS. We fine-tune for 3 epochs per dataset. BLiMP is evaluated by minimal-pair likelihood ranking, where an example is correct if the grammatical sentence receives a better rank than the ungrammatical sentence. WikiText-2 and Alpaca are evaluated using teacher-forced gold-continuation likelihood, reported as byte-normalized NLL (see Appendix E.1 and E.3).

4.2 Baselines and Controlled Training Setup

We conduct our experiments on the SmolLM2 model family (Allal et al., 2025), utilizing two distinct model scales—SmolLM2-135M and SmolLM2-360M—to verify whether the impact of fact-conditional pretraining (FCP) remains consistent across parameter sizes.

To strictly isolate the effects of our training objective, we perform a controlled ablation study comparing **vanilla pretraining (vanilla)** with our **fact-conditional pretraining (FCP)**. For each scale, both the baseline and FCP models are initialized from the identical SmolLM2 seed checkpoint, pretrained on the same 20B token subset of FineWeb-Edu (Penedo et al., 2024), and optimized using identical training budgets and hyperparameters (see Appendix E.4). The difference only lies in the input formatting: Vanilla pretraining uses the original raw text as input, whereas FCP uses the same text but prepends template-wrapped factual triples as additional context. (see Appendix D for template details).

4.3 FCT Reduces Factual Knowledge

Table 1 shows that FCP consistently reduces parametric factual knowledge across both model scales. The largest reduction occurs on CUB-CounterFact, where Acc@5 drops from 31.20 to 19.20 for SmolLM2-135M and from 71.60 to 56.60 for SmolLM2-360M, corresponding to absolute reductions of 12.00 and 15.00 points, respectively. MyriadLAMA also shows clear reductions under Acc@20, decreasing from 42.27 to 37.25 for SmolLM2-135M and from 50.11 to 41.61 for SmolLM2-360M. Together, these results indicate that FCP substantially weakens direct subject–relation–object recall from model parameters, not only reducing the top-ranked factual prediction but

Model	Method	Knowledge Probing		
		CUB-CounterFact Acc@5↓	**MyriadLAMA** Acc@20↓	**PopQA** Acc@20↓
SmolLM2-135M	Vanilla	31.20	42.27	6.60
	FCP(ours)	19.20	37.25	6.40
	Δ	-12.00	-5.02	-0.20
SmolLM2-360M	Vanilla	71.60	50.11	8.00
	FCP(ours)	56.60	41.61	6.80
	Δ	-15.00	-8.50	-1.20

Table 1: Factual knowledge reduces performance. Lower Acc@K indicates that our method successfully reduces the amount of parametric factual knowledge in LLMs. Δ denotes FCP minus Vanilla Baseline.

also pushing correct factual objects further down the likelihood ranking. PopQA shows a weaker but directionally consistent reduction, with Acc@20 decreasing from 6.60 to 6.40 for SmolLM2-135M and from 8.00 to 6.80 for SmolLM2-360M. This smaller effect is mainly due to a floor effect: both baselines already retrieve very few PopQA answers within the top 20 candidates.

4.4 FCP Retains Linguistic Competency

Table 2 shows that the linguistic competency difference between the baseline and our FCP. Overall, FCP successfully reduces factual memorization without causing a broad collapse in language understanding and generation abilities. This is expected because FCP merely alters how factual knowledge is structured and provided during training, leaving the foundational distributions of general syntactic structures and functional language patterns completely unperturbed.

On CoLA and BLiMP, FCP leads to only moderate decreases: CoLA changes by -0.010 for SmolLM2-135M and -0.006 for SmolLM2-360M, while BLiMP decreases by 3.13 and 2.09 points, respectively. This is likely due to the distributional shift introduced by the structured factual prior, rather than a fundamental loss of grammar understanding competence. FCP even performs slightly better than the baseline on semantic matching tasks. This improvement may come from the auxiliary alignment signal introduced by FCP. By learning to generate text from explicit factual triples, the model must align structured factual priors with their natural-language realizations. This may improve its sensitivity to whether two sentences express the same underlying meaning, which is central to MRPC and PAWS.

Experiments on WikiText-2 and Alpaca show that FCP largely retains its language-generation capability, with only a moderate increase in ByteNLL. To better understand this increase, we decompose the loss by token type. Our hypothesis is that if FCP mainly attenuates factual memorization, the increase in ByteNLL should concentrate on fact-bearing tokens, such as entities, rather than spread uniformly across the sequence. In other words, the generation degradation should reflect lower confidence in factual predictions, not a general decline in linguistic fluency. To test this, we use spaCy to identify named-entity tokens in WikiText-2 and compute ByteNLL separately on these tokens, denoted as Factual ByteNLL. We compare it with the ByteNLL over all tokens, denoted as Global ByteNLL.

Table 3 shows a clear asymmetry. For SmolLM2-135M, Factual ByteNLL increases by +0.1353, compared with a Global ByteNLL increase of only +0.0846. The same pattern holds for SmolLM2-360M, where Factual ByteNLL increases by +0.1218, while Global ByteNLL increases by +0.0706. In both cases, the increase in loss for named entities is substantially larger than the average increase across all tokens.

This result sharpens the interpretation of the generation regression: FCP does not mainly degrade general language modeling, but makes the model less confident when predicting factual or entity-bearing tokens. This is exactly the expected behavior if factual knowledge has been attenuated from the model parameters. Therefore, the moderate increase in generation ByteNLL provides additional evidence of factual forgetting rather than a broad loss of linguistic competence.

4.5 Model Scaling Amplifies Separation

As demonstrated in Table 1, the attenuation of parametric factual knowledge becomes more pronounced as the model capacity expands. On CUB-

Model	Method	Language Understanding				Language Generation	
		MRPC Acc/F1 ↑	**PAWS** Acc/F1 ↑	**CoLA** MCC ↑	**BLiMP** Rank Acc. ↑	**WikiText-2** ByteNLL ↓	**Alpaca** ByteNLL ↓
SmolLM2-135M	Vanilla	78.92 / 84.91	88.11 / 87.14	0.323	77.16	0.9179	0.6268
	FCP(ours)	79.66 / 85.66	91.23 / 90.33	0.313	74.03	1.0025	0.6970
	Δ	+0.74 / +0.75	+3.12 / +3.19	-0.010	-3.13	+0.0846	+0.0702
SmolLM2-360M	Vanilla	81.37 / 86.85	91.59 / 90.84	0.473	78.96	0.7678	0.4955
	FCP(ours)	83.33 / 88.47	93.33 / 92.58	0.467	76.87	0.8384	0.5535
	Δ	+1.96 / +1.62	+1.74 / +1.74	-0.006	-2.09	+0.0706	+0.0580

Table 2: Language competency preservation across model scales. Higher scores indicate better performance on MRPC, PAWS, CoLA, and BLiMP; lower ByteNLL indicates better performance on WikiText-2 and Alpaca.

Model	Factual ByteNLL Δ	Global ByteNLL Δ
135M	+0.1353	+0.0846
360M	+0.1218	+0.0706

Table 3: Analysis of ByteNLL increase on WikiText-2. Factual ByteNLL is computed only on named-entity tokens, while Global ByteNLL is computed over all tokens.

CounterFact, Acc@5 drops by 12.00 points for SmolLM2-135M, from 31.20 to 19.20, and by 15.00 points for SmolLM2-360M, from 71.60 to 56.60. A similar pattern appears on MyriadLAMA, where Acc@20 decreases by 5.02 points for 135M and by 8.50 points for 360M. PopQA shows a weaker reduction due to its low baseline accuracy, but the drop is still larger for 360M (-1.20) than for 135M (-0.20). These results suggest that larger models store more recoverable factual associations in their parameters, leaving more room for FCP to attenuate factual knowledge.

Scaling up also reduces FCP's linguistic side effects. As shown in Table 2, the side effects of FCP are generally milder overall. For language understanding, the 360M model has smaller drops on CoLA (-0.006 vs. -0.010) and BLiMP (-2.09 vs. -3.13), while obtaining larger gains on MRPC (+1.96/+1.62 vs. +0.74/+0.75). For language generation, the increase in ByteNLL is also smaller for 360M than for 135M on both WikiText-2 (+0.0706 vs. +0.0846) and Alpaca (+0.0580 vs. +0.0702).

4.6 Data Scaling Amplifies Separation

In this section, we investigate how the decoupling of factual knowledge and linguistic competence evolves as the volume of training data increases. We use SmolLM2-135M as the testbed and evaluate intermediate checkpoints from training on 2B to 20B tokens. For each checkpoint, we compare the vanilla pretraining baseline with FCP under the same model size and training budget. We track three indicators: training loss, factual retention (measured by CUB-CounterFact Acc@5), and linguistic competency (measured by MRPC F1).

As shown in Figure 2(a), FCP consistently achieves much lower training loss than the baseline across all training sizes. At 20B tokens, the baseline training loss is 3.3977, while FCP reaches 1.7649. This indicates that conditioning generation on explicit factual triples substantially simplifies the training objective. Instead of forcing the model to both memorize factual content and learn how to express it, FCP shifts factual content into the input context, allowing the model to focus more on learning how to use and verbalize the provided facts.

Figure 2(b) further shows that the factual-retention gap between FCP and the baseline model becomes larger as training size increases. The baseline gradually accumulates more factual knowledge: CUB-CounterFact Acc@5 increases from 9.40 at 2B tokens to 31.00 at 20B tokens. In contrast, FCP remains much lower, rising only from 5.20 to 19.00. As a result, the factual-retention gap between the baseline and FCP widens from 4.20 points at 2B tokens to 12.00 points at 20B tokens. This suggests that more training data gives the baseline more opportunities to store factual associations in parameters, whereas FCP learns to rely more on explicit factual context.

At the same time, Figure 2(c) shows that linguistic competency remains comparable. MRPC F1 stays close between the two methods across most checkpoints. This small difference suggests that the reduction in factual retention does not come from a broad degradation of language understanding. Instead, FCP primarily reduces parametric factual memorization while preserving sentence-

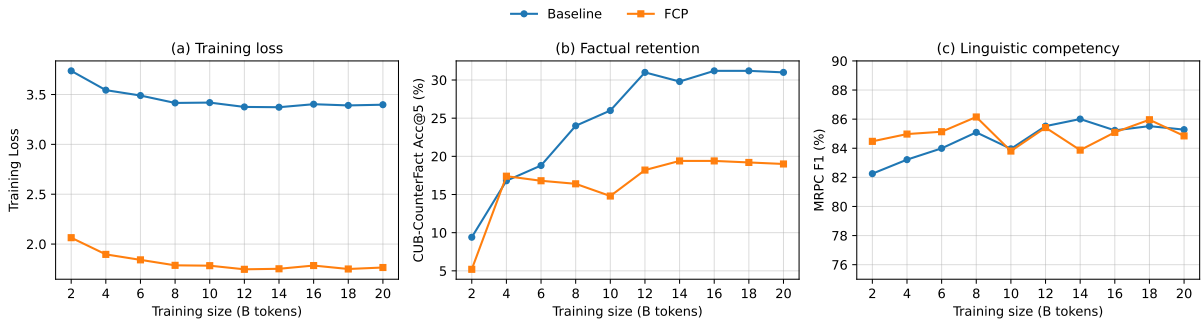

Figure 2: Scaling curves of baseline and FCP on SmolLM2-135M. From left to right: (a) training loss, (b) CUB-CounterFact Acc@5, and (c) MRPC F1. As training size increases, FCP achieves lower training loss and lower factual retention, while maintaining comparable linguistic competency.

Model	Method	Context Faithfulness		
		Overall Acc@1 ↑	Low-Freq Obj Acc@1 ↑	High-Freq Obj Acc@1 ↑
SmolLM2-135M	Vanilla	91.00	94.68	88.78
	FCP(ours)	**98.80**	**100.00**	**98.08**
SmolLM2-360M	Vanilla	96.40	99.47	94.55
	FCP(ours)	**99.40**	**100.00**	**99.04**

Table 4: Context Faithfulness Evaluation. Higher Acc@1 indicates a stronger tendency to follow the context.

level semantic matching.

4.7 FCP Improves Context Faithfulness

In this section, we evaluate whether FCP enables the model to faithfully use contextual factual knowledge, rather than relying on its memorized parametric priors.

To evaluate this, we construct counterfactual contexts using the edited objects in CUB-CounterFact. For each query, we replace the original object with a counterfactual object and include the edited fact in the prompt. For example, before answering "Paris is the capital of", the model is given the edited fact that "Paris is the capital of Japan". The baseline receives edited facts as a natural-language statement, while FCP receives them as factual triples wrapped up in the same facts-to-text template used during training.

Table 4 shows that FCP improves the model's context faithfulness, making the model more reliably follows the provided factual context rather than relying on its parametric knowledge. For SmolLM2-135M, overall Acc@1 increases from 91.00 to 98.80, yielding a +7.80 point improvement. For SmolLM2-360M, Acc@1 increases from 96.40 to 99.40, yielding a +3.00 point improvement. This makes FCP valuable for evidence-grounded applications, such as medical decision support, where dosage recommendations must strictly follow the supplied clinical guideline.

We further split edited objects into high- and low-frequency groups based on their occurrence counts in the training corpus. Objects above the median are labeled high-frequency; the rest are labeled low-frequency. As shown in Table 4, the improvement is especially pronounced on high-frequency objects. For SmolLM2-135M, high-frequency object Acc@1 improves from 88.78 to 98.08, a +9.30 point gain. For SmolLM2-360M, it improves from 94.55 to 99.04, a gain of 4.49 points. This pattern is expected because high-frequency facts are more strongly memorized by baseline models and are therefore harder to override. By separating factual knowledge from the model parameters, FCP reduces this parametric stubbornness and makes it easier to update even frequently observed factual associations in context.

5 Conclusion

We propose *Fact-Conditional Pretraining* (FCP), a novel paradigm that exposes factual knowledge as contextual priors rather than requiring the model to store it in its parameters. Experiments show that FCP reduces parametric factual knowledge while largely preserving linguistic competence. Scaling analyses suggest that, as model size or training data increases, FCP achieves stronger factual attenuation with limited linguistic side effects.

6 Limitation

Although FCP shows promising results, several limitations remain.

First, FCP depends on the quality and coverage of the extracted factual priors. If the triple extractor misses important facts, extracts noisy triples, or fails to capture implicit factual information, the model may generate incomplete or incorrect outputs because it is trained to rely on the provided factual context. This makes FCP particularly suitable for scenarios where the factual context is reliable, curated, or externally verifiable, such as enterprise knowledge bases, clinical guidelines, legal documents, and product manuals. In contrast, FCP may be less suitable for settings where the available context is noisy, incomplete, rapidly assembled, or difficult to verify, such as open-web retrieval, social media content, user-generated forums, or ambiguous multi-source evidence.

Second, FCP does not guarantee the complete removal of all factual information from model parameters. Some facts may still be learned indirectly through repeated linguistic patterns, co-occurrence statistics, or incomplete factual priors.

Third, FCP introduces additional preprocessing and context-length overhead. Constructing factual priors requires running a fact extraction pipeline over the training corpus, and prepending triples increases the input length during pretraining. Although this cost is manageable in our experimental setup, more efficient extraction, filtering, and compression methods are needed for training at larger web-scale regimes.

7 Acknowledgments

AI tools were used only for language polishing and readability improvement. The authors retain full responsibility for the content, analysis, and conclusions of this work.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

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, and 3 others. 2025. Smollm2: When smol goes big – data-centric training of a small language model. *Preprint*, arXiv:2502.02737.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Antoine Boutet, Lucas Magnana, and Juliette Sénéchal. 2026. Towards the anonymization of the language modeling. *Preprint*, arXiv:2501.02407.

Luis Adrián Cabrera-Diego and Akshita Gheewala. 2024. PSILENCE: A pseudonymization tool for international law. In *Proceedings of the Workshop on Computational Approaches to Language Data Pseudonymization (CALD-pseudo 2024)*, pages 25–36, St. Julian's, Malta. Association for Computational Linguistics.

Roi Cohen, Yvan Carré, Nick Lechtenbörger, Hendrik Droste, Lucas Kerschke, Russa Biswas, Gerard de Melo, and Jan Buys. 2026. Knowledgeless language models: Decoupling linguistic competence and factual knowledge.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Ronen Eldan and Mark Russinovich. 2023. Who's harry potter? approximate unlearning in llms. *Preprint*, arXiv:2310.02238.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Lovisa Hagström, Youna Kim, Haeun Yu, Sang-goo Lee, Richard Johansson, Hyunsoo Cho, and Isabelle Augenstein. 2025. Cub: Benchmarking context utilisation techniques for language models. *arXiv preprint arXiv:2505.16518*.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. 2024. Tofu: A task of fictitious unlearning for llms. *Preprint*, arXiv:2401.06121.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Thomas Vakili, Aron Henriksson, and Hercules Dalianis. 2024. End-to-end pseudonymization of fine-tuned clinical bert models: privacy preservation with maintained data utility. *BMC Medical Informatics and Decision Making*, 24.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Yuanshun Yao and Xiaojun Xu. 2024. Large language model unlearning. *Advances in Neural Information Processing Systems*, 37:105425–105475.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. *Preprint*, arXiv:2404.05868.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. Paws: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308.

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. 2025. Pre-training limited memory language models with internal and external knowledge. *arXiv preprint arXiv:2505.15962*.

A Guideline and Prompt for Factual Prior Extraction

We use the following prompt to extract factual prior from a given passage. The prompt requires the model to extract only explicitly stated facts and linearize each fact into a fixed subject-relation-object format. All four annotators are authors or collaborators of this work. No external crowdworkers were recruited. The annotation was conducted as part of the annotators' research contribution, and no additional monetary compensation was provided.

System Prompt

```
breaklines
You are an information extraction system. Extract atomic factual triples from the passage.
```

User Prompt: Task and Output Format

```
breaklines
# 1. Task
Extract factual triples that are explicitly stated in the passage.

Do not:
- infer unstated facts
- add outside knowledge

# 2. Output Format
Output ONLY triples in this format:

[Start]subject[Mid-SR]relation[Mid-RO]object[End]

Do not output numbering, bullet points, introductions, or comments.
```

User Prompt: Faithfulness and Atomicity

```
breaklines
# 3. Faithfulness
- Every triple must be directly supported by the passage.
- If the passage conflicts with real-world knowledge, follow the passage.

# 4. Atomicity
- Each triple must express exactly ONE independent fact.
- Split sentences with multiple facts into multiple triples.
- Split listed items into separate triples.
- Subjects and objects must not contain "and" or "or".
- If listed items share a modifier, repeat the modifier in every relevant triple.

Example:
"LLMs inherit inaccuracies and biases in training data."

Output:
[Start]large language model[Mid-SR]inherits[Mid-RO]inaccuracies in training data[End]
[Start]large language model[Mid-SR]inherits[Mid-RO]biases in training data[End]
```

User Prompt: Subject Selection and Stand-alone Triples

```
breaklines
# 5. Subject Selection
- Choose concise and central subjects.
- Prefer subjects under 5 words when possible.
- Reuse the same central subject across triples when appropriate.
- Prefer the central entity as the subject rather than the object.

# 6. Stand-alone Triples
- Do not use pronouns such as "it", "they", "this", "these", or "those".
- Replace pronouns with the specific entity or concept they refer to.
- Keep important modifiers in subjects and objects.
- Each triple should be understandable without reading the original passage.
```

User Prompt: Normalization

```
breaklines
# 7. Normalization
- Use natural English phrases with spaces for subjects and objects.
- Use short snake_case relations.
- Do not include commas inside subjects, relations, or objects.
- Do not output duplicate triples.

Canonical forms:
- LLM -> large language model
- GPT -> generative pre-trained transformer
```

User Prompt: Special Cases

```
breaklines
# 8. Special Cases

## Negation
Use negative relations such as lacks, does_not_have, does_not_change, or cannot.

Do not put "no" or "not" inside subjects or objects.

Example:
"Chameleons don't change colors to match their surroundings."

Output:
[Start]Chameleons[Mid-SR]does_not_change[Mid-RO]colors to match surroundings[End]

## Conditions
For conditional facts, include the condition in the subject or object.

Example:
"If water reaches 100 degrees C, it boils."

Output:
[Start]water at 100 degrees C[Mid-SR]undergoes[Mid-RO]boiling[End]

## Category-level facts
For general facts about a class, use the class as the subject.

Example:
"All stars emit light."

Output:
[Start]star[Mid-SR]emits[Mid-RO]light[End]

## Examples and instances
For examples or specific instances, extract both the instance relation and the stated property.

Example:
"Some planets, like Mars, have ice."

Output:
[Start]Mars[Mid-SR]instance_of[Mid-RO]planet[End]
[Start]Mars[Mid-SR]has[Mid-RO]ice[End]
```

User Prompt: Full Examples

```
breaklines
# 9. Full Examples

Input:
"LLMs, such as GPT, are powerful tools developed by OpenAI. They can generate text and code."

Output:
[Start]generative pre-trained transformer[Mid-SR]instance_of[Mid-RO]large language model[End]
[Start]large language model[Mid-SR]is_a[Mid-RO]powerful tool[End]
[Start]large language model[Mid-SR]developed_by[Mid-RO]OpenAI[End]
[Start]large language model[Mid-SR]can_generate[Mid-RO]text[End]
[Start]large language model[Mid-SR]can_generate[Mid-RO]code[End]

Input:
"Smartphones use lithium-ion batteries and OLED displays that are manufactured in specialized factories."

Output:
[Start]smartphone[Mid-SR]uses[Mid-RO]lithium-ion battery[End]
[Start]smartphone[Mid-SR]uses[Mid-RO]OLED display[End]
[Start]lithium-ion battery[Mid-SR]manufactured_in[Mid-RO]specialized factory[End]
[Start]OLED display[Mid-SR]manufactured_in[Mid-RO]specialized factory[End]

Input:
"Average life span in the wild: 12 years
Size: 21 in (50 cm)
Weight: 14.4 oz (408 g)
Did you know? Chameleons don't change colors to match their surroundings."

Output:
[Start]Chameleons[Mid-SR]has_average_life_span[Mid-RO]12 years in the wild[End]
[Start]Chameleons[Mid-SR]has_size[Mid-RO]21 in (50 cm)[End]
[Start]Chameleons[Mid-SR]has_weight[Mid-RO]14.4 oz (408 g)[End]
[Start]Chameleons[Mid-SR]does_not_change[Mid-RO]colors to match surroundings[End]

# Passage
<<<
{TEXT}
>>>
```

B LLM Annotators Evaluation

Figure 3 compares the macro-F1 scores of different LLM extractors for triple extraction. Models are ranked according to their performance under the LLM-as-the-judge evaluation, while fuzz string matching is included as an additional automatic matching metric.

Overall, larger, stronger instruction-tuned models achieve substantially higher macro-F1 scores, with Qwen3-14B, Qwen3-8B, and gemma-3-12b-it among the best-performing extractors. In contrast, smaller models generally obtain much lower scores. A closer inspection suggests that their weaker performance is mainly due to low recall: these models tend to extract fewer triples per article, thereby missing many factual statements covered by the human annotations.

C Pretraining Corpus Statistics

D Fact-Conditional Prompt Template

FCT uses a chat-level facts-to-text prompt. The factual triples are inserted into the user message, and the original passage is used as the assistant response during training. The prompt template is:

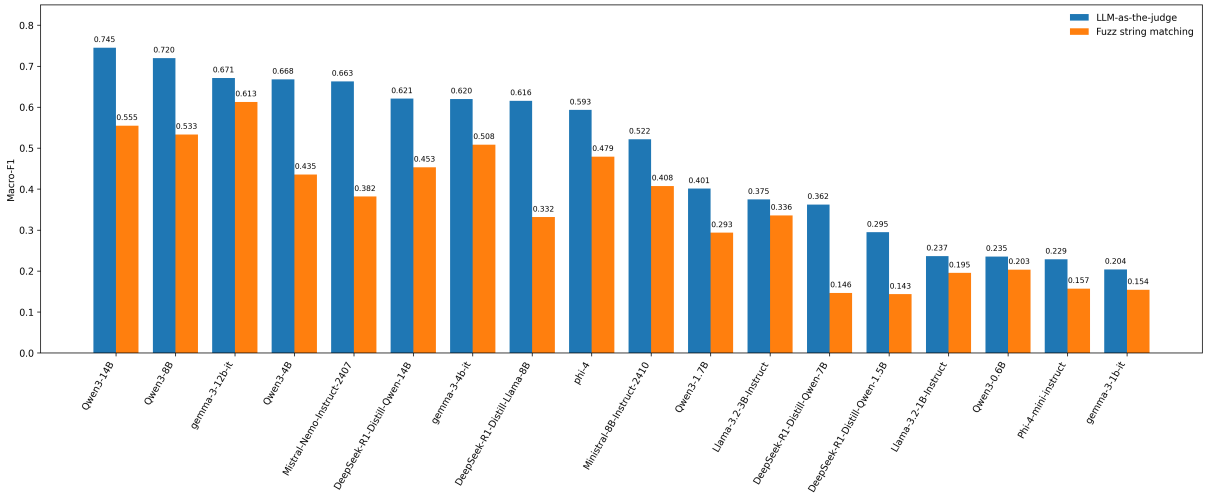

Figure 3: Performance of LLMs on Triple Extraction. LLMs with worse performance mainly because they extract fewer triples from the given text.

Table 5: Pretraining Corpus Statistics.

Metric	**Value**
Total Documents in Corpus	49,946,998
Sampled Size for Audit	1,000
Average Triples per Document	28.00
Median Triples per Document	27.00
Minimum Triples per Document	1
Maximum Triples per Document	101
Non-empty Factual Documents	1,000 / 1,000 (100%)
Zero-triple Documents	0 / 1,000 (0%)

Fact-Conditional Prompt Template

```
<|im_start|>user
Reconstruct the original passage from the following factual triples.

Requirements:
1. Preserve all factual content expressed by the triples.
2. Produce a fluent and coherent natural-language passage.
3. Do not add facts that are not supported by the triples.
4. Do not output the triples themselves.

Factual triples:
{facts}
<|im_end|>
<|im_start|>assistant
{text}
<|im_end|>
```

Here, `{facts}` denotes the extracted factual triples for the passage, and `{text}` denotes the original passage. In the baseline setting, the model is trained directly on `{text}` raw text without the fact-conditional chat prompt.

E Evaluation

E.1 Likelihood-based Candidate Ranking

For a prompt p and a candidate continuation $c = (x_1, \ldots, x_T)$, we compute the average negative log-likelihood of the candidate conditioned on the prompt:

$$s(c \mid p) = -\frac{1}{T} \sum_{t=1}^{T} \log P_\theta(x_t \mid p, x_{<t}).$$

Only candidate tokens are scored; prompt tokens are masked out. The candidate with the lowest score is selected:

$$\hat{c} = \arg\min_{c \in \mathcal{C}} s(c \mid p).$$

This scoring rule is used for CUB-CounterFact, MyriadLAMA, PopQA, and BLiMP. For BLiMP, the prompt is empty, and the model compares the unconditional likelihood of the two minimal-pair sentences.

E.2 Acc@k Metrics

For CUB-CounterFact, MyriadLAMA, and PopQA, we additionally report top-k accuracy:

$$\mathrm{Acc@}k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\mathrm{Top}k_i \cap \mathcal{G}_i \neq \emptyset\right],$$

with $k \in \{1, 5, 10, 20\}$ whenever top-k candidate rankings are available. This is useful for factual probing because a model may assign high probability to the correct entity even when it is not ranked first. $\mathcal{G}_i$ is the set of acceptable gold answers. For datasets with multiple valid aliases, such as PopQA and MyriadLAMA, a prediction is correct if it exactly matches any normalized gold answer. Normalization lowercases the text, removes punctuation, and collapses whitespace.

E.3 Continuation Likelihood Metrics

For WikiText-2 and Alpaca, we evaluate the likelihood of the gold continuation rather than generating free-form outputs. Given a prompt p and gold continuation $y = (y_1, \ldots, y_T)$, we compute the total negative log-likelihood:

Hyperparameter	Value
Model family	SmolLM2
Model sizes	135M, 360M, 1.7B
Training corpus	FineWeb-Edu
Baseline input field	`text` (512)
FCT input fields	`factual triple`, `text` (2048)
Number of nodes	4
GPUs per node	4
Total GPUs	16
CPUs per task	32
Memory per node	240G
Training steps	87,500
Per-device batch size	16
Gradient accumulation steps	2
Effective global batch size	512 sequences
Learning rate	5×10^{-4}
Data loader workers	4
Logging interval	50 steps
Checkpoint interval	8,750 steps
Maximum saved checkpoints	10

Table 6: Training hyperparameters used for controlled continued pretraining. The vanilla pretraining baseline and FCT use the same initialization, corpus, training budget, and optimization settings, differing only in the input-output formulation.

$$\mathrm{NLL} = -\sum_{t=1}^{T} \log P_\theta(y_t \mid p, y_{<t}),$$

the token-normalized average NLL:

$$\mathrm{AvgNLL} = \frac{\mathrm{NLL}}{T},$$

and perplexity:

$$\mathrm{PPL} = \exp(\mathrm{AvgNLL}).$$

Because tokenizers may differ across models, we also report byte-normalized likelihood. Let B be the number of UTF-8 bytes in the gold continuation. We compute

$$\mathrm{ByteNLL} = \frac{\mathrm{NLL}}{B}, \qquad \mathrm{BPB} = \frac{\mathrm{ByteNLL}}{\log 2}.$$

Lower AvgNLL, PPL, ByteNLL, and BPB indicate better language modeling ability.

E.4 Training Hyperparameter

We use the same training configuration for the vanilla pretraining baseline and FCT to ensure a controlled comparison. The main experiment results are trained on about 20B fineweb-edu.

We run each experiment 3 times and report the average results. We train each model on 4 nodes, with 4 GPUs per node, for a total of 16 GPUs. We use GH200 with 144GB of GPU memory. For vanilla pretraining, we use around 2000 GPU-hours; for FCP (ours), we use around 4000 GPU-hours. The maximum sequence length is set to 4096 tokens. We train for 87,500 optimization steps with a per-device batch size of 16 and gradient accumulation of 2, giving an effective global batch size of 512 sequences per optimization step. We use a learning rate of 5×10^{-4}. The same hyperparameters are used for both the vanilla pretraining baseline and FCT; the only difference is the input-output formulation.