

000 PROPAGATING KNOWLEDGE IN LLMs WITH HYPER- 001 NETWORKS 002 003 004

005 **Anonymous authors**

006 Paper under double-blind review

007 008 ABSTRACT 009 010

011 Knowledge editing techniques for large language models (LLMs) can inject
012 knowledge that is later reproducible verbatim, but fall short on propagating that
013 knowledge; that is, LLMs can't answer questions that involve reasoning about it.
014 In this paper, we study hypernetwork-based knowledge editing techniques (i.e.,
015 MEND (Mitchell et al., 2022)) for knowledge propagation. We find that vanilla
016 hypernetwork-based editing methods do not effectively propagate knowledge. We
017 propose a simple fix to optimize hypernetworks for knowledge propagation, which
018 is to explicitly include propagation questions as the objective during hypernetwork
019 training. This achieves a substantial performance gain in the RippleEdit dataset,
020 almost $2\times$ accuracy on challenging multi-hop questions whose answer strings do
021 not appear in the injected fact. We further introduce a new synthetic dataset,
022 Controlled RippleEdit, that isolates a confounding factor in knowledge
023 propagation evaluation and further supports evaluating the generalization of knowl-
024 edge propagation. Our approach outperforms all other approaches for knowledge
025 propagation, including more computationally intensive methods such as continued
026 fine-tuning on synthetic data. Hypernetworks demonstrate some scaling to multi-
027 edit settings (up to 20 edits), achieving performance on par with or higher than
028 CPT-based approaches. Yet, we observe significant limitations in the performance
029 for out-of-domain propagation, suggesting future work in propagating knowledge
030 to a wide range of relations.

031 032 1 INTRODUCTION

033 Knowledge editing methods (Meng et al., 2022; Mitchell et al., 2022; De Cao et al., 2021; Sinitzin
034 et al., 2020) can transform large language models (LLMs) to *reproduce* injected knowledge, but
035 induce very limited *propagation* of that knowledge (Cohen et al., 2024; Zhong et al., 2025). This
036 failure stands in disappointing contrast to LLMs' ability to propagate knowledge that is given in
037 context at inference time (Onoe et al., 2022; Zheng et al., 2023). One promising path for propagation
038 is through training on data that explicitly demonstrates that propagation (Padmanabhan et al., 2023;
039 Akyürek et al., 2024; Chang et al., 2024), but these methods require large-scale data augmentation
040 for each piece of knowledge to be injected (Yang et al., 2024).

041 Hypernetwork-based editing methods (i.e., MEND (Mitchell et al., 2022)) present an attractive
042 solution for knowledge editing by introducing auxiliary hypernetworks to make efficient, effective,
043 and local edits to LMs. Yet, prior work (Cohen et al., 2024) established that these methods fail to
044 achieve meaningful knowledge propagation. In this work, we explore adapting a hypernetwork-based
045 editing method for knowledge propagation. We rectified the learning objective of the hypernetwork
046 and substantially improved its results in knowledge propagation. Our study builds upon Model Editor
047 Networks using Gradient Decomposition (MEND (Mitchell et al., 2022)), proposing to train these
048 hypernetworks with knowledge propagation as the core objective. Taking in a model's gradient from
049 the language modeling objective on the injected fact as input, now hypernetworks are trained to
050 modify that gradient to enable LMs to answer propagation questions involving that fact correctly
051 when the output gradient is applied; see Figure 1. We further improve hyperparameter choices
052 for MEND (e.g., layers in which model updates are applied), significantly improving propagation
053 performance.

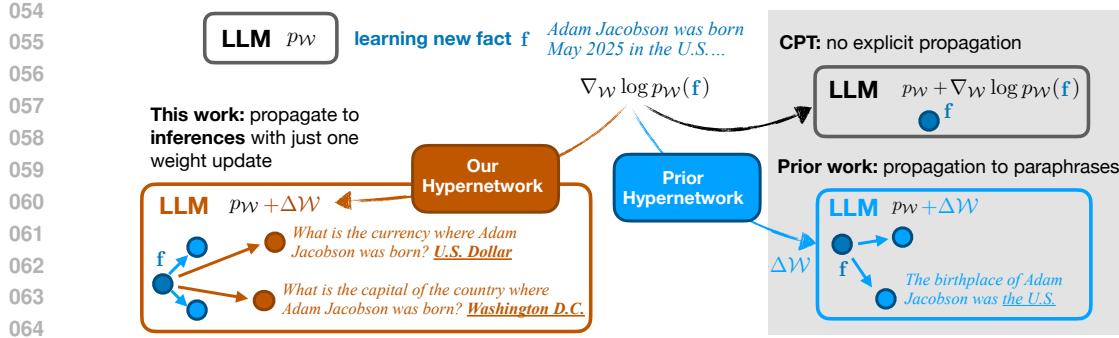


Figure 1: Our modification to enable propagation of injected knowledge. Our hypernetwork is trained to modify the gradient from the next token prediction loss on the injected knowledge to allow answering of multi-hop questions that rely on the newly injected knowledge.

We first evaluate our approach on RippleEdit (Cohen et al., 2024), a knowledge propagation question answering dataset. Existing methods excel in instances where the target answer appears verbatim in the injected facts, while achieving negligible improvement on non-verbatim questions. We show our variants outperform all other approaches, showing almost $2\times$ accuracy (22.4% compared to 12.7% of the next best system) in non-verbatim cases.

To better understand the extent of knowledge propagation, we design a new synthetic dataset Controlled RippleEdit. We focus on injecting facts related to well-known entities, allowing us to test propagation through the information already known to LLMs. We design test sets to evaluate propagation relations and entities seen during hypernetwork training and those that are unseen. In this new dataset, we observe that our approach outperforms other approaches (Gururangan et al., 2020; Lin et al., 2025; Meng et al., 2023; Mitchell et al., 2022) consistently, in both in-domain settings and on out-of-domain generalization. However, our model performance degrades significantly from the in-domain setting to the hardest out-of-domain setting (76.7% to 18.3% accuracy), leaving ample headroom for further work. Lastly, we show that our design changes can be applied to RLEdit (Li et al., 2025), a recent method that focuses on improving MEND for multiple edit scenarios.

Our contributions are: (1) A new learning objective and data condition, which meta-trains a hypernetwork explicitly for propagation; we show this applies to both single-edit and multi-edit settings. (2) An analysis and evaluation on RippleEdit, showing that our variant achieves substantial improvement on questions whose answers are not verbatim in the injected fact. (3) A new dataset Controlled RippleEdit, which allows us to evaluate out-of-domain settings in knowledge propagation. Our model shows improvement over baselines in this challenging setting.

2 BACKGROUND

2.1 TASK

We define a language model \mathcal{M} with parameters \mathcal{W} modeling a probability distribution $p_{\mathcal{W}}(x_i | \mathbf{x}_{<i})$ of current token x_i given the previous tokens $\mathbf{x}_{<i}$. Such an LM is defined by its architecture and parameters, which are real-valued weight tensors $\mathcal{W} = \{W_{\ell,k}, \dots\}$, where ℓ denotes the layer index and k ranges over the number of weights per layer (e.g., the MLP matrices and projection matrices for self-attention).

The task of knowledge editing is to inject a previously unknown fact or facts represented by f into the model. In this work, f consists of raw text (e.g., $f = \text{"Keir Starmer was elected prime minister of the UK"}$). The weights are updated by $\Delta\mathcal{W} = \{\Delta W_{\ell,k}, \dots\}$, yielding $\tilde{\mathcal{W}} = \{W_{\ell,k} + \Delta W_{\ell,k}, \dots\}$ as the final weights which should reflect f . Ideally, the model should be able to use this fact in various contexts (efficacy of the edit) without modifying unrelated facts (locality of the edit).

We introduce a set of propagation questions associated with each injected set of facts: our data is of the form $\{(f_i, \{(q_{ij}, a_{ij})\})\}$. For instance, given the f in the previous paragraph, propagation questions might be (Q : *What year was the prime minister of the UK born?* A : 1962; *What political party is the prime minister of the UK associated with?* A : Labour Party). These questions evaluate

```

108 Training hypernetwork  $g_\phi$ 
109
110 Input: Pretrained Language Model  $p_W$ 
111 Set of weights to update  $\mathcal{W}$ 
112 Coefficient  $c_{\text{edit}}$  Edit dataset  $D_{\text{edit}}^{\text{tr}}$ 
113
114 Repeat
115    $\mathcal{W}_1 = \mathcal{W}$ 
116   for  $t = 1$  to  $n$  do
117     Sample from edit dataset  $D_{\text{edit}}^{\text{tr}}$ 
118     Calculate update  $\Delta\mathcal{W}_t = g_\phi(\nabla w_t)$  ←
119     Update model  $\mathcal{W}_{t+1} = \mathcal{W}_t + \Delta\mathcal{W}_t$ 
120     Calculate editing loss  $L_{e_t}$  at turn  $t$  ←
121     Calculate locality loss  $L_{\text{loc}_t}$ 
122      $L_t = c_{\text{edit}} \cdot L_{e_t} + L_{\text{loc}_t}$ 
123     [Optional regularization]
124     Compute loss  $L$  by summing  $L_t$ 
125      $\phi \leftarrow \text{Adam}(\phi, \nabla_\phi L)$ 
126   until  $g_\phi$  converges

```

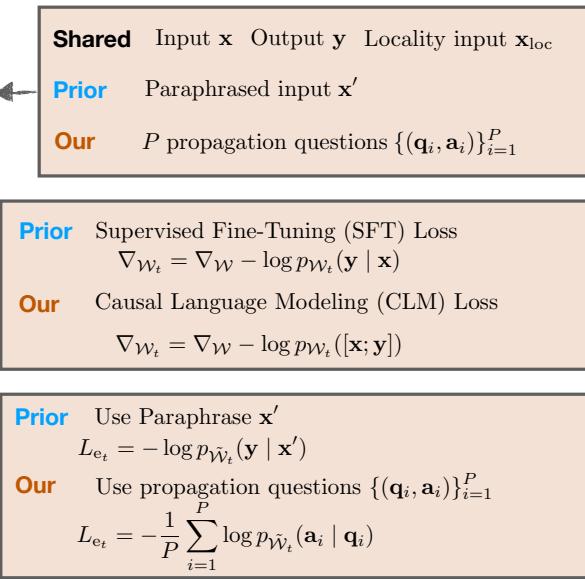


Figure 2: We modify the hypernetwork to take a gradient from causal language modeling of a new fact and transform it such that, when applied to the model, the model can answer propagation questions. The pseudocode skeleton depicts the general process used by hypernetwork-based editing methods.

that an updated LM should use its knowledge of the fact f . Such questions have been explored in past work where they have been harvested from knowledge bases (Cohen et al., 2024) or by prompting LMs (Akyürek et al., 2024).

A natural approach is to compute an update to the weight $\Delta\mathcal{W}$ as the gradient of a language modeling loss or SFT loss computed on \mathbf{f} ; for instance, $\Delta\mathcal{W} = \alpha \nabla p_{\mathcal{W}}(\mathbf{f})$, where α is the learning rate. However, training a model on some text is typically insufficient to inject that knowledge in a way that leads to strong performance on the (\mathbf{q}, \mathbf{a}) pairs (Chang et al., 2024; Berglund et al., 2024).

2.2 HYPERNETWORK-BASED EDITING WITH MEND

Our work first builds on MEND (Mitchell et al., 2022), a hypernetwork-based method for knowledge editing. MEND will compute an update $\Delta\mathcal{W}$ via a modification of the basic gradient.

The hypernetwork g_ϕ is parameterized by ϕ and meta-trained on an editing dataset $D_{edit}^{tr} = \{(\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{x}_{loc})_i\}$. As depicted in Figure 2, the training of the hypernetwork involves an inner-loop update which (1) computes the gradient of the injected fact; (2) modifies that gradient with the hypernetwork g_ϕ ; (3) applies the gradient to the base network \mathcal{W} to form an updated network $\tilde{\mathcal{W}}$. In standard MEND, the gradient in (1) is computed over an input-output pair (\mathbf{x}, \mathbf{y}) (e.g., a QA pair) as $\nabla_{\mathcal{W}} L^I(\mathbf{x}, \mathbf{y}) = \nabla_{\mathcal{W}} [-\log p_{\mathcal{W}}(\mathbf{y} \mid \mathbf{x})]$.

In the outer loop, the desiderata of generalization and locality are specified by an SFT loss (as editing loss L_e) with paraphrased input \mathbf{x}' and Kullback–Leibler divergence (as locality loss L_{loc}) with a random input \mathbf{x}_{loc} from the NaturalQuestions dataset (Kwiatkowski et al., 2019). An additional coefficient c_e (typically 0.1) is used to balance between the two desired properties.

$$L^O = c_e L_e(\tilde{\mathcal{W}}) + L_{\text{loc}}(\mathcal{W}, \tilde{\mathcal{W}}) = -c_e \log p_{\tilde{\mathcal{W}}}(\mathbf{y} \mid \mathbf{x}') + \text{KL}(p_{\mathcal{W}}(\cdot \mid \mathbf{x}_{\text{loc}}) \| p_{\tilde{\mathcal{W}}}(\cdot \mid \mathbf{x}_{\text{loc}})) \quad (1)$$

The full pseudocode for MEND can be found in Appendix B.4. MEND makes a key observation that the gradient of L^I with respect to weights \mathcal{W} is a rank-1 matrix. This allows more efficient parameterization of the hypernetwork q_ϕ and efficient computation of the final weight update.

A major limitation of MEND is the structure of the inner- and outer-loop losses. As described in the paper, the inner loop injects a single QA pair (x, y) , and the outer loop only encourages propagation to paraphrases of that QA pair. In the next section, we describe our method, which extends MEND and relaxes these assumptions.

162 2.3 HYPERNETWORK-BASED MULTI-EDITING WITH RLEdit
163

164 For a practical application of editing methods, the editing method should be able to handle injecting
165 multiple facts robustly. Previous work has shown that MEND is not effective out-of-the-box at
166 injecting facts sequentially (Meng et al., 2023; Li et al., 2025). Recent work, RLEdit (Li et al.,
167 2025), has suggested that the key issue for multi-edit failing is MEND’s training process, which does
168 not accommodate the cascading parameter changes after each turn of editing. To address this, they
169 meta-train the gradient modification by taking into account the cumulative delta over multiple steps.

170 RLEdit frames the multi-edit process for n facts as editing m facts for $\frac{n}{m}$ turns. Concretely, at the
171 t -th turn of editing, RLEdit edits a “base model” obtained by applying the hypernetwork update to
172 the previous model turns. The meta-training objective then also includes a sum over losses from
173 k previous turns; this is to encourage the updated hypernetwork to also perform well despite the
174 cascading changes of model weights. Finally, RLEdit adds a weight normalization and a decay term
175 to the summation of losses over the sequence. In Section 6, we present results on applying our
176 approach on top of RLEdit.

177 3 PROPAGATING FACTS WITH HYPERNETWORKS
178

180 We propose to modify the training and loss shared by conventional hypernetwork-based algorithms,
181 described below and visualized in Figure 2. There are two principal modifications: training data and
182 learning objective.

183 **Training data of outer loop** First, the outer loop loss is computed over the propagation questions:
184

$$185 \quad 186 \quad 187 \quad L_e = -\frac{1}{P} \sum_{i=1}^P \log p_{\tilde{W}}(\mathbf{a}_i \mid \mathbf{q}_i) \quad (2)$$

188 Critically, this loss encourages the trained hypernetwork to make modifications that enable the final
189 model to correctly answer propagation questions. This property does not hold for the conventional
190 hypernetwork training objective; there, the objective in the outer loop is to predict simple paraphrases
191 of the injected fact.

192 **Learning objective of inner loop** Second, we make the structure of the inner loop more flexible: we
193 use the standard causal language model (CLM) loss to enable the model to inject any new knowledge
194 expressible as text, rather than requiring it to be structured as QA pairs as in conventional objectives:

$$195 \quad 196 \quad 197 \quad L^I = -\log p_{\mathcal{W}}([\mathbf{x}; \mathbf{y}]) = -\log p_{\mathcal{W}}(\mathbf{f}) \quad (3)$$

198 where $[\cdot ; \cdot]$ means the concatenation of two strings. This objective resembles the inner loop loss used
199 in past editing work (Chen et al., 2023).

200 Together, these two losses reflect the chief objective of knowledge editing: taking raw knowledge
201 expressed in text (which can be trained on with next token prediction loss) and adapting the learning
202 of that knowledge to support answering propagation questions. This goal is more ambitious than that
203 of a conventional objective, which propagates QA pairs to paraphrases of those questions. Injection
204 under a conventional objective may underperform on knowledge that is not expressed as QA pairs,
205 and it may propagate less than a model explicitly trained to be able to answer propagation questions.

206 4 EVALUATION ON RIPPLEEDIT
207

208 4.1 EXPERIMENTAL SETTINGS

209 We evaluate on instances from RippleEdit with the following procedure. An LLM \mathcal{M} receives an
210 edited fact $\mathbf{e} = (s, r, o^*)$ to be injected into LLM, yielding an updated model $\mathcal{M}^{(\mathbf{e})}$. After that, the
211 model is evaluated on a set of P propagation queries (including all propagation types) in the format
212 $\{(\mathbf{q}_i, \mathcal{A}_i)\}_{i=1}^P$, where \mathbf{q}_i is a query string from one of the 6 propagation types, and \mathcal{A}_i is the set of
213 valid answers for the query \mathbf{q}_i . See the detailed description of the task in Appendix D.

214 **Data Setup** RippleEdit has three subsets, Popular, Random, and Recent. We do not
215 distinguish these subsets for simplicity, and form the dataset splits out of the union of all of them. We

randomly sample 500 examples for a validation set, 500 examples for a test set, and use the remaining 3,686 examples for training. We filter examples in the validation and test sets, such that each instance has at least 1 test query for efficacy and 1 test query for specificity. The training dataset here is used for meta-training our hypernetwork and not for learning specific knowledge. See the statistics for a number of propagation questions in Table 6.

Following prior work (Scialanga et al., 2025), we categorize six propagation types in RippleEdit into two: (1) *efficacy* queries, since these test the effectiveness of knowledge injection and propagation of a test fact. (2) *specificity* queries, whose answer should not change after the edit. An example can be found in Table 4c in the appendix.

Our analysis of the dataset revealed that the answer to the propagated fact frequently appears verbatim in the edit fact (overall 31.9% of propagation questions in the test set; see breakdown per propagation type in Table 5 in the Appendix). Models can trivially answer these questions correctly by learning to copy from edited facts. Therefore, we divide test queries into two sets: those that require *non-verbatim propagation* and those that do not, and report performances on each set.

Evaluation Metrics We greedily decode a maximum of 20 new tokens. We use two evaluation metrics, **Exact Match (EM)**, following the original paper, and **LLM-as-Judge (LLM-Score)**, a more robust metric that can handle lexical variations. **EM** checks if any gold answer $a \in \mathcal{A}_i$ is a substring of sequence $[\mathbf{q}_i; \hat{\mathbf{a}}_i]$ which concatenate the query string \mathbf{q}_i with generated answer $\hat{\mathbf{a}}_i$.¹ For **LLM-as-Judge (LLM-Score)**, an LLM (GPT-4o-mini) takes the query string \mathbf{q}_i , the generated answer $\hat{\mathbf{a}}_i$, and one answer from valid answers $a \in \mathcal{A}_i$, and gives a numerical score of whether the generated answer matches the valid answer. If the generated answer matches any of the valid answers, we count it as correct. See the LLM prompt in Appendix A.1.

4.2 COMPARISON SYSTEMS

All our model variants use a 16-layer pre-trained transformer model, `Llama-3.2-1B-base`. We conduct a light-weight supervised fine-tuning on the TriviaQA dataset (Joshi et al., 2017) on this model to teach the model to answer in short answer format: $L_{\text{SFT}}(\mathcal{M}) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \text{TriviaQA}} [\log p_{\mathcal{M}}(\mathbf{y} | \mathbf{x})]$. We call the tune model `Llama-3.2-1B-base-QA`.

- **Prepend:** This is not an editing method, simply prepending the new fact \mathbf{f} to the test query \mathbf{q}_i at inference time. Past work has shown this method to be a competitive baseline (Cohen et al., 2024).
- **Continued Pretraining (CPT)** is frequently used to adapt an off-the-shelf LM to new domains or tasks (Gururangan et al., 2020). We continue training the base model with the next token prediction loss (Equation 3) on the new fact \mathbf{x} . We report two variants, differing in which parameters are updated — all parameters in the model (denoted CPT (Full)), or parameters associated with layers 10-12 (denoted CPT (Mid-Upper)).
- **Active-Reading CPT** (Lin et al., 2025) augments the injected fact by prompting a language model to generate more data, aiming to emulate the way humans actively engage with new information. We report results of fully finetuning the parameters on this data (denoted Active-Reading CPT (Full)). See details about the method in Appendix B.2.
- **MEMIT** (Meng et al., 2023) creates a weight update by solving an closed-form optimization problem; it requires precomputed covariance matrices from a reference corpus, typically on `wikitext-103` (Merity et al., 2017). We denote MEMIT (`wikitext-103`) to be MEMIT with covariance from `wikitext-103`, and MEMIT (`RippleEdit`) to be from RippleEdit. See more details in Appendix B.3.
- **MEND** (Mitchell et al., 2022): We present two versions of MEND. One follows the standard practice (denoted MEND (with standard config)); the other follows our practice in MEMIT and edits MLP weights at layer 10-12 (denoted MEND (Mid-Upper)). See more details in Appendix B.4
- **MEND+Propagation:** we apply the modification described in Section 3 to MEND, targeting layers 4-15. We also include a variant denoted by MEND+Propagation (Mid-Upper), targeting layers 10-12.

¹In the original paper (Cohen et al., 2024), the evaluation pipeline filters test queries based on edit success, performance on prerequisite test queries, making the set of evaluation queries different for different models. We do not filter to ensure each method is evaluated on the same test set.

270
 271
 272
 273
 274
 275
 Table 1: LLM-Score Results on the RippleEdit dataset, stratified by verbatim / non-verbatim,
 and token usage relative to CPT. MEND+Propagation improves over baselines on verbatim questions
 whose answers are in the injected facts, and on non-verbatim questions whose answers are not in the
 injected facts. Improvement of existing baselines mostly comes on questions with verbatim answers.
[†]means the system is outperformed by MEND+Propagation on that metric according to a paired
 bootstrap test ($p = 0.05$).

# tokens		Efficacy		Specificity	
		Verbatim (1373)	Non-Verbatim (1586)	Verbatim (165)	Non-Verbatim (2099)
Llama-3.2-1B-base-QA	0×	11.6 [†]	9.2 [†]	13.2 [†]	27.7 [†]
Prepend	1×	36.7 [†]	22.4	18.8	28.7 [†]
CPT (Full)	1×	76.0	7.8 [†]	15.8 [†]	16.0 [†]
CPT (Mid-Upper)	1×	41.8 [†]	9.7 [†]	20.7	26.3 [†]
Active-Reading CPT (Full)	265×	81.6	9.7 [†]	20.5	17.7 [†]
MEMIT (wikitext-103)	1×	17.0 [†]	12.7 [†]	17.7 [†]	24.5 [†]
MEMIT (RippleEdit)	1×	22.5 [†]	12.7 [†]	22.0	21.4 [†]
MEND (with standard config)	1×	64.5 [†]	8.2 [†]	24.3	23.6 [†]
MEND (Mid-Upper)	1×	63.5 [†]	8.2 [†]	21.6	21.6 [†]
MEND+Propagation (Mid-Upper)	1×	71.1 [†]	19.3 [†]	27.3	32.0 [†]
MEND+Propagation	1×	75.7	22.4	24.1	35.4

4.3 RESULTS

Table 1 presents the results on RippleEdit dataset. MEND+Propagation performs strongly on both efficacy and specificity. Especially on non-verbatim questions, our system is the only one that shows substantial gain (9.2 → 22.4), while the best other system achieves only 12.7 (MEMIT). For existing methods, improvement in efficacy mostly comes from questions whose answer is verbatim in the edits (11.6 → 76.0, CPT (full)), but offers negligible improvement on questions whose answers are not in the edits.

We note the Active-Reading method achieves the best performance on questions with verbatim answers, but very low improvement on questions with non-verbatim answers. Although it generates a large number of augmented tokens (265× the size of the CPT data), these tokens are not aligned with the challenges of questions with non-verbatim answers.

The Prepend baseline is the strongest on non-verbatim questions (9.2 → 22.4) more substantially than other methods. We report exact match in Table 26 and performance by propagation types in Table 27 in the appendix.

Limitation of RippleEdit While RippleEdit provides an initial testbed for our work, we find this dataset is not ideal for testing knowledge propagation. Many questions involve tail entities, where the base LM does not parametrically know the relevant information. For example, if LM does not know who the siblings of Keir Starmer are, it would not be able to answer the propagation question “*who is the sibling of the prime minister of the United Kingdom*” even if it could propagate the new fact “*Keir Starmer is the new PM of the UK*”. In the following section, we present a new synthetic dataset that centers around entities and relationships that the model is familiar with.

5 EVALUATION ON CONTROLLED RIPPLEEDIT

We introduce a new dataset called Controlled RippleEdit, which allows a focused evaluation of the model’s knowledge propagation ability. This dataset also allows evaluating out-of-domain performance, propagating along relations unseen during training, unseen entities, or both.

Data Instances Figure 3 illustrates an instance of Controlled RippleEdit. Each instance has a new fact f centering around a fake entity s_f and involving three real-world entities o_1, o_2, o_3 . It also has a set of propagation questions $\{(q_i, a_i)\}_{i=1}^P$ built from P unique knowledge base relations (e.g., `capital_of`) associated with one of the real-world entities (o_1, o_2, o_3). Instead of referring

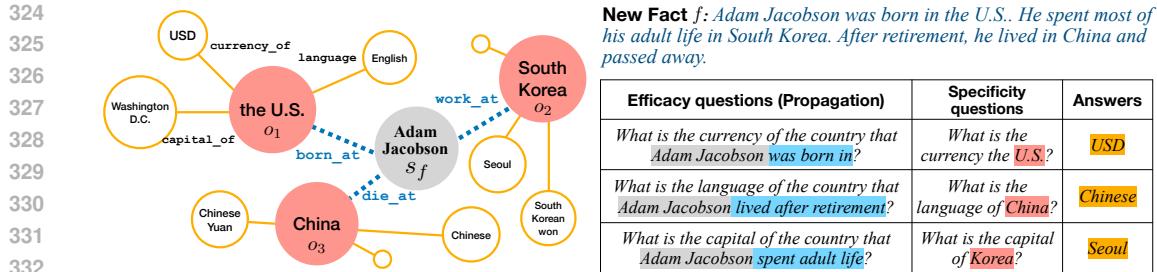


Figure 3: Illustration of our Controlled RippleEdit dataset, designed to evaluate knowledge propagation on well-known entities and relations. Each instance consists of (1) a fictional story (f) relating a fake entity s_f to three real-world entities (o_1, o_2, o_3); and (2) a set of P propagation question-answer pairs $\{(q_i, a_i)\}_{i=1}^P$. Each q_i inquires about a knowledge base relation on one of the real-world entities o_j , but referring to it via its relation to the fake entity.

to the real-world entity directly, the propagation question will refer to it using its relation to the fake entity s_f (e.g., *the country where Adam Jacobson was born*). Therefore, the LM must be able to combine its prior knowledge about real-world entities and the injected fake entity s_f to answer the question correctly. The dataset generation process is further described in Appendix E.1.

Final Dataset We generate 5K instances of Controlled RippleEdit and randomly split these into 4K for training the hypernetwork, 500 for validation, and 500 for testing. To evaluate out-of-domain (OOD) generalization, we generate three additional test sets. We generate 350 instances where their real-world entities (o_i) do not appear in the training dataset (but knowledge base relations occur in the training dataset), calling this set OOD (Entity). Analogously, we generate an OOD (Relation) dataset. Lastly, we generate an OOD (Both) dataset, consisting of 350 instances where neither real-world entities nor the knowledge base relations appear in the training dataset.

5.1 EXPERIMENTAL SETUP

Model We further train the model in Section 4.2 with 500 QA pairs involving real-world entities and relations in Controlled RippleEdit to make the propagation easier by reinforcing the model’s knowledge of the propagation relations. We call this model Llama-3.2-1B-base-QA, and this model is used for all comparison methods in this section.

Metric We use LLM-as-a-Judge (with GPT-4o-mini) to evaluate the correctness of the predicted answer against the reference answer, as in the prior section. For efficacy measure, we use the model’s performance on multi-hop questions, e.g., “*Q: What is the currency of [the country that Adam Jacobson was born in]? A: U.S. Dollar*”. To measure specificity, we evaluate whether the model retains its ability to answer simplified versions of our questions that do not require any updated knowledge, e.g., “*What is the currency of the United States?*”. See examples in Figure 3. Ideally, updates to the model should not degrade its ability to answer these questions.

Comparison methods We use the same set of comparison methods described in Section 4.2. For fair comparison, we modify MEMIT and MEND. As they require the fact f to be in an input-output format (\mathbf{x}, \mathbf{y}) , we map f into three atomic facts (e.g., *(Adam Jacobson was born in, the U.S.)*); and conduct multi-edit to inject those facts. See examples in Table 7 and details in Appendix E.3. Lastly, we add one more baseline, described below.

Meta-Aug CPT This CPT method explicitly uses how the meta-training set was constructed. We augment an injected fact using the gold in-domain relations for that fact, essentially reproducing what would be the propagation questions in the meta-training set, paired with gold answers. For instance, for the example in Figure 1, we would generate “*What is the currency of the country that Adam Jacobson was born? US Dollar*”, “*What is the currency of the country that Adam Jacobson lived in after retirement? Chinese yuan*”, and “*What is the currency of the country that Adam Jacobson spent adult life? South Korean won*”. We denote this method as Meta-Aug CPT. For In-Domain and OOD (Entity) test sets, note that the test queries are included directly in the CPT data. Because the data is dynamically generated for each example, In-domain and OOD (Entity) are distributionally equivalent for this approach.

378 Table 2: **Single-edit results on Controlled RippleEdit** with Llama-3.2-1B-base-QA.
 379 We use the model’s LLM-Score on multi-hop questions for efficacy, and the model’s LLM-Score
 380 on single-hop questions for specificity. MEND+Propagation significantly outperforms most of the
 381 baselines on the In-Domain test set and also OOD test sets where components of the injected facts
 382 do not appear during meta-training. \dagger means the system is out-performed by MEND+Propagation
 383 according to a paired bootstrap test ($p = 0.05$). Gray cells do not match the data or learning
 384 condition of other cells; see text. For all CPT methods, we conduct full finetuning.

	#token	In-Domain (2284)		OOD (Entity) (1368)		OOD (Rel) (421)		OOD (Both) (447)	
		Effi.	Spec.	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.
Llama-3.2-1B-base-QA	0 \times	8.3 \dagger	94.7 \dagger	7.1 \dagger	94.3	8.9 \dagger	94.2	10.9 \dagger	90.7
Prepend	1 \times	38.1 \dagger	86.2 \dagger	41.5	88.2	29.4 \dagger	82.4	31.7	79.5
CPT	1 \times	18.1 \dagger	80.2 \dagger	17.0 \dagger	79.9 \dagger	15.6 \dagger	79.3 \dagger	12.9 \dagger	71.1 \dagger
Meta-Aug CPT	7 \times	80.3	75.4 \dagger	79.1	73.3	26.1 \dagger	57 \dagger	12.9 \dagger	51.7 \dagger
Active-Reading CPT	95 \times	19.6 \dagger	69.6 \dagger	19.1 \dagger	67.8 \dagger	23.3 \dagger	65.4	16.7	62.8
MEMIT (wikitext-103)	1 \times	12.8 \dagger	94.4 \dagger	14.4 \dagger	94.4	12.0 \dagger	93.9	13.8 \dagger	90.0
MEMIT (Ctrl RippleEdit)	1 \times	12.0 \dagger	94.6 \dagger	13.3 \dagger	94.5	11.1 \dagger	94.3	11.6 \dagger	90.2
MEND (with standard config)	1 \times	14.7 \dagger	89.0 \dagger	14.2 \dagger	89.4	10.1 \dagger	91.8	10.7 \dagger	86.3
MEND (Mid-Upper)	1 \times	12.3 \dagger	91.8 \dagger	11.5 \dagger	92.9	11.5 \dagger	92.2	12.0 \dagger	88.1
MEND+Propagation (Mid-Upper)	1 \times	60.8 \dagger	91.3 \dagger	36.0	85.4	28.4 \dagger	87.4	18.3	84.0
MEND+Propagation	1 \times	76.7	95.5	35.2	81.6	34.5	84.0	18.3	77.5

5.2 RESULTS: EFFECTIVENESS OF PROPAGATION

We report the results on Controlled RippleEdit in Table 2. Our variant, MEND+Propagation, substantially outperforms other parametric methods consistently for various settings. On the in-domain test set, MEND+Propagation even outperforms Prepend, showing that parametric propagation can be as effective as in-context augmentation.

We observe MEND+Propagation’s performance degrades in out-of-domain settings when either entities or relations are unobserved during training. However, MEND+Propagation still outperforms other methods substantially. For example, on OOD (Entity), the best-performing baseline MEMIT (wikitext-103) achieves 20.8% lower performance than our variant. We observe that our variant’s performance improvement in OOD (Entity) tends to be higher than OOD (Relation). On OOD (Both), where our variant does not observe any entity or relation in the test, MEND+Propagation is able to offer better propagation than others, but the gap is smaller.

Additional Results In Appendix F, we report further results on Controlled RippleEdit, which we summarize here. In Table 14 and 19, we conduct ablation studies and show design elements proposed in Section 3 are essential. In Table 11 and 18, we conducted experiments that scale hypernetwork in terms of meta-training data and parameter usage, and show OOD performances are not fixable by mere scaling and require some algorithmic innovation. In Table 15, we report performance on two evaluation subsets (where the answer occurred in the meta-training set or not), showing that the hypernetwork generalizes to queries whose answers are not in the meta-training set. In Table 28 and 21, we show that gains from MEND+Propagation are robust to models of different size and family. Lastly, we show MEND+Propagation is cost-effective by comparing memory usage and runtime in Table 20 and 17.

6 PROPAGATING MULTIPLE FACTS AT ONCE

Knowledge propagation can be thought as making multiple knowledge edits at once, including injected facts and facts that are impacted by the injected facts. Being able to propagate multiple facts to the language model will have an additional multiplier effect, thus making the learning of knowledge very data-efficient and the updated language model more generalizable. In this section, we extend our approach to RLEdit (Li et al., 2025), a recent hypernetwork training recipe for sequentially injecting multiple facts.

Setting We investigate injecting a total of 10 or 20 facts into the language model and measure the propagation performance. With RLEdit, both settings, we inject 5 facts at each turn, and change

432 Table 3: Multi-edit results on Controlled RippleEdit with Llama-3.2-1B-base-QA.
 433 Training MEND+Propagation with 20 edits leads to out-of-memory in our hardware configuration.
 434 Gray cells do not match the data condition of other cells; see text. For all CPT baselines, we conduct
 435 full finetuning. We bold the **best** system and underline the second best system.
 436

437 LLM-Score (↑)	#token	In-Domain (2284)		OOD (Entity) (1368)		OOD (Rel) (421)		OOD (Both) (447)	
		Effi.	Spec.	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.
439 Llama-3.2-1B-base-QA	440 0×	8.3	94.7	7.1	94.3	8.9	94.2	10.9	90.7
<i># injected fact = 10</i>									
442 CPT	1×	11.2	88.6	9.3	85.9	14.5	89.0	12.8	80.7
443 Meta-Aug CPT	7×	77.3	88.6	76.8	86.6	31.8	73.6	17.9	63.0
444 Active-Reading CPT	95×	12.8	75.9	10.3	77.0	17.3	78.0	14.7	74.0
445 MEND+Propagation	1×	25.5	68.2	15.5	48.4	10.9	54.6	11.4	56.5
RLEdit+Propagation	1×	<u>48.6</u>	92.9	<u>25.8</u>	<u>82.0</u>	8.6	<u>87.8</u>	<u>17.6</u>	79.5
<i># injected fact = 20</i>									
447 CPT	1×	10.3	89.8	8.3	88.2	13.8	89.9	12.5	82.5
448 Meta-Aug CPT	7×	72.2	89.2	68.8	85.9	28.3	74.5	17.0	62.7
449 Active-Reading CPT	95×	10.5	77.0	9.1	76.2	17.1	77.3	16.4	76.8
RLEdit+Propagation	1×	29.9	88.5	<u>18.3</u>	<u>78.3</u>	12.0	82.6	13.0	82.5

451 the number of turns to control the total number of facts. Since MEMIT and MEND underperform
 452 CPT-based methods on the single-edit scenario, and prior work (Meng et al., 2023; Mitchell et al.,
 453 2022) shows that multi-edit performance mostly decreases from single-edit performance, we focus
 454 on CPT-based baselines in this section.
 455

456 **Results** Table 3 reports performances. All methods show degraded performances compared to
 457 single edit scenarios, especially as the number of edits grows. Meta-Aug CPT shows strongest
 458 efficacy overall, especially in-domain. RLEdit+Propagation outperforms MEND+Propagation in
 459 most settings by a large margin. RLEdit+Propagation shows competitive performance in OOD
 460 (Entity), and comparable performance in OOD (Both), but lags in OOD (Relation), compared to CPT
 461 methods. Active-Reading CPT and Meta-Aug CPT, while showing stronger efficacy performance in
 462 OOD (Both) setting, also show a specificity drop.
 463

7 RELATED WORK

464 **Knowledge Propagation** Recent work has studied the propagation of injected knowledge, finding
 465 that existing methods are largely lacking. A line of work (Ma et al., 2024; Berglund et al., 2024)
 466 studied reversal curse — the model knows “A is B”, but not “B is A”. Other work (Qin et al., 2024;
 467 Nishi et al., 2025) analyzes unintended ripple effects of different editing methods. Hase et al. (2024)
 468 surveys a wide range of open problems regarding revising the belief of the model. We discuss recent
 469 benchmarks for evaluating knowledge edits in Appendix H.1.
 470

471 **Continual Learning** Knowledge editing can be viewed as continual learning, injecting new knowl-
 472 edge gradually. Continual learning has been studied in domain adaptation scenarios (Gururangan
 473 et al., 2020; Ke et al., 2023). A line of work studies catastrophic forgetting during continual learning
 474 (Chen et al., 2025; Franke et al., 2024; Jin & Ren, 2024a;b). They evaluate the performance on
 475 downstream tasks, rather than changes in parametric knowledge. We include more discussion related
 476 to continued pretraining in Appendix H.2.
 477

8 CONCLUSION

478 In this work, we introduce a simple but effective modification that addresses the critical challenge of
 479 propagating edits to related facts in current knowledge editing techniques. We show the effectiveness
 480 of our method on RippleEdit, a widely-adopted dataset measuring propagation. We present
 481 a controlled dataset centered around well-known entities and relations to further demonstrate the
 482 effectiveness when propagated knowledge is known by the model. We also show that our modification
 483 is generalizable to multi-edit editing methods, and the modified variant maintains strong performance
 484 improvement over baselines.
 485

486 **Reproducibility Statement** Our code and dataset will be made publicly available once.
 487 See anonymized code at <https://anonymous.4open.science/r/propmend-84F6/README.md>. We provided the description of RippleEdit at Appendix D; and Controlled RippleEdit at Section 5 and Appendix E.
 488
 489

490
 491 **REFERENCES**
 492

493 Afra Feyza Akyürek, Ekin Akyürek, Leshem Choshen, Derry Wijaya, and Jacob Andreas. De-
 494 ductive closure training of language models for coherence, accuracy, and updatability. In Lun-
 495 Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Compu-
 496 tational Linguistics: ACL 2024*, pp. 9802–9818, Bangkok, Thailand, August 2024. Associa-
 497 tion for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.584. URL <https://aclanthology.org/2024.findings-acl.584/>.
 498

499 Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz
 500 Korbak, and Owain Evans. The reversal curse: LLMs trained on “a is b” fail to learn “b is
 501 a”. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=GPKTIktA0k>.
 502

503 Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and
 504 Minjoon Seo. How do large language models acquire factual knowledge during pretraining? In
 505 *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL
 506 <https://openreview.net/forum?id=TYdzj1EvBP>.
 507

508 Howard Chen, Jiayi Geng, Adithya Bhaskar, Dan Friedman, and Danqi Chen. Continual memorization
 509 of factoids in language models, 2025. URL <https://arxiv.org/abs/2411.07175>.
 510

511 Zeming Chen, Gail Weiss, Eric Mitchell, Asli Celikyilmaz, and Antoine Bosselut. RECKONING: Rea-
 512 soning through dynamic knowledge encoding. In *Thirty-seventh Conference on Neural Information
 513 Processing Systems*, 2023. URL <https://openreview.net/forum?id=dUAcAtCuKk>.
 514

515 Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects
 516 of knowledge editing in language models. *Transactions of the Association for Computational
 517 Linguistics*, 12:283–298, 2024. doi: 10.1162/tacl_a_00644. URL <https://aclanthology.org/2024.tacl-1.16/>.
 518

519 Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing Factual Knowledge in Language Models. In
 520 *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*,
 521 2021.
 522

523 Jörg K.H. Franke, Michael Hefenbrock, and Frank Hutter. Preserving principal subspaces to reduce
 524 catastrophic forgetting in fine-tuning. In *ICLR 2024 Workshop on Mathematical and Empirical
 525 Understanding of Foundation Models*, 2024. URL <https://openreview.net/forum?id=XoWtroECJU>.
 526

527 Omer Goldman, Uri Shaham, Dan Malkin, Sivan Eiger, Avinatan Hassidim, Yossi Matias, Joshua
 528 Maynez, Adi Mayrav Gilady, Jason Riesa, Shruti Rijhwani, Laura Rimell, Idan Szpektor, Reut
 529 Tsarfaty, and Matan Eyal. ECLeKTic: a Novel Challenge Set for Evaluation of Cross-Lingual
 530 Knowledge Transfer, 2025. URL <https://arxiv.org/abs/2502.21228>.
 531

532 Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and
 533 Noah A. Smith. Don’t stop pretraining: Adapt language models to domains and tasks. *Proceedings
 534 of the Annual Meeting of the Association for Computational Linguistics (ACL)*, abs/2004.10964,
 535 2020.
 536

537 Peter Hase, Thomas Hofweber, Xiang Zhou, Elias Stengel-Eskin, and Mohit Bansal. Fundamental
 538 problems with model editing: How should rational belief revision work in LLMs? *Transactions
 539 on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=LRf19n5Ly3>.
 540

540 Zhengbao Jiang, Zhiqing Sun, Weijia Shi, Pedro Rodriguez, Chunting Zhou, Graham Neubig, Xi Lin,
 541 Wen-tau Yih, and Srini Iyer. Instruction-tuned language models are better knowledge learners.
 542 In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual*
 543 *Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5421–
 544 5434, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/
 545 v1/2024.acl-long.296. URL <https://aclanthology.org/2024.acl-long.296/>.

546 Xisen Jin and Xiang Ren. Demystifying forgetting in language model fine-tuning with statistical
 547 analysis of example associations. In *NeurIPS 2024 Workshop on Scalable Continual Learning*
 548 *for Lifelong Foundation Models*, 2024a. URL <https://openreview.net/forum?id=0d03UdUY0w>.

549 Xisen Jin and Xiang Ren. What Will My Model Forget? Forecasting Forgotten Examples in Language
 550 Model Refinement, 2024b. URL <https://openreview.net/forum?id=uleynu9DVf>.

551 Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale
 552 distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and
 553 Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Com-
 554 putational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July
 555 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL <https://aclanthology.org/P17-1147/>.

556 Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. Continual
 557 pre-training of language models. In *The Eleventh International Conference on Learning Represen-
 558 tations*, 2023. URL https://openreview.net/forum?id=m_GDIItaI3o.

559 Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris
 560 Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion
 561 Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav
 562 Petrov. Natural questions: A benchmark for question answering research. *Transactions of the*
 563 *Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL
 564 <https://aclanthology.org/Q19-1026>.

565 Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via
 566 reading comprehension. In Roger Levy and Lucia Specia (eds.), *Proceedings of the 21st Conference*
 567 *on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada,
 568 August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL
 569 <https://aclanthology.org/K17-1034/>.

570 Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and
 571 Xiang Wang. Reinforced lifelong editing for language models, 2025. URL <https://arxiv.org/abs/2502.05759>.

572 Jessy Lin, Vincent-Pierre Berges, Xilun Chen, Wen-Tau Yih, Gargi Ghosh, and Barlas Oguz. Learning
 573 facts at scale with active reading. *CoRR*, abs/2508.09494, 2025. doi: 10.48550/ARXIV.2508.09494.
 574 URL <https://doi.org/10.48550/arXiv.2508.09494>.

575 Zeyu Leo Liu, Shrey Pandit, Xi Ye, Eunsol Choi, and Greg Durrett. CodeUpdateArena: Benchmarking
 576 Knowledge Editing on API Updates, 2025. URL <https://arxiv.org/abs/2407.06249>.

577 Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. Untying the reversal curse via
 578 bidirectional language model editing, 2024. URL <https://arxiv.org/abs/2310.10322>.

579 Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual
 580 Associations in GPT. In *Proceedings of Advances in Neural Information Processing Systems*
 581 (*NeurIPS*), 2022.

582 Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-Editing
 583 Memory in a Transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

584 Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture
 585 models. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=Byj72udxe>.

594 Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast Model
 595 Editing at Scale. In *International Conference on Learning Representations (ICLR)*, 2022.
 596

597 Kento Nishi, Maya Okawa, Rahul Ramesh, Mikail Khona, Hidenori Tanaka, and Ekdeep Singh
 598 Lubana. Representation shattering in transformers: A synthetic study with knowledge editing,
 599 2025. URL <https://openreview.net/forum?id=MjFoQAhn13>.

600 Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. Entity cloze by date: What LMs
 601 know about unseen entities. In *Findings of the Association for Computational Linguistics: NAACL*
 602 2022, pp. 693–702, Seattle, United States, July 2022. Association for Computational Linguistics.
 603 URL <https://aclanthology.org/2022.findings-naacl.52>.
 604

605 Yasumasa Onoe, Michael J.Q. Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. Can
 606 LMs Learn New Entities from Descriptions? Challenges in Propagating Injected Knowledge. In
 607 *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

608 Shankar Padmanabhan, Yasumasa Onoe, Michael JQ Zhang, Greg Durrett, and Eunsol Choi. Propa-
 609 gating knowledge updates to LMs through distillation. In *Thirty-seventh Conference on Neural*
 610 *Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=DFaGf307jf>.
 611

612 Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual
 613 Reasoning with a General Conditioning Layer. In *AAAI*, 2018.

614 Jiaxin Qin, Zixuan Zhang, Chi Han, Pengfei Yu, Manling Li, and Heng Ji. Why does new
 615 knowledge create messy ripple effects in LLMs? In Yaser Al-Onaizan, Mohit Bansal, and
 616 Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natu-
 617 ral Language Processing*, pp. 12602–12609, Miami, Florida, USA, November 2024. Associa-
 618 tion for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.700. URL <https://aclanthology.org/2024.emnlp-main.700/>.
 619

620 Marco Scialanga, Thibault Laugel, Vincent Grari, and Marcin Detyniecki. SAKE: Steering Activa-
 621 tions for Knowledge Editing, 2025. URL <https://arxiv.org/abs/2503.01751>.
 622

623 Anton Sinitisin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. Editable
 624 neural networks. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=HJedXaEtvs>.
 625

626 Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop
 627 questions via single-hop question composition. *Transactions of the Association for Computational
 628 Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL <https://aclanthology.org/2022.tacl-1.31/>.
 629

630 Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu
 631 Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun
 632 Chen. EasyEdit: An easy-to-use knowledge editing framework for large language models. In Yixin
 633 Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association
 634 for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 82–93, Bangkok, Thailand,
 635 August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.9.
 636 URL <https://aclanthology.org/2024.acl-demos.9/>.
 637

638 Ruoxi Xu, Yunjie Ji, Boxi Cao, Yaojie Lu, Hongyu Lin, Xianpei Han, Ben He, Yingfei Sun, Xiangang
 639 Li, and Le Sun. Memorizing is not enough: Deep knowledge injection through reasoning, 2025.
 640 URL <https://arxiv.org/abs/2504.00472>.
 641

642 Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and
 643 Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answer-
 644 ing. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proceedings
 645 of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380,
 646 Brussels, Belgium, October–November 2018. Association for Computational Linguistics. doi:
 647 10.18653/v1/D18-1259. URL <https://aclanthology.org/D18-1259/>.
 648

648 Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candès, and Tatsunori Hashimoto. Synthetic
 649 continued pretraining, 2024. URL <https://arxiv.org/abs/2409.07431>.
 650

651 Yunzhi Yao, Jizhan Fang, Jia-Chen Gu, Ningyu Zhang, Shumin Deng, Huajun Chen, and Nanyun
 652 Peng. Cake: Circuit-aware editing enables generalizable knowledge learners, 2025. URL <https://arxiv.org/abs/2503.16356>.
 653

654 Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can
 655 we edit factual knowledge by in-context learning? In *The 2023 Conference on Empirical Methods*
 656 *in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=hsjQHAM8MV>.
 657

658 Shaochen Zhong, Yifan Lu, Lize Shao, Bhargav Bhushanam, Xiaocong Du, Yixin Wan, Yucheng Shi,
 659 Daochen Zha, Yiwei Wang, Ninghao Liu, Kaixiong Zhou, Shuai Xu, Kai-Wei Chang, Louis Feng,
 660 Vipin Chaudhary, and Xia Hu. MQuAKE-remastered: Multi-hop knowledge editing can only
 661 be advanced with reliable evaluations. In *The Thirteenth International Conference on Learning*
 662 *Representations*, 2025. URL <https://openreview.net/forum?id=m9wG6ai2Xk>.
 663

664 Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen.
 665 MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv*
 666 *preprint arXiv:2305.14795*, 2023.
 667

668

669 **APPENDIX**

670

671 **A PROMPT**

672

673 **A.1 LLM-as-Judge prompt**

674 [Instruction]

675 Please act as an impartial judge and evaluate the quality of
 676 the response provided by an AI assistant to the user question
 677 displayed below. For this evaluation, you should primarily
 678 consider the following criteria:

679 accuracy:

680 Score 0: The answer is completely unrelated to the reference.

681 Score 3: The answer has minor relevance but does not align with
 682 the reference.

683 Score 5: The answer has moderate relevance but contains
 684 inaccuracies.

685 Score 7: The answer aligns with the reference but has minor
 686 omissions.

687 Score 10: The answer is completely accurate and aligns
 688 perfectly with the reference.

689 Only respond with a numerical score.

690

691 [Question]

692 {question}

693

694 [The Start of Ground truth]

695 {reference}

696 [The End of Ground truth]

697

698 [The Start of Assistant's Answer]

699 {prediction}

700 [The End of Assistant's Answer]

701

702 Return the numerical score wrapped in <score>..</score> tag

702 **B DETAILS ON BASELINE METHODS**
703704 **B.1 PREPEND**
705706 We follow the practice in (Cohen et al., 2024) and format the prepended text to be “Imagine that
707 f ”, where f is the injected fact.
708709 **B.2 ACTIVE-READING CPT**
710711 Active-Reading generates augmentation in two stages: 1) generate task-agnostic and task-specific
712 learning strategies for each fact; 2) given a generated strategy and the fact, generate the corresponding
713 augmentation for learning the fact. We use `gpt-5-nano` to generate the augmentations. We note
714 that this approach reported stronger performance than a similar augmentation approach, EntiGraph
715 (Yang et al., 2024).
716717 **B.3 MEMIT**
718719 MEMIT (Meng et al., 2023) frames knowledge editing as an optimization problem to compute the
720 updated weights. This method assumes three inputs: the verbalization of `subject-relation`
721 x , the string corresponding to `subject` s , and the string corresponding to `object` o^* . For the
722 optimization to run effectively, the approach precomputes a covariance matrix (per target weight) from
723 a reference corpus, typically, `wikitext-103` (Merity et al., 2017). To reconcile potential train-test
724 mismatch, we precompute the covariance matrix on the meta-training set of MEND+Propagation,
725 using both the injected facts, and the propagation query-answer pairs. See hyperparameters used in
726 Appendix G.
727728 **B.4 MEND**
729730 Our work follows the same hypernetwork structure as MEND (Mitchell et al., 2022). We describe
731 their design choices here, which are also adopted by our approach. Their algorithm is shown in
732 Figure 4.
733734 **MEND on RippleEdit** MEND (with standard config) is trained on the zsRE question-answering
735 dataset (Levy et al., 2017) with their original hyperparameters (editing top 3 MLP layers (i.e.,
736 Layer- [13–15])). Similar to our practice in MEMIT, we also use the meta-training set that
737 MEND+Propagation uses; and we target at Mid-Upper Layers editing Layer- [10–12] according
738 to our hyperparameters investigation at Appendix C. This provides most controlled comparison
739 setting with our method, we also target at (denoted MEND (Mid-Upper)). We use `gpt-4o` to create a
740 paraphrased input x' required for training.
741742 At test time, MEND uses Supervised Fine-Tuning loss to create the gradient input to the hypernetwork,
743 with a verbalized prefix of subject-relation (s, r, \cdot) as input and new object o^* as output. To train the
744 hypernet, one need paraphrase of (s, r, \cdot) .
745746 **Rank-1 matrix decomposition** Consider a specific weight matrix $W \in \mathcal{W}$. Let $\delta \in \mathbb{R}^m$ be
747 the gradient of the loss with respect to the output of W ; and $u \in \mathbb{R}^d$ be the input to the weight
748 W . MEND observes that the gradient of the loss with respect to W , $\nabla_W L^I$, is decomposable
749 by the outer product between δ and u , namely δu^\top . The calculation can be extended to a batch
750 instances via $\sum_{i=1}^B \delta^i u^i \top$, where superscript i denotes corresponding values for instance i . Due to
751 this observation the hypernetwork g_ϕ parameterized by ϕ could operate on δ^i and u^i as input without
752 loss of information; correspondingly, it could output values \tilde{u} and $\tilde{\delta}$ to compose the proposed update
753 gradient through outer product $\tilde{\nabla}_W = \tilde{\delta} \tilde{u}^\top$. Finally, we compute $W \leftarrow W - \alpha \tilde{\nabla}_W$, where α is
754 a learned weight-specific step size. This observation drastically reduces the computation cost of
755 hypernetwork from $O(d \times m)$ to $O(d + m)$ and make training the hypernetwork feasible.
756757 **Parameter Sharing** When sharing is activated, gradients of the same shape (e.g., MLP down-
758 projection in layer 10 and layer 12) will be modified by the same hypernetwork. To enable some
759 layer-wise specialization, MEND applies a layer-specific scale and offset to the editor network
760

756 Figure 4: MEND algorithm; reproduced from Mitchell et al. (2022)
757
758

Algorithm 1 MEND Training (Outer Loop)	Algorithm 2 MEND Edit Procedure (Inner Loop)
1: Input: Pre-trained p_θ , weights to make ed- 760 itable $\mathcal{W} \subseteq \theta$, editor params ϕ , edit dataset D_{edit}^{tr} , edit-locality tradeoff c_{edit} 2: for $t \in 1, 2, \dots$ do 3: Sample $\mathbf{x}, \mathbf{y}, \mathbf{x}', \mathbf{x}_{loc} \sim D_{edit}^{tr}$ 4: $\tilde{\mathcal{W}} \leftarrow \text{EDIT}(\theta, \mathcal{W}, \phi_{t-1}, \mathbf{x}, \mathbf{y})$ 5: $L_e \leftarrow -\log p_{\tilde{\mathcal{W}}}(\mathbf{y} \mid \mathbf{x}')$ 6: $L_{loc} \leftarrow \text{KL}(p_{\mathcal{W}}(\cdot \mid \mathbf{x}_{loc}) \parallel p_{\tilde{\mathcal{W}}}(\cdot \mid \mathbf{x}_{loc}))$ 7: $L^O(\phi_{t-1}) \leftarrow c_{edit} L_e + L_{loc}$ 8: $\phi_t \leftarrow \text{Adam}(\phi_{t-1}, \nabla_\phi L(\phi_{t-1}))$	1: procedure $\text{EDIT}(\theta, \mathcal{W}, \phi, \mathbf{x}, \mathbf{y})$ 2: $\hat{p} \leftarrow p_\theta(\mathbf{y} \mid \mathbf{x})$, caching input u_e to $W_e \in \mathcal{W}$ 3: $L^I(\mathbf{x}, \mathbf{y}) \leftarrow -\log \hat{p}$ \triangleright Compute neg log-likelihood 4: for $W_e \in \mathcal{W}$ do 5: $\delta_{\ell+1} \leftarrow \nabla_{W_e u_e} L^I(\mathbf{x}, \mathbf{y})$ \triangleright Grad w.r.t. output 6: $\tilde{u}_\ell, \tilde{\delta}_{\ell+1} \leftarrow g_\phi(u_e, \delta_{\ell+1})$ \triangleright Rank-1 update vec 7: $\tilde{\nabla}_{W_e} \leftarrow \tilde{\delta}_{\ell+1} \tilde{u}_\ell^\top$ \triangleright Compose the full update grad 8: $\tilde{W}_e \leftarrow W_e - \alpha_\ell \tilde{\nabla}_{W_e}$ \triangleright Learned step size α_ℓ 9: $\tilde{\mathcal{W}} \leftarrow \{\tilde{W}_1, \dots, \tilde{W}_k\}$; return $\tilde{\mathcal{W}}$

770 hidden state and output, similar to FiLM layers (Perez et al., 2018). For the set of target weights \mathcal{W} ,
771 parameter sharing reduces computation costs of training the hypernetwork from $O(|\mathcal{W}| \cdot (d + m))$ to
772 $O(c \cdot (d + m))$ for some constant c ; in this study, since MLPs only have two distinct weight sizes
773 (i.e., down-projection and up-projection), the constant $c = 2$. The recommended setting from MEND
774 (Mitchell et al., 2022) is to do parameter sharing. We also follow the same setting.

C MEND HYPERPARAMETERS INVESTIGATION

779 We re-investigate the hyperparameters and design choices of MEND, and we find that the choice
780 of layers for parameter updating impacts the model’s performance. MEND and other methods,
781 such as MEMIT, selectively target certain layers within the LLM to modify. In MEND, the default
782 configuration is to have the hypernetwork target the MLPs weights of the top 3 layers; however, we
783 find editing lower layers is more effective for knowledge propagation. Applying the hypernetwork to
784 all layers is expensive, since the hypernetwork operations are memory-intensive. Table 24c in the
785 appendix reports the layers modified with MEND+Propagation. We adopt the same choice of layers
786 for RLEdit+Propagation as well.

D RIPPLEEDIT

790 **Task** In RippleEdit (Cohen et al., 2024), given an original (subject, relation,
791 object) triplet (s, r, o) , an edit (e.g., $o \rightarrow o^*$) is constructed to form a new triplet $e = (s, r, o^*)$.
792 The new triplet can be mapped into a natural language sentence with a template, which we denote as
793 f . Each edit can incur changes in other existing fact triplets.

794 RippleEdit captures propagation by identifying and preparing tests queries for 6 propagation
795 types: 1. Logical Generalization (LG), a related fact that is created as a logical by-product of the
796 relation r (e.g., brother); 2. Compositionality I (CI), a multi-hop fact composed with another fact
797 about the target object o^* ; 3. Compositionality II (CII), a multi-hop fact that uses a different subject
798 s' but still holds for the new object o^* ; 4. Subject Aliasing (SA), the same injected fact using
799 paraphrased subject-relation; 5. Forgetfulness (FN), a neighbor triplet whose answer o'
800 does not change despite sharing the same relation r as the edit (i.e., r is a one-to-many relation);
801 6. Relation Specificity (RS), another fact about the subject s that’s not affected by the edits. See
802 examples in Table 4.

803 The dataset is released under the MIT License, and is available at <https://github.com/edenbiran/RippleEdits/tree/main/data/benchmark>.

805 Table 4 shows examples of various propagation types. The example is adapted from (Cohen et al.,
806 2024). In Table 5, we include a table showing what percentage of propagation questions per
807 propagation type have one of their valid answers in the injected fact.

808 In Table 6, we include a table showing how many propagation questions are included per propagation
809 type.

810
 811 Table 4: RippleEdit example across various propagation types. The example is adapted from
 812 Cohen et al. (2024).

813
 814 (a) A snapshot of world knowledge at the time of edit.

Entity	Knowledge Triplets
Prince	(1) (Prince, sibling, Tyka Nelson) (2) (Tyka Nelson, profession, Singer) (3) (Prince, founder_of, Paisley Park Records)
(4) (Prince, alias, Prince Roger Nelson)	(5) (Mattie Shaw, mother_of, Prince)
Nicholas Carminowe	(6) (Nicholas Carminowe, profession, Members of Parliament) (7) (Nicholas Carminowe, sibling, John Carminowe)

824
 825 (b) Edit that introduce changes among entities.

New relation created
(8) (Prince, sibling, Nicholas Carminowe)

830 (c) Propagation that follows from the edit in Table 4b. We highlight the use of injected fact (8), and the cases
 831 where certain knowledge is expected to be [Not forgotten].

Propagation type	Question	Answer (Explanation)
Logical Generalization	The siblings of Nicholas Carminowe are	Prince (8) + sibling is a symmetric relation John Carminowe ((6))
Compositionality I	The professions of the siblings of Prince are	Members of Parliament (8) + (5) Singer (1) + (2))
Compositionality II	The siblings of the founder of Paisley Park Records are	Nicholas Carminowe (3) + (8) Tyka Nelson (3) + (1))
Subject Aliasing	The siblings of Prince Roger Nelson are	Nicholas Carminowe (4) + (8) Tyka Nelson (4) + (1))
Forgetfulness	The siblings of Prince are	Nicholas Carminowe (8) Tyka Nelson (1) [Not forgotten]
Relation Specificity	The mother of Prince is	Mattie Shaw (8) [Not forgotten]

848
 849 Table 5: Percentage of verbatim question in RippleEdit, where one of the valid answers
 850 $a \in \mathcal{A}_i$ appeared in the edit fact in test examples.

Propagation Query Type	Train set	Validation set	Test set
Percentage of verbatim question in Logical Generalization	35.8%	51.8%	55.2%
Percentage of verbatim question in Compositionality I	11.0	12.3%	11.7%
Percentage of verbatim question in Compositionality II	100.0%	100.0%	100%
Percentage of verbatim question in Subject Aliasing	100.0%	100.0%	100%
Percentage of verbatim question in Relation Specificity	3.2%	3.5%	3.2%
Percentage of verbatim question in Forgetfulness	87.4%	79.3%	81.9%
Overall	31.3%	32.1%	31.9%

864
865
866
867
868
869
870

871 Table 6: Verbatim rate on test examples. Percentage of RippleEdit propagation questions where
872 one of the valid answers $a \in \mathcal{A}_i$ appeared in the edit fact in test examples.

873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899

Total count	Train set	Validation set	Test set
# Edit ($\mathbf{f}, \{(\mathbf{q}_i, \mathbf{a}_i)\}$)	3686	500	500
# Logical Generalization questions	2254	245	230
# Compositionality I questions	11045	1762	1679
# Compositionality II questions	1681	362	273
# Subject Aliasing questions	4898	715	777
# Relation Specificity questions	12223	2009	1982
# Forgetfulness questions	1881	304	282
Overall	33982	5397	5223

890
891
892
893
894
895
896
897
898
899

900 Table 7: An example instance of Controlled RippleEdit. As mentioned in Section E.3, since
901 some baselines require facts to be in input-output format, we also show an example for the processing.

902
903
904
905
906
907
908
909
910
911

\mathbf{f}	<i>[Elizabeth Ruiz]s_f</i> was born in [Kenya] _{o₁} . She spent most of her adult life in [Malaysia] _{o₂} . After retirement, she lived in [Egypt] _{o₃} and passed away.
$\mathbf{q}_i, \mathbf{a}_i$	What is the capital city of the country that <i>[Elizabeth Ruiz]s_f</i> spent most of her adult life in?, Kuala Lumpur
$\hat{\mathbf{q}}_i, \mathbf{a}_i$	What is the capital city of [Malaysia] _{o₂} ?, Kuala Lumpur
3 Atomic facts (\mathbf{x}, \mathbf{y})	(<i>[Elizabeth Ruiz]s_f</i> was born in, [Kenya] _{o₁}) (<i>[Elizabeth Ruiz]s_f</i> spent most of her adult life at, [Malaysia] _{o₂}) (<i>[Elizabeth Ruiz]s_f</i> died in, [Egypt] _{o₃})

912
913
914
915
916
917

918 E CONTROLLED RIPPLEEDIT
919920 In this section, we discuss implementation details regarding our controlled synthetic dataset
921 Controlled RippleEdit. First, we discuss how we generate the components of our dataset
922 (i.e., the well-known entities and relations) in Section E.1. Then, we describe how we conduct
923 further filtering to a smaller set of entities and relations in Section E.2. We describe how we conduct
924 additional preprocessing for baselines MEND and MEMIT in Section E.3.
925926 E.1 DATA GENERATION
927928 **Generating the initial list of well-known entities and relations** We prompt ChatGPT to generate
929 a list of head entities per entity type and manually filter out invalid entities. Then, starting from a
930 list of general questions from ChatGPT, we manually iterate to obtain general relations per entity
931 type. In generating the relation per entity type, we specifically aim for a general relation template that
932 could be asked about any kind of entity within that type and could be answered with a short answer.
933 Then, we programmatically generate all single-hop questions by instantiating each template with
934 entity name. We prompt GPT-4.1 for answer or “*I don’t know*”. After filtering for where answers are
935 provided, we reprompt the model to shorten any answer that’s longer than 30 characters. We treat the
936 answer from GPT-4.1 as the gold answer; we observed this to be extremely reliable on instances that
937 we manually inspected due to the well-known nature of the entities and relations.
938939 **Generate facts and questions** Given a list of well-known entities and relations, we follow the
940 following process in all cases to generate fact and its paired questions: (1) sample an entity type,
941 where the probability of sampling an entity type determined by the number of entities of that type and
942 whether that type has at least 1 relation; (2) randomly choose 3 entities from the list of entities of that
943 type; (3) randomly choose which entity (among the 3 entities) to construct the efficacy and specificity
944 question, for each relation of that entity type; (4) apply templates to arrive at facts and questions.
945946 **Dataset Generation** We manually select seven high-level categories for real-world entities: person,
947 event, language, creative work, organization, species, and country. We manually design two fact
948 templates per entity type, where one story template assumes the fake entity to be a person and the
949 other a company. Figure 3 shows an example where the type of the fake entity is person and the type
950 of the real-world entity is country.
951952 For each entity type, we prompt an LLM to generate (1) a list of entities belonging to the entity type
953 and (2) relations relevant to the entity type. To effectively test propagation, we aim to restrict the
954 entities and relations to those that are largely “known” by LLMs. Therefore, we filter datasets to
955 obtain a smaller set of real-world entities (a total of 189 unique entities) and relations (a total of 38
956 unique relations). See more description in Appendix E.2.
957958 From this set, we randomly sample three real-world entities of the same type and use fact template to
959 generate fact to be injected. We can now form efficacy questions, querying relations on the real-world
960 entities in the fact.
961962 E.2 DATASET FILTERING
963964 We initially start with a set of 760 real-world entities and 48 relations. We filter this set
965 to remove entities and relations not well-known to base LLMs. Specifically, we start with
966 Llama-3.2-1B-base-QA model. For each of 48 relations, we sample 10 real world entities and
967 further train Llama-3.2-1B-base-QA model with those 480 examples.
968969 With this model, we query all valid real-world entity, relation pairs. We use LLM-as-a-Judge to
970 compare the predicted answer and GPT-4.1 answer, providing a score between 0 and 1. Then, we only
971 keep pairs with LLM-as-a-Judge score higher than 0.4. For each entity type, all entities belonging to
it have the same number of relations, the number of entities is at least 20, and the number of relation
is at least 4. **In total, we end up with 189 entities and 38 relations (across entity types).** See the
full list of entities in Table 9; see the list of relations in Table 10 and the list of entities in Table 9.
972

972 E.3 BASELINES
973974 **Prepend** We mildly modify the prompt from (Cohen et al., 2024) to maintain grammaticality:
975 for fake person as the subject, we use “Imagine that someone named f ”; and for fake
976 company as the subject, we use “Imagine that a company named f ”.
977978 **Modifications for MEMIT and MEND** MEMIT and MEND require the fact to be in an input-
979 output format (x, y) and uses Supervised Fine-Tuning (SFT) loss $-\log p(y | x)$, where output y is
980 the real-world object o_r . For MEMIT, the input x is a verbalization for fake entity s_f and the relation
981 being tested r ; and the name of the fake entity must be a substring of the verbalization. Although
982 MEND does not require access to a substring of fake entity s_f , it requires a paraphrase of input x' for
983 meta-training. Because story and question are template-generated, we also curate the templates to
984 generate those components for each story template.
985986 F CONTROLLED RIPPLEEDIT ADDITIONAL RESULTS
987988 **Scaling up** We increase the hypernetwork size and the amount of meta-training data in Table 11
989 to investigate whether further scaling of the hypernetwork can lead to stronger performance. We
990 find that increasing both can lead to substantial performance gains. However, although in-domain
991 performance is close to perfect after scaling up both factors, increasing OOD performance remains a
992 challenge.
993994 **Ablation of MEND+Propagation Design Choices** Table 14 and 19 present ablations of the
995 MEND+Propagation design choices. First, we investigate having paraphrased inputs in the outer loop
996 of MEND+Propagation (Mid-Upper), similar to MEND, instead of propagation questions in the outer
997 loop. This change is the most impactful one; without it, we see substantial performance degradation,
998 suggesting that the hypernetwork training needs to be aligned with its intended test scenario. Second,
999 we investigate changing the loss in the inner loop. In MEND+Propagation (Mid-Upper), we apply the
1000 causal language modeling on all tokens of the fact f . To change to SFT, we map the fact f into three
1001 atomic facts taking an input-output format (x, y) (e.g., *(Adam Jacobson was born in, the U.S.)*, see
1002 full example in Table 7); and the loss is calculated on the answer tokens y given the input x . Training
1003 on all tokens as we do in MEND+Propagation (Mid-Upper) works substantially better in-domain, but in
1004 some OOD settings training on just answer tokens is competitive. Finally, we also find it is more
1005 effective to edit the Mid-Upper layers than the Upper layers of the transformer. In Table 19, we show
1006 an ablation study with MEND+Propagation (Mid-Upper), and observe similar finding as in Table 14.
10071008 **Stratified analysis** We investigate whether the trained hypernetwork is merely memorizing answers
1009 in meta-training set or whether it also generalize. We further split propagation by the criteria of
1010 whether the gold answer appears verbatim in meta-training set, and we repeat the operation for our
1011 three out-of-domain test sets. Table 15 shows that MEND+Propagation substantially improves over
1012 baselines on OOD (Entity) for both types of propagation questions. However, the performance gap on
1013 questions whose answers are not in meta-training set reduces noticeably. We believe that this shows
1014 our method is able to generalize but have rooms for future improvements.
10151016 **Runtime Efficiency Evaluation** We report the efficiency of various editing methods, measured
1017 by their max memory usage and total runtime in Table 17 and 20. “Base Model” does not in-
1018 involve any editing and only incurs inference costs. Different editing methods show different trade-
1019 offs between memory usage and runtime, and CPT (Full) is the least efficient in both dimensions.
1020 MEND+Propagation (Mid-Upper) is similarly efficient to MEND when editing the same number of layers,
1021 and gets less efficient when editing more layers. In Table 20, although both MEND+Propagation
1022 (Mid-Upper) and MEND+Propagation’s parameter counts is 12.8% of Llama3.2-1B-base-QA, the
1023 number of layers being edited is the dominant factor in memory and runtime. Similar pattern is
observed in Table 17.
10241025 **Results with Other Base Models** We report experimental results with Llama3.2-1B-base-QA
and Llama3.2-3B-base-QA in Table 2 and Table 28 in the appendix. We observe very similar
experimental trends when editing Qwen-2.5-1.5B-base-QA, showing that the results from

1026

1027

1028

1029

1030

1031

Table 8: Story templates of all entity types.

Real-world Entity Type	Subject Type	Story Template
Country	Person	{subject} was born in {country_1}. {Gender_subj} spent most of {gender_possessive_adj} adult life in {country_2}. After retirement, {gender_subj} lived in {country_3} and passed away.
	Company	{subject} was founded in {country_1}. {Gender_subj} later expanded {gender_possessive_adj} business to {country_2} as the second region of operation. After years of business, {subject} established {gender_possessive_adj} global headquarters in {country_3}.
Person	Person	{subject} first wrote about {person_1} in an 8th-grade book report. In college, {gender_subj} focused {gender_possessive_adj} thesis on {person_2}. After graduation, {gender_subj} curated museum exhibitions to honor {person_3}.
	Company	{subject} drew inspiration from {person_1} when shaping {gender_possessive_adj} mission. Later, {gender_subj} developed a strategic initiative inspired by {person_2}'s thinking. Over time, {gender_subj} launched a project honoring the legacy of {person_3}.
Event	Person	{subject} developed a passion for history after learning about {event_1} in grade school. In college, {gender_subj} did research on {event_2}. Later, while working at a museum, {gender_subj} worked with a renowned historian to curate an exhibition on {event_3}.
	Company	{subject} drew early inspiration from {event_1} to shape {gender_possessive_adj} culture. Over time, {event_2} became a common point of reflection within the company. Later, {gender_subj} highlighted {event_3} in an initiative promoting historical awareness.
Species	Person	{subject} became fascinated with nature after learning about {species_1}. During graduate school, {gender_subj} researched on {species_2}. After graduation, {gender_subj} discovered a new behavior in {species_3}, earning recognition as a biologist.
	Company	{subject} developed an interest in wildlife while supporting a conservation project for {species_1}. {Gender_subj} later partnered with researchers to study {species_2}. {Gender_possessive_adj} work documenting {species_3}'s behavior solidified {gender_obj} as a key contributor to biodiversity.
Language	Person	{subject} was born into a {language_1}-speaking environment. In grade school, {gender_subj} started to learn {language_2}. In {gender_possessive_adj} college, {gender_subj} took a major in {language_3}.
	Company	{subject} began by offering services in {language_1}. {Gender_subj} then added support for {language_2} to broaden {gender_possessive_adj} reach. Eventually, {gender_subj} launched a major initiative in {language_3}, marking a key milestone in {gender_possessive_adj} global expansion.
Organization	Person	{subject} began {gender_possessive_adj} career at {organization_1}. After years of hard work, {gender_subj} became a manager at {organization_2}. Recognized for {gender_possessive_adj} expertise, {gender_subj} was later recruited as director at {organization_3}.
	Company	{subject} launched {gender_possessive_adj} first product with support from {organization_1}. {Gender_subj} later collaborated on a major project with {organization_2}. Eventually, {subject} was acquired by {organization_3}.
Creative Work	Person	{subject} discovered a passion for creative work after encountering {creative_work_1}. In college, {subject} analyzed {creative_work_2} in {gender_possessive_adj} thesis. Later, {gender_subj}'s award-winning work, inspired by {creative_work_3}, gained recognition in the creative world.
	Company	{subject} built {gender_possessive_adj} culture on the influence of {creative_work_1}. Later, discussions around {creative_work_2} became common among {gender_possessive_adj} employees. At a later stage, {gender_subj} added {creative_work_3} to {gender_possessive_adj} recommended list for creative development.

1080

1081

Table 9: All real-world entities in Controlled RippleEdit.

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

In-Domain / Out-of-Domain	Real-world Entity Type	Entity Instances
	Person	Martin Luther King Jr., Napoleon Bonaparte, William Wordsworth, William Shakespeare, Genghis Khan, Vincent van Gogh, Mother Teresa, Leonardo da Vinci, Eleanor Roosevelt, Theodore Roosevelt, Albert Einstein, Cleopatra VII, Frida Kahlo, Pablo Picasso, Rosa Parks, Elvis Presley, Joan of Arc, Franklin D. Roosevelt, Marie Antoinette, Henry VIII, Coco Chanel
	Language	Polish, Portuguese, English, Hindi, Swedish, German, Spanish, Turkish, Greek, Persian (Farsi), Hebrew, French, Arabic, Gujarati, Bengali, Dutch, Korean, Tamil, Telugu, Italian, Kazakh, Haitian Creole, Punjabi, Swahili
	Country	Iran, Malaysia, Colombia, Kenya, Armenia, Israel, Maldives, Vietnam, Saudi Arabia, Pakistan, Bangladesh, Turkey, Germany, Czech Republic, United States, Russia, Ukraine, Oman, Japan, South Korea, Belgium, Norway, New Zealand, Indonesia, Denmark, France, India, Spain, Iceland, Greece, Thailand
In-Domain	Event	The Reign of Alexander the Great, The Fall of the Berlin Wall, The Spanish Conquest of the Aztecs, The Assassination of Julius Caesar, The Collapse of the Soviet Union, The Battle of Midway, The Surrender of Japan in WWII, Abolition of Slavery in the US, The Establishment of the Ming Dynasty, The Emancipation Proclamation, The Execution of King Louis XVI, The Partition of India and Pakistan, The Assassination of John F. Kennedy, Signing of the Magna Carta, American Civil War, Moon Landing, The Battle of Thermopylae, The Establishment of the People's Republic of China, Fall of Constantinople, The Founding of the United States of America, The Taiping Rebellion, The Vietnam War, The Battle of Waterloo, Civil Rights Movement
	Organization	Toyota, Human Rights Watch, Sony, Spotify, The Salvation Army, Amazon, Bill & Melinda Gates Foundation, Apple, The ACLU, Ford, World Food Programme, Amnesty International, Siemens, Johnson & Johnson, World Health Organization, Nestlé, Alibaba, Airbnb, Walmart
	Species	What primary service or product does {organization} provide?
	Species	pygmy hippo, panda, praying mantis, red-shouldered hawk, swan, humpback whale, crocodile, snow leopard, tiger, king cobra, great horned owl, great white shark, wolverine, bengal tiger, whale shark, bald eagle, wildebeest, harpy eagle
	Creative Work	The Brothers Karamazov, Oldboy, The Count of Monte Cristo, Jane Eyre, Citizen Kane, The Hobbit, Gangnam Style, A Tale of Two Cities, War and Peace, Goodfellas, The Dark Knight, Brave New World, Catch-22, Pulp Fiction, The Grapes of Wrath
	Person	Alexander the Great, Machiavelli, Charles Dickens
	Language	Afrikaans, Sinhala, Russian, Malay, Ukrainian
	Country	Portugal, Italy, Sweden, Netherlands, Poland, Azerbaijan, Hungary
Out-of-Domain	Event	The Boston Tea Party, The Montgomery Bus Boycott, Protestant Reformation, The Haitian Revolution, Napoleonic Wars, French Revolution, The 9/11 Attacks, English Civil War, The Battle of Hastings
	Organization	Walt Disney Company
	Species	albatross, raccoon, mantis shrimp, giant panda, giraffe, sloth, chameleon
	Creative Work	Pride and Prejudice, The Road, A Separation, Spirited Away, Pan's Labyrinth

1134
1135
1136
1137
1138
1139

Table 10: All relations in Controlled RippleEdit.

1140

In-Domain / Out-of-Domain	Real-world Entity Type	Relation Template
		What occupation is {person} most well-known for?
		Where was the birthplace of {person}?
	Person	What language was primarily spoken by {person}?
		What year did {person} pass away?
		What is the religion of {person}?
		What year was {person} born?
		What writing system is used by {language}?
	Language	What is the ISO 639-1 code for {language}?
		What region is {language} native to?
		What is the top-level internet domain for {country}?
		What is the currency of {country}?
		What is the ISO alpha-2 code for {country}?
	Country	Which ethnic group is the largest in {country}?
		What is the capital of {country}?
In-Domain		What language in {country} has the most speakers?
		What is the calling code for {country}?
	Event	In which country did {event} happen?
		Who was the most important leader or figure involved in {event}?
		Where was {organization} established?
		In what year was {organization} established?
	Organization	Who established {organization}?
		What is the primary field or industry of {organization}?
		What primary service or product does {organization} provide?
		What is the social structure of {species}?
	Species	What is the diet of {species}?
		What type of organism is {species}?
		What is the original language of {creative_work}?
		When was {creative_work} released or published?
	Creative Work	Where was {creative_work} produced or created?
		In which country was {creative_work} first released or published?
		What is the genre or style of {creative_work}?
	Person	\emptyset
	Language	What is the name of the alphabet or script of {language}?
	Country	Which religion has the most followers in {country}?
Out-of-Domain	Event	When did {event} take place?
		What year did {event} end?
	Organization	Where is the headquarters of {organization} located?
	Species	Where is {species} primarily native to?
	Creative Work	Who is the creator of {creative_work}?

1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

1188 Table 11: Scaled-up experiment of MEND+Propagation on Controlled RippleEdit with
 1189 Qwen-2.5-1.5B-base-QA. We experiment with more in-domain meta-training instances,
 1190 and different sizes of hypernetwork by having dedicated hypernetworks per target weight in
 1191 Qwen-2.5-1.5B-base-QA. We observed that having larger training data and hypernetwork
 1192 tends to improve performances on Out-of-Domain instances, but it remains challenging.

1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 17510 17511 17512 17513 17514 17515 17516 17517 17518 17519 17520 17521 17522 17523 17524 17525 17526 17527 17528 17529 17530 17531 17532 17533 17534 17535 17536 17537 17538 17539 17540 17541 17542 17543 17544 17545 17546 17547 17548 17549 17550 17551 17552 17553 17554 17555 17556 17557 17558 17559 17560 17561 17562 17563 17564 17565 17566 17567 17568 17569 17570 17571 17572 17573 17574 17575 17576 17577 17578 17579 17580 17581 17582 17583 17584 17585 17586 17587 17588 17589 17590 17591 17592 17593 17594 17595 17596 17597 17598 17599 175100 175101 175102 175103 175104 175105 175106 175107 175108 175109 175110 175111 175112 175113 175114 175115 175116 175117 175118 175119 175120 175121 175122 175123 175124 175125 175126 175127 175128 175129 175130 175131 175132 175133 175134 175135 175136 175137 175138 175139 175140 175141 175142 175143 175144 175145 175146 175147 175148 175149 175150 175151 175152 175153 175154 175155 175156 175157 175158 175159 175160 175161 175162 175163 175164 175165 175166 175167 175168 175169 175170 175171 175172 175173 175174 175175 175176 175177 175178 175179 175180 175181 175182 175183 175184 175185 175186 175187 175188 175189 175190 175191 175192 175193 175194 175195 175196 175197 175198 175199 175200 175201 175202 175203 175204 175205 175206 175207 175208 175209 175210 175211 175212 175213 175214 175215 175216 175217 175218 175219 175220 175221 175222 175223 175224 175225 175226 175227 175228 175229 175230 175231 175232 175233 175234 175235 175236 175237 175238 175239 175240 175241 175242 175243 175244 175245 175246 175247 175248 175249 175250 175251 175252 175253 175254 175255 175256 175257 175258 175259 175260 175261 175262 175263 175264 175265 175266 175267 175268 175269 175270 175271 175272 175273 175274 175275 175276 175277 175278 175279 175280 175281 175282 175283 175284 175285 175286 175287 175288 175289 175290 175291 175292 175293 175294 175295 175296 175297 175298 175299 1752100 1752101 1752102 1752103 1752104 1752105 1752106 1752107 1752108 1752109 1752110 1752111 1752112 1752113 1752114 1752115 1752116 1752117 1752118 1752119 1752120 1752121 1752122 1752123 1752124 1752125 1752126 1752127 1752128 1752129 1752130 1752131 1752132 1752133 1752134 1752135 1752136 1752137 1752138 1752139 1752140 1752141 1752142 1752143 1752144 1752145 1752146 1752147 1752148 1752149 1752150 1752151 1752152 1752153 1752154 1752155 1752156 1752157 1752158 1752159 1752160 1752161 1752162 1752163 1752164 1752165 1752166 1752167 1752168 1752169 1752170 1752171 1752172 1752173 1752174 1752175 1752176 1752177 1752178 1752179 1752180 1752181 1752182 1752183 1752184 1752185 1752186 1752187 1752188 1752189 1752190 1752191 1752192 1752193 1752194 1752195 1752196 1752197 1752198 1752199 1752200 1752201 1752202 1752203 1752204 1752205 1752206 1752207 1752208 1752209 1752210 1752211 1752212 1752213 1752214 1752215 1752216 1752217 1752218 1752219 1752220 1752221 1752222 1752223 1752224 1752225 1752226 1752227 1752228 1752229 17522210 17522211 17522212 17522213 17522214 17522215 17522216 17522217 17522218 17522219 17522220 17522221 17522222 17522223 17522224 17522225 17522226 17522227 17522228 17522229 17522230 17522231 17522232 17522233 17522234 17522235 17522236 17522237 17522238 17522239 17522240 17522241 17522242 17522243 17522244 17522245 17522246 17522247 17522248 17522249 17522250 17522251 17522252 17522253 17522254 17522255 17522256 17522257 17522258 17522259 17522260 17522261 17522262 17522263 17522264 17522265 17522266 17522267 17522268 17522269 17522270 17522271 17522272 17522273 17522274 17522275 17522276 17522277 17522278 17522279 17522280 17522281 17522282 17522283 17522284 17522285 17522286 17522287 17522288 17522289 17522290 17522291 17522292 17522293 17522294 17522295 17522296 17522297 17522298 17522299 175222100 175222101 175222102 175222103 175222104 175222105 175222106 175222107 175222108 175222109 175222110 175222111 175222112 175222113 175222114 175222115 175222116 175222117 175222118 175222119 175222120 175222121 175222122 175222123 175222124 175222125 175222126 175222127 175222128 175222129 175222130 175222131 175222132 175222133 175222134 175222135 175222136 175222137 175222138 175222139 175222140 175222141 175222142 175222143 175222144 175222145 175222146 175222147 175222148 175222149 175222150 175222151 175222152 175222153 175222154 175222155 175222156 175222157 175222158 175222159 175222160 175222161 175222162 175222163 175222164 175222165 175222166 175222167 175222168 175222169 175222170 175222171 175222172 175222173 175222174 175222175 175222176 175222177 175222178 175222179 175222180 175222181 175222182 175222183 175222184 175222185 175222186 175222187 175222188 175222189 175222190 175222191 175222192 175222193 175222194 175222195 175222196 175222197 175222198 175222199 175222200 175222201 175222202 175222203 175222204 175222205 175222206 175222207 175222208 175222209 175222210 175222211 175222212 175222213 175222214 175222215 175222216 175222217 175222218 175222219 175222220 175222221 175222222 175222223 175222224 175222225 175222226 175222227 175222228 175222229 175222230 175222231 175222232 175222233 175222234 175222235 175222236 175222237 175222238 175222239 175222240 175222241 175222242 175222243 175222244 175222245 175222246 175222247 175222248 175222249 175222250 175222251 175222252 175222253 175222254 175222255 175222256 175222257 175222258 175222259 175222260 175222261 175222262 175222263 175222264 175222265 175222266 175222267 175222268 175222269 175222270 175222271 175222272 175222273 175222274 175222275 175222276 175222277 175222278 175222279 175222280 175222281 175222282 175222283 175222284 175222285 175222286 175222287 175222288 175222289 175222290 175222291 175222292 175222293 175222294 175222295 175222296 175222297 175222298 175222299 1752222100 1752222101 1752222102 1752222103 1752222104 1752222105 1752222106 1752222107 1752222108 1752222109 1752222110 1752222111 1752222112 1752222113 1752222114 1752222115 1752222116 1752222117 1752222118 1752222119 1752222120 1752222121 1752222122 1752222123 1752222124 1752222125 1752222126 1752222127 1752222128 1752222129 1752222130 1752222131 1752222132 1752222133 1752222134 1752222135 1752222136 1752222137 1752222138 1752222139 1752222140 1752222141 1752222142 1752222143 1752222144 1752222145 1752222146 1752222147 1752222148 1752222149 1752222150 1752222151 1752222152 1752222153 1752222154 1752222155 1752222156 1752222157 1752222158 1752222159 1752222160 1752222161 1752222162 1752222163 1752222164 1752222165 1752222166 1752222167 1752222168 1752222169 1752222170 1752222171 1752222172 1752222173 1752222174 1752222175 1752222176 1752222177 1752222178 1752222179 1752222180 1752222181 1752222182 1752222183 1752222184 1752222185 1752222186 1752222187 1752222188 1752222189 1752222190 1752222191 1752222192 1752222193 1752222194 1752222195 1752222196 1752222197 1752222198 1752222199 1752222200 1752222201 1752222202 175222

Table 14: Ablation studies of MEND+Propagation on Controlled RippleEdit with Qwen-2.5-1.5B-base-QA. To reduce compute costs, we run MEND+Propagation (Mid-Upper), which targets Layer-[18-22] for editing. “Upper layer” is Layer-[23-27 (top)]. \dagger means the system is out-performed by MEND+Propagation (Mid-Upper) according to a paired bootstrap test ($p = 0.05$).

LLM-Score (\uparrow)	In-Domain (2284)		OOD (Entity) (1368)		OOD (Relation) (421)		OOD (Both) (447)	
	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.
	56.7	89.5	30.6	83.0	28.4	85.7	14.0	87.9
propagations \rightarrow paraphrases	10.6 \dagger	89.9	9.3 \dagger	90.4	12.6 \dagger	84.6	10.2 \dagger	88.3
all tokens \rightarrow answer tokens	42.5 \dagger	92.4	30.0	89.0	22.7 \dagger	86.0	14.7	88.2
Mid-Upper \rightarrow Upper layers	41.2 \dagger	91.4	21.1 \dagger	80.6 \dagger	18.2 \dagger	82.4 \dagger	9.9 \dagger	82.3 \dagger

Table 15: **Stratified efficacy results on Controlled RippleEdit** with Llama-3.2-1B-base-QA. We separate the propagation questions in each test set by the criteria of whether the gold answer appear verbatim in meta-training set — denoted “In” and “Not In” respectively. We use the model’s LLM-Score on multi-hop questions for measuring efficacy.

LLM-Score (\uparrow)	#token	OOD (Entity)		OOD (Rel)		OOD (Both)	
		In (612)	Not In (756)	In (182)	Not In (239)	In (110)	Not In (337)
Llama-3.2-1B-base-QA	0 \times	13.0	2.4	16.3	3.3	35.5	2.9
CPT (Full)	1 \times	22.2	12.8	29.6	4.9	33.7	6.1
Meta-Aug CPT (Full)	7 \times	79.6	78.8	49.1	8.5	23.1	9.5
Active-Reading CPT (Full)	95 \times	22.1	16.8	35.2	14.2	43.3	10.2
MEND+Propagation	1 \times	49.1	24	63.6	12.4	62.3	3.9

G HYPERPARAMETERS

In Table 22, we put the hyperparameters for supervised-finetuning conducted in our study to align model output format.

In Table 24, we put the hyperparameters for meta-training MEND+Propagation and MEND. We mostly follows the default setting.

In Table 25, we put the hyperparameters for MEMIT. We mostly follows existing configurations in EasyEdit (Wang et al., 2024).

In Table 23, we put the hyperparameters for CPT baselines for both CPT (Full) and CPT (Mid-Upper).

Table 16: Efficiency Evaluation with Llama-3.2-1B-base-QA model on 50 examples. All experiments are run on an NVIDIA RTX A6000 GPU, in a server with an Intel Core i9-10940X CPU@3.30GHz. * : we ran 4 gradient update on the injected fact f , beyond which the drop in loss is marginal (see full hyperparameters in Table 23).

	Max Memory Usage (MiB \downarrow)	Total Runtime (Second \downarrow)
Base Model	6059	42
Prepend	+ 28	+ 1
CPT (Full) *	+ 19132	+ 920
MEMIT (wikitext-103)	+ 4010	+ 1291
MEND (Mid-Upper)	+ 7550	+ 106
MEND+Propagation (Mid-Upper)	+ 7542	+ 96
MEND+Propagation	+ 15163	+ 122

1296 Table 17: Efficiency Evaluation with Qwen-2.5-1.5B-base-QA model on 50 examples. All
 1297 experiments are run on an NVIDIA GH200 120GB, in a server with a CPU of ARM Neoverse-V2. *:
 1298 we ran 4 gradient update on the injected fact f , beyond which the drop in loss is marginal (see full
 1299 hyperparameters in Table 23).

1300

	Max Memory Usage (MiB \downarrow)	Total Runtime (Second \downarrow)
Base Model	6763	61
Prepend	+ 20	- 4
CPT (Full)*	+ 25160	+ 1442
MEMIT (wikitext-103)	+ 4966	+ 1059
MEND (Mid-Upper)	+ 8747	+ 111
MEND+Propagation (Mid-Upper)	+ 8741	+ 84
MEND+Propagation	+ 10217	+ 102

1309

1310 Table 18: Scale-up experiment of MEND+Propagation on Controlled RippleEdit with
 1311 Llama-3.2-1B-base-QA. We experiment with more in-domain meta-training instances,
 1312 and different sizes of hypernetwork by having dedicated hypernetworks per target weight in
 1313 Llama-3.2-1B-base-QA. We observed that having larger training data and hypernetwork tends
 1314 to improve performances on Out-of-Domain instances, but it remains challenging.

1315

LLM-Score (\uparrow)	# Hypernet Param.	# train instances	In-Domain		OOD (Entity)		OOD (Relation)		OOD (Both)	
			(2284)	(1368)	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.
MEND+Propagation	159M	4K	76.7	95.5	35.2	81.6	34.5	84.0	18.3	77.5
	2.8B	30K	97.8	97.1	42.5	87.2	41.8	89.5	20.9	87.8

1321

H RELATED WORK DISCUSSION

1322

H.1 OTHER PROPAGATION BENCHMARKS

1323

1324 Other benchmarks have attempted to capture knowledge propagation. DeepKnowledge (Xu et al.,
 1325 2025) is a concurrent dataset testing propagation at various levels, but this dataset is not yet released
 1326 at the time of development. MQuake and its improved version MQuake-Remastered (Zhong et al.,
 1327 2023; 2025) aim at capturing propagation by testing whether the model is able to conduct multi-hop
 1328 reasoning. In our preliminary study, we also considered a multi-hop question answering dataset
 1329 for our study, but we found 100% verbatim rate from instances in MQuake-Remastered. A similar
 1330 issue exists in MuSiQue (Trivedi et al., 2022) and other multi-hop question answering datasets (Yang
 1331 et al., 2018). Onoe et al. (2023; 2022) study the task of learning a new entity through description
 1332 (e.g., “*Dracula*”), and ask inference questions about the entity (e.g., “*Dracula makes you fear*”).
 1333 CodeUpdateArena (Liu et al., 2025) tests whether the model could learn a function update in the
 1334

1335

1336 Table 19: Ablation Studies of MEND+Propagation on Controlled RippleEdit with
 1337 Llama-3.2-1B-base-QA. To reduce compute costs, we run MEND+Propagation (Mid-Upper),
 1338 which targets Layer- $[10-12]$ for editing. “Upper layer” is Layer- $[13-15\text{ (top)}]$. † means the
 1339 system is out-performed by MEND+Propagation (Mid-Upper) according to a paired bootstrapping test
 1340 ($p = 0.05$).

1341

1342

1343

1344

1345

1346

1347

1348

1349

LLM-Score (\uparrow)	In-Domain		OOD (Entity)		OOD (Relation)		OOD (Both)	
	(2284)	(1368)	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.
MEND+Propagation (Mid-Upper)	60.8	91.3	36.0	85.4	28.4	87.4	18.3	84.0
propagations \rightarrow paraphrases	12.4 †	91.8	10.5 †	93.1	11.8 †	93.2	12.9 †	89.1
all tokens \rightarrow answer tokens	45.9 †	91.7	34.8	89.5	20.5 †	89.7	16.2	88.3
Mid-Upper \rightarrow Upper layers	42.5 †	93.8	19.4 †	84.1	20.6 †	89.1	11.5 †	82.5

1350 Table 20: Efficiency Evaluation with `Llama-3.2-1B-base-QA` model on 50 examples. All
 1351 experiments are run on an NVIDIA RTX A6000 GPU, in a server with an Intel Core i9-10940X
 1352 CPU@3.30GHz. *: we ran 4 gradient update on the injected fact f , beyond which the drop in loss is
 1353 marginal (see full hyperparameters in Table 23).

1354

	Max Memory Usage (MiB \downarrow)	Total Runtime (Second \downarrow)
Base Model	6059	42
Prepend	+ 28	+ 1
CPT (Full)*	+ 19132	+ 920
MEMIT (<code>wikitext-103</code>)	+ 4010	+ 1291
MEND (Mid-Upper)	+ 7550	+ 106
MEND+Propagation (Mid-Upper)	+ 7542	+ 96
MEND+Propagation	+ 15163	+ 122

1363

1364 Table 21: Results on Controlled RippleEdit with `Qwen-2.5-1.5B-base-QA`. We report
 1365 the model’s LLM-Score on the dataset for efficacy, and the model’s performance on a collection of
 1366 single-hop questions for specificity. OOD (Entity) means using ID relation with OOD entity; OOD
 1367 (Relation) means using ID entity with OOD relation. Prepend is not a parametric method. † means
 1368 the system is outperformed by MEND+Propagation according to a paired bootstrap test ($p = 0.05$).

1369

LLM-Score (\uparrow)	In-Domain (2284)		OOD (Entity) (1368)		OOD (Relation) (421)		OOD (Both) (447)	
	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.	Effi.	Spec.
<code>Qwen-2.5-1.5B-base-QA</code>	8.0 †	91.2 †	6.8 †	89.9	10.5 †	87.3	9.1 †	91.1
Prepend	63.1	86.2 †	59.4	86.9	58.6	82.9	51.9	81.5 †
CPT (Full)	12.0 †	88.2 †	9.6 †	86.8	12.0 †	82.7	11.2 †	82.0 †
Meta-Aug CPT (Full)	85.2	87.5	–	–	32.1	73.6	18.2	73.8
Active Reading CPT (Full)	16.1	80	15.2	79.9	19.5	73.8	14.3	75.3
MEMIT (<code>wikitext-103</code>)	16.0 †	91.3 †	16.1 †	90.1	13.9 †	87.2	9.6 †	90.3
MEMIT (Ctrl RippleEdit)	11.6 †	91.2 †	12.6 †	90.0	10.3 †	86.6	10.1 †	89.7
MEND (with standard config)	12.3 †	87.1 †	9.9 †	88.2	11.1 †	83.5	10.9 †	86.2
MEND (Mid-Upper)	9.1 †	58.3 †	8.9 †	56.6 †	4.8 †	61.4 †	5.2 †	69.4 †
MEND+Propagation (Mid-Upper)	56.7 †	89.5 †	30.6 †	83.0	28.4 †	85.7	14.0 †	87.9
MEND+Propagation	64.0	93.6	34.7	83.0	33.3	84.8	17.7	85.8

1385

1386 docstring difference and apply the updated function in program synthesis. ECLeKTic (Goldman
 1387 et al., 2025) focuses on cross-lingual knowledge transfer.

1388

1389 H.2 CONTINUED PRETRAINING

1390

1391 Continued pretraining (CPT) on documents to be injected serves as a strong baseline in these scenarios.
 1392 A line of work (Padmanabhan et al., 2023; Akyürek et al., 2024) proposes to improve knowledge
 1393 propagation in CPT by modifying data scenarios or learning objectives. Yao et al. (2025) uses circuit
 1394 analysis to arrive at the template for data augmentation. Jiang et al. (2024) finds instruction-tuning
 1395 LMs on question-answering pairs prior to CPT is beneficial for knowledge injection. Yang et al.
 1396 (2024) proposes to synthesize large-scale data from the document to be injected and perform CPT
 1397 on those documents, showing improved propagation. Unlike this line of work, hypernetwork-based
 1398 methods do not synthesize additional data at test time.

1399

1400 I COMPUTATIONAL RESOURCES

1401

1402 We conducted experiments with `Llama-3.2-1B-base` primarily on a server with NVIDIA A40
 1403 48GB GPUs and an AMD EPYC 7413 24-Core Processor. For larger models, our experiments were
 conducted on a server with NVIDIA GH200 120GB and ARM Neoverse-V2.

1404
1405 Table 22: Hyperparameters used for Supervised Fine-Tuning (SFT). The same set of parameters
1406 was used for Llama-3.2-1B-base, Qwen-2.5-1.5B-base, and Llama-3.2-3B-base
1407 (suffixed by -QA).

(a) SFT on TriviaQA.rc.		(b) SFT on Controlled RippleEdit.	
Hyperparamter	Value	Hyperparamter	Value
Learning rate	1e-5	Learning rate	2e-6
Scheduler	linear	Scheduler	linear
Epoch	2	Epoch	2
Max seq. length	256	Max seq. length	256
Batch size	128	Batch size	10
Weight decay	0.1	Weight decay	0.1
Max Gradient Norm	1.0	Max Gradient Norm	1.0
WarmUp ratio	0.03	WarmUp ratio	0.03
Optimizer	AdamW	Optimizer	AdamW

1420
1421 Table 23: Hyperparameters used for Continue Pretraining baselines, CPT (Full) and CPT (Mid-Upper),
1422 when injecting one fact f .

Hyperparamter	Value
Learning rate	1e-5
Scheduler	linear
Epoch	4
Max seq. length	1024
Batch size	1
Weight decay	0.1
Max Gradient Norm	1.0
Optimizer	AdamW

1433
1434 Table 24: Hyperparameters used for MEND+Propagation and MEND.

1435
1436 (a) Hyperparameters for training
1437 MEND+Propagation and MEND.

Hyperparameter	Value
c_{edit}	0.1
learning rate to learn test-time learning rate α_ℓ	0.0001
Learning rate for hypernetwork weight ϕ	1.0e-06
Batch size (after gradient accumulation)	10
Validation step	100
Early stop patience (# steps)	2000
Maximum training step	1000000
Optimizer	Adam

1438
1439 (b) Hyperparameters for hypernetwork
1440 (MLP) in MEND+Propagation and MEND.

Hyperparameter	Value
Activation	ReLU
# hidden	1
# hidden dim	1920
# parameter sharing	False

1441
1442 (c) Target MLP layers used for various comparison system

Base Model	Total # layers	Comparison system	Layer indices (min: 0)
Llama-3.2-1B-base	16	MEND+Propagation	4-15
		MEND+Propagation (Mid-Upper)/ MEND (Mid-Upper)	10-12
Qwen2.5-1.5B-base	28	MEND+Propagation	13-27
Llama-3.2-3B-base	28	MEND+Propagation	15-27

1458
1459
1460
1461
1462
1463

Table 25: Hyperparameters used for MEMIT.

(a) For Llama-3.2-1B-base		(b) For Qwen-2.5-1.5B-base	
Hyperparameter	Value	Hyperparameter	Value
Target layer	[1, 2, 3, 4, 5]	Target layer	[4, 5, 6, 7, 8]
rewrite_module_tmp	"layers.{ }.mlp.down_proj"	rewrite_module_tmp	"layers.{ }.mlp.down_proj"
clamp_norm_factor	0.75	clamp_norm_factor	4
fact_token	"subject_last"	fact_token	"subject_last"
v_num_grad_steps	20	v_num_grad_steps	25
v_lr	5e-1	v_lr	5e-1
v_loss_layer	15	v_loss_layer	27
v_weight_decay	0.5	v_weight_decay	1e-3
kl_factor	0.0625	kl_factor	0.0625
mom2_adjustment	true	mom2_adjustment	true
mom2_update_weight	20000	mom2_update_weight	15000
mom2_n_samples	100000	mom2_n_samples	100000

1482
1483
1484
1485
1486
1487
1488
1489
1490Table 26: **Exact Match (EM) Results on RippleEdit with Llama-3.2-1B-base-QA.** We report the total number of test queries in brackets. Prepend is not a parametric method. The other metric (LLM-Score) is reported in Table 1 in the main paper.1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511

EM (↑)	Efficacy		Specificity	
	Verbatim	Non-Verbatim	Verbatim	Non-Verbatim
(1373)	(1586)	(165)	(2099)	
Llama-3.2-1B-base-QA	17.0	4.0	90.9	23.2
Prepend	36.0	12.4	94.5	21.6
CPT (Full)	87.8	3.4	99.4	17.3
CPT (Mid-Upper)	48.7	4.0	93.3	24.1
MEMIT (wikitext-103)	21.1	5.6	93.3	24.1
MEMIT (RippleEdit)	26.6	5.9	98.2	19.3
MEND (with standard config)	72.7	3.0	98.2	21.3
MEND (Mid-Upper)	69.7	3.1	97.0	17.8
MEND+Propagation (Mid-Upper)	73.8	14.9	97.6	31.8
MEND+Propagation	78.7	17.3	95.2	35.1

1512

1513

1514

1515

1516 Table 27: **Results on RippleEdit with Llama-3.2-1B-base-QA.** Performances are reported
 1517 in the format of Exact Match (EM) / LLM-Score. We notice the EM and LLM-Score strongly disagree
 1518 with each other on Forgetfulness (FN); after spotchecking, we found EM is high because one of the
 1519 valid answers $a \in \mathcal{A}_i$ is a substring of the propagation question q_i . Prepend is not a parametric
 1520 method.

1521

1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540				1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	
	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540				1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	
	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540	1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540
Llama-3.2-1B-base-QA	13.0/13.5	13.0/11.0	4.4/9.3	4.6/8.2	24.9/29.0	51.1/10.4
Prepend	20.0/31.9	21.1/24.9	18.3/22.6	30.9/39.2	23.3/30.0	52.5/13.6
CPT (Full)	16.1/11.4	12.7/10.4	93.8/89.3	97.0/93.0	19.9/17.8	47.5/3.3
CPT (Mid-Upper)	13.9/15.8	13.3/12.0	32.6/32.2	50.1/51.7	26.4/28.0	48.6/10.9
Active-Reading CPT (Full)	36.5/29.9	13.2/12.7	93.8/95.1	97.7/93.8	20.7/19.7	48.9/5.4
MEMIT (wikitext-103)	14.3/13.8	14.5/14.6	7.3/11.6	10.6/16.2	24.1/26.3	49.6/7.9
MEMIT (RippleEdit)	14.3/13.3	14.8/14.8	7.7/13.9	20.2/24.9	21.6/23.5	48.9/7.3
MEND (with standard config)	14.8/11.7	12.1/10.2	68.9/69.8	79.9/80.8	24.0/25.8	47.5/8.4
MEND (Mid-Upper)	13.5/13.8	12.4/10.8	59.0/64.1	77.9/79.2	20.1/23.6	47.5/8.1
MEND+Propagation (Mid-Upper)	27.0/12.8	22.9/25.9	72.5/74.3	77.7/79.3	33.3/33.1	59.9/21.5
MEND+Propagation	30.9/25.0	25.3/27.7	83.5/85.7	81.3/82.1	35.7/35.6	65.6/27.3

1541

1542

1543

1544

1545

1546

1547

1548

1549 Table 28: Results on Controlled RippleEdit with Llama-3.2-3B-base-QA. We use the
 1550 model’s LLM-Score on multi-hop questions for efficacy, and the model’s performance on single-hop
 1551 questions for specificity. OOD (Entity) means using ID relation with OOD entity; OOD (Relation)
 1552 means using ID entity with OOD relation. Prepend is not a parametric method.

1553

1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565		1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565		1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	
	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565		1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565		1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	
	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565	1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565
LLama-3.2-3B-base-QA	8.1	91.8	6.9	93.0	8.1	92.4
Prepend	66.1	90.3	62.5	92.1	61.3	90.3
CPT (Full)	18.4	86.2	16.8	86.0	16.1	86.7
MEND+Propagation	69.9	94.6	42.4	89.8	34.0	93.2

1566 Though the runtime varies depending on the datasets, the meta-training of hyper networks typically
1567 takes around 10 hours, or as little as 4 hours for some experiments.
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619