BEYOND REPRODUCTION: EDIT-CONDITIONED ADAPTATION FOR UNSTRUCTURED KNOWLEDGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

In unstructured knowledge editing, successfully reproducing a target document does not necessarily ensure that a model can use its individual facts to answer new queries. To narrow this reproduction–usability gap, we propose Edit-Conditioned Adaptation (ECoA), which trains lightweight adapters conditioned on the weight updates produced by an existing editor. ECoA freezes the edited base weights and trains low-rank adapters on continuation and cloze views constructed from the target document. It selects adapter layers using gradients of a document-derived training objective evaluated before and after editing, and modulates adapter rank-channel contributions using the responses of the editing updates to the current input activations. Evaluations across three language models and three unstructured knowledge editing benchmarks show consistent improvements in question answering over the corresponding base editors on UnKEBench and AKEW-CounterFact, with gains maintained in batch settings where multiple documents are edited into a single model. These results suggest that conditioning subsequent adaptation on existing editing updates provides an effective approach to improving query-conditioned use of edited knowledge.

1 INTRODUCTION

Knowledge stored in large language models (LLMs) requires updates after deployment as facts change, errors in pretraining data are discovered, and new information becomes available (Petroni et al., 2019; Hu et al., 2024; Zhang et al., 2025). *Knowledge editing* addresses this need by selectively modifying specific knowledge without retraining the entire model (De Cao et al., 2021; Mitchell et al., 2022; Meng et al., 2022; 2023). Early work primarily formulated editing requests as structured (subject, relation, object) triples. In practice, however, new knowledge often arrives as paragraphs or documents describing multiple entities, relations, events, and numerical details (Wu et al., 2024). This has motivated research on *unstructured knowledge editing*, which extends the editing target from individual structured facts to free-form text containing multiple facts (Deng et al., 2025; Jiang et al., 2025).

The goal of unstructured knowledge editing extends beyond reproducing a target document given the editing prompt. An edited model should also use relevant facts from the document to answer questions that differ from the original editing prompt (Fu et al., 2026; Zhong et al., 2023). We refer to this ability as *knowledge usability*. An unstructured document contains multiple facts, and questions about individual facts provide different contexts from those encountered when generating the full document. Thus, successful document reproduction under the original prompt does not guarantee accurate answers to fact-specific questions. For example, a model may reproduce an edited passage nearly verbatim yet revert to its prior knowledge or generate information that contradicts the passage when directly asked about an occupation or the details of an event described therein. We focus on this gap between *document reproduction* and *query-conditioned factual use*, aiming to improve knowledge usability after editing.

Unstructured knowledge editing methods such as UnKE (Deng et al., 2025) and AnyEdit (Jiang et al., 2025) optimize the generation of target text conditioned on an editing prompt.

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.

We evaluate document reproduction and fact-specific question answering in models edited with these methods. Figure 1(a) shows high document reproduction scores across all four editor–model combinations, yet substantially lower question-answering scores, with a ROUGE-L gap exceeding 0.45 in each. Figure 1(b) examines rescaling the editing update: increasing γ from 1 to 1.25 reduces question-answering ROUGE-L from 0.361 to 0.303, while document reproduction also declines. These observations show that successful document reproduction can coexist with limited query-conditioned use, and that amplifying the update alone may not resolve this limitation. We therefore focus on enabling individual facts in an edited document to be used across different query contexts, beyond reproducing the document itself.

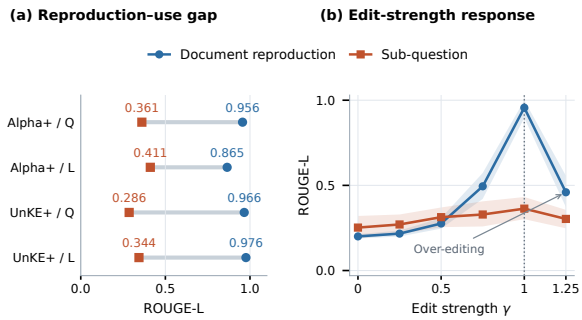

Figure 1: The reproduction-usability gap. (a) Document reproduction vs. fact-specific QA in ROUGE-L. Alpha+ denotes AlphaEdit+; Q/L denote Qwen/Llama. (b) Update scaling via $W_\gamma = W_0 + \gamma\Delta$ for AlphaEdit+ on Qwen2.5-7B-Instruct and UnKEBench. Shading indicates document-level bootstrap 95% confidence intervals.

To address this problem, we propose **Edit-Conditioned Adaptation (ECoA)**, which conditions post-edit adaptation on the weight updates produced by an existing editor. ECoA constructs continuation and cloze views from each sentence in the target document and trains lightweight adapters to reconstruct document content from diverse partial contexts. It selects adapter locations by comparing MLP gradients of the document-derived objective before and after editing. It further applies the editing weight updates to current input activations and projects the resulting responses into a low-dimensional conditioning signal that modulates the contribution of each adapter rank channel. By combining existing edit information with document-derived training, ECoA improves query-conditioned factual use without using evaluation queries during training.

We evaluate ECoA under single-document and batch editing settings on three unstructured knowledge editing benchmarks (UnKEBench (Deng et al., 2025), AKEW-CounterFact, and AKEW-MQuAKE (Wu et al., 2024)) using three models (Qwen2.5-7B-Instruct (Yang et al., 2025b), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-14B (Yang et al., 2025a)). Applied to two distinct base editors, ECoA consistently improves query-conditioned factual use across all three models on UnKEBench and AKEW-CounterFact. These gains persist as the number of jointly edited documents increases, reaching approximately $2.0\times$ the ROUGE-L score of the corresponding base editor on Qwen2.5-7B-Instruct.

Our contributions are as follows:

- We show that high document reproduction performance in unstructured knowledge editing does not guarantee query-conditioned factual use, and that simply amplifying editing updates does not close this gap.
- We propose Edit-Conditioned Adaptation (ECoA), which conditions post-edit adaptation on the weight updates produced by an existing editor. ECoA trains adapters using document-derived training views and edit-derived signals for layer selection and input-dependent modulation.
- We evaluate ECoA across three models and three benchmarks, demonstrating consistent improvements in query-conditioned factual use for two base editors on UnKEBench and AKEW-CounterFact. These gains also persist in batch editing.

2 Related Work

Locate-and-edit Knowledge Editing. Knowledge editing aims to update specific knowledge in a pretrained language model while preserving unrelated knowledge and behavior (Zhang et al., 2024). Locate-and-edit methods achieve this by identifying model components associated with target knowledge and modifying their weights. ROME (Meng et al., 2022) uses causal mediation analysis to identify a layer important for factual associations and applies a closed-form rank-one update to

its feed-forward weights. MEMIT (Meng et al., 2023) extends these updates across multiple layers to edit many facts simultaneously, while AlphaEdit (Fang et al., 2025) projects updates onto the null space of preserved knowledge representations to reduce interference with non-target knowledge. These methods primarily formulate editing requests as structured facts and evaluate editing success, generalization across paraphrases, and preservation of non-target knowledge.

Unstructured Knowledge Editing. UnKE (Deng et al., 2025) extends editing beyond structured facts to directly edit paragraph-level unstructured knowledge. AnyEdit (Jiang et al., 2025) decomposes knowledge in diverse formats into sequential chunks and autoregressively edits key tokens within each chunk, enabling edits of arbitrary length and format. COIN (Zhou et al., 2026) aligns predictions under local and global contexts to reduce the dependence of individual facts on surrounding context. While these methods focus on the editing process for incorporating unstructured knowledge into a model, our work focuses on adaptation after editing. ECoA uses the weight updates produced by an existing editor as conditioning signals to improve the use of individual facts from edited documents across diverse query contexts.

3 METHOD

We first formulate the task (Section 3.1), then introduce document-derived training views and edit-aware layer selection (Section 3.2). We next describe input-dependent adapter conditioning (Section 3.3) and the training objective (Section 3.4).

3.1 PROBLEM FORMULATION

Let f_θ be a pretrained language model and $d = (q_d, a_d)$ an unstructured editing request, where q_d is the editing prompt and a_d is the target document containing multiple facts. We denote the set of evaluation queries and reference answers for individual facts by $\mathcal{Q}_d = \{(q_d^{(j)}, y_d^{(j)})\}_{j=1}^{m_d}$, where m_d is the number of queries and $y_d^{(j)}$ is the reference answer to $q_d^{(j)}$.

We distinguish two aspects of post-edit performance. *Document reproduction* is the ability to reproduce a_d given the editing prompt q_d, whereas *query-conditioned factual use* is the ability to generate responses consistent with the reference answers $y_d^{(j)}$ to individual queries $q_d^{(j)}$. Our goal is to improve factual use in the edited model. $\mathcal{Q}_d$ is used exclusively for evaluation, while ECoA constructs its training views directly from the target document a_d.

3.2 EDIT-CONDITIONED ADAPTATION

Given an editing request d, an existing unstructured knowledge editor E yields:

$$\theta_d = E(\theta, d), \qquad \Delta_d = \theta_d - \theta, \tag{1}$$

where θ_d denotes the edited model parameters, and Δ_d is the corresponding weight update. ECoA attaches adapters to f_{θ_d} without modifying the internal editing procedure of E. During subsequent training, θ_d and Δ_d remain fixed, and only the adapter and conditioning projection parameters are optimized.

Because ECoA explicitly conditions on Δ_d, its current formulation requires access to editing-induced MLP weight updates and the corresponding input activations. It also requires eligible layers beyond the last edited layer for adapter placement. The AnyEdit-based configurations (Jiang et al., 2025) evaluated in our experiments satisfy these requirements.

Document-derived multi-view training. We split the target document a_d into sentences and construct two continuation views and one cloze view from each sentence s. Each view is an input–target sequence pair (x, y). Let $|s|$ denote the number of tokens in s, indexed from 0, and define the split position as $b_\rho(s) = \lfloor \rho |s| \rfloor$.

For $\rho \in \{0.4, 0.6\}$, a continuation view uses the sentence prefix to predict the remaining tokens:

$$v_{\text{cont}}^{\rho}(s) = \left(p_{\text{cont}} \oplus s_{<b_\rho(s)},\ s_{\geq b_\rho(s)}\right). \tag{2}$$

A cloze view masks the middle span and uses the masked tokens as the target. Writing $b_1 = b_{0.4}(s)$ and $b_2 = b_{0.6}(s)$, we define:

$$v_{\text{cloze}}(s) = \Big(p_{\text{cloze}} \oplus s_{<b_1} \oplus [\text{MISSING}] \oplus s_{\geq b_2},\ s_{[b_1,b_2)}\Big), \tag{3}$$

where $\oplus$ denotes sequence concatenation. The instructions p_{cont} and p_{cloze} are *"Continue this learned factual statement."* and *"Restore the missing span."*, respectively. We denote the collection of views from all sentences by $\mathcal{V}_d$.

Continuation uses the sentence prefix, while cloze uses context on both sides of the missing span. Together, these complementary views train the model to recover document facts from different partial contexts rather than relying solely on the original editing context, without using evaluation queries.

Edit-aware layer selection. ECoA selects adapter locations by comparing MLP gradients before and after editing. We define the average log-likelihood over the training views as:

$$\mathcal{J}_{\text{view}}(\vartheta; d) = \frac{1}{|\mathcal{V}_d|} \sum_{(x,y)\in\mathcal{V}_d} \log p_\vartheta(y \mid x). \tag{4}$$

Let $\vartheta_\ell^{\text{MLP}}$ denote the MLP parameters at layer ℓ. We compute the gradients for the edited and original models:

$$g_\ell(d) = \nabla_{\vartheta_\ell^{\text{MLP}}} \mathcal{J}_{\text{view}}(\vartheta; d)\Big|_{\vartheta=\theta_d}, \qquad \bar{g}_\ell(d) = \nabla_{\vartheta_\ell^{\text{MLP}}} \mathcal{J}_{\text{view}}(\vartheta; d)\Big|_{\vartheta=\theta}. \tag{5}$$

We then assign each layer a score:

$$I_\ell(d) = \langle g_\ell(d) - \bar{g}_\ell(d), g_\ell(d)\rangle, \tag{6}$$

where the inner product is computed over vectorized MLP gradients. The gradient $g_\ell(d)$ gives a local direction for increasing view log-likelihood in the edited model. The score measures the alignment of the editing-induced gradient change with this direction, weighted by their magnitudes.

Let $\mathcal{L}_{\text{edit}}$ denote the set of layers modified by the base editor. Since the conditioning signal described below uses responses from all edited layers, candidate adapter locations are restricted to:

$$\mathcal{C} = \{\ell : \ell > \max(\mathcal{L}_{\text{edit}})\}. \tag{7}$$

We select the top k layers in $\mathcal{C}$ by $I_\ell(d)$, with $k = 5$ by default.

For joint editing of a batch $\mathcal{B}$ containing N documents, we use the jointly edited parameters θ_B and update $\Delta_B = \theta_B - \theta$. Each document's score is computed by replacing θ_d with θ_B in Eq. (5). We select a common set of top k layers based on the mean score across documents:

$$\bar{I}_\ell = \frac{1}{N} \sum_{d\in\mathcal{B}} I_\ell(d). \tag{8}$$

3.3 Input-Dependent Adapter Conditioning

ECoA constructs a conditioning signal from the editing weight updates and current input activations. This signal modulates adapters at the selected layers, linking post-edit adaptation to the updates applied to the model.

Edit-response code. For each edited layer $e \in \mathcal{L}_{\text{edit}}$, let $\Delta_d^{(e)}$ denote the MLP weight update used for conditioning and $h_e(x)$ the activation entering the corresponding weight matrix. All operations below are performed token-wise, with token indices omitted for brevity. We first compute the normalized response of the weight update to the current activation:

$$u_e(x) = \text{LayerNorm}\left(\Delta_d^{(e)} h_e(x)\right). \tag{9}$$

We map each response to the adapter rank dimension r through a learnable linear projection P_e and average across edited layers to obtain the edit-response code:

$$c(x) = \frac{1}{|\mathcal{L}_{\text{edit}}|} \sum_{e\in\mathcal{L}_{\text{edit}}} P_e u_e(x) \in \mathbb{R}^r. \tag{10}$$

Although Δ_d remains fixed, $c(x)$ varies across inputs and across token positions within each input through token-specific activations.

Rank-channel modulation. We attach low-rank adapters (Hu et al., 2022) to the MLP projections $s \in \{\text{gate}, \text{up}, \text{down}\}$ at each selected layer ℓ. For input and output dimensions $d^{\text{in}}_{\ell,s}$ and $d^{\text{out}}_{\ell,s}$, the adapter parameters are:

$$A_{\ell,s} \in \mathbb{R}^{r \times d^{\text{in}}_{\ell,s}}, \qquad B_{\ell,s} \in \mathbb{R}^{d^{\text{out}}_{\ell,s} \times r}. \tag{11}$$

We transform the edit-response code into a rank-wise gate and apply it to the adapter's low-dimensional activations:

$$g(x) = \mathbf{1} + \tanh(c(x)), \qquad \text{Adapter}_{\ell,s}(x_{\ell,s}; x) \quad = \eta B_{\ell,s} \left[(A_{\ell,s} x_{\ell,s}) \odot g(x) \right], \tag{12}$$

where $x_{\ell,s}$ is the input activation of the corresponding projection, $\odot$ denotes element-wise multiplication, and η is the LoRA scaling coefficient. The adapter output is added to the base projection output. Since $g(x) \in (0, 2)^r$, the gate amplifies or attenuates each rank channel relative to its ungated contribution. When $c(x) = 0$, $g(x) = \mathbf{1}$, recovering the standard LoRA form. The gate conditions adapter contributions on input-dependent edit responses rather than detecting editing success.

For batch editing, both training and inference use the joint update Δ_B in place of Δ_d, without selecting separate document-specific updates for individual inputs. Although all inputs share Δ_B, the code and gate vary with the activations $h_e(x)$. Appendix C.6 compares this joint conditioning scheme with conditioning on document-specific editing updates.

3.4 Training Objective

ECoA freezes the edited base weights θ_d and editing updates Δ_d, optimizing only the parameters ϕ of the low-rank adapters and conditioning projections. The training objective is:

$$\mathcal{L}(\phi) = \mathcal{L}_{\text{CE}} + \lambda_d \mathcal{L}_{\text{KD}}, \tag{13}$$

where $\mathcal{L}_{\text{CE}}$ is the token-level cross-entropy over the target span y of each view $(x, y) \in \mathcal{V}_d$. Each target token y_t is predicted conditioned on the input x and preceding target tokens $y_{<t}$. Input tokens are excluded from the loss, and λ_d controls the strength of distillation.

To limit deviations from the edited model's predictive distribution, we use f_{θ_d} with adapters disabled as the teacher. Distillation is computed only at target token positions:

$$\mathcal{L}_{\text{KD}} = \mathbb{E}_{(x,y),\, t} \left[D_{\text{KL}} \left(p_{\theta_d}(\cdot \mid x, y_{<t}) \,\|\, p_{\theta_d,\phi}(\cdot \mid x, y_{<t}) \right) \right]. \tag{14}$$

The expectation is taken over training views and token positions within their target spans. This term constrains deviations from the edited model while the adapters learn to reconstruct document content from partial contexts.

For batch editing, we replace θ_d and Δ_d with θ_B and Δ_B, respectively, and train a shared set of adapters and conditioning projections on views constructed from all documents in the batch.

4 Experiment

4.1 Experimental Setup

Models and Datasets. We evaluate unstructured knowledge editing using Qwen2.5-7B-Instruct (Yang et al., 2025b), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-14B (Yang et al., 2025a). We use three benchmarks: UnKEBench (Deng et al., 2025), AKEW-CounterFact, and AKEW-MQuAKE (Wu et al., 2024), with 500 documents each for UnKEBench and AKEW-CounterFact and 277 for AKEW-MQuAKE. We compare generated answers to fact-specific questions associated with each edited document against the provided reference answers. Unless otherwise specified, all subsequent analyses use Qwen2.5-7B-Instruct on UnKEBench. Further dataset details are provided in Appendix B.1.

Edit Setups and Baselines. In single-document editing, each document is edited and evaluated independently, whereas batch editing jointly incorporates multiple documents into a single model. Our baselines include the pre-edited model, FT-L (Zhu et al., 2020), LoRA (Hu et al., 2022), MEMIT (Meng et al., 2023), AlphaEdit (Fang et al., 2025), UnKE (Deng et al., 2025), COIN (Zhou et al., 2026), and UnKE and AlphaEdit combined with AnyEdit (Jiang et al., 2025). We denote the latter two configurations as UnKE+ and AlphaEdit+, respectively, and their ECoA-augmented versions as UnKE+ w/ ECoA and AlphaEdit+ w/ ECoA. Further baseline descriptions and implementation details are provided in Appendix B.2 and Appendix B.3.

Table 1: Single-document editing results across three models and three datasets. Each document is edited and evaluated independently. Bold indicates the best score for each model-dataset-metric combination. Parenthesized values below ECoA scores indicate absolute score changes relative to the corresponding base editor.

Model	Method	UnKEBench			AKEW-CounterFact			AKEW-MQuAKE		
		R-L	BERT	Judge	R-L	BERT	Judge	R-L	BERT	Judge
Qwen2.5-7B	Pre-edited	0.194	0.238	0.151	0.169	0.197	0.208	0.393	0.306	0.494
	FT-L	0.175	0.220	0.108	0.247	0.249	0.197	0.226	0.257	0.192
	LoRA	0.139	0.190	0.032	0.240	0.218	0.024	0.158	0.191	0.017
	MEMIT	0.276	0.256	0.274	0.246	0.210	0.257	0.484	0.324	0.526
	AlphaEdit	0.278	0.254	0.294	0.264	0.216	0.280	0.463	0.322	0.520
	UnKE	0.289	0.263	0.286	0.291	0.228	0.325	0.500	0.331	0.542
	COIN	0.390	0.317	0.486	0.349	0.256	0.384	0.594	0.367	**0.616**
	UnKE+	0.286	0.260	0.272	0.307	0.226	0.339	0.569	0.335	0.587
	AlphaEdit+	0.361	0.273	0.400	0.448	0.268	0.475	0.563	0.334	0.575
	UnKE+ **w/ ECoA**	0.431 (+0.145)	0.321 (+0.061)	0.523 (+0.251)	**0.496** (+0.189)	0.290 (+0.064)	**0.516** (+0.177)	**0.607** (+0.038)	**0.375** (+0.040)	0.609 (+0.022)
	AlphaEdit+ **w/ ECoA**	**0.451** (+0.090)	**0.325** (+0.052)	**0.554** (+0.154)	0.467 (+0.019)	**0.295** (+0.027)	0.512 (+0.037)	0.569 (+0.006)	0.351 (+0.017)	0.586 (+0.011)
Llama-3.1-8B	Pre-edited	0.094	0.215	0.070	0.129	0.195	0.168	0.388	0.311	0.502
	FT-L	0.176	0.270	0.296	0.206	0.314	0.394	0.342	0.375	0.467
	LoRA	0.045	0.090	0.149	0.063	0.104	0.074	0.048	0.092	0.087
	MEMIT	0.180	0.241	0.187	0.217	0.226	0.245	0.498	0.328	0.585
	AlphaEdit	0.145	0.234	0.141	0.197	0.223	0.224	0.471	0.323	0.551
	UnKE	0.341	0.269	0.413	0.384	0.261	0.452	0.468	0.331	0.552
	COIN	0.201	0.254	0.175	0.327	0.285	0.354	0.430	0.352	0.465
	UnKE+	0.344	0.273	0.397	0.427	0.264	0.487	0.495	0.307	0.549
	AlphaEdit+	0.411	0.279	0.495	0.492	0.283	0.541	0.486	0.312	0.551
	UnKE+ **w/ ECoA**	0.492 (+0.148)	**0.334** (+0.061)	0.676 (+0.279)	**0.567** (+0.140)	**0.318** (+0.054)	0.614 (+0.127)	**0.609** (+0.114)	**0.399** (+0.092)	0.628 (+0.079)
	AlphaEdit+ **w/ ECoA**	**0.508** (+0.097)	0.321 (+0.042)	**0.681** (+0.186)	0.566 (+0.074)	0.317 (+0.034)	**0.632** (+0.091)	0.604 (+0.118)	0.360 (+0.048)	**0.631** (+0.080)
Qwen3-14B	Pre-edited	0.180	0.221	0.079	0.159	0.206	0.122	0.260	0.303	0.449
	FT-L	0.279	0.285	0.188	0.269	0.240	0.201	0.356	0.300	0.355
	LoRA	0.067	0.122	0.040	0.106	0.146	0.031	0.091	0.140	0.029
	MEMIT	0.287	0.260	0.212	0.301	0.237	0.266	0.443	0.314	0.457
	AlphaEdit	0.284	0.256	0.218	0.290	0.233	0.247	0.415	0.308	0.421
	UnKE	0.296	0.257	0.227	0.323	0.247	0.303	0.476	0.327	0.484
	COIN	0.265	0.292	0.302	0.295	0.282	0.348	0.364	**0.355**	**0.553**
	UnKE+	0.355	0.270	0.279	0.397	0.258	0.359	0.537	0.354	0.546
	AlphaEdit+	0.354	0.270	0.272	0.393	0.258	0.361	0.519	0.353	0.518
	UnKE+ **w/ ECoA**	**0.397** (+0.042)	**0.312** (+0.042)	**0.411** (+0.132)	**0.434** (+0.037)	**0.300** (+0.042)	**0.409** (+0.050)	0.544 (+0.007)	0.331 (-0.023)	0.530 (-0.016)
	AlphaEdit+ **w/ ECoA**	0.394 (+0.040)	0.311 (+0.041)	0.398 (+0.126)	0.405 (+0.012)	0.290 (+0.032)	0.373 (+0.012)	**0.548** (+0.029)	0.333 (-0.020)	0.528 (+0.010)

Evaluation Metrics. We report ROUGE-L (R-L) (Lin, 2004) for lexical similarity, BERTScore (BERT) (Zhang et al., 2020) for semantic similarity,[1] and LLM-as-a-judge (Judge) (Zheng et al., 2023) for factual correctness. The judge assesses whether the generated response contains the key facts in the reference answer without contradicting them, returning a binary yes/no judgment. The evaluation prompt is provided in Appendix B.4, and additional ROUGE-1/2 results are reported in Appendix C.3.

4.2 Main Results

Table 1 compares single-document editing performance across three models and three datasets. ECoA consistently improves ROUGE-L, BERTScore, and Judge for both base editors across all

[1]We follow the AnyEdit (Jiang et al., 2025) configuration using `all-MiniLM-L6-v2`.

three models on UnKEBench and AKEW-CounterFact. On UnKEBench with Qwen2.5-7B, for example, applying ECoA to UnKE+ increases the Judge score from 0.272 to 0.523. Both ECoA configurations also outperform COIN across all three metrics in this setting. The improvements across lexical similarity, semantic similarity, and judged factual correctness indicate that the benefits are not restricted to lexical overlap. Their consistency across models and base editors supports the effectiveness of post-edit adaptation for improving query-conditioned factual use.

Among existing baselines, LoRA underperforms the pre-edited model in most settings, while FT-L yields improvements in some settings but generally trails dedicated editing methods. UnKE, COIN, and editors combined with AnyEdit achieve comparatively stronger performance. These comparisons suggest that directly fine-tuning on target documents is insufficient to achieve strong query-conditioned factual use in the evaluated settings. ECoA provides further gains on top of UnKE+ and AlphaEdit+, showing that factual use can be improved through subsequent adaptation even after the document has been incorporated by an existing editor.

On AKEW-MQuAKE, the gains are less consistent across models and metrics. ECoA improves all three metrics for both base editors on Qwen2.5-7B and Llama-3.1-8B, with larger ROUGE-L gains on the latter. On Qwen3-14B, however, ROUGE-L improves while BERTScore declines for both editors, and Judge also decreases for UnKE+. This divergence shows that improvements in lexical overlap do not always coincide with improvements in the other evaluation metrics. Thus, ECoA delivers consistent gains on UnKEBench and AKEW-CounterFact, while its effectiveness on AKEW-MQuAKE depends on the model and metric.

4.3 Batch Editing and Scalability

Figure 2 shows ROUGE-L on Qwen2.5-7B as the number of jointly edited documents increases to 500. ECoA maintains gains over both base editors even at this scale. With 500 documents, UnKE+ w/ ECoA achieves 0.376 on UnKEBench, approximately twice the score of UnKE+ (0.187). Both ECoA configurations also outperform their corresponding base editors and COIN on both datasets at this scale. These results show that the benefits of post-edit adaptation extend beyond independently edited documents to settings where hundreds of documents share a single edited model.

Performance trends nevertheless differ across methods as the editing scale increases. LoRA, FT-L, AlphaEdit, and MEMIT degrade substantially, whereas UnKE, COIN, and editors combined with AnyEdit retain comparatively higher performance. The two ECoA configurations also exhibit distinct trends. On UnKEBench, AlphaEdit+ w/ ECoA performs best at the smallest batch size but declines as more documents are edited. UnKE+ w/ ECoA remains comparatively stable and performs better at the largest scale. Thus, ECoA provides gains over both base editors, but the underlying editor remains important for performance as the number of jointly edited documents grows.

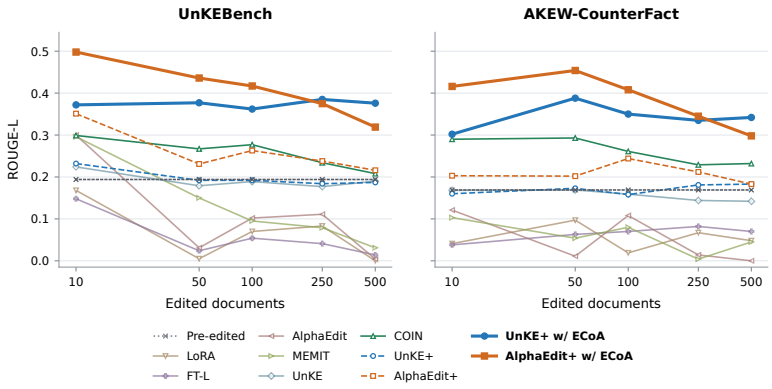

Figure 2: Batch editing performance on UnKEBench and AKEW-CounterFact as the number of documents edited into a single model increases to 500. Thick blue and orange solid lines represent UnKE+ and AlphaEdit+ with ECoA; dashed lines in the corresponding colors represent the base editors.

4.4 Query Usability Across Document Positions

Figure 3 compares query-conditioned factual use across relative answer positions within documents. Both ECoA configurations outperform their respective base editors in all five position bins, showing that the gains are distributed across the document rather than concentrated near its beginning. The advantage is particularly clear in the later bins: while COIN's performance declines toward the end of the document, both ECoA configurations maintain ROUGE-L above 0.4 and outperform all baselines. The fitted regression trends provide a complementary view of position-dependent variation. Both ECoA configurations exhibit flatter trends than COIN and their corresponding base editors, with UnKE+ w/ ECoA showing a nearly flat trend. Together, the higher scores across position bins and flatter trends indicate improved factual use throughout the document, including facts appearing near its end.

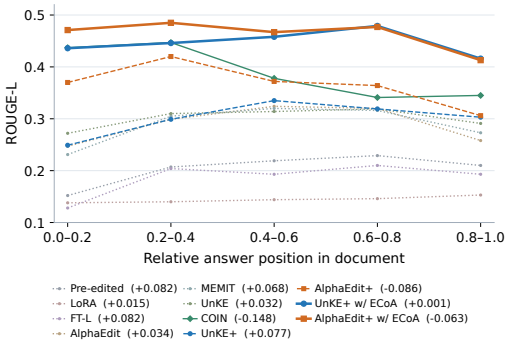

Figure 3: Query-conditioned factual use across document positions. ROUGE-L scores are grouped into five bins according to the normalized position of the reference answer in the edited document. Values in parentheses in the legend indicate regression slopes with respect to normalized answer position.

4.5 Edit-Response Signal Analysis

Table 2 evaluates whether the trained adapters use conditioning signals corresponding to the current edit. Keeping the trained model and adapters fixed, we compare three settings: Correct uses the original signal, Shuffled shuffles signals across documents, and Zero removes the signal. Correct achieves the highest ROUGE-L across both datasets and both base editors. Replacing the correct signal with another document's edit signal reduces performance to a level close to Zero, suggesting that the trained adapters benefit from signals corresponding to the current edit. Appendix C.6 further shows that document-specific updates yield higher performance than the shared batch update. Together, these results support the importance of edit-signal correspondence. Section 4.6 complements this analysis by retraining adapters without conditioning to assess its contribution beyond document-derived training alone.

		ROUGE-L			BERT	
Dataset	Base editor	Correct	Shuffled	Zero	Correct	Zero
UnKEBench	UnKE+	**0.431**	0.355	0.342	**0.321**	0.292
	AlphaEdit+	**0.451**	0.403	0.389	**0.325**	0.307
AKEW-CounterFact	UnKE+	**0.496**	0.397	0.398	**0.290**	0.268
	AlphaEdit+	**0.467**	0.387	0.379	**0.295**	0.251

Table 2: Effect of the conditioning signal at evaluation time.

4.6 Component Analysis

Table 3 analyzes the contributions of combining editing with post-edit adaptation, edit conditioning, and adaptive layer selection on UnKEBench. Full ECoA outperforms all compared variants for both base editors. Adapter only trains adapters on the pre-edited model while using the layer selections and conditioning signals obtained from the corresponding editor. Editing only provides the same document-derived views to the base editor without training additional adapters. Both variants underperform Full ECoA, demonstrating that combining editing with post-edit adaptation is more effective than either alone in the evaluated settings. Removing edit conditioning and retraining the adapters lowers ROUGE-L by 0.059 for UnKE+ and 0.036 for AlphaEdit+. Since these variants use the same training views, the comparison supports the contribution of edit-derived conditioning beyond document-derived training alone. Random layer selection also underperforms Full ECoA, supporting the benefit of gradient-based layer selection. Appendix C.4 examines sensitivity to the number of selected layers, training steps, and distillation weight. Additional analyses of adapter target modules, conditioning variants, and training view composition are provided in Appendix C.5.

Table 3: Component ablations. Values in parentheses indicate absolute score changes relative to Full ECoA.

	UnKE+ w/ ECoA		AlphaEdit+ w/ ECoA	
Variant	ROUGE-L	BERT	ROUGE-L	BERT
Full ECoA	**0.431**	**0.321**	**0.451**	**0.325**
Adapter only	0.305 (−0.126)	0.278 (−0.043)	0.290 (−0.161)	0.268 (−0.057)
Editing only	0.365 (−0.066)	0.286 (−0.035)	0.418 (−0.033)	0.267 (−0.058)
w/o edit conditioning	0.372 (−0.059)	0.296 (−0.025)	0.415 (−0.036)	0.308 (−0.017)
Random layer selection	0.388 (−0.043)	0.317 (−0.004)	0.415 (−0.036)	0.313 (−0.012)

4.7 General Capability Preservation

Knowledge editing can affect a model's general capabilities beyond the targeted factual updates (Gu et al., 2024). We evaluate post-edit language understanding on SST-2, MRPC, CoLA, and RTE from GLUE (Wang et al., 2018). As shown in Figure 4, UnKE+ w/ ECoA achieves a mean accuracy only 0.005 below the pre-edited model and remains close to UnKE+ and COIN. It therefore improves query-conditioned factual use while largely maintaining performance on these tasks. For AlphaEdit+ w/ ECoA, mean accuracy decreases by 0.024 relative to its base editor, indicating that preservation varies across editors. Low baseline scores should also be interpreted in light of generation behavior: LoRA exhibits repetitive outputs (Appendix C.7), while AlphaEdit frequently produces Chinese explanations that fail English-label-based scoring.

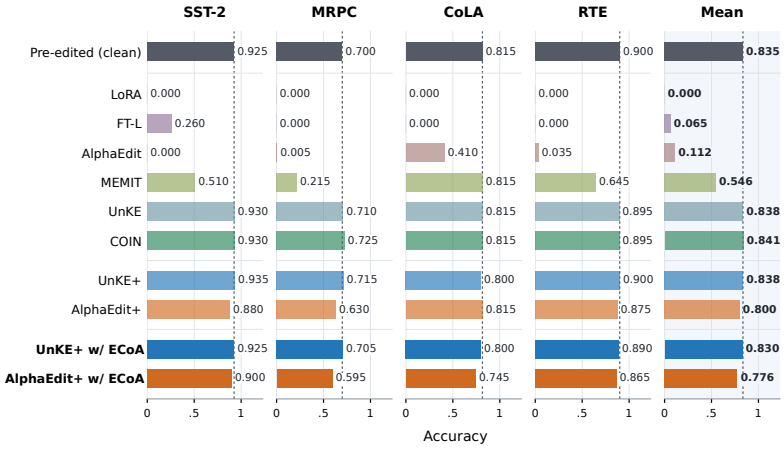

Figure 4: Post-editing accuracy on four GLUE tasks after batch editing. Mean denotes the unweighted average accuracy across SST-2, MRPC, CoLA, and RTE. Values are shown at the ends of the bars, and all panels share the same accuracy scale.

5 Conclusion

This work addresses the gap between successful document reproduction and query-conditioned factual use in unstructured knowledge editing. We propose Edit-Conditioned Adaptation (ECoA), which conditions post-edit adaptation on existing editing updates. ECoA combines document-derived multi-view training with edit-aware layer selection and input-dependent adapter modulation to improve the use of edited knowledge without training on evaluation queries. Experiments across three models and three benchmarks show consistent gains for both base editors on UnKEBench and AKEW-CounterFact. These gains extend to batch editing with 500 documents and facts at different document positions, while the UnKE+ configuration also maintains GLUE performance close to that of the pre-edited model. Our findings suggest that editing updates can serve not only to incorporate knowledge but also as useful signals for guiding subsequent adaptation. More broadly, they support extending the goal of unstructured knowledge editing beyond document reproduction to factual use across diverse query contexts.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and language editing. We also used GPT-4o-mini (OpenAI, 2024) as an automated evaluator of generated answers. We take responsibility for the final manuscript, including the accuracy of its claims, reported results, and AI-assisted content.

ETHICS STATEMENT

This work studies unstructured knowledge editing using existing benchmarks. The counterfactual statements in these benchmarks serve as controlled editing targets and should not be interpreted as verified real-world facts. Although knowledge editing can support the correction and updating of model knowledge, it may also be misused to introduce false or misleading information. Our evaluation focuses on the use of edited facts and the preservation of selected language understanding capabilities; it does not establish the factual reliability or safety of edited models in deployment. Practical applications therefore require verification of editing targets and assessment of unintended effects on model behavior.

REPRODUCIBILITY STATEMENT

We describe the ECoA framework and training objective in Section 3. The appendix provides additional details on the datasets, baseline implementations, hyperparameters, and evaluation procedures. We will publicly release our implementation and experiment configurations to facilitate reproduction of our results.

REFERENCES

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.522. URL https://aclanthology.org/2021.emnlp-main.522/.

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. Everything is editable: Extend knowledge editing to unstructured data in large language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 628–652, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/02763667a5761ff92bb15d8751bcd223-Paper-Conference.pdf.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations*, volume 2025, pp. 16366–16396, 2025.

Jinhu Fu, Yan Bai, Longzhu He, Yihang Lou, Yanxiao Zhao, Li Sun, and Sen Su. Learning to edit knowledge via instruction-based chain-of-thought prompting. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24694–24711, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1133. URL https://aclanthology.org/2026.acl-long.1133/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the*

2024 Conference on Empirical Methods in Natural Language Processing, pp. 16801–16819, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.934. URL https://aclanthology.org/2024.emnlp-main.934/.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Xuming Hu, Junzhe Chen, Xiaochuan Li, Yufei Guo, Lijie Wen, Philip Yu, and Zhijiang Guo. Towards understanding factual knowledge of large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 28680–28715, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7b7d7985f62284060d65f532ed2ea5fa-Paper-Conference.pdf.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Anyedit: Edit any knowledge encoded in language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=aJIoBur0Ef.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

OpenAI. Chatgpt: Optimizing language models for dialogue. https://openai.com/index/chatgpt/, 2022.

OpenAI. GPT-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, July 2024. Released July 18, 2024.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250/.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. EasyEdit: An easy-to-use knowledge editing framework for large language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 82–93, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.9. URL https://aclanthology.org/2024.acl-demos.9/.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. AKEW: Assessing knowledge editing in the wild. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15118–15133, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.843. URL https://aclanthology.org/2024.emnlp-main.843/.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, volume 2024, pp. 35623–35646, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report, 2025b. URL https://arxiv.org/abs/2412.15115.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models, 2024. URL https://arxiv.org/abs/2401.01286.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, 51(4):1373–1418, 12 2025. ISSN 0891-2017. doi: 10.1162/COLI.a.16. URL https://doi.org/10.1162/COLI.a.16.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15686–15702, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.971. URL https://aclanthology.org/2023.emnlp-main.971/.

Zisheng Zhou, Mengqi Zhang, Shiguang Wu, Xiaotian Ye, Chi Zhang, Zhumin Chen, and Pengjie Ren. Uncovering context reliance in unstructured knowledge editing, 2026. URL https://arxiv.org/abs/2602.19043.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models, 2020. URL https://arxiv.org/abs/2012.00363.

A LIMITATIONS

While ECoA improves query-conditioned factual use through post-edit adaptation, it has several limitations. ECoA requires access to MLP weight updates and the corresponding input activations, as well as candidate adapter layers after all edited layers. Its current formulation is therefore not directly applicable to external-memory editors that do not provide such updates or settings where model internals are inaccessible. We focus our evaluation on UnKE+ and AlphaEdit+ because they build on widely used locate-and-edit methods for knowledge editing. Whether ECoA provides additional gains when combined with other editors remains untested. ECoA also introduces computational overhead beyond the base editing procedure, including gradient computation before and after editing and adapter training. Appendix C.1 reports per-document runtime and peak GPU memory usage.

B EXPERIMENTAL SETUP

B.1 DATASETS

We use three unstructured knowledge editing benchmarks with distinct characteristics: UnKEBench (Deng et al., 2025), AKEW-CounterFact, and AKEW-MQuAKE (Wu et al., 2024). Unlike conventional settings that directly edit individual $(subject, relation, object)$ triples, all three benchmarks use documents containing multiple facts and natural language context as editing targets. Our primary evaluation focuses on whether individual facts in edited documents can be used to answer separate queries, rather than whether the documents can be reproduced given the editing prompts. We therefore evaluate query-conditioned factual use by prompting the model with the provided subquestions and comparing its free-form responses against the corresponding reference answers.

UnKEBench. UnKEBench is designed for unstructured knowledge editing, using free-form passages containing multiple facts and contextual information as editing targets. These passages include entities and relations beyond the target fact, making them difficult to represent as a single key–value pair. The benchmark constructs unstructured texts from counterfactual knowledge collected from ConflictQA (Xie et al., 2024). This counterfactual setting helps distinguish newly edited knowledge from facts the model may have acquired during pretraining. The benchmark also provides questions and paraphrased questions derived from the documents to evaluate whether edited knowledge can be retrieved across different input formulations.

AKEW-CounterFact. AKEW (Assessing Knowledge Editing in the Wild) addresses the reliance of conventional editing evaluations on curated structured triples. It evaluates the same knowledge updates in three forms: structured facts, unstructured texts, and triples extracted from those texts. AKEW-CounterFact converts counterfactual updates from CounterFact (Meng et al., 2022) into unstructured editing targets. CounterFact constructs these updates by replacing the object in a factual triple (s, r, o) with an alternative value. AKEW uses ChatGPT (OpenAI, 2022) to generate Wikipedia-style paragraphs incorporating the updates, together with additional entities and contextual details. This setting allows us to examine the reproduction–usability gap: reproducing a paragraph does not necessarily imply that its target fact can be retrieved through an independent question.

AKEW-MQuAKE. AKEW-MQuAKE converts counterfactual knowledge updates from MQuAKE-CF (Zhong et al., 2023) into AKEW's unstructured format. The original MQuAKE benchmark evaluates whether factual edits propagate to answers requiring multi-hop reasoning. For example, changing a country's prime minister should also change answers to questions involving attributes of that prime minister. AKEW transforms these structured updates into Wikipedia-style paragraphs containing the edited facts. In this work, we evaluate the provided fact-specific subquestions, focusing on whether facts embedded in these paragraphs can be used in separate query contexts.

GLUE. We use four tasks from GLUE (Wang et al., 2018) to evaluate general language understanding after knowledge editing: SST-2 for sentiment classification, MRPC for paraphrase identification, CoLA for linguistic acceptability, and RTE for textual entailment. These tasks complement our evaluation of edited knowledge by assessing performance beyond the targeted factual updates. We randomly sample 200 examples from each task's validation split and evaluate all methods on

the same examples. Task-specific classification prompts instruct the model to output a single-word label. We report accuracy for each task and the unweighted mean across the four tasks, comparing the pre-edited model with models after batch editing.

B.2 BASELINES

We compare ECoA with a broad range of baselines, including parameter-efficient fine-tuning, structured and unstructured knowledge editing methods, and a generalized editing framework.

Pre-edited. We use the original language model without editing as a reference. Its responses reflect the model's knowledge and behavior before any editing updates are applied.

FT-L (Zhu et al., 2020). FT-L is a gradient-based editing baseline that directly fine-tunes selected model layers on the target knowledge. It modifies existing model parameters and serves to assess how effectively direct optimization on the target document supports editing and subsequent query-conditioned factual use.

LoRA (Hu et al., 2022). LoRA is a parameter-efficient fine-tuning method that adapts the model to target data by adding low-rank updates to weight matrices while keeping the original parameters fixed.

MEMIT (Meng et al., 2023). MEMIT treats transformer feed-forward modules as associative memories and computes weight updates across multiple layers to incorporate new factual associations. Originally designed for structured factual editing and large-scale batch editing, it provides a comparison for assessing the applicability of structured editors to paragraph-level unstructured knowledge.

AlphaEdit (Fang et al., 2025). AlphaEdit is a locate-and-edit method that projects editing updates onto the null space of preserved knowledge representations. This projection aims to reduce interference with previously stored knowledge when incorporating new facts.

UnKE (Deng et al., 2025). UnKE extends knowledge editing from structured triples to paragraph-level unstructured knowledge. It uses non-local representations spanning multiple blocks rather than relying on local key-value storage within a single layer. Its cause-driven optimization directly optimizes the target sequence while retaining its context, instead of requiring an explicitly identified entity term as the editing target.

COIN (Zhou et al., 2026). COIN addresses context reliance in unstructured editing based on next-token prediction, where newly learned facts depend excessively on preceding document context. Under document-level training, facts appearing later in a document are repeatedly observed alongside earlier context, which can hinder independent retrieval when that context is absent at inference time. COIN aligns predictions under local and global contexts to reduce this dependence and support factual retrieval without the original surrounding context.

AnyEdit (Jiang et al., 2025). AnyEdit is a generalized editing framework that extends locate-and-edit methods to knowledge of diverse lengths and formats. It decomposes target knowledge into sequential chunks and edits key target tokens within each chunk autoregressively, enabling updates beyond individual triples. By retaining the underlying editor's weight-update rule, AnyEdit can be combined with editors such as ROME, MEMIT, AlphaEdit, and UnKE. In our experiments, we combine AnyEdit with UnKE and AlphaEdit, denoting these configurations as UnKE+ and AlphaEdit+.

B.3 IMPLEMENTATION DETAILS

All experiments use the Hugging Face checkpoints `Qwen/Qwen2.5-7B-Instruct`, `meta-llama/Llama-3.1-8B-Instruct`, and `Qwen/Qwen3-14B`, with each experiment run on a single NVIDIA A100 80GB GPU. ECoA and the editing configurations incorporating AnyEdit are implemented using the public AnyEdit codebase (Jiang et al., 2025), while the remaining baselines use the public EasyEdit codebase (Wang et al., 2024).

Table 4: Editing cost on Qwen2.5-7B. Runtime is reported per document. For ECoA configurations, runtime includes base editing and subsequent adaptation, while memory values marked with † measure only the adapter training stage and exclude base editing.

Method	Time (s/doc)	Peak GPU memory (GiB)
FT-L	7.2	30.7
LoRA	27.4	31.4
MEMIT	19.9	38.3
AlphaEdit	20.9	39.6
UnKE	33.8	40.8
COIN	31.6	32.2
UnKE+	53.6	40.1
UnKE+ w/ ECoA	75.5	31.9†
AlphaEdit+	35.8	39.3
AlphaEdit+ w/ ECoA	57.9	31.9†

By default, adapters are attached to the MLP projections `gate_proj`, `up_proj`, and `down_proj` in the top $k = 5$ layers chosen by edit-aware layer selection. We set the LoRA rank to 4, alpha to 8, and dropout to 0. We train with AdamW (Loshchilov & Hutter, 2019) using a learning rate of 4×10^{-4}, weight decay of 0, and a gradient clipping threshold of 1.0. Training runs for 30 steps with three views per step, and the distillation weight λ_{d} is set to 0.5. During evaluation, we use greedy decoding with a maximum of 96 new tokens. We use `gpt-4o-mini` (OpenAI, 2024) as the LLM judge, with the evaluation prompt provided in the following section.

B.4 LLM-AS-A-JUDGE PROMPT

The prompt used for LLM-as-a-judge is shown below. It instructs the judge to return "yes" when the key facts in the generated response are semantically consistent with the reference answer and do not contradict it. This criterion avoids penalizing additional, non-contradictory context that naturally occurs in free-form responses.

```
Question: {question}
Reference (correct) answer: {gold}
Model's response: {pred}

The model's response may contain extra elaboration beyond
the core answer. Judge ONLY whether the core fact asked in
the question is stated correctly somewhere in the response,
matching the reference (paraphrases are fine). Answer 'yes'
if the correct core answer is present and not contradicted,
otherwise 'no'. Answer with only one word: "yes" or "no".
```

C EXPERIMENTAL RESULTS

C.1 EDITING COST

Table 4 compares per-document runtime and peak GPU memory usage across editing methods on Qwen2.5-7B. ECoA trains additional adapters on top of the edited model, introducing a trade-off between improved query-conditioned factual use and computational cost. Per-document runtime increases from 35.8 to 57.9 seconds for AlphaEdit+ and from 53.6 to 75.5 seconds for UnKE+.

The adapter training stage reaches a peak GPU memory usage of 31.9 GiB for both base editors, comparable to LoRA and COIN. This measurement excludes the base editing stage. Taking the maximum across the editing and adaptation stages, the overall peak remains 39.3 GiB for AlphaEdit+ and 40.1 GiB for UnKE+, as the base editing stage requires more memory. Thus, by freezing the edited base weights and training only low-rank adapters and conditioning projections, ECoA improves factual use without increasing peak GPU memory beyond that required by the base editor.

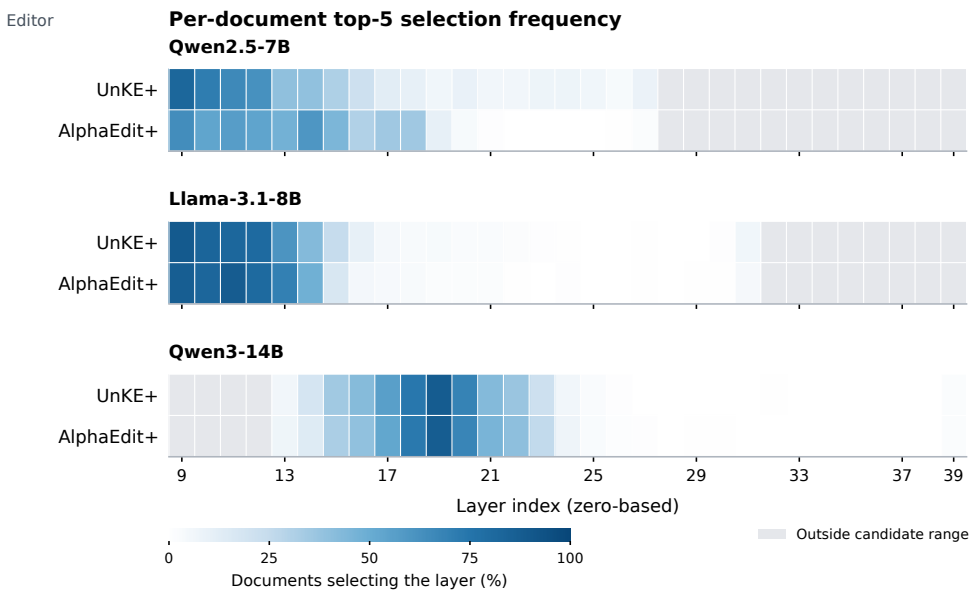

Figure 5: Distribution of document-specific layer selections. Each cell shows the percentage of documents for which the layer is selected among the top-5, with a shared color scale across all panels. Layer indices are zero-based. Gray indicates layers outside the candidate range, while white indicates eligible layers that are never selected. Percentages sum to 500% per row because five layers are selected for each document.

C.2 Distribution of Selected Layers

Figure 5 summarizes the top-5 layers selected for each edited document across models and base editors. For Qwen2.5-7B and Llama-3.1-8B, the five lowest-index layers in the candidate range account for 55.1%–81.6% of all selections, indicating relatively concentrated placement. For Qwen3-14B, this proportion drops to 29.1%–31.4%, with selections distributed across a broader range of layers. Within Qwen3-14B, layer 19 is nevertheless the most frequently selected layer for both base editors, indicating a shared preference despite the broader distribution. These distributions show that edit-aware layer selection adapts adapter locations to the document and model rather than assigning the same fixed layers to every document. The broader distribution observed for Qwen3-14B also highlights differences in selection behavior across models. Complementing this descriptive analysis, Table 3 evaluates the performance benefit of edit-aware selection over random layer selection.

C.3 Additional Lexical Similarity Metrics

Table 5 reports ROUGE-1 and ROUGE-2 alongside ROUGE-L for single-document and batch editing on UnKEBench. In single-document editing, AlphaEdit+ w/ ECoA achieves the highest scores across all three metrics, with ROUGE-L, ROUGE-1, and ROUGE-2 of 0.451, 0.466, and 0.233, respectively. These results show that ECoA's gains extend beyond ROUGE-L to unigram and bigram overlap.

In batch editing, UnKE+ w/ ECoA performs best across all three metrics, achieving ROUGE-L, ROUGE-1, and ROUGE-2 of 0.376, 0.386, and 0.159, respectively, compared with 0.187, 0.196, and 0.031 for UnKE+. AlphaEdit+ w/ ECoA also improves all three metrics over its base editor, although the UnKE+ configuration achieves higher absolute scores in this setting. LoRA, FT-L, MEMIT, and AlphaEdit exhibit substantial declines in ROUGE-1/2 under batch editing, whereas UnKE, COIN, UnKE+, and the ECoA configurations retain comparatively higher lexical overlap. These findings are consistent with the main ROUGE-L results and show that ECoA's improvements are not specific to a single ROUGE variant.

Table 5: Additional lexical similarity metrics. R-L, R-1, and R-2 denote ROUGE-L, ROUGE-1, and ROUGE-2.

	Single-document editing			Batch editing		
Method	R-L	R-1	R-2	R-L	R-1	R-2
Pre-edited	0.194	0.203	0.036	0.194	0.203	0.036
LoRA	0.139	0.150	0.014	0.000	0.000	0.000
FT-L	0.175	0.182	0.040	0.014	0.014	0.000
MEMIT	0.276	0.288	0.085	0.031	0.032	0.000
AlphaEdit	0.278	0.290	0.092	0.004	0.004	0.000
UnKE	0.289	0.300	0.092	0.190	0.199	0.032
COIN	0.390	0.399	0.225	0.208	0.219	0.051
UnKE+	0.286	0.295	0.088	0.187	0.196	0.031
AlphaEdit+	0.361	0.371	0.153	0.216	0.227	0.041
UnKE+ w/ ECoA	0.431	0.443	0.220	**0.376**	**0.386**	**0.159**
AlphaEdit+ w/ ECoA	**0.451**	**0.466**	**0.233**	0.319	0.329	0.106

C.4 Hyperparameter Analysis

Number of selected layers. Figure 6 compares performance for $k \in \{3, 5, 7, 10\}$, including the default setting of $k = 5$ used in Table 1. For UnKE+, increasing k from 3 to 5 improves ROUGE-L from 0.371 to 0.431. The score then rises slightly to 0.436 at $k = 7$ and remains similar at 0.433 for $k = 10$. For AlphaEdit+, ROUGE-L reaches 0.410, 0.451, 0.441, and 0.430, respectively, declining beyond $k = 5$, whereas BERTScore peaks at 0.330 with $k = 7$. Increasing the number of selected layers therefore does not yield monotonic improvements across both base editors and metrics. We use $k = 5$ as a common default throughout our experiments.

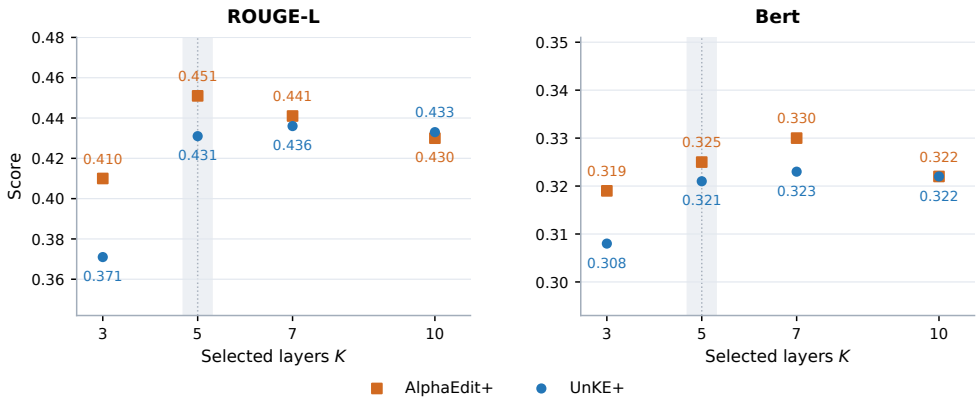

Figure 6: Query-conditioned factual use as a function of the number of selected layers k. The gray band indicates the default setting of $k = 5$ used in Table 1.

Number of training steps. Figure 7 shows performance as the number of adapter training steps increases to 50. For UnKE+, ROUGE-L improves up to 40 steps and remains nearly unchanged at 50 steps. For AlphaEdit+, it increases steadily across the five settings, reaching 0.350, 0.420, 0.451, 0.461, and 0.468. Both base editors show larger gains between 10 and 30 steps, followed by smaller changes. Although further training can improve performance, we use 30 steps by default to balance performance gains against additional training cost.

Distillation weight. Figure 8 examines the effect of the distillation weight λ_d. Setting $\lambda_d = 0$ removes distillation and trains the adapters using only cross-entropy on document-derived views.

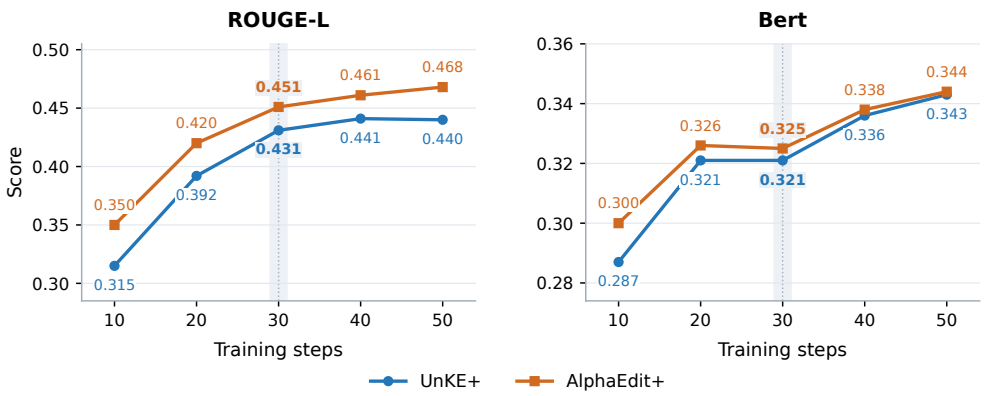

Figure 7: Performance as a function of the number of training steps.

Larger values place greater weight on limiting deviations from the edited model's predictive distribution.

Removing distillation causes a substantial performance drop for AlphaEdit+, while increasing λ_d from 0.5 to 1.0 reduces ROUGE-L for both base editors. For BERTScore, $\lambda_d = 0.5$ gives the best result of 0.325 for AlphaEdit+, while differences across the three settings are relatively small for UnKE+. These results suggest that regularization toward the edited model's predictions benefits post-edit adaptation, but excessive distillation can restrict adaptation to the document-derived views.

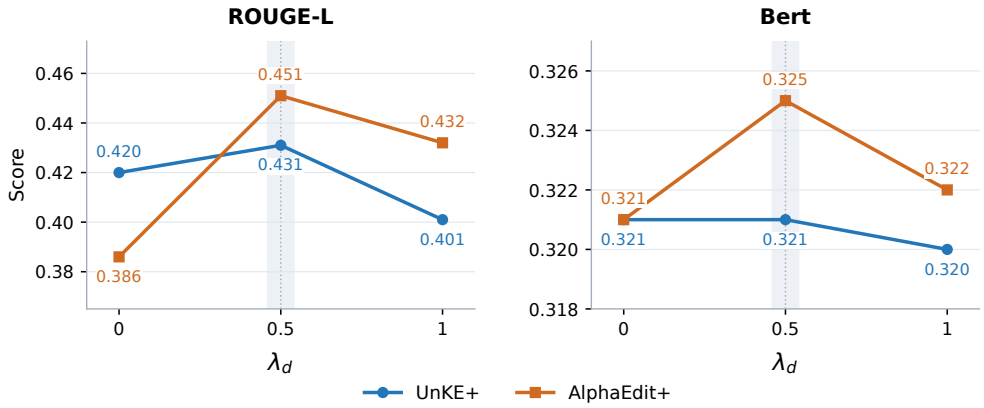

Figure 8: Effect of the distillation weight λ_d. The default setting is $\lambda_d = 0.5$.

C.5 Additional Component Analysis

Table 6 presents additional analyses of adapter target modules, conditioning variants, and training view composition. Attaching adapters to attention instead of MLP projections reduces ROUGE-L from 0.431 to 0.346 for UnKE+ and from 0.451 to 0.398 for AlphaEdit+. BERTScore also decreases for both configurations, indicating that MLP adaptation is more effective in the evaluated settings.

Replacing the learnable conditioning projections with Identity/SVD projections reduces ROUGE-L to 0.375 and 0.409 for the two base editors, respectively. This supports the contribution of learning to map edit responses into the adapter rank space. Entity-level conditioning slightly improves BERTScore for AlphaEdit+ from 0.325 to 0.328, but yields lower ROUGE-L than the default configuration for both base editors. The default token-level conditioning therefore provides more consistent performance across the evaluated settings.

Table 6: Additional component analysis. Variants modify the adapter target module, conditioning mechanism, or training views. Values in parentheses indicate absolute score changes relative to Full ECoA.

Variant	UnKE+ w/ ECoA		AlphaEdit+ w/ ECoA	
	ROUGE-L	BERT	ROUGE-L	BERT
Full ECoA	0.431	0.321	0.451	0.325
Attention instead of MLP	0.346 (−0.085)	0.290 (−0.031)	0.398 (−0.053)	0.307 (−0.018)
Identity/SVD projection	0.375 (−0.056)	0.308 (−0.013)	0.409 (−0.042)	0.324 (−0.001)
Entity-level conditioning	0.372 (−0.059)	0.306 (−0.015)	0.435 (−0.016)	0.328 (+0.003)
Continuation only	0.392 (−0.039)	0.308 (−0.013)	0.418 (−0.033)	0.319 (−0.006)
Cloze only	0.292 (−0.139)	0.271 (−0.050)	0.342 (−0.109)	0.280 (−0.045)

Table 7: Joint versus individual edit signals for batch editing. Individual uses document-specific editing updates for conditioning, whereas Joint uses the shared update of the jointly edited model. Joint is the default setting of ECoA.

Signal	UnKE+ w/ ECoA		AlphaEdit+ w/ ECoA	
	ROUGE-L	BERT	ROUGE-L	BERT
Individual	0.393	0.312	0.341	0.260
Joint (default)	0.376	0.292	0.319	0.237

Combining continuation and cloze views also proves effective. Using only continuation reduces ROUGE-L to 0.392 and 0.418 for the two base editors, respectively, while using only cloze leads to larger decreases, to 0.292 and 0.342. Although continuation outperforms cloze among the single-view variants, Full ECoA achieves the highest ROUGE-L and BERTScore among these view configurations. These results suggest that the two view types provide complementary training signals for improving query-conditioned factual use from different partial contexts.

C.6 Joint versus Individual Edit Signals

Table 7 compares conditioning signal configurations for batch editing. Individual uses document-specific editing updates for conditioning, whereas Joint shares the update of the jointly edited model, $\Delta_B = \theta_B - \theta$, across all inputs. Individual achieves higher performance for both base editors. With UnKE+, it reaches ROUGE-L of 0.393 and BERTScore of 0.312, compared with 0.376 and 0.292 for Joint. The same trend holds for AlphaEdit+, suggesting that document-specific edit signals can provide additional information for conditioning. Nevertheless, Joint achieves ROUGE-L of 0.376 and 0.319 for the two base editors without selecting a corresponding document update for each input. These scores exceed the base editors' respective scores of 0.187 and 0.216, demonstrating that the joint update alone supports effective post-edit adaptation. We therefore adopt Joint as the default configuration, as it provides substantial gains without requiring document-specific signal selection.

C.7 Case Study

Table 8 illustrates how ECoA improves query-conditioned factual use through a comparison of responses to a question about specific crops mentioned in the edited document. The target document states that Nathan Purdee managed the cultivation and harvesting of grapes, berries, and other fruits. The pre-edited model and FT-L fail to provide the requested information, while several editing baselines generate plausible but unsupported crop types, such as wheat, corn, and soybeans. COIN produces an unrelated description, and LoRA generates repetitive text. Both UnKE+ and AlphaEdit+ also name crops inconsistent with the target document. Adding ECoA to either editor yields an answer that correctly identifies grapes, berries, and other fruits.

Table 8: Qualitative comparison. Blue highlights the reference facts; red highlights inconsistent crop types. ✓ and ✗ denote positive and negative LLM judgments, respectively.

EDIT TARGET **Document 160**

Nathan Purdee is a raiser, as evidenced by his extensive experience in the agriculture industry. He has worked for over 10 years on a family-owned farm, where he has been responsible for managing the cultivation and harvesting of various crops, including **grapes, berries, and other fruits**. . . .

Question What crops has Nathan Purdee managed the cultivation and harvesting of?

Reference Grapes, berries, and other fruits.

Method	Generated response	Eval.
Pre-edited	As of my knowledge cutoff in July 2024, there is no widely known public information about a person named Nathan Purdee managing the cultivation and harvesting of specific crops.	✗
FT-L	I don't have specific information on a person named Nathan Purdee and the crops he has managed.	✗
LoRA	is is is is is is is is is is is is is is is is is is is is is is...	✗
MEMIT	Nathan Purdee has worked as a farmer, and he has managed the cultivation and harvesting of a variety of crops, including wheat, corn, soybeans, and various fruits and vegetables.	✗
UnKE	Nathan Purdee has managed the cultivation and harvesting of a variety of crops, including wheat, corn, soybeans, and barley.	✗
AlphaEdit	Nathan Purdee has managed the cultivation and harvesting of a diverse range of crops, including wheat, corn, soybeans, and various fruits and vegetables.	✗
COIN	Nathan Purdee is a fictional character from the video game *Harvest Moon 64* (known as *Bouquet* in Japan).	✗
UnKE+	Nathan Purdee has managed the cultivation and harvesting of a variety of crops, including corn, wheat, and soybeans.	✗
AlphaEdit+	Nathan Purdee has managed the cultivation and harvesting of a variety of crops, including corn, wheat, soybeans, and barley.	✗
UnKE+ w/ ECoA	Nathan Purdee has extensive experience in managing the cultivation and harvesting of various crops, including **grapes, berries, and other fruits**.	✓
AlphaEdit+ w/ ECoA	Nathan Purdee has extensive experience in managing the cultivation and harvesting of various crops, including **grapes, berries, and other fruits**.	✓