Knowledge Editing for Large Language Models: A Survey

SONG WANG, University of Virginia, Charlottesville, United States
YAOCHEN ZHU, University of Virginia, Charlottesville, United States
HAOCHEN LIU, University of Virginia, Charlottesville, United States
ZAIYI ZHENG, University of Virginia, Charlottesville, United States
CHEN CHEN, University of Virginia, Charlottesville, United States
JUNDONG LI, University of Virginia, Charlottesville, United States

Large Language Models (LLMs) have recently transformed both the academic and industrial landscapes due to their remarkable capacity to understand, analyze, and generate texts based on their vast knowledge and reasoning ability. Nevertheless, one major drawback of LLMs is their substantial computational cost for pre-training due to their unprecedented amounts of parameters. The disadvantage is exacerbated when new knowledge frequently needs to be introduced into the pre-trained model. Therefore, it is imperative to develop effective and efficient techniques to update pre-trained LLMs. Traditional methods encode new knowledge in pre-trained LLMs through direct fine-tuning. However, naively re-training LLMs can be computationally intensive and risks degenerating valuable pre-trained knowledge irrelevant to the update in the model. Recently, **Knowledge-based Model Editing** (KME), also known as **Knowledge Editing** or **Model Editing**, has attracted increasing attention, which aims to precisely modify the LLMs to incorporate specific knowledge, without negatively influencing other irrelevant knowledge. In this survey, we aim to provide a comprehensive and in-depth overview of recent advances in the field of KME. We first introduce a general formulation of KME to encompass different KME strategies. Afterward, we provide an innovative taxonomy of KME techniques based on how the new knowledge is introduced into pre-trained LLMs, and investigate existing KME strategies while analyzing key insights, advantages, and limitations of methods from each category. Moreover, representative metrics, datasets, and applications of KME are introduced accordingly. Finally, we provide an in-depth analysis regarding the practicality and remaining challenges of KME and suggest promising research directions for further advancement in this field.

CCS Concepts: • **Computing methodologies** → **Natural language processing**.

Additional Key Words and Phrases: Model Editing, Knowledge Update, Fine-tuning, Large Language Models

1 Introduction

Recently, large language models (LLMs) have become a heated topic that revolutionizes both academia and industry [10, 109, 144, 173]. With the substantial factual knowledge and reasoning ability gained from pre-training on large corpora, LLMs have exhibited an unprecedented understanding of textual information, which are able to analyze and generate texts akin to human experts [84, 87, 135, 138, 176]. Nevertheless, one main drawback of LLMs is the extremely high computational overhead of the training process due to the large amounts of parameters [59, 64, 179]. This is exacerbated by the continuous evolvement of the world where the requirement
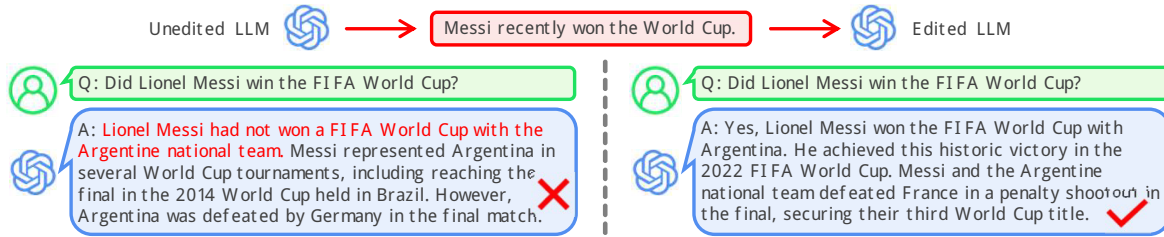
Fig. 1. An example of KME for efficient update of knowledge in LLMs.

of updating pre-trained LLMs to rectify obsolete information or incorporate new knowledge to maintain their relevancy is constantly emerging [85, 92, 128, 134]. For example, as in Fig. 1, the outdated LLM, GPT-3.5, cannot precisely describe the latest achievements of the famous soccer player Lionel Messi, which requires an explicit injection of new knowledge to generate the correct answers.

One feasible while straightforward strategy for updating pre-trained LLMs is through naive fine-tuning [20, 31, 141, 161], where parameters of pre-trained LLMs are directly optimized to encode new knowledge from new data [6, 99, 111, 173]. For example, various instruction-tuning methods are proposed to fine-tune pre-trained LLMs on newly collected data in a supervised learning manner [100, 112, 157, 159]. Although such fine-tuning techniques are widely used and capable of injecting new knowledge into LLMs, they are known for the following disadvantages: (1) Even with parameter-efficient strategies to improve efficiency [89, 158, 170], fine-tuning LLMs may still require intensive computational resources [97, 102, 174]. (2) Fine-tuning LLMs alters the pre-trained parameters without constraints, which can lead to the overfitting problem, where LLMs face the risk of losing valuable existing knowledge [172].

To address the drawbacks of updating LLMs with naive fine-tuning, more attention has been devoted to **Knowledge-based Model Editing**[1] (KME). In general, KME aims to *precisely* modify the behavior of pre-trained LLMs to update specific knowledge, without negatively influencing other pre-trained knowledge *irrelevant* to the updates [116, 152, 167]. In KME, the update of a specific piece of knowledge in LLMs is typically formulated as an **edit**, such as rectifying the answer to "*Who is the president of the USA?*" from "*Trump*" to "*Biden*". Regarding a specific edit, KME strategies typically modify the model output by either introducing an auxiliary network (or set of parameters) into the pre-trained model [52, 79, 175] or updating the (partial) parameters to store the new knowledge [22, 49, 51, 83]. Through these strategies, KME techniques can store new knowledge in new parameters or locate it in model parameters for updating, thereby precisely injecting the knowledge into the model. In addition, certain methods further introduce optimization constraints to ensure that the edited model maintains consistent behaviors on unmodified knowledge [13, 106, 177]. With these advantages, KME techniques can provide an efficient and effective way to constantly update LLMs with novel knowledge without explicit model re-training [172].

While sharing certain similarities with fine-tuning strategies, KME poses unique advantages in updating LLMs, which are worthy of deeper investigations. Particularly, both KME and model fine-tuning seek to update pre-trained LLMs with new knowledge. However, aside from this shared objective, KME focuses more on two crucial properties that cannot be easily addressed by fine-tuning. (1) **Locality** requires that KME does not unintentionally influence the output of other *irrelevant inputs* with distinct semantics. For example, when the edit regarding the president of the USA is updated, KME should not alter its knowledge about the prime minister of the UK. The practicality of KME methods largely relies on their ability to maintain the outputs for unrelated inputs, which serves as a major difference between KME and fine-tuning [117]. (2) **Generality** represents whether the edited model can generalize to a broader range of *relevant inputs* regarding the edited knowledge. Specifically, it

---

[1]The concept is also termed as Knowledge Editing or Model Editing. For clarity, we refer to it as KME in this paper.

indicates the model's capability to present consistent behavior on inputs that share semantic similarities. For example, when the model is edited regarding the president, the answer to the query about the leader or the head of government should also change accordingly. In practice, it is important for KME methods to ensure that the edited model can adapt well to such related input texts. To summarize, due to these two unique objectives, KME remains a challenging task that requires specific strategies for satisfactory effectiveness.

Differences between this survey and existing ones. Several surveys have been conducted to examine various aspects of (large) language models [12, 34, 71, 73, 142, 173]. Nevertheless, there is still a dearth of thorough investigations of existing literature and continuous progress in editing LLMs. For example, recent works [100, 159] have discussed the fine-tuning strategies that inject new knowledge in pre-trained LLMs with more data samples. However, the distinctiveness of KME, i.e., locality and generality, is not adequately discussed, which will be thoroughly analyzed in this survey. Two other surveys [35, 63] review knowledge-enhanced language models. However, they mainly focus on leveraging external knowledge to enhance the performance of the pre-trained LLMs, without addressing the editing task based on specific knowledge. To the best of our knowledge, the most related work [167] to our survey provides a brief overview of KME and concisely discusses the advantages of KME methods and their challenges. Nevertheless, the investigation lacks a thorough examination of more details of KME, e.g., categorizations, datasets, and applications. The following work [172] additionally includes experiments with classic KME methods. Another recent work [152] proposes a framework for KME that unifies several representative methods. This work focuses on the implementation of KME techniques, with less emphasis on the technical details of different strategies. A more recent study [116] discusses the limitations of KME methods regarding the faithfulness of edited models, while it is relatively short and lacks a more comprehensive introduction to all existing methods. Considering the rapid advancement of KME techniques, we believe it is imperative to review the details of all representative KME methods, summarize the commonalities while discussing the uniqueness of each method, and discuss open challenges and prospective directions in the domain of KME to facilitate further advancement.

Contributions of this survey. This survey provides a comprehensive and in-depth analysis of techniques, challenges, and opportunities associated with the editing of pre-trained LLMs. We first provide an overview of KME tasks along with an innovative formulation. Particularly, we formulate the general KME task as a constrained optimization problem, which simultaneously incorporates the goals of **accuracy**, **locality**, and **generality**. We then classify the existing KME strategies into three main categories, i.e., **external memorization**, **global optimzation**, and **local modification**. More importantly, we demonstrate that methods in each category can be formulated as a specialized constrained optimization problem, where the characteristics are theoretically summarized based on the general formulation. In addition, we provide valuable insights into the effectiveness and feasibility of methods in each category, which can assist practitioners in selecting the most suitable KME method tailored to a specific task. Our analysis regarding the strengths and weaknesses of KME methods also serves as a catalyst for ongoing progress within the KME research community. In concrete, our key contributions can be summarized into three folds as follows:

- **Novel Categorization.** We introduce a comprehensive and structured categorization framework to systematically summarize the existing works for LLM editing. Specifically, based on how the new knowledge is introduced into pre-trained LLMs, our categorization encompasses three distinct categories: external memorization, global optimization, and local modification, where their commonality and differences are thoroughly discussed in this survey.
- **In-Depth Analysis.** We formulate the task of KME as a constrained optimization problem, where methods from each category can be viewed as a special case with refined constraints. Furthermore, we emphasize the primary insights, advantages, and limitations of each category. Within this context, we delve deep into representative methods from each category and systematically analyze their interconnections.

- **Future Directions.** We analyze the practicality of existing KME techniques regarding a variety of datasets and applications. We also comprehensively discuss the challenges of the existing KME techniques and suggest promising research directions for future exploration.

The remainder of this paper is organized as follows. Section 2 introduces the background knowledge for KME. Section 3 provides a general formulation of the KME task, which can fit into various application scenarios. Section 4 provides a comprehensive summary of evaluation metrics for KME strategies, which is crucial for a fair comparison across various methods. Before delving into the specific methods, we provide a comprehensive categorization of existing methods into three classes in Section 5.1, where their relationship and differences are thoroughly discussed. Then we introduce the methods from the three categories in detail, where the advantages and limitations of each category are summarized. Section 6 introduces the prevalently used public datasets. Section 7 provides a thorough introduction to various realistic tasks that can benefit from KME techniques. Section 8 discusses the potential challenges of KME that have not been addressed by existing techniques. This section also provides several potential directions that can inspire future research. Lastly, we conclude this survey in Section 9.

2 Background

In this section, we provide an overview of the editing strategies for machine learning models and the basics of large language models (LLMs) as background knowledge to facilitate the understanding of technical details in KME. In this survey, we use bold uppercase letters (e.g., $\mathbf{K}$ and $\mathbf{V}$) to represent matrices, use lowercase bold letters (e.g., $\mathbf{k}$ and $\mathbf{v}$) to represent vectors, and use calligraphic uppercase letters (e.g., $\mathcal{X}$ and $\mathcal{Y}$) to represent sets. We summarize the primary notations used in this survey in Table 1 for the convenience of understanding.

2.1 Editing of Machine Learning Models

Machine learning models [41, 54, 74] pre-trained on large datasets frequently serve as foundation models for various tasks in the real-world [26, 126]. In practical scenarios, there is often a need to modify these pre-trained models to enhance the performance for specific downstream tasks [18, 20, 103, 164, 178], reduce biases or undesirable behaviors [39, 104, 113, 123], tailor models to align more closely with human preferences [44, 72, 88], or incorporate novel information [101, 167, 177].

Model Editing is a special type of model modification strategy where the modification should be as *precise* as possible. Nevertheless, it should accurately modify the pre-trained model to encode specific knowledge while maximally preserving the existing knowledge, without affecting their behavior on unrelated inputs [68]. First explored in the computer vision field, Bau et al. [8] investigate the potential of editing generative adversarial networks (GAN) [45] by viewing an intermediate layer as a linear memory, which can be manipulated to incorporate novel content. Afterward, **Editable Training** [133] is proposed to encourage fast editing of the trained model in a model-agnostic manner. The goal is to change the model predictions on a subset of inputs corresponding to misclassified objects, without altering the results for other inputs. In [125], the authors propose a method that allows for the modification of a classifier's behavior by editing its decision rules, which can be used to correct errors or reduce biases in model predictions. In the field of natural language processing, several works [22, 102] have been proposed to perform editing regarding textual information. Specifically, Zhu et al. [177] propose a constrained fine-tuning loss to explicitly modify specific factual knowledge in transformer-based models [146]. More recent works [42, 43] discover that the MLP layers in transformers actually act as key-value memories, thereby enabling the editing of specific knowledge within the corresponding layers.

Table 1. Important notations used in this survey.

Notations	Detailed Descriptions
x	Input (prompt) to LLMs
y	Output of LLMs
(x, y)	Input-output pair
$t = (s, r, o)$	Original knowledge triple (before editing)
$s/r/o$	Subject/Relation/Object in a knowledge triple
$t^* = (s, r, o^*)$	Target knowledge triple (after editing)
$e = (s, r, o \rightarrow o^*)$	Edit descriptor
$\mathcal{X}_e$	In-scope input space
$\mathcal{Y}_e$	Original output space (before editing)
$\mathcal{Y}_e^*$	Target output space (after editing)
$\mathcal{E} = \{e_i\}$	Set of edits
O_e	Out-scope input space
$\mathbf{q}_i^{(l)}/\mathbf{k}_i^{(l)}/\mathbf{v}_i^{(l)}$	Query/Key/Value vector for the i-th head of the l-th attention module in Transformer
$\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}$	Weights of the fully connected layers of the l-th attention module in Transformer
$\mathbf{h}^{(l)}$	Output from the l-th self-attention module in Transformer
$\parallel$	Vector concatenation

2.2 Language Models

*2.2.1 **Transformers.*** Transformers lie in the core of large language models (LLMs) [27, 121, 146]. The fully-fledged transformer possesses an encoder-decoder architecture initially designed for the neural machine translation (NMT) task [137]. Nowadays, transformers have found wide applications in most fields of the NLP community, beyond their original purpose. Generally, a transformer network is constructed from multiple stacks of the self-attention module with residual connections, which is pivotal for capturing contextual information from textual sequences. The self-attention module is composed of a **self-attention layer (SelfAtt)** and a **point-wise feed-forward neural network layer (FFN)** formulated as follows:

$$
\begin{aligned}
\mathbf{h}_i^{A,(l-1)} &= \text{SelfAtt}_i\left(\mathbf{h}_i^{(l-1)}\right) = \text{Softmax}\left(\mathbf{q}_i^{(l)}\left(\mathbf{k}_i^{(l)}\right)^{\top}\right)\mathbf{v}_i^{(l)}, \\
\mathbf{h}^{F,(l-1)} &= \text{FFN}\left(\mathbf{h}^{(l-1)}\right) = \text{GELU}\left(\mathbf{h}^{(l-1)}\mathbf{W}_1^{(l)}\right)\mathbf{W}_2^{(l)}, \mathbf{h}^{(0)} = \mathbf{x}, \\
\mathbf{h}^{(l)} &= \mathbf{h}^{A,(l-1)} + \mathbf{h}^{F,(l-1)} = \Big\|_i \text{SelfAtt}_i\left(\mathbf{h}_i^{(l-1)}\right) + \text{FFN}\left(\mathbf{h}^{(l-1)}\right),
\end{aligned}
\tag{1}
$$

where $\mathbf{q}_i^{(l)}$, $\mathbf{k}_i^{(l)}$, and $\mathbf{v}_i^{(l)}$ represent the sequences of query, key, and value vectors for the i-th attention head of the l-th attention module, respectively. GELU is an activation function. They are calculated from $\mathbf{h}_i^{(l-1)}$, the i-th slice of the outputs from the $(l-1)$-th self-attention module (i.e., $\mathbf{h}^{(l-1)}$), and $\mathbf{x}$ denotes the input sequence of token embeddings. $\parallel$ represents vector concatenation. Normalizing factors in the self-attention layer are omitted for simplicity.

Generally, multi-head self-attention directs the model to attend to different parts of the sequence to predict the next token. Specifically, the prediction is based on different types of relationships and dependencies within the textual data, where the output $\mathbf{h}_i^{A,(l-1)}$ is a weighted sum of the value vector of other tokens. In contrast,

FFN adds new information $\mathbf{h}_i^{F,(l-1)}$ to the weighted sum of the embeddings of the attended tokens based on the information stored in the weights of the fully connected layers, i.e., $\mathbf{W}_1^{(l)}$ and $\mathbf{W}_2^{(l)}$. The final layer outputs of the transformer, i.e., $\mathbf{h}^{(L)}$, can be used in various downstream NLP tasks. For token-level tasks (e.g., part-of-speech tagging [19]), the entire hidden representation sequence $\mathbf{h}^{(L)}$ can be utilized to predict the target sequence. For the sequence-level tasks (e.g., sentiment analysis [160]), the hidden representation of the last token, i.e., $\mathbf{h}_{-1}^{(L)}$, can be considered as a summary of the sequence and thus used for the predictions.

2.2.2 ***Large Language Models (LLMs).*** Transformers with billions of parameters trained on large corpora have demonstrated **emergent ability**, showcasing an unprecedented understanding of factual and commonsense knowledge [173]. Consequently, these models are referred to as large language models (LLMs) to indicate their drastic distinction from traditional small-scale language models [34, 142]. Generally, based on the specific parts of the transformer utilized for language modeling, existing LLMs can be categorized into three classes: encoder-only LLMs, such as BERT [74], encoder-decoder-based LLMs such as **T5** [119], and decoder-only models (also the most common structure in LLMs) such as different versions of **GPT** [118] and **LLaMA** [144].

2.3 Relevant Topics

KME intersects with several extensively researched topics, yet these techniques cannot effectively address KME-specific challenges [141, 161]. The most relevant approach is model fine-tuning [6, 20, 99], including parameter-efficient fine-tuning [89, 158, 170], which requires fewer parameter updates. However, fine-tuning remains computationally intensive and is often impractical for black-box LLMs [172, 173]. Another related area is machine unlearning [105], which aims to remove the influence of individual samples from models. Unlike KME, which focuses on abstract and generalized knowledge updates, machine unlearning targets the elimination of specific training data, making it unsuitable for KME. On the other hand, external memorization KME methods share similarities with RAG (retrieval-augmented generation) [40], where a large repository of documents is stored and retrieved as needed to provide contextually relevant information for generating responses. While RAG can introduce new knowledge into LLMs by retrieving recently added documents, it does not effectively update the inherent knowledge within LLMs. Thus, RAG is not suitable for the fundamental knowledge updates that KME seeks to achieve.

3 Problem Formulation

In this section, we provide a formal definition for the knowledge-based model editing (KME) task for pre-trained LLMs, where a general formulation of the KME objective is formulated to encompass specific KME strategies. The task of KME for LLMs can be broadly defined as the process of precisely modifying the behavior of pre-trained LLMs, such that new knowledge can be incorporated to maintain the currentness and relevancy of LLMs can be maintained, without negatively influencing other pre-trained knowledge irrelevant to the edits. To provide a clear formulation, we present the definitions of different terms used in KME, where the overall process is illustrated in Fig. 2.

Editing Target. In this survey, we represent the knowledge required to be injected into LLMs as a **knowledge triple** $t = (s, r, o)$, where s is the subject (e.g., *president of the USA*), r is the relation (e.g., *is*), and o is the object (e.g., *Biden*). From the perspective of knowledge triple, the objective of KME for LLMs is to modify the **original knowledge triple** $t = (s, r, o)$ encoded in the pre-trained weights of the model into the **target knowledge triple** $t^* = (s, r, o^*)$, where o^* is the target object different from o. In this manner, we can define an **edit** as a tuple $e = (t, t^*) = (s, r, o \rightarrow o^*)$, which denotes the update of the obsolete old knowledge t into the new knowledge t^*.

Input and Output Space. Given a pair of subject s and relation r, in order to query LLMs to obtain the object o, (s, r) needs to be transformed into natural language, which we denoted as x. x is also referred to as the prompt in this survey. The LLM output y is also textual and can be converted back to an object o as the query result. In this way, (x, y) can be considered as the natural language input-output pair associated with the knowledge triple $t = (s, r, o)$.

Fig. 2. The formulation of the KME objective.

For example, the prompt x transformed from s and r can be *"The president of the USA is"*, and y is the model output *"Joe Biden"*. Note that due to the diversity of natural language, multiple (x, y) pairs can be associated with the same knowledge triple t. We denote the set of textual inputs associated with subject s and relation r in an edit e as $\mathcal{X}_e = I(s, r)$, referred to as **in-scope input space**. Similarly, we define the set of textual outputs that can be associated with the object o in the same edit e as $\mathcal{Y}_e^* = O^*(s, r, o^*)$ (i.e., **target output space**), and the original textual output space as $\mathcal{Y}_e = O(s, r, o)$ (i.e., **original output space**). Given an edit e, the aim of KME is to modify the behavior of language models from $\mathcal{Y}_e$ to $\mathcal{Y}_e^*$, regarding the input in $\mathcal{X}_e$. To accommodate the scenarios where multiple edits are performed, we can define the union of $\mathcal{X}_e$ over a set of edits $\mathcal{E} = \{e_1, e_2, \dots\}$ as $\mathcal{X}_\mathcal{E} = \bigcup_{e \in \mathcal{E}} \mathcal{X}_e$. Similarly, we can define $\mathcal{Y}_\mathcal{E} = \bigcup_{e \in \mathcal{E}} \mathcal{Y}_e$ and $\mathcal{Y}_\mathcal{E}^* = \bigcup_{e \in \mathcal{E}} \mathcal{Y}_e^*$.

Formulation. We denote the pre-trained LLM with parameter ϕ as $f : \mathcal{X} \to \mathcal{Y}$ and the edited model with updated parameter ϕ^* as $f^* : \mathcal{X} \to \mathcal{Y}^*$. The objective of knowledge-based model editing is to precisely update the pre-trained LLM f into f^* according to edits in the edit set $\mathcal{E}$ such that for each edit e and for each $y \in \mathcal{Y}_e$, the changes to the input-output pairs irrelevant to the edits is minimized. The problem of KME can be formulated as follows:

DEFINITION 1. *The objective for KME on a series of edits $\mathcal{E}$ is represented as follows:*

$$\min \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x, y^* \in \mathcal{X}_e, \mathcal{Y}_e^*} \mathcal{L}(f^*(x), y^*), \text{ where } f^* = M(f; \mathcal{E}),$$
$$\text{s.t. } f^*(x) = f(x), \ \ \forall x \in \mathcal{X} \setminus \mathcal{X}_\mathcal{E}, \tag{2}$$

where $\mathcal{L}$ is a specific loss function that measures the discrepancy between the model output $f^*(x)$ and y^* from the desirable response set $\mathcal{Y}_e^*$. $M(f; \mathcal{E})$ denotes the modification applied to f based on the desirable edits $\mathcal{E}$.

From the above definition, we can summarize two crucial perspectives regarding the objective of KME: (1) **Generality**, which requires that the correct answers in the target output space $\mathcal{Y}_e^*$ can be achieved, provided prompts in the in-scope input space, i.e., $\mathcal{X}_e$, where the target knowledge triple $t^* \in e$ can be updated into the pre-trained model; (2) **Locality**, which requires the consistency of model output regarding unrelated input, i.e., $\mathcal{X} \setminus \mathcal{X}_\mathcal{E}$, where valuable pre-trained knowledge can be maximally preserved after the editing. Here, we note that locality is especially important for editing LLMs, as the knowledge that needs to be updated often occupies only a small fraction of all knowledge encompassed by the pre-trained model. In other words, the output of an edited model regarding most input prompts should remain consistent with the output before editing.

4 Evaluation Metrics

Before introducing the taxonomy of KME and the exemplar methods in detail, in this section, we first discuss various metrics commonly used to evaluate the effectiveness of different KME strategies from varied perspectives. We summarize the metrics to facilitate the understanding in terms of the properties and advantages of different methods.
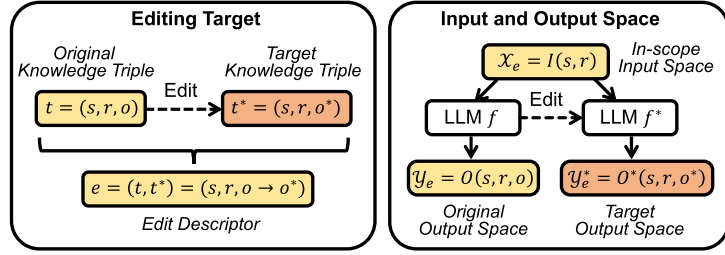
4.1 Accuracy

Accuracy is a straightforward metric for evaluating the effectiveness of KME techniques [17, 29, 79, 101, 106, 174, 175], defined as the success rate of editing in terms of a specific set of *pre-defined* input-output pairs (x_e, y_e^*) associated with all the edited knowledge. Accuracy can be easily defined to evaluate the performance of KME on classification tasks, e.g., fact checking [102, 114], where the answers y are categorical. Defining the prompt and ground truth related to an edit e as x_e and y_e^*, respectively, the metric of the accuracy of an edited model f^* is formulated as follows:

$$\mathbf{Acc}(f^*; \mathcal{E}) = \mathbb{E}_{e \in \mathcal{E}} \mathbb{1}\{f^*(x_e) = y_e^*\}. \tag{3}$$

Since accuracy is defined on a deterministic set of prompt-answer pairs, it provides a fair comparison between KME methods [22, 97, 98]. Nevertheless, it is non-trivial to evaluate the practicality of KME methods with accuracy, as there is no consensus on how to design the $\mathcal{E}$, especially when the task needs to output a long sequence such as question answering or text generation [29, 97, 98].

4.2 Locality

One crucial metric for the KME strategies is ***locality*** [17, 25, 83, 101], which reflects the capability of the edited model f^* to preserve the pre-trained knowledge in f irrelevant to the edits in $\mathcal{E}$. Note that in most KME applications, the number of required edits makes for an extremely small fraction of the entire knowledge learned and preserved in the pre-trained LLMs [167, 172]. Consequently, the locality measurement is of great importance in assessing the capability of edited models to preserve unrelated knowledge [49, 95, 104]. Given an edit e, the edited model f^*, and the original pre-trained model f, the locality of f^* can be defined as the expectation of matched agreement between the edited model and unedited model for out-scope inputs, which can be defined as follows:

$$\mathbf{Loc}(f^*, f; e) = \mathbb{E}_{x \notin \mathcal{X}_e} \mathbb{1}\{f^*(x) = f(x)\}. \tag{4}$$

We can also consider the locality regarding the entire edit set $\mathcal{E}$, which can be defined as follows:

$$\mathbf{Loc}(f^*, f; \mathcal{E}) = \mathbb{E}_{x \notin \mathcal{X}_\mathcal{E}} \mathbb{1}\{f^*(x) = f(x)\}, \quad \text{where } \mathcal{X}_\mathcal{E} = \bigcup_{e \in \mathcal{E}} \mathcal{X}_e. \tag{5}$$

Although the above metric measures the overall locality of f^* based on all inputs that are not in $\mathcal{X}_\mathcal{E}$, it is difficult to compute in realistic scenarios, as the entire input space can be excessively large or even infinite [167]. Therefore, existing methods generally resort to alternative solutions that pre-define the specific range of out-scope inputs to calculate the locality metric [15, 22, 25, 82, 97]. For example, in **SERAC** [102], the authors generate hard out-scope examples from the dataset **zsRE** [78] by selectively sampling from training inputs with high semantic similarity with the edit input, based on embeddings obtained from a pre-trained semantic embedding model. Denoting the out-scope input space related to the input $\mathcal{X}_e$ as O_e, we can similarly define the feasible out-scope input space for multiple edits as $O_\mathcal{E} = \bigcup_{e \in \mathcal{E}} O_e$. In this manner, we define a specific metric of locality of f^* regarding $\mathcal{E}$ as follows:

$$\mathbf{Loc}(f^*, f; O_e) = \mathbb{E}_{x \in O_e} \mathbb{1}\{f^*(x) = f(x)\}, \tag{6}$$

$$\mathbf{Loc}(f^*, f; O_\mathcal{E}) = \mathbb{E}_{x \in O_\mathcal{E}} \mathbb{1}\{f^*(x) = f(x)\}, \quad \text{where } O_\mathcal{E} = \bigcup_{e \in \mathcal{E}} O_e. \tag{7}$$

4.3 Generality

Aside from locality, another crucial metric is ***generality***, which indicates the capability of the edited model f^* to correctly respond to semantically similar prompts [13, 101, 106, 130, 177]. This requires the generalization of the updated knowledge to other in-scope inputs that do not appear in the training set while conveying similar or related meanings [50, 163]. As such, ensuring the generality of edited models prevents the edited model

from overfitting to a particular input [172]. Specifically, in the scenarios of knowledge-based model editing, the inherent diversity of natural language determines that various in-scope inputs x can correspond to a specific knowledge triple t [152]. These semantically equivalent inputs can involve differences in aspects such as syntax, morphology, genre, or even language. Existing works mostly pre-define a specific in-scope input space of each edit via different strategies [61, 86, 136, 166, 168]. For example, in the CounterFact dataset proposed in **ROME** [97], the authors utilize prompts that involve distinct yet semantically related subjects as the in-scope input. In general, the generality of an edited model f^* is defined as the expectation of exact-match agreement between the output of the edited model and true labels for in-scope inputs, which can be defined on either an edit e or the edit set $\mathcal{E}$ as:

$$\mathbf{Gen}(f^*; e) = \mathbb{E}_{x \in \mathcal{X}_e} \mathbb{1}\{f^*(x) \in \mathcal{Y}_e^*\}, \tag{8}$$

$$\mathbf{Gen}(f^*; \mathcal{E}) = \mathbb{E}_{x \in \mathcal{X}_{\mathcal{E}}} \mathbb{1}\{f^*(x) \in \mathcal{Y}_e^*\}, \quad \text{where } \mathcal{X}_{\mathcal{E}} = \bigcup_{e \in \mathcal{E}} \mathcal{X}_e. \tag{9}$$

4.4 Portability

In addition to generality, another vital metric is **portability**, which measures the effectiveness of the edited model f^* in transferring a conducted edit to other *logically related edits* that can be interpreted via reasoning [172]. For example, if an edit is conducted towards the President of the USA, the edit regarding the query "Which political party does the current President of the USA belong to?" should also be achieved. This ensures that the edited model is not limited to responding to specific input formats. In concrete, the transfer of knowledge is crucial for robust generalization of the edited model. In practice, portability can be assessed with logically related edits obtained in different ways [21, 167]. Denoting an edit as $e = (s, r, o \rightarrow o^*)$, hereby we introduce two common types of logically related edits $\tilde{e}$. (1) Reversed Relation: $\tilde{e} = (o \rightarrow o^*, \tilde{r}, s)$, where $\tilde{r}$ is the reversed relation of r, and (2) Neighboring Relation: $\tilde{e} = (s, r \oplus r_\epsilon, \epsilon \rightarrow \epsilon^*)$, where both $(o, r_\epsilon, \epsilon)$ and $(o^*, r_\epsilon, \epsilon^*)$ exist in the pre-trained knowledge, and $r \oplus r_\epsilon$ is a combined relation from r and r_ϵ. In this manner, we define portability as the edited model performance on one or multiple logically related edits as follows:

$$\mathbf{Por}(f^*; \tilde{e}) = \mathbb{E}_{x \in \mathcal{X}_{\tilde{e}}} \mathbb{1}\{f^*(x) \in \mathcal{Y}_{\tilde{e}}^*\}, \tag{10}$$

$$\mathbf{Por}(f^*; \widetilde{\mathcal{E}}) = \mathbb{E}_{x \in \mathcal{X}_{\widetilde{\mathcal{E}}}} \mathbb{1}\{f^*(x) \in \mathcal{Y}_{\tilde{e}}^*\}, \quad \text{where } \mathcal{X}_{\widetilde{\mathcal{E}}} = \bigcup_{\tilde{e} \in \widetilde{\mathcal{E}}} \mathcal{X}_{\tilde{e}}. \tag{11}$$

4.5 Retainability

Retainability characterizes the ability of KME techniques to preserve the desired properties of edited models after **multiple consecutive edits** [47, 69, 169]. In the presence of ever-evolving information, practitioners may need to frequently update a conversational model (i.e., sequential editing). Such a KME setting requires that the model does not forget previous edits after each new modification [81]. It is essential to distinguish retainability from scalability, which evaluates the model's ability to handle a vast number of edits [15]. In contrast, retainability assesses the consistent performance of the model after each individual edit, presenting a more challenging objective to achieve. Recently, **T-Patcher** [66] first explores the sequential setting of KME and observes that many existing approaches significantly fall short in terms of retainability. In **SLAG** [53], the authors also discover a significant drop in editing performance when multiple beliefs are updated continuously. To assess the retainability of an edited language model f^*, we define it as follows:

$$\mathbf{Ret}(M; \mathcal{E}) = \frac{1}{|\mathcal{E}| - 1} \sum_{i=1}^{|\mathcal{E}|-1} \mathbf{Acc}(M(f; \{e_1, e_2, \dots, e_{i+1}\})) - \mathbf{Acc}(M(f; \{e_1, e_2, \dots, e_i\})) \tag{12}$$

where **Acc** is the accuracy measurement, $|\mathcal{E}|$ is the number of edits in the edit set, and M denotes the editing strategy that modifies the pre-trained model f into f^* with $i/i + 1$ consecutive edits $\{e_1, e_2, \ldots, e_i, (e_{i+1})\}$. The retainability metric aims to quantify the effect of applying consecutive edits to a model and measures how the performance will change the editing strategy M, where a higher retainability means that after each edit, the less the change in the overall performance of the edited model f^* is required.

4.6 Scalability

The ***scalability*** of an editing strategy refers to its capability to incorporate a large number of edits simultaneously [15]. Recently, several works have emerged that can inject multiple new knowledge into specific parameters of pre-trained LLMs [168, 172]. For instance, **SERAC** [102] can perform a maximum of 75 edits. In addition, **MEMIT** [98] is proposed to enable thousands of edits without significant influence on editing accuracy. When there is a need to edit a model with a vast number of edits concurrently, simply employing the current knowledge-based model editing techniques in a sequential manner is proven ineffective in achieving such scalability [167]. To effectively evaluate the scalability of edited language models, we define the scalability of an edited model as follows:

$$\mathbf{Sca}(M; \mathcal{E}) = \mathbb{E}_{e \in \mathcal{E}} \mathbf{Acc}(M(f; e)) - \mathbf{Acc}(M(f; \mathcal{E})), \tag{13}$$

where $\mathbf{Acc}(M(f; \mathcal{E}))$ denotes the accuracy of the edited model after conducting all edits in $\mathcal{E}$, whereas $\mathbf{Acc}(M(f; e))$ is the accuracy of only performing the edit e. **Sca** demonstrates the model performance and practicality in the presence of multiple edits. Nevertheless, we note that baseline value $\mathbf{Acc}(M(f; \{e\}))$ is also important in evaluating the scalability of various models. This is because, with higher accuracy for each e, the retainment of such performance after multiple edits is more difficult. Therefore, we further define the relative version of Eq. (13) as follows:

$$\mathbf{Sca}_{rel}(M; \mathcal{E}) = (\mathbb{E}_{e \in \mathcal{E}} \mathbf{Acc}(M(f; \{e\})) - \mathbf{Acc}(M(f; \mathcal{E}))) / \mathbb{E}_{e \in \mathcal{E}} \mathbf{Acc}(M(f; \{e\})). \tag{14}$$

The introduced scalability measurement further considers the magnitude of the original accuracy to provide a fairer evaluation.

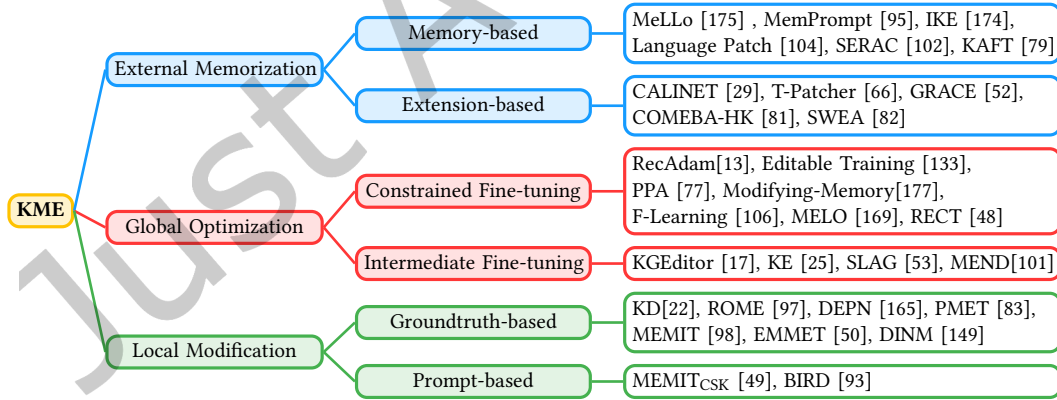Fig. 3. The categorization of KME techniques for LLMs and the corresponding works.

5 Methodologies

In this section, we introduce existing knowledge-based model editing (KME) strategies in detail. We first provide an innovative taxonomy of existing KME strategies based on how and where the new knowledge is injected into the pre-trained LLMs, where the advantages and drawbacks are thoroughly discussed. We then introduce various

methods from each category, with an emphasis on analyzing the technical details, insights, shortcomings, and their relationships.

5.1 Categorization of KME Methods

Faced with the rapid deprecation of old information and the emergence of new knowledge, various KME methodologies have been proposed to update the pre-trained LLMs to maintain their updatedness and relevancy. KME ensures that new knowledge can be efficiently incorporated into the pre-trained LLMs without negatively influencing the pre-trained knowledge irrelevant to the edit. In this survey, we categorize existing KME methods into three main classes as follows:

- **External Memorization**-based methods **leverage an external memory** to store the new knowledge for editing without modifying the pre-trained weights, where the pre-trained knowledge can be fully preserved in the LLM weights. By storing new knowledge with external parameters, the memory-based strategies enable precise representation of new knowledge with good scalability, as the memory is easily extensible to incorporate new knowledge.
- **Global Optimization**-based methods seek to **achieve generalizable incorporation** of the new knowledge into pre-trained LLMs via optimization with the guidance of new knowledge, where tailored strategies are introduced to limit the influence of other pre-trained knowledge, distinguishing it from naive fine-tuning. Nevertheless, these methods may fall short in editing efficiency when applied to LLMs due to the large number of parameters to be optimized.
- **Local Modification**-based methods aim to **locate the related parameters** of specific knowledge in LLMs and update it accordingly to incorporate the new knowledge relevant to the edit. The main advantage of local modification is the possibility of only updating a small fraction of model parameters, thereby providing considerable memory efficiency compared to memorization-based methods and computational efficiency compared to global optimization.

The above categorization is achieved based on **where** (e.g., external parameters or internal weights) and **how** (e.g., via optimization or direct incorporation) new knowledge is introduced into the LLM during editing. Methods in each category exhibit different strengths and weaknesses regarding the four crucial evaluation metrics introduced in Sec. 4. For example, external memorization prevails in scenarios that require massive editing while the computational resources are limited, as the size of the memory is controllable to fit into different requirements. On the other hand, global optimization is advantageous when practitioners focus more on the generality of edited knowledge, as the optimization can promote the learning of relevant knowledge [2]. The taxonomy is visually illustrated in Fig. 3, and a more detailed demonstration of each category is presented in Fig. 4.

5.2 External Memorization

*5.2.1 **Overview**.* The editing approaches via external memorization aim to modify the current model f_ϕ (with parameter ϕ) via introducing external memory represented by additional trainable parameters ω that encodes the new knowledge, resulting in an edited LLM model $f^*_{\phi,\omega}$. The rationale behind the external memorization strategy is that storing new knowledge in additional parameters is intuitive and straightforward to edit the pre-trained LLMs with good scalability, as the parameter size can be expanded to store more knowledge. In addition, the influence on the pre-trained knowledge can be minimized as this strategy does not alter the original parameters ϕ. Based on the general formulation of KME in Eq. (2), the objective of external memorization approaches can be formulated as follows:

$$
\begin{aligned}
&\min \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x,y^* \in \mathcal{X}_e, \mathcal{Y}_e^*} \mathcal{L}(f^*_{\phi,\omega}(x), y^*), \text{ where } f^*_{\phi,\omega} = M(f_\phi, \omega; \mathcal{E}), \\
&\text{s.t. } f^*_{\phi,\omega}(x) = f_\phi(x), \quad \forall x \in \mathcal{X} \setminus \mathcal{X}_{\mathcal{E}},
\end{aligned}
\tag{15}
$$

Fig. 4. The illustration of three categories of KME methods: **External Memorization**, **Global Optimization**, and **Local Modification**.

where f_ϕ denotes the LLM before editing with the pre-trained parameter ϕ, and $f^*_{\phi,\omega}$ denotes the edited LLM with ϕ and additional parameter ω as the external memorization. Moreover, based on whether the introduced parameters are directly incorporated into the model process or not, external memorization strategies can be divided into two categories, i.e., **memory-based methods** and **extension-based methods**.

5.2.2 Memory-based Strategies. In memory-based strategies, the external memory, outside the intrinsic architecture of the pre-trained LLM, functions as a repository to store edited knowledge. Here the edits are generally converted to text via pre-defined templates [154, 174, 175]. The LLM can access and update this memory as required during inference.

One exemplar work is **SERAC** [102], which stores the edited samples $x, y^* \in \mathcal{X}_e, \mathcal{Y}^*_e$ in a cache without performing modifications on the original model. When presented with a new prompt x', SERAC uses a scope classifier to determine whether the prompt falls within the scope of any cached instances. If yes, the desirable output y' associated with the new prompt x' is predicted via a counterfactual model f_c which utilizes the most relevant edit example as follows:

$$f^*_{\phi,\omega}(x) = \begin{cases} f_\phi(x), & \text{if } x \text{ is not in scope of any edit,} \\ f_c(x, \mathcal{E}), & \text{otherwise.} \end{cases} \tag{16}$$

SERAC is a gradient-free approach to KME without relying on gradients of the target label y^* w.r.t. the pre-trained model parameters. In addition to using memory as an external repository, the desirable edits can also be stored in the form of human feedback. For example, **Language Patch** [104] performs editing by integrating patches in natural language, and **MemPrompt** [95] involves human feedback prompts to address the issue of lacking commonsense knowledge regarding a particular task. An integral feature of the Language Patch [104] framework is its ability to empower practitioners with the capability to create, edit, or remove patches without necessitating frequent model re-training. This trait not only streamlines the development process but also enhances the adaptability and versatility of the edited model. To enable the automatic correction in memory, MemPrompt [95] equips the language model with a memory bank containing corrective feedback to rectify misunderstandings. Specifically, MemPrompt leverages question-specific historical feedback to refine responses on novel and unencountered instances through prompt adjustments.

In **KAFT** [79], controllability is achieved through the utilization of counterfactual data augmentations. In this approach, the entity representing the answer within the context is substituted with an alternative but still plausible entity. This substitution is intentionally designed to introduce a conflict with the genuine ground truth, thereby enhancing the controllability and robustness of LLMs with respect to their working memory. The aim is

to ensure that LLMs remain responsive to pertinent contextual information while filtering out noisy or irrelevant data.

In addition to relying on parameter-based memory, recent works also leverage prompting techniques of LLMs, e.g., in-context learning [30] and chain-of-thought prompting [162], to promote editing performance of external memorization. Specifically, **IKE** [174] introduces novel factual information into a pre-trained LLM via in-context learning, where a set of k demonstrations, i.e., $\omega = \{x_i, y_i^*\}_{i=1}^k$, is selected as the reference points. These demonstrations will alter the prediction of a target factual detail when the input is influenced by an edit. Particularly, IKE guarantees a balance between generality and locality via storing factual knowledge as prompts. The process can be formulated as follows:

$$f_{\phi,\omega}^*(x) = f_\phi(\omega \| x), \text{ where } \omega = \{x_i, y_i^*\}_{i=1}^k. \tag{17}$$

Here $\|$ denotes the concatenation of the reference points in ω and the input x, which follows an in-context learning manner. Note that in this process, the framework first transforms all new facts into natural language to input them into LLMs. Similar methods of knowledge editing based on prompts [15, 131, 136, 154] can also update and modify knowledge within large language models (LLMs). These approaches allow users to guide the model to generate desired outputs by providing specific prompts, and effectively and dynamically adjusting the model's knowledge base. By leveraging the flexibility of prompts and the contextual understanding of LLMs, users can correct or update information in real-time. These methods offer immediacy, flexibility, and cost-efficiency, making them powerful tools for maintaining the accuracy and relevance of language models in rapidly evolving knowledge domains. Although the prompt approaches effectively edit factual knowledge via in-context learning, they cannot solve more complex questions that involve multiple relations. To deal with this, **MeLLo** [175] first explores the evaluation of the editing effectiveness in language models regarding multi-hop knowledge. For example, when editing knowledge about the president of the USA, the query regarding the president's children should change accordingly. MeLLo proposes to enable multi-hop editing by breaking down each query into subquestions, such that the model generates a provisional answer. Subsequently, each subquestion is used to retrieve the most pertinent fact from the memory to assist the model in answering the query.

5.2.3 ***Extension-based Strategies****.* Extension-based strategies utilize supplementary parameters to assimilate modified or additional information into the original language model. These supplementary parameters are designed to represent the newly introduced knowledge or necessary adjustments tailored for specific tasks or domains. Different from memory-based methods, by incorporating new parameters into the language model, extension-based approaches can effectively leverage and expand the model's functionality.

Extension-based methods can be implemented through various means, and one representative way is to modify the Feed-forward Neural Network (FFN) output. For example, **CALINET** [29] uses the output from sub-models fine-tuned specifically on factual texts to refine the original FFN output produced by the base model. Another technique **T-Patcher** [66] introduces a limited number of trainable neurons, referred to as "patches," in the final FFN layer to alter the model's behavior while retaining all original parameters to avoid reducing the model's overall performance. Generally, these methods that refine the structure of FFN can be formulated as follows:

$$\text{FFN}(\mathbf{h}) = \text{GELU}(\mathbf{h}\mathbf{W}_1)\mathbf{W}_2 + \text{GELU}(\mathbf{h} \cdot \mathbf{k}_p + b_p) \cdot \mathbf{v}_p, \tag{18}$$

where $\mathbf{k}_p$ is the patch key, $\mathbf{v}_p$ is the patch value, and b_p is the patch bias scalar. The introduced patches are flexible in size and can be accurately activated to edit specific knowledge without affecting other model parameters.

Alternatively, a different technique involves integrating an adapter into a specific layer of a pre-trained model. This adapter consists of a discrete dictionary comprising keys and values, where each key represents a cached activation generated by the preceding layer and each corresponding value decodes into the desired model output. This dictionary is systematically updated over time. In line with this concept, **GRACE** [52] introduces an adapter

that enables judicious decisions regarding the utilization of the dictionary for a given input, accomplished via the implementation of a deferral mechanism. It is crucial to achieve a balance between the advantages of preserving the original model's integrity and the practical considerations associated with storage space when implementing this approach. **COMEBA-HK** [81] incorporates hook layers within the neural network architecture. These layers allow for the sequential editing of the model by enabling updates to be applied in batches. This approach facilitates the integration of new knowledge without requiring extensive retraining of the entire model, making it a scalable solution for continuous learning and adaptation. **SWEA** [82] focuses on altering the embeddings of specific subject words within the model. By directly updating these embeddings, the method can inject new factual knowledge into the LLMs. This approach ensures that the updates are precise and relevant, thereby enhancing the model's ability to reflect current information accurately.

5.2.4 **Summary**. The eternal memorization methodology operates by preserving the parameters within the original model while modifying specific output results through external interventions via memory or additional model parameters. One notable advantage of this approach is its minimal perturbation of the original model, thereby ensuring the consistency of unedited knowledge. It allows for precise adjustments without necessitating a complete overhaul of the model's architecture. However, it is imperative to acknowledge a trade-off inherent in this methodology. Its efficacy is contingent upon the storage and invocation of the edited knowledge, a factor that leads to concerns regarding storage capacity. Depending on the scale of knowledge to be edited, this approach may entail substantial storage requisites. Therefore, cautiously seeking a balance between the advantages of preserving the original model's integrity and the practical considerations of storage capacity becomes a pivotal concern when employing this particular approach.

5.3 Global Optimization

5.3.1 **Overview**. Different from external memorization methods that introduce new parameters to assist the editing of pre-trained LLMs, there also exist branches of works that do not rely on external parameters or memory. Concretely, global optimization strategies aim to inject new knowledge into LLMs by updating all parameters, i.e., ϕ in Eq. (15). Through fine-tuning model parameters with specific designs to ensure the preservation of knowledge irrelevant to the target knowledge t^*, the LLMs are endowed with the ability to absorb new information without altering unedited knowledge. Generally, the goal of global optimization methods can be formulated as follows:

$$
\min \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x, y^* \in \mathcal{X}_e, y_e^*} \mathcal{L}(f_{\phi^*}(x), y^*), \text{ where } f_{\phi^*} = M(f_\phi; \mathcal{E}),
$$
$$
\text{s.t. } f_{\phi^*}(x) = f_\phi(x), \quad \forall x \in \mathcal{X} \setminus \mathcal{X}_{\mathcal{E}},
$$
(19)

where f_ϕ denotes the LLM before editing with the pre-trained parameter ϕ, and f_{ϕ^*} denotes the edited LLM with updated parameter ϕ^*. Generally, these methods focus more on the precision and generality of desirable knowledge, as the fine-tuning process ensures that the LLMs achieve satisfactory results regarding the edits and relevant knowledge. Nevertheless, as fine-tuning affects all parameters, they cannot easily preserve the locality of edited models, i.e., maintaining consistent output for unedited knowledge [167]. In practice, directly applying fine-tuning strategies typically exhibits suboptimal performance on KME due to overfitting concerns [98, 152]. Furthermore, fine-tuning large language models is also time-consuming and lacks scalability for multiple edits. Therefore, recently, motivated by these two challenges in fine-tuning, several global optimization works have been proposed and can be categorized as **constrained fine-tuning methods** and **intermediate fine-tuning methods**. Note that this section primarily focuses on methods from the model training perspective. Additionally, certain studies [38, 69] address the overfitting challenge by constructing more a comprehensive $\mathcal{X}'_{\mathcal{E}}$ with the following fine-tuning goal:

$$
\min \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x, y^* \in \mathcal{X}'_e, y_e^*} \mathcal{L}(f_{\phi^*}(x), y^*), \text{ where } f_{\phi^*} = M(f_\phi; \mathcal{E}),
$$
$$
\text{s.t. } \mathcal{X}_{\mathcal{E}} \subset \mathcal{X}_{\mathcal{E}}', \mathcal{X}_{\mathcal{E}}' \subseteq \mathcal{X}.
$$
(20)

5.3.2 ***Constrained Fine-tuning***. Constrained fine-tuning strategies generally apply specific constraints to prevent updating on non-target knowledge in $\{X \setminus X_{\mathcal{E}}, \mathcal{Y} \setminus \mathcal{Y}_{\mathcal{E}}\}$. In this manner, the objective in Eq. (20) is transformed into a constrained optimization problem:

$$
\begin{aligned}
&\min \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x, y^* \in X_e, \mathcal{Y}_e^*} \mathcal{L}(f_{\phi^*}(x), y^*), \text{ where } f_{\phi^*} = M(f_\phi; \mathcal{E}), \\
&\text{s.t. } \|\mathcal{L}(f_{\phi^*}(x), y) - \mathcal{L}(f_\phi(x), y)\| \le \delta, \forall x, y \in X \setminus X_{\mathcal{E}}, \mathcal{Y} \setminus \mathcal{Y}_{\mathcal{E}},
\end{aligned}
\tag{21}
$$

where ϕ, ϕ^* are the parameters before and after updating, respectively. δ is a scalar hyper-parameter to restrict the difference between losses of f_{ϕ^*} and f_ϕ. The constraint in Eq. (21) restricts the change of the edited model on unmodified knowledge. Zhu et al. [177] first propose an approximate optimization constraint that is easier for implementation and computation:

$$
\begin{aligned}
&\min \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x, y^* \in X_e, \mathcal{Y}_e^*} \mathcal{L}(f_{\phi^*}(x), y^*), \text{ where } f_{\phi^*} = M(f_\phi; \mathcal{E}), \\
&\text{s.t. } \|\phi^* - \phi\| \le \delta.
\end{aligned}
\tag{22}
$$

The updates are regularized by restricting the norm of parameters before and after updating. **RECT** [48] adopts a similar yet simpler approach, specifically modifying only the top-k% of parameters with the largest numerical updates during fine-tuning. Although restricting the norm is helpful in preventing the forgetting of original knowledge, the fine-tuning process can be less effective. To deal with this, **RecAdam** [13], in addition to the norm constraint, applies an annealing technique to control the ratio between the parameter norm and the fine-tuning loss as follows:

$$
\mathcal{L}_{total} = \lambda(t) \mathcal{L}_{FT} + (1 - \lambda(t)) \|\phi^* - \phi\|, \text{ where } \lambda(t) = \frac{1}{1 + \exp(-k \cdot (t - t_0))}.
\tag{23}
$$

Here k and t_0 are hyper-parameters. t is the number of fine-tuning steps. Such a design enables a gradual fine-tuning process that prevents massive parameter updates at the beginning. Motivated by the intuition of regularization to preserve original knowledge, **PPA** [77] employs LoRA [62] in the feed-forward (FFN) layers of the transformer decoder. LoRA is proposed to train the expansion/reduction matrix, instead of the model parameter ϕ, to improve training speed by only updating parameters with a low intrinsic rank via dimensionality reduction. PPA leverages plug-in modules trained with constraints via LoRA to keep original knowledge intact. Moreover, the authors assess whether the content of the inputs falls within the scope of $X_{\mathcal{E}}$ using the K-adapter module [153], and redirect such inputs to the new plug-in modules. This information is then used to determine whether to employ LoRA within the FFN layers. Furthermore, **MELO** [169] clusters the edits and employs multiple non-overlapping LoRA blocks for fine-tuning each cluster separately, thereby mitigating the issue of catastrophic forgetting. **F-Learning** (Forgetting before Learning) [106] proposes another approach to preserve original knowledge, which learns knowledge parameters $\Delta\phi$ that indicates old knowledge to be forgotten, defined as follows:

$$
\phi^* = \phi - \lambda \Delta\phi, \text{ where } \Delta\phi = \text{FT}(\phi; \mathcal{K}_{old}) - \phi.
\tag{24}
$$

Here $\mathcal{K}_{old}$ denotes the dataset composed of old knowledge that we desire to forget, and $\text{FT}(\phi; \mathcal{K}_{old})$ is the supervised fine-tuning process of parameters ϕ on dataset $\mathcal{K}_{old}$. λ is a hyper-parameter used to control the rate of forgetting. Based on the assumption that subtracting the parameters $\Delta\phi$ from ϕ can help the model forget this part of old knowledge [68], F-Learning defines the forgetting process as a subtraction operation to obtain the updated model parameter ϕ^*.

On the other hand, other works also resort to meta-learning [36, 145] to apply more flexible constraints. Meta-learning addresses the issue of overfitting by training a model that can quickly adapt to new tasks [60]. By exposing the model to a variety of tasks during training, meta-learning improves the model's ability to generalize from limited data and reduces the risk of overfitting individual tasks [67]. In the scenario of KME, the optimal

model parameters ϕ^* should minimize the expected loss over a variety of meta-tasks [120]:

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\, \mathbb{E}_{D \sim \mathcal{D}}[\mathcal{L}_\phi(D)], \tag{25}$$

where $\mathcal{D}$ corresponds to the sample set for each meta-task D. Moreover, each meta task D contains multiple (x^*, y^*) pairs for editing. In practice, such methods often introduce additional objective functions or networks to regulate parameter updates. As a typical meta-learning method for KME, **Editable Training** [133] focuses on effectively rectifying errors within models while preserving their performance on other irrelevant data instances. Following a model-agnostic training manner, the authors introduce additional constraints to restrict parameter updates in a different way. Specifically, the loss function is separated into $\mathcal{L}_{base}$ (task-specific objective function), $\mathcal{L}_{edit}$ (computed on the edit set $\mathcal{X}_\mathcal{E}$), and $\mathcal{L}_{local}$ (computed on samples in $\mathcal{X} \setminus \mathcal{X}_\mathcal{E}$). Moreover, the models are updated in a meta-learning manner, where k steps of gradient descent would be applied for parameters before computing the objective function.

*5.3.3 **Intermediate Fine-tuning Strategies**.* While constrained fine-tuning techniques have demonstrated remarkable efficacy in a variety of NLP tasks [7, 164, 179], they still exhibit instability and high computational cost when applied to KME, primarily due to the necessity of altering all parameters [167]. A potential solution to address this challenge is to utilize an intermediate model to obtain the updated parameters in an efficient manner. Such an intermediate model is required to maintain significantly fewer parameters to ensure efficiency [17]. In general, recent works have widely adopted the **Hyper-Network** [51] as the intermediate model. Specifically, the Hyper-Network is a small network that generates the weights for a larger network, referred to as the main network. Specifically, the Hyper-Network takes inputs that contain information about the structure of the weights and generates the weights for layers in the main network. With the generated weights, the main network is updated to map input data to desired output targets. The updating process for the main network, denoted as ϕ, can be defined as follows:

$$\phi^* = \phi + \Delta\phi, \quad \text{where } \Delta\phi = \text{H}(\nabla_\phi \mathcal{L}(f_\phi(x), y^*)) \text{ and } x, y^* \in \mathcal{X}_\mathcal{E}, \mathcal{Y}_\mathcal{E}^*, \tag{26}$$

where $\text{H}(\cdot)$ denotes the hyper-network. $\Delta\phi$ is the weight deviation calculated by the hyper-network. According to a recent study [147], task-specific Hyper-Networks (i.e., networks that generate target model weights based on task attributes) are effective in mitigating catastrophic forgetting issues. Therefore, such methods are suitable for the setting of KME, which requires the preservation of unedited knowledge.

Recently, researchers have proposed to adopt hyper-networks in various ways for parameter updates in KME. As a classic example, **KE** [25] first proposes to edit knowledge and rectify erroneous or unexpected predictions without expensive fine-tuning. Specifically, it trains a hyper-network via constrained optimization to modify facts without affecting pre-trained knowledge irrelevant to the edit. The trained hypernetwork is then used to predict the weight update at the inference time. Based on KE, **SLAG** [53] further appends metrics for two types of input texts: (1) Inputs that are not in the desired edit set $\mathcal{X}_\mathcal{E}$ but logically related to $\mathcal{E}$; (2) Inputs that share a formal resemblance to edited knowledge, but do not lead to changes in the prediction outcomes.

However, hyper-networks are generally not capable of updating large language models due to the massive parameter size. To tackle this challenge, **MEND** [101] adopts a mechanism referred to as gradient decomposition. In particular, it leverages small auxiliary editing networks to transform the gradients obtained by standard fine-tuning into edits of weights in a pre-trained model. As gradients are generally high-dimensional objects, a low-rank decomposition of the gradients is utilized to achieve the transformation. Particularly, MEND parameterizes the gradient mapping functions as MLPs with a single hidden layer, such that a significantly small number of parameters are required, compared with the edited models. In this manner, MEND enables fast model editing that can operate on considerably large pre-trained language models. Moreover, **KGEditor** [17] proposes to combine the benefits of memory-based methods and hyper-networks to ensure flexibility and further reduce computation

costs. Particularly, KGEditor introduces an additional layer with the same architecture of FFN layers for storing knowledge. Then it constructs a hyper-network based on a bi-directional LSTM [58] that encodes embeddings of triples. In this manner, KGEditor becomes an efficient way to edit knowledge graph embeddings.

5.3.4 **Summary**. Global optimization methods typically apply specific fine-tuning restrictions to regularize parameter updates, namely constrained fine-tuning strategies. This is to prevent overfitting and ensure the model's performance on the unedited knowledge. One crucial advantage of such strategies is its generality regarding the relevant knowledge, i.e., in-scope inputs $\mathcal{X}_e$ of edit e. As the global optimization affects all parameters in a language model, the relevant knowledge in it will also be edited, thereby generalizing to such knowledge. On the other hand, the high computation costs of fine-tuning all parameters also motivate researchers to propose intermediate fine-tuning strategies that leverage hyper-networks. Furthermore, global optimization methods are mostly model-agnostic, which means they can be applied to other editing methods. Nevertheless, such possibilities are less explored in the context of KME. In terms of the drawbacks, global optimization methods are suboptimal in maintaining the locality of edited models, as the optimization can easily influence unedited knowledge. Hence, it is crucial to achieve a balance between generality and locality when optimizing language models with specific constraints or intermediate designs.

5.4 Local Modification

5.4.1 **Overview**. To tackle the challenge of fine-tuning methods with respect to locality, extensive research has been conducted on the *local modification* strategy for KME tasks [102, 167]. These techniques originate from the concept of identifying and modifying specific relevant weights in a pre-trained model to achieve desirable outputs. The primary objective is to first locate the weights ϕ_k that store the knowledge in a pre-trained model f_ϕ regarding the input x. Afterward, by adjusting these weights, it becomes possible to generate the correct output y^* from the same input x without re-training or fine-tuning the whole model. Recently, researchers have generalized the local modification strategy to LLMs, where the efficiency of information updates for pre-trained LLMs can be substantially improved. Generally, the goal of the local modification strategy of KME can be formulated as a constrained optimization problem with refined constraints as follows:

$$\min_{\phi_k^*} \mathbb{E}_{e \in \mathcal{E}} \mathbb{E}_{x, y^* \in \mathcal{X}_e, y_e^*} \mathcal{L}(f_{\bar{\phi}_k, \phi_k^*}^*(x), y^*),$$

$$\text{s.t. } f_{\bar{\phi}_k, \phi_k^*}^*(x) = f(x), \quad \forall x \in \mathcal{X} \setminus \mathcal{X}_{\mathcal{E}}, \tag{27}$$

$$\text{where } \phi_k = L(f_\phi, \mathcal{E}), \ \bar{\phi}_k = \phi \setminus \phi_k, \ f_{\bar{\phi}_k, \phi_k^*}^* = M(f_\phi, \mathcal{E}).$$

Here ϕ^* denotes the edited weights related to the new knowledge, and $\bar{\phi}_k$ denotes the unedited weights. Eq. (27) breaks down the local modification strategy for KME into two steps: (1) The *locating step*, denoted by function L, locates the relevant weights ϕ_k in pre-trained model f_ϕ that store the obsolete information regarding the query x. (2) The *editing step*, denoted by function M, edits the located weights ϕ_k into new weights ϕ_k^* such that the correct answer y^* given the query x can be generated by the model with ϕ_k^*. By only updating a small fraction of model weights, the editing step avoids negatively influencing other irrelevant information, (i.e., $x \in \mathcal{X} \setminus \mathcal{X}_{\mathcal{E}}$).

In the following subsections, we first introduce the concept of **knowledge neuron** in LLMs, which are specific neurons that store factual knowledge and can be activated to generate the desirable answer based on a certain query x. Then we discuss two local modification strategies for KME: (1) the groundtruth-based strategies, which identify and edit knowledge neurons based on the supervision signal provided by the groundtruth; (2) the prompt-based strategies, which locate knowledge neurons based on the input prompts.

Knowledge Neurons. LLMs pre-trained on large corpora can be viewed as databases that store factual and common-sense knowledge in the pre-trained model weights [49]. To update such knowledge by locally modifying

the weights in the pre-trained LLMs, it is imperative to identify which weights store such information, i.e., locating the knowledge neurons. This can be challenging due to the complex transformer architecture of LLMs [7].

As described in Section 2.2.1, the transformer structure of LLMs consists of two primary types of layers, i.e., (1) the self-attention layer and (2) the point-wise feed-forward (FFN) layer, which is implemented as a two-layer multi-layer perceptron (MLP). Particularly, given a prompt x, the self-attention layers of the LLMs use the query vector of the last token and the key vectors of the previous tokens to calculate a weighted sum of their value vectors. Therefore, given the input x, these layers provide information about which previous tokens we should consider when generating the answer. Here we provide a simplified example for illustration. To answer the question *"Who is the current president of the USA?"*, the self-attention layer indicates that the model should attend to words *"president"* and *"USA"*, i.e., $\mathbf{v}_{president}, \mathbf{v}_{USA}$, to determine the answer. This provides us with a start-up embedding $\mathbf{h}^{start}$ to generate the answer token, which is the weighted sum of the values of the two attended words, i.e., $w_1\mathbf{v}_{president} + w_2\mathbf{v}_{USA}$. However, the information regarding the current president of the USA is not provided. In contrast, recent works [42, 43, 97, 98] claim that the residual added to $\mathbf{h}^{start}$ by the outputs of FNN layers, i.e., $\mathbf{h}^{next} = \mathbf{h}^{start} + \text{FFN}(\mathbf{h}^{start})$, injects the information *"Biden"* to $\mathbf{h}^{start}$ and leads to the generation of correct answers. Therefore, neurons in the FFN can be viewed as the **knowledge neurons** that store the factual knowledge. The role of FFN in storing knowledge can be theoretically analyzed by revisiting their formulation in Eq. (1), which we rewrite as follows:

$$\text{SelfAtt}_i(\mathbf{x}) = \text{Softmax}\left(\mathbf{q}_i\mathbf{k}_i^\top\right)\mathbf{v}_i, \quad \text{FFN}(\mathbf{h}) = \text{GELU}\left(\mathbf{h}\mathbf{W}_1\right)\mathbf{W}_2. \tag{28}$$

Specifically, comparing the above two equations, we observe that the input $\mathbf{h}$ to the FFN acts similarly to the query $\mathbf{q}$ to the SelfAtt. Moreover, the weights of the first layer $\mathbf{W}_1$ can be viewed as the key $\mathbf{v}$, where GELU $(\mathbf{h}\mathbf{W}_1)$ can be viewed as calculating an unnormalized attention score over the row vectors of $\mathbf{W}_2$. Finally, the weights of the second layer $\mathbf{W}_2$ can be viewed as the value (or the memory) that stores the knowledge, which can be retrieved according to the unnormalized weights calculated by the first layer.

5.4.2 ***Groundtruth-based Strategies.*** Based on the knowledge neuron view of the FFN layer weights in pre-trained LLMs, various groundtruth-based methods are proposed to locate and edit the pre-trained LLMs. Generally, they perform editing in a top-down manner, utilizing the supervision signal provided by the correct groundtruth y^*. As an exemplar work, **KD** [22] proposes to change each weight $w_i^{(l)}$ (i.e., the i-th weight in the l-th layer of FFN) from 0 to the pre-trained value $\hat{w}_i^{(l)}$ and calculates the cumulative change in the probability of predicting the output y^* with input x, where the weights with a high cumulative probability are considered relevant for knowledge regarding y^*. DEPN [165] proposes a similar cumulative probability-based strategy to detect knowledge neurons that store privacy knowledge. In contrast to locating and editing an individual weight $w_i^{(l)}$, **ROME** [97] proposes to update an entire FFN layer to encode the new knowledge of y^*. Specifically, they view the second layer weights $\mathbf{W}_2$ in the FFN layer in Eq. (28) as a linear associative memory [3, 75] in the form of $\mathbf{K}\mathbf{W}_2 = \mathbf{V}$, where the keys $\mathbf{K}$ and values $\mathbf{V}$ associated with $\mathbf{W}_2$ can be directly calculated via pseudoinverse. With such a view of $\mathbf{W}_2$ in the FFN layer, the optimization objective of updating it into $\hat{\mathbf{W}}_2$ to encode new knowledge in the edit $e = (s, r, o \rightarrow o^*)$ can be formulated as follows:

$$\min \|\mathbf{K}\hat{\mathbf{W}}_2 - \mathbf{V}\| \text{ s.t. } \hat{\mathbf{W}}\mathbf{k}^* = \mathbf{h}^*. \tag{29}$$

Here $\mathbf{k}^*$, which should encode the information of the subject s, is calculated by sampling multiple $x \sim \mathcal{X}_e$ and taking the average of the outputs from the first dense layer of the FFN. The target activation $\mathbf{h}^*$ is calculated via optimizing the probability of outputting the correct answers $y^* \in \mathcal{Y}_e$ of the pre-trained LLM via the subsequent layers. Then, an efficient rank-one update is conducted on the weights $\mathbf{W}_2$ according to Eq. (29), such that after the update, the edited FFN layer can output the correct hidden representation $\mathbf{h}^*$ conducive to the generation of the right answer y^* from $\mathbf{k}^*$. The **ROME** framework has been shown to generalize to the large Mamba model

[130]. Recently, **MEMIT** [98] proposes to further generalize the above editing strategy of the FFN layers of pre-trained LLMs to the mass editing of different knowledge. Particularly, with u new edits $\{e_1, e_2, \ldots, e_u\}$ that are required to be updated in the weights $\mathbf{W}_2$, the mass knowledge editing problem can be formulated as the following optimization problem:

$$\min \left(\sum_{i=1}^{n} \left\| \mathbf{k}_i \hat{\mathbf{W}}_2 - \mathbf{v}_i \right\|^2 + \sum_{i=n+1}^{n+u} \left\| \mathbf{k}_i^* \hat{\mathbf{W}}_2 - \mathbf{v}_i^* \right\|^2 \right), \tag{30}$$

where $\mathbf{k}_i, \mathbf{v}_i$ are the original key, value pairs associated with the weights $\mathbf{W}_2$ (i.e., row vectors in matrices $\mathbf{K}$, $\mathbf{V}$ in Eq. (29)), whereas $\mathbf{k}_i^*, \mathbf{v}_i^*$ are the updated key, value pairs calculated from the i-th edit e_i as with Eq. (29). In addition, since multiple edits are required, the update is shared among different MLP layers, which is conducted in a top-down manner to prevent the potential issue of editing layers that could affect the ones that have already been edited. The residual for each edit is spread evenly over the range of the critical FFN layer. The strategy of residual attribution has recently been improved by **PMET** [83], which adopts a square root strategy to spread residuals to bottom FFN layers such that more precise information can be conveyed to critical layers. Furthermore, **EMMET** [50] generalized ROME and MEMIT by formulating the mass knowledge editing problem as a preservation (of irrelevant knowledge)-memorization (of new knowledge) constrained optimization problem, where they derive closed form weight update formulae when the edit is exact, i.e., $\mathbf{k}_i^* \hat{\mathbf{W}}_2 = \mathbf{v}_i^*$ instead of minimizing the MSE in Eq. (30).

From the application's perspective, to remove toxic knowledge of LLM, **DINM** [149] identifies layers that store toxic knowledge with the discrepancy of toxic/non-toxic sequence embeddings, and uses the non-toxic samples to locally modify the weights of identified layers.

5.4.3 **Prompt-based Strategies**. Tailored to characteristics of LLMs that provide answer y^* based on the prompt x, the operation of locating and editing knowledge neurons can also be conducted in a bottom-up manner, which aims to change the prompt to detect neurons to be edited. Specifically, by masking out the key information and observing the difference of activations in the intermediate layers of the LLM, the weights that store the information regarding the query x can be located and updated to store the new information y^*. For example, **ROME** [97] proposes a corruption-and-restore based strategy to identify relevant layers (or their hidden output variables $\mathbf{h}$) that store the information based on the prompt x. It first randomly masks the hidden representations of the key vectors $\mathbf{k}$ (as described in Eq. (1)) of the tokens in the prompts from a certain intermediate layer of the pre-trained LLM. Then it calculates the reduced probability of predicting y (i.e., the obsolete outputs) as the causal mediation effects of x on y mediated by $\mathbf{h}$. Consequently, the weights in layers with large mediated effects are viewed as knowledge neurons that store the information of y. **MEMIT$_{\text{CSK}}$** [49] extends the above corruption-based strategy to editing common sense knowledge. The authors argue that, different from the factual knowledge that can be directly retrieved by the subject s, the object o and relation r also matter for commonsense knowledge. Therefore, three types of corruption and edit locations, i.e., subject, verb, and object, are thoroughly analyzed, where the performance of editing commonsense knowledge can be improved. Moreover, **BIRD** [93] studies the novel problem of bidirectional KME, which requires the edited model to possess reversibility. For example, if the phrase "The capital of France is" is edited to a counterfactual "London" within a model, it should logically be able to retrieve the inverse fact. That is, when presented with "London is the capital of," the model should respond with "France" rather than "England". Based on the strategy of ROME, BIRD introduces a novel objective that involves the bidirectional relationships between subject and object in an edit. In this manner, the updated model weights can preserve reversibility by learning such information.

5.4.4 **Summary**. In this part, we introduce the local modification strategy for pre-trained LLMs for efficient updates of new information without adding new weights or optimizing the whole network. We start by analyzing

the pivotal role of the point-wise feedforward layers, i.e., the FFNs, to store the factual information in pre-trained LLMs, with the knowledge neurons associated with the FFN layer thoroughly analyzed. We then discuss the groundtruth-based strategies, which achieve the modification in a top-down manner, generally based on least squares objectives computed from the output y. We further discuss the prompt-based strategies, which conduct modifications in a bottom-up manner based on the input prompt x. Nevertheless, the scalability and retainability of local modification methods lack further improvements, as the performance might deteriorate with more edits performed [98].

6 Datasets

Recently, multiple datasets have been established to facilitate the evaluation of KME methods, and we summarize the commonly-used datasets in Table 2 to benefit future KME research. Specifically, these datasets can be divided into two groups: **generation datasets** (i.e., textual output) and **classification datasets** (i.e., categorical output). The datasets are obtained from a variety of sources, including knowledge graphs, Wikipedia pages, crowd-sourced responses, etc., which are adapted by researchers to fit into the KME setting.

6.1 Generation Datasets

For generation datasets, the target is in the form of textual content that is required to be generated by LLMs. Serving as pivotal resources to evaluate KME methods, most generation datasets are based on relational knowledge and used for assessing the ability of editing techniques to inject new factual knowledge. This is because relational datasets preserve more definitive answers for each input and thus are more convenient and precise for evaluation [167, 172]. Specifically, these datasets are generally curated from the corresponding relational datasets to encompass diverse relational contexts, ranging from question-answer pairs to intricate multi-hop queries. Therefore, the most prevalent output format is an object to be predicted.

In this subsection, we present the most representative generation datasets, shedding light on their unique attributes, the nature of their content, and the specific challenges they present for evaluating KME methods on factual knowledge as follows:

- **zsRE** [78]: zsRE is one of the most prevalent Question Answering (QA) datasets extended and adopted by [25, 101] for KME evaluation. zsRE is suitable for evaluating KME due to its annotations of human-generated question paraphrases, which allow researchers to assess the model resilience to semantically equivalent inputs. In zsRE, each relation is associated with a set of crowd-sourced template questions, such as "What is Albert Einstein's alma mater?". Each entry cites a Wikipedia sentence, serving as the factual basis or provenance. The dataset also contains negative examples that are generated by pairing a valid question with a random sentence.
- **CounterFact** [97]: CounterFact is established to distinguish superficial alterations in the word selections and significant, generalized modifications in its foundational factual knowledge. Proposed in ROME [97], each entry in CounterFact originates from a related record in ParaRel [32], containing a knowledge triple and meticulously crafted prompt templates. It is important to note that all subjects, relations, and objects in this tuple are recognized entities in Wikidata [148].
- **WikiGen** [101]: Firstly proposed in MEND [101], WikiGen consists of approximately 68k question-answer pairs, with a similar size to zsRE. Here, each question corresponds to a sentence randomly sampled from Wikitext-103, and each answer is a 10-token sample obtained from a pre-trained distilGPT-2 model [94]. It is noteworthy that greedy 10-token prediction of the base model only aligns with edit targets for less than 1% of samples.
- **T-REx-100 & T-REx-1000** [33]: First used in CALINET [29], the authors adopt the classic relational dataset T-REx [33] for evaluating model editors by extracting factual triplets of varying sizes (100 and 1,000). Particularly, for each triplet, the authors insert the head and tail entities into the template in LAMA [115] based on the

Table 2. Statistics of prevalent KME datasets, including generation and classification datasets.

Dataset	Type	# Train	# Test	Input	Output	Used in
zsRE	Relational	244,173	244,173	Factual Statement	Object	[25, 38, 48, 50, 52, 66, 69, 77, 81, 97, 98, 101, 102, 106, 136, 151, 156, 169]
CounterFact	Relational	N/A	21,919	Factual Question	Object	[15, 38, 50, 61, 81, 97, 98, 106, 130, 136, 156, 168, 174]
WikiGen	Generation	N/A	68k	Wiki Passage	Continuation	[101]
T-REx-100/-1000	Relational	N/A	100/1,000	Factual Statement	Object	[29, 79]
ParaRel	Relational	N/A	253,448	Factual Question	Object	[22]
NQ-SituatedQA	QA	N/A	67.3k	User Query	Answer	[23, 77]
MQuAKE-CF/-T	Relational	N/A	9,218/1,825	Multi-hop Question	Object	[47, 69, 82, 131, 155, 175]
Hallucination	Hallucination	N/A	1,392	(Fake) Biography	Biography	[52, 151, 169]
MMEdit-E-VQA	Multimodal	6,346	2,093	Image & Question	Answer	[16]
MMEdit-E-IC	Multimodal	2,849	1,000	Image	Description	[16]
ECBD	Relational	N/A	1000	Reference to Entity	Completion	[108]
Conflict Edit	Relational	N/A	7,500	Factual Statement	Object	[86]
Round Edit	Relational	N/A	5,000	Factual Statement	Object	[86]
UKE	Relational	N/A	2,478	Factual Question	Object	[166]
RippleEdits	Relational	N/A	5,000	Factual Question	Object	[21, 69]
VLKEB	Multimodal	5,000	3,174	Image	Description	[65]
MLaKE	Multilingual	N/A	9,432	Question	Answer	[163]
FEVER	Fact Checking	104,966	10,444	Fact Description	Binary Label	[15, 25, 66, 101]
ConvSent	Sentimental	287,802	15,989	Topic Opinion	Sentiment	[102]
Bias in Bio	Biographical	5,000	5,000	Biographical Sentence	Occupation	[57]
VitaminC-FC	Fact Checking	370,653	55,197	Fact Description	Binary Label	[102]
SCOTUS	Categorization	7,400	931	Court Documents	Dispute Topic	[52, 169]

relation they share, which results in two datasets with 100 and 1,000 facts, respectively, for the purpose of false knowledge detection. It should be noted that each fact in these datasets is represented by several paraphrased sentences.

- **ParaRel** [32]: ParaRel is an expert-curated dataset that comprises diverse prompt templates for 38 relations, sourced from the T-REx dataset [33]. Firstly used in KN [22], the authors insert the head entity into each relational fact and set the tail entity as a blank for prediction. To ensure a rich variety in templates, relations with less than four prompt templates are excluded, resulting in 34 relations in total. Each of these relations, on average, preserves 8.63 distinct prompt templates, leading to a total of 253,448 knowledge-revealing prompts for 27,738 relational facts.

- **NQ-SituatedQA** [76]: NQ (Natural Questions) is a comprehensive question-answering dataset originating from user searches. In PPA [77], the authors utilize NQ as the source knowledge while excluding any outdated information as identified by SituatedQA [171] to create a novel dataset NQ-SituatedQA. SituatedQA is a dataset containing questions within a subset of NQ that are dependent on specific time and location. The authors then incorporate the time-dependent QA pairs from this subset, annotated using the 2021 Wikipedia [148] dump.

- **MQuAKE** [175]: MQuAKE is constructed from Wikidata [148] for evaluating the effectiveness of KME methods on multi-hop questions. In particular, it is designed to assess whether the edited models can correctly answer questions generated by chains of facts in plain text. MQuAKE consists of two datasets. (1) MQuAKE-CF is a diagnostic dataset, specifically crafted to evaluate KME methods in the context of counterfactual edits. (2) MQuAKE-T focuses on temporal-based knowledge updates and is aimed at assessing the effectiveness of KME techniques in updating outdated information with contemporary factual data.

- **Hallucination** [52]: Firstly processed in GRACE [52], Hallucination is created from the dataset released in SelfCheckGPT [96], where the authors prompt GPT-3 to generate biographies based on concepts extracted from WikiBio. The sentences are annotated regarding the factual accuracy, and hallucinations in them are identified. Then in GRACE, the authors process this dataset by further extracting Wikipedia summaries from WikiBio and thereby acquire the correct entry of each sentence. In this manner, every edit consists of a potentially false biography generated by GPT-3 as the prompt, and a ground truth output, which is the correct next sentence extracted from Wikipedia. There exist 1,392 potential edits for test.

- **MMEdit** [16]: This dataset is the first to explore the possibility of editing multimodal LLMs. Specifically, MMEdit consists of two prevalent multimodal tasks: Visual Question Answering (VQA) [4] and Image Captioning [56]. VQA involves developing algorithms that can analyze an image's visual content, comprehend questions asked in natural language about the image, and accurately respond to those questions. Image Captioning aims to understand an image and then generate a detailed and coherent natural language description of that image. To create dataset MMEdit, the authors utilize BLIP-2 OPT [80] and extract edit data from the evaluation datasets VQAv2 [46] and COCO Caption [14], specifically focusing on their suboptimal entries.

- **ECBD** [108]: Based on the original dataset ECBD (Entity Cloze By Date) [107], the authors process this dataset for a novel task, namely Entity Knowledge Propagation (EKP). The task aimed at updating model parameters to incorporate knowledge about newly emerged entities that are not present in the pre-training data of the language models. For instance, BERT [27], trained in 2018, does not recognize "COVID-19" as it is a more recent entity. The processed dataset aims to provide evaluation for such a task with the help of definition sentences as input to update knowledge about new entities. The entities are taken from the date between 2020/01 and 2021/09 to ensure that they are not in training data. Each edit consists of a new entity, a description sentence, a probe sentence, and a ground truth completion.

- **VLKEB** [65]: VLKEB (Large Vision-Language Model Knowledge Editing Benchmark) aims to address the unique challenges of editing large vision-language models, which face additional difficulties due to different data modalities and complex model components with limited data for LVLM editing. VLKEB collects data from the multi-modal knowledge graph MMKG [90] and extends the Portability metric for evaluation. With MMKG, VLKEB binds image data with knowledge entities, which can be used to extract entity-related knowledge for editing data.

- **MLaKE** [163]: MLaKE (Multilingual Language Knowledge Editing) is proposed to evaluate the capability of KME methods in multilingual contexts and multi-hop reasoning across five languages: English, Chinese, Japanese, French, and German. MLaKE aggregates fact chains from Wikipedia in multiple languages and utilizes LLMs to generate questions in both free-form and multiple-choice formats. Notably, existing methods show relatively high generalization for languages within the same language family compared to those from different families. These findings underscore the need for advancements in multilingual knowledge editing.

- **UKE** [166]: UKE (Unstructured Knowledge Editing) is proposed to evaluate the capability of KME methods in updating knowledge based on unstructured texts. Updating LLMs with texts appears to be a more realistic application, which is also more complex and difficult. The authors leverage subjects and objects in Wikidata [148] and retrieve the corresponding Wikipedia article summaries as unstructured texts. The authors also utilize LLMs to generate summaries for edits in two existing datasets, CounferFact [97] and MQuAKE-CF [175], to obtain unstructured texts.

- **RippleEdits** [21]: This dataset proposes a novel evaluation criterion, which assesses the performance of KME methods on additional edits brought by an existing edit. In particular, injecting new knowledge (e.g., "Jack Depp is the son of Johnny Depp") introduces a "ripple effect," which necessitates the model to update related knowledge as well (e.g., "Jack Depp is the sibling of Lily-Rose Depp"). Based on this, the authors construct RippleEdits, consisting of 5,000 edits with various types of ripple effects.

- **Conflict/Round Edit** [86]: This dataset pioneers in investigating the potential side effects of KME methods for LLMs. The proposed dataset and evaluation metrics underline two primary concerns: (1) Knowledge Conflict: Modifying sets of logically conflicting facts can amplify the existing inconsistencies within LLMs. (2) Knowledge Distortion: Altering model parameters to update factual knowledge can permanently disrupt the inherent knowledge framework of LLMs. The dataset is constructed from WikiData [148] with specific logical rules.

6.2 Classification Datasets

Classification datasets are also widely adopted to evaluate the effectiveness of KME. These datasets consist of prompt-target pairs, where the target is a discrete label instead of a textual sentence. In the context of KME, these labels help ascertain the alignment of model performance with desired edits. The advantages of classification datasets also involve their preciseness in evaluation without the need to define the specific output space. In this section, we summarize notable classification datasets that have been tailored and leveraged for assessing KME techniques as follows:

- **FEVER** [143]: FEVER is a fact-checking dataset originally processed in KILT [114] for verifying factual knowledge in the form of binary classification. It necessitates the retrieval of sentence-level evidence to determine whether a claim is supported or refuted, and is widely used for evaluating the performance of KME. Specifically, FEVER excludes claims labeled as lacking sufficient information, as they typically do not provide any evidence to evaluate the claim.
- **ConvSent** [102]: Firstly processed in SERAC [102], ConvSent is used to evaluate the capability of an editor to modify a dialog agent's sentiment about a particular topic without influencing its responses to other topics. ConvSent is obtained from a list of 15,000 non-numeric entities from zsRE [25, 78], combined with 989 noun phrases from GPT-3 [10] for 15,989 topics. Particularly, for each entity, there are ten positive and ten negative sentiment completions, which can be noisy, from the BlenderBot model with 3B parameters [124]. The refined sentiment labels are achieved by a sentiment classifier [55] pre-trained on RoBERTa [91].
- **Bias in Bios** [24]: Bias in Bios is a dataset originally proposed for fairness-related machine learning, containing approximately 397k short professional biographies of online individuals, which are not relatively famous. Each biographical sentence is assigned an associated occupation label for the described person. To adopt this dataset for evaluating the performance of KME methods, the authors of REMEDI [57] extract a single sentence, modify it to display only the person's first name, and then query the language model with the prompt that follows the structure: "Person has the occupation of...". Then they evaluate the relative probabilities of the language model assigned to 28 potential occupations, where the language model is considered to be correct if the ground-truth occupation is ranked top-1.
- **VitaminC-FC** [127]: Firstly processed in SERAC [102], VitaminC-FC is constructed based on a fact-checking dataset, VitaminC [127]. Particularly, VitaminC consists of more than 400,000 evidence-claim pairs, each of which is assigned a binary label to indicate whether the evidence entails the claim. The dataset was gathered from over 100,000 Wikipedia revisions that modify an underlying fact, along with additional synthetic ones. In SERAC, the authors convert VitaminC into a KME dataset by using the evidence as the edit descriptor and using claims from the same Wiki pages accordingly as in-scope samples.
- **SCOTUS** [52]: Firstly proposed in GRACE [52], SCOTUS is processed with label shift based on the dataset with the same name from Fairlex [11]. This classification task is to categorize U.S. Supreme Court documents from various decades into one of 11 topics. The topics are clustered based on the specific matter of dispute, such as Criminal Procedure, Civil Rights, and First Amendment. Due to the evolution of categorization rules over time, the label distributions in this dataset also shift. Specifically, 7.4k cases from 1946-1982 are used for training, and 931 cases from the 1991-2009 period are for test.

Table 3. Examples of different downstream applications of KME: Question Answering (QA), Fact Checking (FC), and Natural Language Generation (NLG).

Task	Edit Descriptor e	In-scope Input $x \sim \mathcal{X}_e$	Original Output $y \sim \mathcal{Y}_e$	Target Output $y \sim \mathcal{Y}_e^*$
QA	(Kazakhstan, Captital, Astana→Nur-Sultan)	What is the capital of Kazakhstan?	Astana	Nur-Sultan
FC	(Marathon, Record, Kipchoge→Kiptum)	Kipchoge holds the men's marathon world record.	True	False
NLG	(Jordan Poole, Play In, Warriors→Wizards)	Provide a short introduction to Jordan Poole, describing his current position.	Jordan Poole entered the Warriors' rotation recently.	In 2023, Jordan Poole transitioned from the Warriors to the Wizards, remarking a significant change.

7 Applications

KME can benefit multiple downstream applications with the ability to precisely and efficiently inject knowledge into pre-trained LLMs. In the following, we introduce several key applications of KME techniques in realistic scenarios, where intuitive examples are provided in Table 3.

7.1 Question Answering

Background. Question Answering (QA) is a core NLP task that aims to comprehend queries posed by users in natural language and provide answers based on the encoded knowledge in the pre-trained language model [132]. Traditional models for QA are generally fixed in their knowledge, capturing only the information available at the training time of [70, 115]. However, in our dynamic world, new information is generated incessantly, which necessitates the constant update of QA models [139]. Fortunately, KME methods enable the modification of QA models to cater to specific questions without disrupting responses to other unrelated inputs. Therefore, with KME strategies, the QA model can be efficiently updated on the run, where the currentness of the model can be guaranteed. Consequently, language model editing techniques have found broad applications across a myriad of QA contexts with potentially distinct requirements [77].

Existing Works. The QA task encompasses various aspects, such as conversational QA, definition-based QA, and notably, relation-based QA [110]. Relation-based QA is primarily adopted as an evaluation benchmark as it necessitates the retrieval of precise real-world facts in response to queries. This particular emphasis on specific information retrieval renders relation-based QA especially conducive to the benefits of KME techniques. For example, **PPA** [77] introduces an innovative task of **CuQA** (Continuously-updated QA), which intentionally emphasizes recurrent, substantial edits for language models to constantly update them with new information. An important aspect of the CuQA task is to ensure that the existing pre-trained knowledge remains unaltered with the integration of new knowledge. Therefore, this property is one important evaluation to assess model editing in CuQA tasks. In **MQuAKE** [175], the authors innovatively propose a multi-hop QA task that involves answering questions generated by chains of facts in plain text. Specifically, the task requires edited models to infer implicit relations that can be several hops away from the objects in the edit. For example, when a language model is modified regarding the president of the USA, an ideal model should also authentically alter answers to "Who is the son of the president of the USA", which is a two-hop relation. Such a task is significantly more challenging as it necessitates the model to alter its reasoning results in addition to the original edit. Nevertheless, the proposed method MeLLo in MQuAKE still exhibits outstanding performance on this difficult task, demonstrating the potential of KME in generalizing edited knowledge to multi-hop relations.

7.2 Fact Checking

Background. Fact-checking (FC) is a pivotal task in journalism, information verification, and combating misinformation that aims to scrutinize and affirm the authenticity of claims, statements, or information in news articles, social media, and other media content [37, 127]. In a world overwhelmed with ever-emerging information, fact-checking facilitates the trustworthiness in the sharing of distributed information, promotes information transparency, and aids individuals in making well-informed decisions [143]. However, it is crucial to constantly update fact-checking models. For instance, during the COVID-19 pandemic, initial understandings and guidelines about the virus evolved as researchers gathered more data [129]. A fact-checking model that cannot adapt to these rapidly changing facts would quickly become outdated and potentially spread misinformation, thereby requiring the application of language model editing. By integrating KME techniques into fact-checking models to consistently update them with the latest information and facts, it becomes possible to ensure the currentness, trustworthiness, and accuracy of the model despite the persistent evolution of information.

Existing Works. Recently, several works have proposed to apply KME techniques in fact-checking models. In [177], the authors first explore the potential of modifying specific factual knowledge within the transformer backbone of the fact-checking model while ensuring that overall model performance remains intact on facts irrelevant to the editing purpose. Particularly, they identify the critical components within the transformer backbones conducive to effective knowledge modifications. In **SERAC** [102], the authors propose to use evidence gathered from Wikipedia as edit descriptors to update potentially outdated knowledge in the model. The proposed method exhibits significant performance improvements over baselines and can be generalized to other in-scope inputs collected from the same Wikipedia page.

7.3 Natural Language Generation

Background. KME techniques are also promising to ensure the relevancy of the Natural Language Generation (NLG) task, which aims to generate coherent and contextually relevant content based on provided instructions [122]. Considering the rapid evolution of the global information landscape, it is essential for NLG models to remain up-to-date and ensure the accuracy of generated text while avoiding potentially false statements that may mislead the users.

Existing Works. In practice, several works have been proposed to apply KME methods to promote model performance in natural language generation tasks. For instance, **FRUIT** [5] proposes to update outdated Wikipedia articles according to the collection of new information about the article's subject. Based on the T5 model [119], the authors utilize a compressed output format to eliminate the necessity of generating the entire update from scratch and promote content structuring, which effectively handles the challenge of incoherence. In **MEND** [101], the authors apply their proposed method in the Wikitext generation task, where the edited model is required to produce credible 10-token extensions based on a provided Wikitext prefix [94]. With modification on multi-layer token-wise activations and gradients, the edited model presents higher coherence on the NLG task, which demonstrates the effectiveness of KME in generating target texts with richer information than QA or FC.

8 Discussion

8.1 Challenges

Despite the continual progress of works on KME, several critical aspects have been inadequately addressed by existing studies. Delving deeper into these challenges could offer researchers fresh insights and pave the way for the further advancement of the field. Consequently, we hereby outline the pressing challenges that await solutions in KME.

Trade-off between Locality and Generality. In KME, it is crucial to balance two objectives, *locality* and *generality* (as defined in Sec. 4), such that a higher edit success rate can be achieved with minimal negative

influence on knowledge irrelevant to the edits. When editing a language model, a potential trade-off might emerge between these two desirable properties. As demonstrated in [167], local modification methods, such as MEMIT [98] and ROME [97] generally preserve a higher level of locality, as they locate precise locations of target knowledge to conduct the edition, which does not largely affect the unrelated weights. In addition, T-Patcher [66] points out that increasing the size of memory increases locality while decreasing the generality. These observations underscore the intricate balance between locality and generality. However, it remains challenging to tackle the trade-off problem and achieve a balance between these two desirable properties of KME methods.

Theoretical Analysis. While many current KME studies focus on developing effective methods to enhance the editing performance regarding various desirable properties, there exists a notable gap between the practical application and the comparatively less discovered theoretical analysis. Recently, in [140], the authors provide theoretical support for the justification of identifying harmful training examples and editing the model by erasing the information from a Bayesian view. LEACE [9] introduces an analytical framework that offers a theoretical perspective for the task of erasing target concept information from every layer in language models. In general, the benefits of incorporating theoretical analysis are multi-faceted. First, theoretical analysis provides a deeper understanding of the mechanics underlying KME, allowing for more principled approaches to editing. Second, a strong theoretical basis sets a solid foundation for future research, encouraging more rigorous and systematic exploration in the field of KME. However, to the best of our knowledge, there still does not exist any comprehensive theoretical analysis regarding the KME problem that involves novel knowledge. We hope that future research will enrich the theoretical discourse that can deliver profound insights into the substantial foundations of KME methods.

Editing at Scale. Another crucial property that hinders the practical application of KME is scalability — the ability of editing strategy to effectively perform a large number of edits simultaneously [101]. For example, conversational systems [174] are expected to be constantly updated to incorporate an enormous number of global events and the information originating from them. However, as the number of applied edits increases, the coherence of language models is severely jeopardized, as multiple edits might contradict a broader spectrum of pre-existing knowledge in the models [152]. This can lead to decreased editing performance in both locality and generality metrics [102]. Although external memorization methods can alleviate such problems with a larger size of memories of additional parameters, they are still vulnerable if thousands of edits are required [97]. Moreover, simply adapting single-edit techniques for a multi-edit environment by merely applying them sequentially has been demonstrated to be proven suboptimal [98]. Therefore, the unique and intricate challenge of coherence renders editing at scale a formidable task.

Unstructured Editing. KME faces significant challenges due to its evaluation strategies that focus on knowledge triples, e.g., $t = (s, r, o)$, which are not reflective of how real-world knowledge updates occur [65, 172]. In reality, updates are often found in unstructured texts such as news articles and scientific papers. To address this gap, a recent benchmark [166], namely UKE (Unstructured Knowledge Editing), is proposed to evaluate editing performance using unstructured texts as knowledge updates. The experimental results demonstrate significant performance declines of state-of-the-art KME methods. Notably, such a decline persists even with knowledge triplets extracted from unstructured texts. As such, it is imperative to develop more robust and adaptable methods that use unstructured texts for editing.

8.2 Future Directions

Despite the recent achievements in the development of KME strategies for effective and efficient updating of new knowledge into LLMs, KME research is still in its emerging stage. Several promising directions could be pursued to further advance this field. Accordingly, we identify five inspiring and important open problems worthy of exploration in the future as follows:

Optimization-Free Editing. Recently, prompt engineering has become a prevalent solution for modifying the behaviors of pre-trained LLMs in a human-preferable manner without the requirement of parameter update [30]. For example, in-context learning provides task descriptions and/or demonstrations in the form of plain text to promote the model performance [10], which makes it a potentially more efficient and practical strategy for language models. We note that IKE [174] proposes a novel framework that relies on demonstration contexts for KME without parameter updating, which explicitly formats the demonstrations that can guide the language model to copy, update, and retain the prediction of different prompts. However, such a strategy is difficult to scale and usually has unsatisfactory retention. Therefore, it remains a crucial while challenging task to develop optimization-free KME methods.

Auto-Discovery of Editing Targets. Current KME methods mainly rely on human expertise to identify and incorporate desirable knowledge into pre-trained LLMs [166, 167, 172]. This approach is inherently labor-intensive and can incur significant costs, especially considering the vast and rapidly expanding new information needed to be integrated into language models. A promising future direction lies in the automation of the edits, which aims to identify, evaluate, and prioritize new knowledge that needs to be integrated from raw resources such as websites and social media. Through this strategy, the application of KME can be streamlined, rendering it more practical and adaptable in real-world scenarios. A straightforward solution would be crawling new knowledge and transforming it into a knowledge base, querying LLMs for each knowledge triple, and editing the wrong answer. However, such a strategy still lacks efficiency. Therefore, it remains a crucial task to discover editing knowledge from various resources without human effort.

Continual Editing. Current KME methods primarily consider one-step offline editing [5, 25]; however, such an approach is not aligned with real-world applications where models might continually encounter novel knowledge to be injected. For example, an online question-answering (QA) model may continually encounter reports of incorrect answers from end users, where the editing needs to be conducted on the run [66]. Therefore, an optimal KME technique should be capable of instantaneously and continuously rectifying emergent issues. We note that continual editing of pre-trained LLMs presents a unique challenge: preventing the edited models from forgetting or contradicting previous edits. Despite the inherent complexities, the persistent demand for continual editing in practice underscores the importance of solving this challenge.

Robust Editing. An important direction for the advancement of KME lies in enhancing its robustness. In an era where misinformation spreads rapidly, it is urgent that edited models not only retain their accuracy but also resist adversarial attacks and misinformation [39]. Here, we should note that the concept of robustness extends beyond just maintaining factual accuracy; it involves fortifying the model against potentially adversarial external perturbations [113]. For example, if KME is maliciously applied to inject harmful knowledge into language models, the edited models can be easily transformed into tools for misinformation [141]. Therefore, to prevent such cases, it is crucial for KME techniques to develop capabilities that can identify and counteract such unwanted inputs, thereby enhancing their resilience against adversarial actions. In practice, as the trend leans towards open-sourcing LLMs, it becomes ever more crucial to safeguard against potential manipulations that can turn these models harmful.

Editable Fairness. With the wide application of large language models (LLMs) to support decisions, the emphasis on fairness has grown significantly [150], which requires LLMs to fairly treat people with diverse background [1]. However, LLMs trained on large datasets inevitably incorporate certain biases during this pre-training phase [28]. Fortunately, the precision and efficiency of KME techniques offer a promising solution to mitigate such biases and promote fairness in pre-trained LLMs. For instance, in a model designed to classify biographical sentences with occupation [24], KME can be used to inject nuanced knowledge about a particular profession, guiding the model towards a more equitable understanding of individuals associated with that profession [57]. However, this remains

a complex challenge, as fairness often entails considering disparate groups of individuals rather than specific people. This broader focus makes knowledge injection via KME a non-trivial task. Despite these difficulties, the enhancement of fairness in language models is paramount, and KME techniques present a promising avenue to achieve this goal.

9 Conclusions

In this survey, we present a comprehensive and in-depth review of knowledge-based model editing (KME) techniques for precise and efficient updating of new knowledge in pre-trained LLMs. We first formulate the KME problem as a constrained optimization objective that simultaneously ensures the accuracy and retention of editing, which is general to encompass different KME strategies. We then provide an overview of the evaluation metrics for KME, which sheds light on the desirable attributes of edited models. Subsequently, we propose a structured taxonomy framework to systematically categorize existing KME techniques. Within each category, we outline the central challenges, elaborate on the representative methods, and discuss their strengths and weaknesses. Furthermore, we summarize the datasets widely utilized to assess KME techniques, highlighting that certain techniques demand specific dataset structures for training or evaluation. To inspire researchers to devise more practical implementations, we also spotlight the real-world applications of KME techniques. Finally, we identify several potential challenges for future research and provide insightful directions that are conducive to further advancement of the field.

Acknowledgments

References

[1] Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Persistent anti-muslim bias in large language models. In *AAAI*.

[2] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *ACL*.

[3] James A Anderson. 1972. A simple neural network generating an interactive memory. *Mathematical biosciences* (1972).

[4] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *ICCV*.

[5] Robert L. Logan IV au2, Alexandre Passos, Sameer Singh, and Ming-Wei Chang. 2022. FRUIT: Faithfully Reflecting Updated Information in Text. In *NAACL*.

[6] Razvan Azamfirei, Sapna R Kudchadkar, and James Fackler. 2023. Large language models and the perils of their hallucinations. *Critical Care* (2023).

[7] Michiel Bakker, Martin Chadwick, Hannah Sheahan, Michael Tessler, Lucy Campbell-Gillingham, Jan Balaguer, Nat McAleese, Amelia Glaese, John Aslanides, Matt Botvinick, et al. 2022. Fine-tuning language models to find agreement among humans with diverse preferences. In *NeurIPS*.

[8] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020. Rewriting a deep generative model. In *ECCV*.

[9] Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. 2023. LEACE: Perfect linear concept erasure in closed form. In *ICLR*.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

[11] Ilias Chalkidis, Tommaso Pasini, Sheng Zhang, Letizia Tomada, Sebastian Schwemer, and Anders Søgaard. 2022. FairLex: A Multilingual Benchmark for Evaluating Fairness in Legal Text Processing. In *ACL*.

[12] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109* (2023).

[13] Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and Learn: Fine-tuning Deep Pretrained Language Models with Less Forgetting. In *EMNLP*.

[14] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. arXiv 2015. *arXiv preprint arXiv:1504.00325* (2015).

[15] Yingfa Chen, Zhengyan Zhang, Xu Han, Chaojun Xiao, Zhiyuan Liu, Chen Chen, Kuai Li, Tao Yang, and Maosong Sun. 2024. Robust and Scalable Model Editing for Large Language Models. In *COLING*.

[16] Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. 2023. Can We Edit Multimodal Large Language Models?. In *EMNLP*.

[17] Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Zelin Dai, Feiyu Xiong, Wei Guo, and Huajun Chen. 2024. Editing Language Model-based Knowledge Graph Embeddings. In *AAAI*.

[18] Cheng-Han Chiang and Hung-yi Lee. 2023. Can Large Language Models Be an Alternative to Human Evaluations? *arXiv preprint arXiv:2305.01937* (2023).

[19] Alebachew Chiche and Betselot Yitagesu. 2022. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data* 9, 1 (2022), 1–25.

[20] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[21] Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *TACL* (2024).

[22] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge Neurons in Pretrained Transformers. In *ACL*.

[23] Damai Dai, Wenbin Jiang, Qingxiu Dong, Yajuan Lyu, and Zhifang Sui. 2023. Neural knowledge bank for pretrained transformers. In *CCF International Conference on Natural Language Processing and Chinese Computing*.

[24] Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *FAccT*.

[25] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing Factual Knowledge in Language Models. In *EMNLP*.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

[27] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[28] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *NeurIPS* (2019).

[29] Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating Factual Knowledge in Pretrained Language Models. In *EMNLP*.

[30] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[31] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387* (2023).

[32] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *TACL* (2021).

[33] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *LREC*.

[34] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046* (2023).

[35] Hao Fei, Yafeng Ren, Yue Zhang, Donghong Ji, and Xiaohui Liang. 2021. Enriching contextualized language model from knowledge graph for biomedical information extraction. *Briefings in Bioinformatics* (2021).

[36] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

[37] Boris A Galitsky. 2023. Truth-O-Meter: Collaborating with LLM in Fighting its Hallucinations. (2023).

[38] Govind Gangadhar and Karl Stratos. 2024. Model Editing by Pure Fine-Tuning. *arXiv preprint arXiv:2402.11078* (2024).

[39] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858* (2022).

[40] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[41] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*.

[42] Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space. In *EMNLP*.

[43] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *EMNLP*.

[44] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375* (2022).

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* (2020).

[46] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*.

[47] Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2023. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194* (2023).

[48] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model Editing Harms General Abilities of Large Language Models: Regularization to the Rescue. *arXiv preprint arXiv:2401.04700* (2024).

[49] Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Lorraine Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing Common Sense in Transformers. In *EMNLP*.

[50] Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024. A unified framework for model editing. *arXiv preprint arXiv:2403.14236* (2024).

[51] David Ha, Andrew Dai, and Quoc V. Le. 2016. HyperNetworks. arXiv:1609.09106 [cs.LG]

[52] Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: Lifelong Model Editing with Discrete Key-Value Adaptors. In *NeurIPS*.

[53] Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2023. Methods for measuring, updating, and visualizing factual beliefs in language models. In *EACL*.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[55] Mark Heitmann. 2020. More than a feeling: Benchmarks for sentiment analysis accuracy. In *More than a Feeling: Benchmarks for Sentiment Analysis Accuracy: Heitmann, Mark*.

[56] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. 2019. Image captioning: Transforming objects into words. *NeurIPS* (2019).

[57] Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023. Inspecting and Editing Knowledge Representations in Language Models. *arXiv preprint arXiv:2304.00740* (2023).

[58] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* (1997).

[59] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural Instructions: Tuning Language Models with (Almost) No Human Labor. *arXiv preprint arXiv:2212.09689* (2022).

[60] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE TPAMI* (2021).

[61] Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987* (2024).

[62] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

[63] Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2023. A Survey of Knowledge Enhanced Pre-Trained Language Models. *IEEE TKDE* (2023).

[64] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv preprint arXiv:2304.01933* (2023).

[65] Han Huang, Haitian Zhong, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2024. KEBench: A Benchmark on Knowledge Editing for Large Vision-Language Models. *arXiv preprint arXiv:2403.07350* (2024).

[66] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-Patcher: One Mistake Worth One Neuron. In *ICLR*.

[67] Mike Huisman, Jan N Van Rijn, and Aske Plaat. 2021. A survey of deep meta-learning. *Artificial Intelligence Review* (2021).

[68] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *ICLR*.

[69] Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, et al. 2024. Learning to edit: Aligning llms with knowledge editing. *arXiv preprint arXiv:2402.11905* (2024).

[70] 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* (2020).

[71] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. 2022. AMMU: a survey of transformer-based biomedical pretrained language models. *Journal of Biomedical Informatics* 126 (2022), 103982.

[72] Atoosa Kasirzadeh and Iason Gabriel. 2023. In conversation with Artificial Intelligence: aligning language models with human values. *Philosophy & Technology* (2023).

[73] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences* (2023).

[74] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[75] Teuvo Kohonen. 1972. Correlation matrix memories. *IEEE Trans. Comput.* (1972).

[76] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* (2019).

[77] Kyungjae Lee, Wookje Han, Seung won Hwang, Hwaran Lee, Joonsuk Park, and Sang-Woo Lee. 2022. Plug-and-Play Adaptation for Continuously-updated QA. In *ACL Findings*.

[78] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *CoNLL 2017*.

[79] Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix X. Yu, and Sanjiv Kumar. 2023. Large Language Models with Controllable Working Memory. In *ACL*.

[80] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597* (2023).

[81] Shuaiyi Li, Yang Deng, Deng Cai, Hongyuan Lu, Liang Chen, and Wai Lam. 2024. Consecutive Model Editing with Batch alongside HooK Layers. *arXiv preprint arXiv:2403.05330* (2024).

[82] Xiaopeng Li, Shasha Li, Bin Ji, Shezheng Song, Xi Wang, Jun Ma, Jie Yu, Xiaodong Liu, Jing Wang, and Weimin Zhang. 2024. SWEA: Changing Factual Knowledge in Large Language Models via Subject Word Embedding Altering. *arXiv preprint arXiv:2401.17809* (2024).

[83] Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. PMET: Precise Model Editing in a Transformer. In *AAAI*.

[84] Xiaonan Li and Xipeng Qiu. 2023. Finding supporting examples for in-context learning. *arXiv preprint arXiv:2302.13539* (2023).

[85] Yuchao Li, Fuli Luo, Chuanqi Tan, Mengdi Wang, Songfang Huang, Shen Li, and Junjie Bai. 2022. Parameter-efficient sparsity for large language models fine-tuning. *arXiv preprint arXiv:2205.11005* (2022).

[86] Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024. Unveiling the Pitfalls of Knowledge Editing for Large Language Models. In *ICLR*.

[87] Q Vera Liao and Jennifer Wortman Vaughan. 2023. AI Transparency in the Age of LLMs: A Human-Centered Research Roadmap. *arXiv preprint arXiv:2306.01941* (2023).

[88] Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. 2023. Chain of hindsight aligns language models with feedback. *arXiv preprint arXiv:2302.02676* 3 (2023).

[89] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *NeurIPS*.

[90] Ye Liu, Hui Li, Alberto Garcia-Duran, Mathias Niepert, Daniel Onoro-Rubio, and David S Rosenblum. 2019. MMKG: multi-modal knowledge graphs. In *ESWC*.

[91] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[92] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747* (2023).

[93] Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. 2023. Untying the Reversal Curse via Bidirectional Language Model Editing. *arXiv preprint arXiv:2310.10322* (2023).

[94] Yuxuan Ma. 2021. distilgpt2-finetuned-wikitext2. https://huggingface.co/MYX4567/distilgpt2-finetuned-wikitext2

[95] Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. *CoRR* (2022).

[96] Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896* (2023).

[97] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *NeurIPS*.

[98] Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-Editing Memory in a Transformer. In *ICLR*.

[99] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. 2022. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147* (2022).

[100] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *Comput. Surveys* 56, 2 (2023), 1–40.

[101] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast Model Editing at Scale. In *ICLR*.

[102] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022. Memory-Based Model Editing at Scale. In *ICML*.

[103] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786* (2022).

[104] Shikhar Murty, Christopher D. Manning, Scott M. Lundberg, and Marco Túlio Ribeiro. 2022. Fixing Model Bugs with Natural Language Patches. In *EMNLP*.

[105] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299* (2022).

[106] Shiwen Ni, Dingwei Chen, Chengming Li, Xiping Hu, Ruifeng Xu, and Min Yang. 2023. Forgetting before Learning: Utilizing Parametric Arithmetic for Knowledge Updating in Large Language Models. *arXiv preprint arXiv:2311.08011* (2023).

[107] Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity Cloze By Date: What LMs Know About Unseen Entities. In *Findings of NAACL*.

[108] Yasumasa Onoe, Michael J. Q. Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can LMs Learn New Entities from Descriptions? Challenges in Propagating Injected Knowledge. In *ACL*.

[109] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[110] Hariom A. Pandya and Brijesh S. Bhatt. 2021. Question Answering Survey: Directions, Challenges, Datasets, Evaluation Matrices. arXiv:2112.03572 [cs.CL]

[111] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813* (2023).

[112] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. *arXiv preprint arXiv:2304.03277* (2023).

[113] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red Teaming Language Models with Language Models. In *EMNLP*.

[114] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a Benchmark for Knowledge Intensive Language Tasks. In *ACL*.

[115] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *EMNLP-IJCNLP*.

[116] Yuval Pinter and Michael Elhadad. 2023. Emptying the Ocean with a Spoon: Should We Edit Models?. In *EMNLP*.

[117] Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, and Jie Zhou. 2022. Exploring Universal Intrinsic Task Subspace via Prompt Tuning. arXiv:2110.07867 [cs.CL]

[118] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[119] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* (2020).

[120] Sachin Ravi and Hugo Larochelle. 2016. Optimization as a model for few-shot learning. In *International conference on learning representations*.

[121] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

[122] EHUD REITER and ROBERT DALE. 1997. Building applied natural language generation systems. *Natural Language Engineering* (1997).

[123] Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive testing and debugging of NLP models. In *ACL*.

[124] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, et al. 2021. Recipes for Building an Open-Domain Chatbot. In *EACL*.

[125] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. 2021. Editing a classifier by rewriting its prediction rules. In *NeurIPS*.

[126] Christoph Schuhmann, Robert Kaczmarczyk, Aran Komatsuzaki, Aarush Katta, Richard Vencu, Romain Beaumont, Jenia Jitsev, Theo Coombes, and Clayton Mullis. 2021. LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs. In *NeurIPS Workshop*

Datacentric AI.

[127] Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get Your Vitamin C! Robust Fact Verification with Contrastive Evidence. In *NAACL*.

[128] Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *EMNLP*.

[129] Gautam Kishore Shahi, Anne Dirkson, and Tim A Majchrzak. 2021. An exploratory study of COVID-19 misinformation on Twitter. *Online social networks and media* 22 (2021), 100104.

[130] Arnab Sen Sharma, David Atkinson, and David Bau. 2024. Locating and editing factual associations in mamba. *arXiv preprint arXiv:2404.03646* (2024).

[131] Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. Retrieval-enhanced knowledge editing for multi-hop question answering in language models. *arXiv preprint arXiv:2403.19631* (2024).

[132] 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 *EMNLP*.

[133] Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable Neural Networks. In *ICLR*.

[134] Chenyang Song, Xu Han, Zheni Zeng, Kuai Li, Chen Chen, Zhiyuan Liu, Maosong Sun, and Tao Yang. 2023. ConPET: Continual Parameter-Efficient Tuning for Large Language Models. *arXiv preprint arXiv:2309.14763* (2023).

[135] Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2023. Preference Ranking Optimization for Human Alignment. *arXiv preprint arXiv:2306.17492* (2023).

[136] Xiaoshuai Song, Zhengyang Wang, Keqing He, Guanting Dong, Jinxu Zhao, and Weiran Xu. 2024. Knowledge Editing on Black-box Large Language Models. *arXiv preprint arXiv:2402.08631* (2024).

[137] Felix Stahlberg. 2020. Neural machine translation: A review. *Journal of Artificial Intelligence Research* (2020).

[138] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975* (2022).

[139] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937* (2018).

[140] Ryutaro Tanno, Melanie F Pradier, Aditya Nori, and Yingzhen Li. 2022. Repairing Neural Networks by Leaving the Right Past Behind. In *NeurIPS*.

[141] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

[142] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine* (2023).

[143] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *ACL*.

[144] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[145] Joaquin Vanschoren. 2018. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548* (2018).

[146] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS* (2017).

[147] Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. 2022. Continual learning with hypernetworks. arXiv:1906.00695 [cs.LG]

[148] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* (2014).

[149] Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024. Detoxifying Large Language Models via Knowledge Editing. *arXiv preprint arXiv:2403.14472* (2024).

[150] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926* (2023).

[151] Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models. *arXiv preprint arXiv:2405.14768* (2024).

[152] Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. 2023. EasyEdit: An Easy-to-use Knowledge Editing Framework for Large Language Models. *arXiv preprint arXiv:2308.07269* (2023).

[153] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808* (2020).

[154] Weixuan Wang, Barry Haddow, and Alexandra Birch. 2023. Retrieval-augmented multilingual knowledge editing. *arXiv preprint arXiv:2312.13040* (2023).

[155] Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024. Deepedit: Knowledge editing as decoding with constraints. *arXiv preprint arXiv:2401.10471* (2024).

[156] Yu Wang, Xiusi Chen, Jingbo Shang, and Julian McAuley. 2024. MemoryLLM: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624* (2024).

[157] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-Instruct: Aligning Language Model with Self Generated Instructions. *arXiv preprint arXiv:2212.10560* (2022).

[158] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410* (2022).

[159] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966* (2023).

[160] Mayur Wankhade, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni. 2022. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review* (2022).

[161] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned Language Models are Zero-Shot Learners.

[162] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

[163] Zihao Wei, Jingcheng Deng, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2024. Mlake: Multilingual knowledge editing benchmark for large language models. *arXiv preprint arXiv:2404.04990* (2024).

[164] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. 2022. Robust fine-tuning of zero-shot models. In *CVPR*.

[165] Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: Detecting and Editing Privacy Neurons in Pretrained Language Models. In *EMNLP*.

[166] Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. 2024. Updating language models with unstructured facts: Towards practical knowledge editing. *arXiv preprint arXiv:2402.18909* (2024).

[167] 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 *EMNLP*.

[168] Junsang Yoon, Akshat Gupta, and Gopala Anumanchipalli. 2024. Is Bigger Edit Batch Size Always Better?–An Empirical Study on Model Editing with Llama-3. *arXiv preprint arXiv:2405.00664* (2024).

[169] Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *AAAI*.

[170] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *ACL*.

[171] Michael Zhang and Eunsol Choi. 2021. SituatedQA: Incorporating Extra-Linguistic Contexts into QA. In *EMNLP*.

[172] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286* (2024).

[173] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[174] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can We Edit Factual Knowledge by In-Context Learning? arXiv:2305.12740 [cs.CL]

[175] Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing Knowledge Editing in Language Models via Multi-Hop Questions. *CoRR* abs/2305.14795 (2023). arXiv:2305.14795

[176] Jiawei Zhou, Yixuan Zhang, Qianni Luo, Andrea G Parker, and Munmun De Choudhury. 2023. Synthetic lies: Understanding ai-generated misinformation and evaluating algorithmic and human solutions. In *CHI*. 1–20.

[177] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying Memories in Transformer Models. arXiv:2012.00363 [cs.CL]

[178] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2020. A comprehensive survey on transfer learning. *Proc. IEEE* (2020).

[179] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593* (2019).