MAKE: Memory-Associated Knowledge Editing

Seongsik Park[1], **Sangmin Park**[2], **Jaieun Kim**[2], **Harksoo Kim**[1]
[1]Konkuk University
Seoul, Republic of Korea
{a163912, nlpdrkim}@konkuk.ac.kr
[2]Saltlux
Seoul, Republic of Korea
{sangmin.park, jaieun.kim}@saltlux.com

Abstract

Since their emergence, large language models (LLMs) have rapidly advanced, exerting substantial influence across various domains. Consequently, the importance of model editing techniques, aimed at locally correcting outdated or incorrect knowledge within language models, has grown significantly. However, traditional model editing methods face limitations: They cannot guarantee that highly related knowledge will transfer to the post-edited model, and they often rely on external knowledge bases to address this issue. In this paper, we propose a novel approach that leverages the internal knowledge of the language model to overcome the shortcomings of existing methods. First, we explore how to recall indirect associated knowledge from the model itself, which can be utilized in the editing process. Building on this, we propose **MAKE (Memory-Associated Knowledge Editing)**, an editing method that takes into account the transfer of associated knowledge. As a result, MAKE successfully updates associated knowledge and achieves state-of-the-art performance in experiments conducted on the zsRE+, CounterFact+ and MQuAKE datasets.

1 Introduction

Since their emergence, large language models (LLMs) have had a tremendous impact on both academic and industrial research fields (Brown et al., 2020; Touvron et al., 2023; Qiao et al., 2023; Zheng et al., 2023b). LLMs are pre-trained on massive amounts of parameters based on large-scale corpora and, through instruction tuning for various tasks, can generate appropriate responses in practical scenarios. Nevertheless, LLMs still face several inevitable limitations. One of these limitations is that LLMs struggle to keep up with the ever-changing factual knowledge of the real world (Wu et al., 2024; Zhao et al., 2024). The knowledge of an LLM is fixed at the time it completed its last training. Therefore, when asked questions about knowledge beyond that point, it may provide inaccurate responses. Another limitation is the issue of bias and hallucinations resulting from flaws in large-scale corpora (Zhang et al., 2023, 2024b; Gallegos et al., 2024; Yang et al., 2024b). The massive corpora used for pre-training LLMs are mostly composed of data collected from the internet. While collecting data online allows for relatively easy construction of large corpora, it is a challenging task to thoroughly review and ensure the corpus is flawless. As a result, biased or incorrect knowledge may be used during pre-training, which can lead to bias and hallucination issues in LLMs. The most intuitive way to address this issue is to collect data containing the correct factual knowledge and retrain the model from scratch each time. However, retraining the model from scratch just to correct a portion of the knowledge is highly inefficient. Therefore, there is a need for methods that can correct the model's incorrect knowledge and generate accurate responses without having to retrain the model.

Recently, various model editing techniques have been proposed to achieve this (Yao et al., 2023; Wang et al., 2024). Model editing (or knowledge editing) aims to effectively modify the model's behavior related to the target of the edit while preserving its behavior in other areas. In one representative approach, ROME (Meng et al., 2022), the authors observed that feed-forward network (FFN) of a specific transformer layer acts as key-value memories that recall factual knowledge (Geva et al., 2021). ROME adjusts the output of the FFN by calculating the optimal latent value needed to generate new objects and edits the

Transactions of the Association for Computational Linguistics, vol. 13, pp. 938–952, 2025. https://doi.org/10.1162/tacl.a.26
Action Editor: Doug Downey. Submission batch: 10/2024; Revision batch: 2/2025; Published 8/2025.

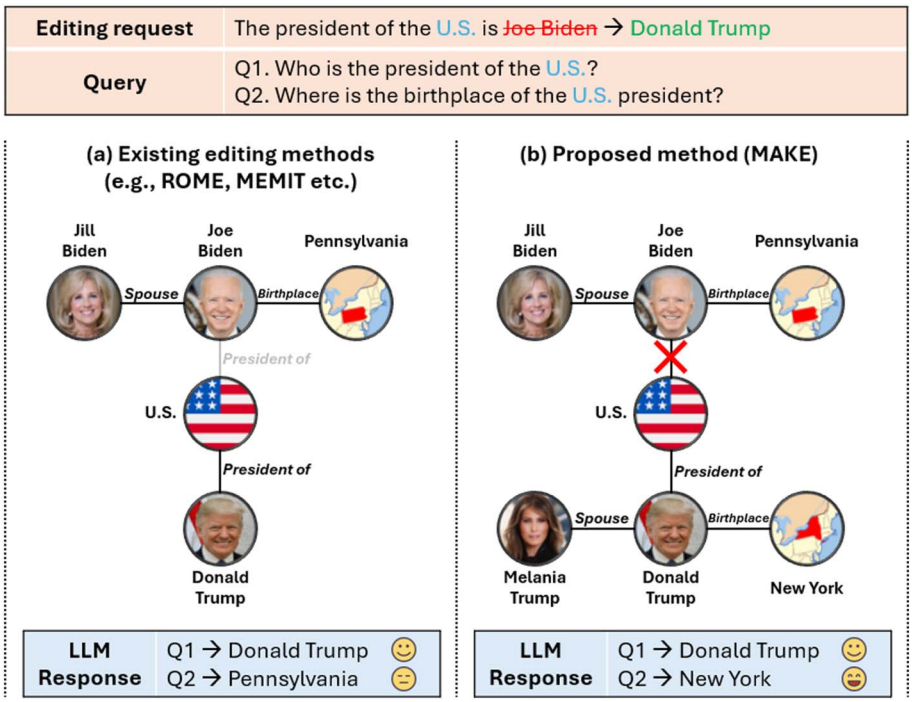

Figure 1: Examples of existing knowledge editing methods and proposed editing methods. (a) visualizes the existing knowledge editing method in the form of a knowledge graph, while (b) illustrates the proposed method (MAKE). For effective knowledge editing in language models, it is necessary to completely sever connections with the existing knowledge and facilitate the transfer of new knowledge along with its associated information.

FFN's weights accordingly. Since it limits modifications to parameters that contribute to the target knowledge, it allows for efficient editing while preserving knowledge unrelated to the target. Subsequent studies similarly focus on calculating latent values that maximize the probability of generating new objects (Meng et al., 2023; Li et al., 2024). However, a method that solely focuses on generation probability has several issues. Figure 1 compares existing knowledge editing methods such as ROME and MEMIT with the proposed method, MAKE (Memory-Associated Knowledge Editing). Knowledge is generally represented as triples in the form of (subject, *relation*, object), which together form a knowledge graph. When existing knowledge editing methods are expressed as a knowledge graph, they correspond to Figure 1(a). These methods perform editing by increasing the probability of generating a new object in response to an ''editing request''. For example, if the model originally knows the knowledge (U.S., *president*, Joe Biden), the editing modifies it so that the new object, ''Donald Trump,'' replaces ''Joe Biden.'' However, this approach has a limitation: It struggles to transfer knowledge associated with ''Donald Trump'' into the edited model. This issue arises because the existing methods merely increase the probability of generating specific words without considering the associated knowledge of ''Donald Trump.'' Additionally, there remains a problem that the connection between ''U.S.'' and ''Joe Biden'' is not entirely severed. Even if the probability of generating ''Donald Trump'' is significantly increased, there is still a residual probability of generating ''Joe Biden,'' and it is difficult to completely remove knowledge associated with ''Joe Biden.'' As a result, when answering a question like Q2 in Figure 1 which queries related knowledge of the edited entity, the model may still provide responses based on the previous object, ''Joe Biden.'' The editing method proposed in this paper, as illustrated in Figure 1(b), severs the connection with the previous object and transfers both the new object and its associated knowledge. This approach is similar to how knowledge is edited in a knowledge base. A language model implicitly stores diverse knowledge learned from large-scale data. Unlike knowledge graphs, this characteristic makes it challenging to directly extract and edit specific knowledge. Therefore, we leverage appropriate contexts to indirectly extract and utilize knowledge. For example, to extract knowledge that the model associates with

‘‘Donald Trump,’’ we can input ‘‘Donald Trump, also known as,’’ into the model and use the generated result as an indirect representation of the associated knowledge. We refer to suffixes like ‘‘also known as’’ as Context Recall Triggers (CRT) and design several such suffixes to aid in knowledge editing. Additionally, we utilize the indirectly acquired knowledge to develop two loss functions that facilitate the transfer of associated knowledge for the new object while severing the associated knowledge of the previous object. The first is the Associated Memory Assignment (AMA) loss, which aims to incorporate the associated memories of the new object (e.g., ‘‘Donald Trump’’ in Figure 1) into the latent value. The AMA loss works by attaching suffixes to the new object and the edit prompt, and then ensuring that the two resulting probability distributions become similar. The second is the Irrelevant Memory Invalidation (IMI) loss, which aims to sever the connection to the associated memories of the previous object (e.g., ‘‘Joe Biden’’ in Figure 1). The IMI loss uses the top-k token sets generated by attaching suffixes to the previous object and the edit prompt. It lowers the generation probabilities of the tokens that belong to the intersection of the two top-k token sets, effectively disconnecting the model from the associated memories of the previous object.

The contributions of this paper are as follows:

- We explore a method to elicit the model’s inherent memory of the target for editing. As a result, we observe that the language model contains sufficient related knowledge that can be effectively utilized for knowledge editing.
- To leverage the inherent memory of language models for knowledge editing, we propose two loss functions with distinct roles: AMA and IMI. Based on these two loss functions, the proposed method can update the model’s associated knowledge of the object.
- The proposed method demonstrates superior performance in knowledge editing across various language models. In particular, it proves to be highly effective in improving the editing of associated knowledge.

2 Related Work

Model editing is a method for modifying a model’s behavior regarding specific knowledge without retraining it. Model editing can be categorized into methods that either preserve or modify the parameters of the backbone model. One of the most representative parameter-preserving approaches is the use of memory modules (Lewis et al., 2020; Mitchell et al., 2022b; Zheng et al., 2023a). This method updates the knowledge to be edited exclusively within an external memory module and retrieves it during inference to modify the model’s output. Recent LLMs can autonomously edit their outputs by incorporating retrieved knowledge through in-context learning. However, this approach has a major limitation: Its effectiveness is highly dependent on retrieval accuracy, and it does not fundamentally alter the knowledge stored within the model. In contrast, parameter-modifying editing methods are relatively free from this limitation. By identifying and modifying only the specific regions of the model responsible for storing the target knowledge, these methods enable stable and efficient knowledge editing. This approach is known as Locate-then-Edit, which has recently been explored through various techniques (Yao et al., 2023; Wang et al., 2024). Dai et al. (2022) consider the feed-forward network (FFN) within a transformer as key-value memories (Geva et al., 2021). They propose a method to identify specific neurons in the FFN that influence the output of the language model and edit them to change the model’s output. Meng et al. (2022) further refine this approach by utilizing causal tracing (Pearl, 2022; Vig et al., 2020a,b) for more precise parameter detection. As a result, they discovered that the FFN in specific transformer layers plays a crucial role in recalling knowledge, and they proposed ROME, a method that updates the weight matrix of the relevant FFN to fit the target of the edit. MEMIT (Meng et al., 2023) builds on this by applying a progressive editing method across multiple layers, allowing for stable knowledge editing even for multiple targets. Li et al. (2024) point out that, in addition to the FFN, the multi-head attention mechanism also contributes to recalling factual knowledge. They propose PMET, an improved editing method that considers the influence of both multi-head attention and FFN. While the previously mentioned studies effectively perform direct edits on the given knowledge, they do not consider the associated knowledge. However, to achieve a complete edit of the model’s behavior, it is necessary to consider the ripple effects that

the target knowledge may have (see the example in Figure 1) (Cohen et al., 2024). To address this issue, Zhang et al. (2024a) propose GLAME, a method that utilizes external knowledge bases to incorporate associated knowledge into the editing process. GLAME extracts a knowledge sub-graph related to the target knowledge from an external knowledge base and applies graph encoding methods (Schlichtkrull et al., 2018; Lv et al., 2021) to integrate the knowledge graph information into existing editing methods. As a result, GLAME achieves model editing that maintains the performance of existing methods while also ensuring portability. However, the process of accessing an external knowledge base and extracting a subgraph each time a new knowledge editing target is introduced is inefficient. According to existing research, language models can serve as knowledge bases (Petroni et al., 2019; Roberts et al., 2020; Jiang et al., 2020; Shin et al., 2020). Taking this characteristic of language models into account, we propose a novel model editing method, which ensures the editing of associated knowledge based on the model's internal knowledge.

3 Methodology

3.1 Preliminaries

3.1.1 Model Editing

The goal of model editing is to replace the knowledge embedded in the language model F, represented as (s, r, o), with new knowledge (s, r, o^*), thereby changing the model's behavior. Here, s, r, and o refer to the subject, relation, and object, respectively, and o^* represents the new object that will replace o. Practically, a prompt x (e.g., ''The capital of France is'') that expresses s and r is used as the editing target, and after editing, the model is expected to generate o^* instead of o when x is provided (e.g., ''Paris'' $\rightarrow$ ''Beijing''). After editing, the model F' must satisfy the following conditions.

$$F'(x) = \begin{cases} o^*, & \text{if } x \in I(s, r, o^*) \\ F(x), & \text{if } x \in O(s, r, o^*) \end{cases} \quad (1)$$

In the equation, $I(s, r, o^*)$ represents all samples within the scope of the editing target. This includes both the direct editing target x and all paraphrase prompts related to x. In other words, the first line indicates that the post-edited model F' must output o^* regardless of how the editing target is expressed in the prompt. $O(s, r, o^*)$ refers to samples outside the scope of the editing target. Since these are not part of the editing target, when their prompts are input, the edited model should produce the same output as the original model F. Considering the editing of associated knowledge, the edited model must satisfy the following conditions.

$$F'(x) = o^E, \quad \text{if } x \in E(s, r^e, o^e) \quad (2)$$

In the equation, $E(s, r^e, o^e)$ denotes the scope of the editing target, which is extended to the associated knowledge. r^e is a relation expressed through r, and o^e refers to the object associated with s and r^e. This implies that the post-edited model must not only account for the editing target but also ensure consistency with all knowledge associated with the new object. For example, after editing the capital of France to Beijing, when the prompt ''The most famous architecture in the capital of France is'' is given, the model should generate ''Forbidden City'' instead of ''Eiffel Tower,'' reflecting the knowledge of Beijing's famous architecture.

3.1.2 Editing Method

MEMIT (Meng et al., 2023) assumes that in language models, the FFNs of specific transformer layers function as key-value mapping memories for recalling knowledge (Geva et al., 2021). The FFN consists of two linear layers, where the output encoded by the first linear layer is defined as the key representation, and the output passing through the second linear layer is defined as the value representation. MEMIT edits the weights of the second linear layer to compute the value representation that induces the generation of o^* when p is input. The post-edited weight that MEMIT seeks to compute is defined as follows.

$$W_1 \triangleq \underset{\hat{W}}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \|\hat{W} k_i - v_i\|^2 + \sum_{i=n+1}^{n+u} \|\hat{W} k_i - v_i\|^2 \right) \quad (3)$$

In the equation, k and v represent the key and value representations, respectively. The term $\sum_{i=1}^{n} \|\hat{W} k_i - v_i\|^2$ represents the n pieces of existing knowledge that the model must retain, while $\sum_{i=n+1}^{n+u} \|\hat{W} k_i - v_i\|^2$ represents the u pieces of knowledge that the model must newly update. Applying the normal equation to the above

Suffixes	Examples of internal memory
is	France is the world leader in nuclear power. France's nuclear power plants generate . . .
which is	France, which is the world's second largest market for wine, has also been . . .
also known as	France, also known as the French Alps or the Côte d'Azur. The French Alps are . . .
related to	France, related to the French Revolution, is a term applied to . . .

Table 1: Examples of suffixes functioning as contextual recall triggers and the corresponding internal memory representations of the model. A total of eight suffixes exist, derived from the four base suffixes by optionally adding the definite article ''the.'' The underlined parts indicate the associated knowledge that the model recalls about the target.

formulation yields the following weight update equation.

$$\begin{aligned} W_1 &= W_0 + \Delta \\ \Delta &= RK_1^\top (C_0 + K_1 K_1^\top)^{-1} \end{aligned} \tag{4}$$

W_0 and W_1 represent the weight matrices before and after the update, respectively. K_1 denotes the set of key representations for the editing target. $R \triangleq V_1 - W_0 K_1$ represents the residual error between the current model and the new knowledge calculated using the pre-edit weight matrix W_0. V_1 is the set of latent value representations that contribute to generating the new object. $C_0 \triangleq K_0 K_0^\top = \lambda \mathbb{E}[kk^\top]$ represents the uncentered covariance of the key representations for the knowledge that the model must retain. This is a constant term inferred from randomly sampled key representations, and in MEMIT, it is computed using 100,000 samples from Wikitext. MEMIT then performs optimization through negative log likelihood loss to calculate the residual error R.

$$\begin{aligned} z_i &= h_i^L + \underset{\delta_i}{\operatorname{argmin}} \mathcal{L}_{nll} \\ \mathcal{L}_{nll} &= \frac{1}{|P|} \sum_{j=1}^{|P|} -\log \mathbb{P}_{F(h_i^L += \delta_i)}[o_i^* | p_j \oplus x_i] \end{aligned} \tag{5}$$

Through this process, the optimal change δ_i, which maximizes the generation probability of o_i^* when the target prompt x_i is given, for the hidden state h_i^L of the last target layer L is calculated. P represents the set of prefix prompts automatically generated by the model to improve generalization performance. Finally, the layer-wise residual error r_i and v_i are calculated, and the layer-wise editing is performed using the following equation.

$$\begin{aligned} v_i^l &= W_{out}^l k_i^l + r_i^l \\ r_i^l &= \frac{z_i - h_i^L}{L - l + 1} \end{aligned} \tag{6}$$

In the equation, l refers to the current target layer for editing, and W_{out}^l represents the second FFN weight matrix of layer l.

3.2 Internal Memory of Language Model

Language models undergo self-supervised pre-training based on large-scale corpora by predicting the next token or masked token (Brown et al., 2020; Devlin et al., 2019). After completing the pre-training process, language models acquire the ability to capture rich contextual information. However, the significance of pre-training is not limited to extracting contextual information alone. When given a piece of knowledge, the language model can generate the most natural sequence of tokens that follow. For example, when given the input ''The capital of France is,'' the language model is highly likely to generate ''Paris.'' This stems from the linguistic patterns the model has memorized. In other words, the model learns and memorizes the patterns that occurred most frequently in the corpus regarding a given piece of knowledge. Due to this characteristic, language models can be utilized as implicit knowledge bases (Petroni et al., 2019; Shin et al., 2020). The most intuitive prompt to recall associated knowledge about a specific entity in a language model is to explicitly express the entity along with any arbitrary relation. For example, to recall knowledge about France, prompts such as ''The president of France is'' or ''The currency of France is'' can be used. However, this method presents practical challenges when applied to all possible relations an entity may have. Instead, we design a suffix that indirectly triggers the model to recall the knowledge it has memorized about the entity. We refer to this as the **Contextual Recall Trigger (CRT)**, with examples provided in Table 1.

In Table 1, the examples of internal memory are the outputs naturally generated by the model

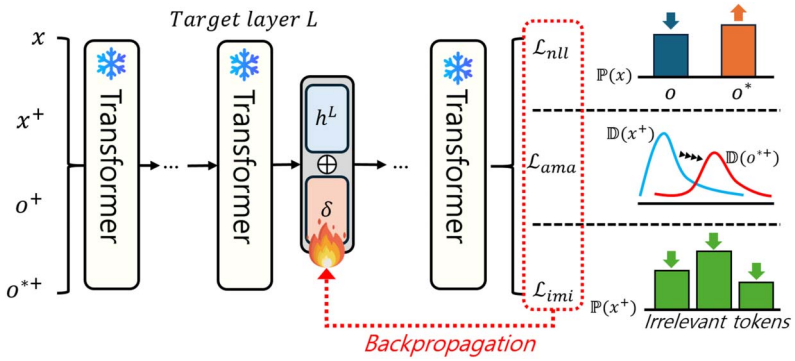

Figure 2: The δ optimization process in MAKE. First, x, x^+, o^+, and o^{*+} are fed into the model. x^+, o^+, and o^{*+} are the results of x, o, and o^* with suffixes attached. Then, the model's next token probability distribution and top-k tokens are obtained, which are used to calculate the loss functions L_{nll}, L_{ama}, and L_{imi}. The red dotted line represents backpropagation, where only δ is optimized through the loss functions.

when each suffix is attached to ''France.'' As a result, the CRT enables the indirect extraction of associated knowledge about the target entity from the model. The knowledge extracted through CRT is used to transfer associated knowledge to the new object or to sever associated knowledge of the previous object. We leverage the probability distribution and top-k tokens generated immediately after the suffix. The token generated right after the suffix acts as a trigger, determining which memory the model recalls. Thus, by utilizing this approach, the model's memory of the target can be selectively activated or deactivated. By guiding the probability distribution of tokens generated after the editing target prompt to resemble the distribution following the new object, the associated knowledge of the new object can be effectively transferred to the language model. Additionally, reducing the generation probabilities of the top-k tokens produced when inputting the previous object with the suffix enables the severance of its connection.

3.3 MAKE: Memory-Associated Knowledge Editing

Figure 2 illustrates the process of optimizing δ for model editing in MAKE. We extend the existing editing mechanism by incorporating an additional optimization that disconnects the model's internal memory from the existing object o and reinforces the association to the new object o^*. In MAKE, three loss functions are used. The first is the negative log likelihood loss function, which is designed to increase the probability of generating o^*.

$$\mathcal{L}_{nll} = \frac{1}{|P|}\sum_{j=1}^{|P|} -\log \mathbb{P}_{F'}[o^*|p_j \oplus x] \quad (7)$$

P is a set of prefixes automatically generated by the model to ensure consistent editing performance, even when x appears in arbitrary contexts, thus aiding generalization. F' refers to the model $F(h^L + \delta)$, where δ is added to the hidden state of the last editing target layer L. $\mathcal{L}_{NLL}$ is the loss function typically used by conventional locate-then-edit methods. The second loss function is the AMA loss, designed to transfer the model's internal knowledge of the new object o^*.

$$\mathcal{L}_{ama} = \frac{1}{|P|}\sum_{j=1}^{|P|} KL(\mathbb{D}_F[p_j \oplus o^{*+}] \| \mathbb{D}_{F'}[p_j \oplus x^+]) \quad (8)$$

In the equation, o^{*+} and x^+ represent o^* and x, respectively, with suffixes that serve as the CRT. $\mathbb{D}_F$ and $\mathbb{D}_{F'}$ denote the next token probability distributions calculated from the original model F and the edited model F', respectively. Specifically, $\mathcal{L}_{ama}$ optimizes δ through KL divergence, ensuring that the next token probability distribution of F' when given x^+ becomes similar to the distribution of F when given o^*. This allows the knowledge that the original model memorized about o^* to be transferred to the edited model. Lastly, the third loss function is the IMI loss,

which aims to sever the connection between the existing object o and the knowledge linked to p.

$$\mathcal{L}_{imi} = \frac{1}{|P|}\frac{1}{|T|}\sum_{j=1}^{|P|}\sum_{t\in T} -\log(1-\mathbb{P}_{F'}[t|p_j \oplus x]) \tag{9}$$

$\mathcal{L}_{imi}$ utilizes the unlikelihood loss (Welleck et al., 2020). In the equation, T represents the intersection of the model's top-k tokens for o and x, excluding those also present in the top-k tokens for o^*. In other words, T consists of tokens that evoke the residual memory of o remaining in the model with respect to the target prompt x. $\mathcal{L}_{imi}$ reduces the generation probability of these tokens, thereby invalidating the association between the previous knowledge and the target of editing. To further enhance the portability of the final edited model, we replace equation (5) with the following.

$$z_i = h_i^L + \underset{\delta_i}{\operatorname{argmin}}(\mathcal{L}_{nll} + \mathcal{L}_{ama} + \mathcal{L}_{imi}) \tag{10}$$

Through these, we can obtain the latent variable z_i, which includes the associated knowledge of the editing target, and perform model editing according to equations (4) and (6).

4 Experiments

4.1 Datasets and Baselines

We use the following three datasets in our experiments to evaluate the model editing effectiveness of MAKE.

The **Zero-Shot Relation Extraction plus (zsRE+)** dataset (Yao et al., 2023) is an extension of a Question Answering dataset (Levy et al., 2017) designed for model editing. It contains a total of 1,037 instances, where each instance consists of a question, its original answer, and an alternate answer meant to replace the original. Additionally, zsRE+ includes rephrased questions generated via back-translation, questions expanded to one-hop relations using GPT-4, and unrelated questions. This setup allows us to evaluate the generalization, specificity, and portability of the post-edited model.

CounterFact plus (CF+) is an extended version of the CounterFact dataset, designed to measure portability in model editing. CF+ is constructed similarly to zsRE+ but is known to be a more challenging dataset. The alternative answers for model editing are composed of counterfactual objects—those that receive the lowest generation scores for the current prompt. To more precisely measure specificity, CF+ uses prompts where the predicate is the same as in the target editing prompt but with a similar subject substituted. CF+ contains a total of 1,031 instances.

Multi-hop Question Answering for Knowledge Editing (MQuAKE) is a dataset composed of multi-hop questions designed to evaluate knowledge editing methods (Zhong et al., 2023). Unlike zsRE+ and CF+, it includes questions with up to four hops, allowing for a more challenging assessment of portability. MQuAKE contains a total of 3,000 instances. Since it does not include specific queries for evaluating specificity, we measure specificity by utilizing prompts from different instances.

We utilize GPT-2 XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), and Qwen-2.5-instruct (7B) (Yang et al., 2024a) as the backbone models for model editing. For performance comparison in model editing, we employ the following baseline models: Constrained Fine-Tuning (FT) (Meng et al., 2022), MEND (Mitchell et al., 2022a), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), PMET (Li et al., 2024), and GLAME (Zhang et al., 2024a). FT is a method that fine-tunes the model directly on the editing target, while MEND is a gradient-based editing approach using an auxiliary model. The remaining methods are well-known locate-then-edit approaches.

4.2 Implementation Details

For GPT-2, the target editing layers are [13, 14, 15, 16, 17], while for GPT-J and Qwen-2.5, they are [3, 4, 5, 6, 7, 8]. In the three backbone models, we optimize δ using a learning rate of 0.5, with a maximum of 35 steps. The value of C_0 in equation (4) is taken from MEMIT, and the set of prefixes P is automatically generated from the language model by creating 5 random prefixes of 20 tokens each. In $\mathcal{L}_{imi}$, the number of top-k tokens was empirically determined and the top 20 tokens are used. And ''also known as the'' is fixed as the CRT (for analysis of CRT, refer to Section 4.5). All scores in the comparative experiments are averaged over five independent trials, with the corresponding standard deviations reported. The results of the t-test indicate that

Method	Efficacy	Generalization	Specificity	Portability	Overall
GPT-2 (1.5B)	24.84	23.85	25.24	14.33	20.88
FT	55.79	31.01	23.10	15.33	25.21$_{(\pm 1.12)}$
MEND	66.57	60.24	25.07	15.94	29.80$_{(\pm 0.86)}$
ROME	**98.87**	**86.90**	24.88	18.15	34.21$_{(\pm 0.61)}$
MEMIT	94.21	82.11	**25.73**	18.27	34.37$_{(\pm 0.53)}$
MAKE	94.02	81.49	25.51	**21.30**	**36.68**$_{(\pm 0.43)}$
GPT-J (6B)	26.18	25.32	27.32	15.84	22.54
FT	60.56	31.56	**27.55**	16.37	27.48$_{(\pm 1.58)}$
MEND	78.98	68.10	27.39	17.14	32.73$_{(\pm 1.08)}$
ROME	99.02	94.82	27.33	24.07	40.49$_{(\pm 0.77)}$
MEMIT	99.18	87.21	27.29	25.44	41.03$_{(\pm 0.73)}$
PMET	95.63	87.08	27.31	24.55	40.29$_{(\pm 0.86)}$
MAKE	**99.47**	**96.59**	27.27	**29.61**	**44.03**$_{(\pm 0.40)}$
Qwen-2.5 (7B)	31.83	30.94	38.64	26.66	31.47
FT	40.74	33.00	38.72	26.67	33.85$_{(\pm 1.27)}$
ROME	98.51	88.39	38.69	42.22	56.34$_{(\pm 0.41)}$
MEMIT	98.46	85.71	38.65	41.75	55.83$_{(\pm 0.59)}$
MAKE	**98.84**	**93.41**	**38.80**	**44.27**	**57.82**$_{(\pm 0.31)}$

Table 2: The results of the comparative experiments on the zsRE+ dataset. The overall score represents the harmonic mean of efficacy, generalization, specificity, and portability. The highest performance for each metric is shown in bold, while the second-highest performance is underlined.

MAKE exhibits statistically significant improvements at the p-value < 0.05 level in the majority of cases.

4.3 Evaluation Metrics

In model editing, the commonly used evaluation metrics are efficacy, generalization, and specificity (Yao et al., 2023).

Efficacy is a metric used to assess how successfully the editing has been performed on the target, and is calculated as follows.

$$\mathbb{E}_{p\sim I(s,r,o^*)}[\mathbb{I}[\mathbb{P}_{F'}(o^*|p) > \mathbb{P}_{F'}(o|p)]] \quad (11)$$

For example, if the model's response to the prompt ''The president of the U.S. is'' is edited from ''Joe Biden'' to ''Donald Trump'' and the edited model F' generate ''Donald Trump'' for the same prompt during inference, the efficacy of the editing is improved.

Generalization is a metric used to evaluate whether the model successfully applies the edit to rephrased prompts that hold the same meaning as the target prompt. The formula for calculating generalization is as follows.

$$\mathbb{E}_{p'\sim I(s,r,o^*)}[\mathbb{I}[\mathbb{P}_{F'}(o^*|p') > \mathbb{P}_{F'}(o|p')]] \quad (12)$$

In the formula, p' refers to a rephrased prompt that holds the same meaning as the target prompt. For example, if the model responds with ''Donald Trump'' to prompts such as ''Who is the president of the U.S.?'' and ''The leader of the U.S. is,'' it can be considered to have high generalization.

Specificity is a metric used to evaluate whether areas outside the scope of the edit are preserved, and its formula is as follows.

$$\mathbb{E}_{p\sim O(s,r,o^*)}[\mathbb{I}[\mathbb{P}_{F'}(o^c|p) > \mathbb{P}_{F'}(o^*|p)]] \quad (13)$$

In the formula, o^c refers to the correct object for prompts outside the scope of the edit. For example, if the edited model provides correct responses to prompts unrelated to the editing target, such as ''The capital of the U.S. is'' and ''Who is the president of France?'', it can be considered to have high specificity.

Method	Efficacy	Generalization	Specificity	Portability	Overall
GPT-2 (1.5B)	19.50	22.79	79.61	12.20	21.09
FT	99.17	46.12	70.01	17.15	$38.33_{(\pm 0.96)}$
MEND	93.57	57.81	44.34	15.12	$34.28_{(\pm 1.13)}$
ROME	**99.99**	**96.71**	77.15	22.53	$51.49_{(\pm 0.63)}$
MEMIT	94.21	81.01	**78.65**	21.10	$48.15_{(\pm 0.61)}$
GLAME	99.84	<u>96.62</u>	76.82	<u>23.95</u>	<u>53.24</u>$_{(-)}$
MAKE	<u>99.87</u>	96.29	<u>77.17</u>	**26.34**	**56.08**$_{(\pm 0.38)}$
GPT-J (6B)	13.29	15.47	84.28	10.58	16.24
FT	<u>99.77</u>	45.09	80.34	15.03	$35.98_{(\pm 1.05)}$
MEND	96.46	54.17	54.21	14.16	$33.93_{(\pm 0.82)}$
ROME	**100.00**	**99.63**	78.91	25.97	$56.17_{(\pm 0.53)}$
MEMIT	**100.00**	95.23	**83.05**	25.31	$55.52_{(\pm 0.55)}$
PMET	**100.00**	97.53	<u>82.40</u>	27.57	$58.26_{(\pm 0.55)}$
GLAME	**100.00**	99.30	81.39	**33.04**	**63.87**$_{(-)}$
MAKE	**100.00**	<u>99.61</u>	81.13	<u>32.78</u>	<u>63.62</u>$_{(\pm 0.49)}$
Qwen-2.5 (7B)	11.74	13.14	86.50	17.87	17.48
FT	56.35	17.85	**86.36**	18.95	$28.96_{(\pm 1.76)}$
ROME	<u>99.90</u>	<u>97.87</u>	84.83	<u>46.81</u>	<u>74.94</u>$_{(\pm 0.28)}$
MEMIT	99.61	90.54	<u>85.83</u>	43.98	$72.11_{(\pm 0.31)}$
MAKE	**100.00**	**99.37**	83.64	**52.36**	**78.25**$_{(\pm 0.26)}$

Table 3: The results of the comparative experiments on the CF+ dataset. The overall score represents the harmonic mean of efficacy, generalization, specificity, and portability. The highest performance for each metric is shown in bold, while the second-highest performance is underlined.

In addition to the above three metrics, portability is also used as an evaluation metric. (Yao et al., 2023; Zhang et al., 2024a). **Portability** evaluates whether knowledge related to the edited target is also correctly updated during the edit, and its formula is as follows.

$$\mathbb{E}_{p \sim P(s,r^P,o^P)}[\mathbb{I}[F'(p) = o^P]] \tag{14}$$

For example, if the model responds based on ‘‘Donald Trump’’ to multi-hop questions related to the editing target, such as ‘‘The birthplace of the U.S. president is’’ and ‘‘Who is the spouse of the U.S. president?’’, portability can be considered ensured.

4.4 Experimental Results

Table 2 presents the comparative experimental results using the zsRE+ dataset. As a result, MAKE demonstrates the highest overall editing capability across all three backbone models. Compared to the previous state-of-the-art, MEMIT, MAKE shows a decrease in efficacy and generalization performance by 0.19 and 0.62 points, respectively, for GPT-2. However, it achieves a significant improvement in portability of 3.03 points, resulting in an overall score increase of 2.31 points. In GPT-J and Qwen-2.5, the performance gap between MAKE and MEMIT becomes more pronounced. Notably, unlike the results observed in GPT-2, a significant improvement in generalization can be observed. With a substantial improvement in portability, the overall performance gap compared to MEMIT increases by 3.74 points and 1.99 points, respectively. This suggests that the AMA and IMI loss functions effectively enable knowledge editing while considering associated knowledge, as intended. Furthermore, the larger the model, the more significant the improvement in MAKE's editing performance, likely due to the higher quantity and quality of internal knowledge in larger models.

Table 3 presents the results of a comparative experiment utilizing CF+. MAKE, similar

Method	Efficacy	Generalization	Specificity	Portability	Overall score
GPT-2 (1.5B)	17.74	17.45	31.49	23.24	21.22
FT	74.56	27.29	30.71	23.70	32.05$_{(\pm 0.86)}$
ROME	93.48	**81.27**	27.72	**26.36**	41.23$_{(\pm 0.25)}$
MEMIT	79.82	55.44	31.80	24.61	38.97$_{(\pm 0.37)}$
MAKE	**93.90**	77.64	**33.10**	25.67	**43.15**$_{(\pm 0.30)}$
Qwen-2.5 (7B)	4.68	10.93	67.90	24.44	11.09
FT	25.26	12.16	**68.57**	24.61	22.59$_{(\pm 1.30)}$
ROME	95.72	80.13	60.40	35.53	59.15$_{(\pm 0.22)}$
MEMIT	**97.80**	67.78	66.84	34.04	57.71$_{(\pm 0.12)}$
MAKE	96.65	**86.51**	66.44	**42.01**	**65.83**$_{(\pm 0.19)}$

Table 4: The results of the comparative experiments on the MQuAKE dataset. The overall score represents the harmonic mean of efficacy, generalization, specificity, and portability. The highest performance for each metric is shown in bold, while the second-highest performance is underlined.

Suffix	Efficacy	Generalization	Specificity	Portability	Overall
None	99.61	94.28	77.68	20.5	48.60
is	99.52	94.52	77.32	25.59	55.07
is the	99.52	94.71	77.07	25.13	54.51
also known as	99.81	95.78	77.12	23.21	52.28
also known as the	99.75	95.87	77.00	26.18	55.83
which is	99.81	95.05	77.07	23.62	52.73
which is the	99.61	94.52	77.17	25.13	54.52
related to	99.81	95.68	77.09	23.22	52.28
related to the	99.61	94.76	77.17	23.7	52.81

Table 5: Performance comparison based on the suffix used as the CRT. GPT-2 is used as the backbone model, and the CF+ dataset is utilized.

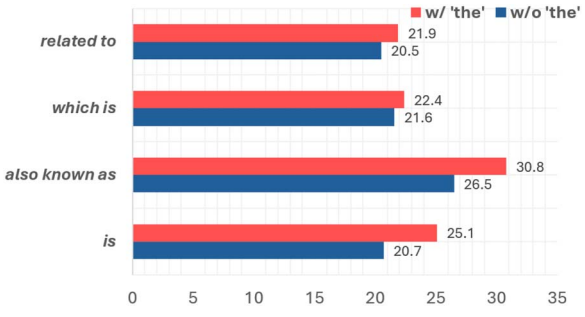

Figure 3: The proportion of generated outputs that include responses to the portability question when a suffix is attached to o^* and input into the model.

to its results on zsRE+, demonstrates significant performance improvements in generalization and portability, achieving state-of-the-art performance in the overall score. Compared to the previous state-of-the-art model, GLAME, MAKE achieves the highest performance with GPT-2, but shows lower performance with GPT-J. GLAME extracts relevant knowledge from external knowledge bases for model editing. While this approach enhances portability, it requires accessing external knowledge bases and constructing sub-graphs from the extracted knowledge for each new knowledge edit. In comparison, GLAME is less efficient than MAKE, which leverages the model's internal knowledge for editing.

Table 4 presents the comparative experimental results on MQuAKE. The results from GPT-2 experiments indicate that MAKE achieves the highest overall performance. However, MAKE shows lower performance on certain metrics, which is likely due to the limited zero-shot

Method	Efficacy	Generalization	Specificity	Portability	Overall
MAKE	99.75	95.87	77.00	26.18	55.83
- IMI loss	99.67	95.47	76.98	25.53	55.04
- AMA loss	99.81	96.65	77.07	21.85	50.56
- CRT	99.61	94.28	77.68	20.50	48.60
- IMI, AMA (MEMIT)	93.60	79.92	78.66	20.65	47.43

Table 6: Ablation study results of MAKE.

question-answering capability of GPT-2. Notably, the portability metric in MQuAKE is measured using multi-hop questions. Since GPT-2 is a pre-trained language model, its reasoning ability for question answering is relatively weak, leading to low portability across all editing methods. In contrast, Qwen-2.5, which has undergone instruction tuning, possesses a meaningful reasoning capability. As a result, it demonstrates higher scores on most metrics compared to GPT-2. Furthermore, MAKE exhibits strong performance in portability, confirming its effectiveness in multi-hop editing as well.

4.5 Analysis of MAKE

Table 5 presents the results of a comparative experiment on the suffix used as the CRT. ''None'' refers to the case where no CRT is used, and the entity is directly input into the model, applying the generated results to MAKE for editing. Compared to using CRT, this shows a significant drop in performance, likely because the model cannot sufficiently extract relevant knowledge for editing without the CRT. Additionally, even when using CRT, performance varies depending on the suffix, with ''also known as the'' showing particularly strong editing performance. Due to the nature of language models, the tokens generated next can vary significantly depending on the context. This may also influence the knowledge invoked by the CRT. In other words, using an appropriate suffix for the CRT can potentially enhance model editing performance. The performance differences between suffixes seem to stem from this characteristic. Notably, performance generally improves when the article ''the'' is attached after the suffix. This is likely because using the article imposes constraints on the next token generation, leading to the elicitation of more precise related knowledge. We conduct additional analysis to further investigate the knowledge elicitation effect of the CRT. Each suffix is attached to o^*, and the top 10 generated outputs are obtained from the language model. Figure 3 presents a chart showing the proportion of generated outputs that include responses related to the portability question. As shown in Figure 3, it can be observed that utilizing ''the'' generally results in a higher proportion of generated outputs containing responses. In particular, using ''also known as the'' as a suffix yields the highest ratio, supporting the results from Table 5. This suggests that the more appropriate the suffix used as the CRT, the more effective the knowledge elicitation, which directly impacts model editing. Table 6 presents the results of the ablation study conducted to measure the contribution of each component of MAKE. As seen in Table 6, removing the CRT from MAKE results in the most significant performance drop, with a decrease of 7.23 points in overall score. This is likely because the CRT is crucial in transferring the relevant knowledge associated with the target being edited. Comparing the two loss functions used in MAKE, the AMA loss contributes more to the model's performance. However, even when only the IMI loss is used, there is still a performance improvement compared to using neither loss function (i.e., MEMIT). This is because the IMI loss effectively removes residual knowledge about the original object, leading to substantial improvements in efficacy and generalization. Nevertheless, since it does not transfer associated knowledge, the overall score is lower than when the AMA loss is used.

5 Conclusion

This paper proposes a new method, MAKE (Memory-Associated Knowledge Editing), that enhances existing model editing techniques by enabling the transfer of associated knowledge alongside the target being edited. We explore

methods to recall the internal memory of the model related to the target entity, which we refer to as CRT (Contextual Recall Trigger). Using this mechanism, we introduce two loss functions, AMA (Associated Memory Alignment) and IMI (Irrelevant Memory Invalidation), to ensure that associated knowledge is also updated. As a result, MAKE demonstrates significant improvements in portability compared to previous editing methods, without relying on external knowledge bases. Future work aims to apply MAKE to larger language models (13B parameters and beyond) to analyze how editing performance scales with model size. Additionally, we plan to explore methods for automatically configuring the optimal CRT for different editing targets.

6 Limitation

MAKE relies heavily on the model's internal knowledge. This poses a limitation in situations where the model lacks sufficient knowledge about the target, resulting in a lack of relevant information to leverage during editing. In such cases, the model is limited to relying solely on the negative log likelihood loss function. However, using larger models could mitigate this issue, as the quantity and quality of internal knowledge are expected to improve, reducing the likelihood of such situations. Additionally, the suffixes used as CRTs in this paper may not always be well-suited to the language model or the specific target being edited. The suffixes we employed were manually constructed as potential CRT phrases, but it cannot be assumed that they are fully optimized for all targets or models. Therefore, future research should focus on developing methods to automatically and dynamically select optimal suffixes based on the target and the language model.

Acknowledgments

We are deeply grateful to the anonymous reviewers and action editor Doug Downey for their invaluable feedback, which significantly contributed to enhancing the quality of this paper. This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00553041, Enhancement of Rational and Emotional Intelligence of Large Language Models for Implementing Dependable Conversational Agents). This work was also supported by an Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (RS-2024-00343989, Enhancing the Ethics of Data Characteristics and Generation AI Models for Social and Ethical Learning).

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298. https://doi.org/10.1162/tacl_a_00644

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.581

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2024. Bias and Fairness in

Large Language Models: A Survey. *Computational Linguistics*, 50(3):1097–1179. https://doi.org/10.1162/coli_a_00524

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.446

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438. https://doi.org/10.1162/tacl_a_00324

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1034

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572. https://doi.org/10.1609/aaai.v38i17.29818

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 1150–1160, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/3447548.3467350

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372. Curran Associates, Inc.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Judea Pearl. 2022. *Direct and Indirect Effects*, 1 edition. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3501714.3501736

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

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5368–5393,

Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.294

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.437

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-93417-4_38

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.346

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. LlaMa 2: Open foundation and fine-tuned chat models.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020a. Causal mediation analysis for interpreting neural NLP: The case of gender bias.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020b. Investigating gender bias in language models using causal mediation analysis. *Advances in Neural Information Processing Systems*, 33:12388–12401.

Ben Wang and Aran Komatsuzaki. 2021. Mesh transformer jax. https://github.com/kingoflolz/mesh-transformer-jax

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024. Easyedit: An easy-to-use knowledge editing framework for large language models. https://doi.org/10.18653/v1/2024.acl-demos.9

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text degeneration with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020*.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men,

Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, et al. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Nakyeong Yang, Taegwan Kang, Stanley Jungkyu Choi, Honglak Lee, and Kyomin Jung. 2024b. Mitigating biases for instruction-following language models via bias neurons elimination. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9061–9073, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.490

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.632

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. https://doi.org/10.18653/v1/2024.emnlp-main.1261

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Lifeng Jin, Linfeng Song, Haitao Mi, and Helen Meng. 2024b. Self-alignment for factuality: Mitigating hallucinations in LLMs via self-evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1965, Bangkok, Thailand. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.107

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. SAPT: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11641–11661, Bangkok, Thailand. Association for Computational Linguistics.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023a. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876. https://doi.org/10.18653/v1/2023.emnlp-main.296

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. 2023b. Secrets of RLHF in large language models part i: Ppo.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702. https://doi.org/10.18653/v1/2023.emnlp-main.971