THE UNIVERSITY OF CHICAGO

CONTINUAL LEARNING AND KNOWLEDGE EDITING OF LARGE LANGUAGE MODELS WITH RELEVANCE-BASED PARAMETER ACTIVATION

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
XIAOTIAN DUAN

CHICAGO, ILLINOIS
DECEMBER 2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation would not have been possible without the guidance of Prof. Rick Stevens and the mentorship of Dr. Fangfang Xia. I am also deeply thankful to my family and friends for their constant support and encouragement, and to my colleagues for their camaraderie and inspiration.

ABSTRACT

Continual learning is essential in fields with evolving data distributions, such as recommendation systems, autonomous vehicles, and social media language processing. The increasing cost of training advanced neural networks, with their growing parameters and data needs, makes continual learning an attractive approach. A key challenge here is *catastrophic forgetting* - a phenomenon where neural networks abruptly lose previously learned information when trained on new data. This occurs because the model's parameters, optimized for new tasks, overwrite representations learned for previous tasks. While various solutions exist, this issue remains unresolved.

Knowledge editing in Large Language Models (LLMs), closely related to continual learning, involves making targeted changes to specific data points. The aim is to fix inaccuracies or biases in these models without mistakenly altering other knowledge or skills. The research of knowledge editing focuses on three key aspects: *generalization* (the model's ability to apply edited information across various contexts), *locality* (precise edits without affecting unrelated information), and *scalability* (maintaining performance efficiency and stability with increasing edits). It's worth noting that the issue of locality is often a direct result of catastrophic forgetting.

This dissertation covers the work in continual learning Hard-Attention-to-the-Task, demonstrating state-of-the-art performance on the CVPR workshop challenge, and makes two contributions in the field of knowledge editing: (1) a new knowledge editing benchmark with expanded patterns and systematic neighbor sampling for comprehensive evaluation, and (2) Relevance-based Parameter Activation (Rel-Par-Act), which combines parameter-efficient fine-tuning with context-aware routing. Experimental results show Rel-Par-Act achieves 98.8% generalization accuracy while maintaining baseline locality performance. Moreover, our results systemically demonstrated that knowledge editing disproportionately affects semantically closely neighboring facts, while facts with strong representation in training data

demonstrate greater resilience to knowledge editing of neighboring facts. These findings highlight the interconnected nature of knowledge storage in LLMs.

CHAPTER 1

INTRODUCTION AND OVERVIEW

In an era where data constantly evolves, the ability of machine learning models to adapt and learn continuously is indispensable. Continual learning, particularly in fields with non-stationary data distributions, stands at the forefront of AI challenges, such as recommendation systems and autonomous vehicles. A primary challenge in this domain is overcoming catastrophic forgetting, wherein a model abruptly loses knowledge from previous tasks when exposed to new ones. The study of continual learning is frequently undertaken in the field of Computer Vision (CV), where one or more image classes are treated as a single task. The assessment of continual learning in this context involves sequentially training on these tasks and consistently evaluating the model's ability to maintain performance on tasks it has previously learned.

In the realm of Natural Language Processing (NLP), the challenges of continual learning take on an added dimension. The dynamic and ever-evolving nature of language, particularly evident in platforms like social media, necessitates LLMs that are capable not only of continuous learning but also of effectively addressing issues such as misinformation and biases. As LLMs increasingly become integral in disseminating information and facilitating decision-making, their ability to remain factually correct and unbiased is crucial for ethical applications. However, the sporadic nature of knowledge editing in LLMs and the opacity of their knowledge storage mechanisms pose the question:

Given the unclear nature of how knowledge is stored in LLMs, how do we precisely edit specific knowledge without impacting other knowledge and capabilities?

Knowledge editing in LLMs, a research field closely related to continual learning, emerges as a solution. Unlike fine-tuning, which is computationally demanding, knowledge editing aims to adjust a small subset of parameters or employ external memory/networks for edits.

However, existing methods fail to simultaneously achieve *generalization*, *specificity/locality*, and *scalability*. Consider the task of editing LLMs to recognize "Joe Biden as the US president in 2023." In this context, we must consider *generalization* to paraphrased or similar prompts, *specificity/locality* to ensure unrelated facts like "Donald Trump is the US president in 2020" remain unaffected, and textitscalability in terms of the number of allowable edits as well as the extra computational resources or memory required. Moreover, current benchmarks often fall short in comprehensive evaluation, suffering from an insufficient number of samples and patterns for both generalization and specificity/locality assessments.

This work presents two novel contributions to address these challenges:

1. **A New Comprehensive Benchmark for Knowledge Editing in LLMs**

   We developed an enhanced benchmark that addresses the limitations of existing datasets, particularly CounterFact. Our benchmark substantially expands the number of paraphrase prompts and neighbor prompts through systematic pattern augmentation and neighbor selection. This augmentation enables more effective fine-tuning by providing a richer set of training tokens, while the carefully curated neighbor selection allows for more rigorous evaluation of both generalization and locality. The benchmark includes diverse prompt patterns, systematic neighbor sampling, and carefully curated prompt prefixes, providing a more comprehensive framework for assessing knowledge editing methods.

2. **Relevance-based Parameter Activation (Rel-Par-Act)**

   We introduce Rel-Par-Act, a novel method for knowledge editing that combines parameter-efficient fine-tuning with context-aware activation. The method employs compact AdaLoRA modules for individual edits and a lightweight embedding-based context detector that determines when to activate these edited parameters. This approach achieves strong performance across all three critical aspects of knowledge editing:

- **Generalization:** By leveraging augmented training prompts, Rel-Par-Act demonstrates robust performance across diverse phrasings and contexts
- **Locality:** Through precise context detection and isolated parameter updates, the method maintains high accuracy on related but unedited facts
- **Scalability:** The parameter-efficient design and modular architecture enable efficient handling of multiple edits with minimal computational overhead

Our experimental results demonstrate that Rel-Par-Act achieves state-of-the-art performance on both our enhanced benchmark and existing datasets. The method shows particular strength in maintaining locality while achieving high generalization accuracy, addressing a key challenge in knowledge editing. Furthermore, its computational efficiency and scalability make it practical for real-world applications where multiple edits may be required.

The remainder of this paper is organized as follows: Chapter 2 provides comprehensive background on continual learning and knowledge editing in LLMs. Chapter 3 discusses key open problems and challenges in the field. Chapter 4 details our benchmark construction and the Rel-Par-Act method. Chapter 5 presents our experimental results and analysis. Chapter 6 provides broader discussion of our findings and their implications for the field. Finally, Chapter A provides additional technical details and supplementary materials.

CHAPTER 2

BACKGROUND AND RELATED WORKS

This chapter delves into the recent advancements in the fields of continual learning and knowledge editing in LLMs. The landscape of continual learning in AI has evolved significantly over the years, marked by a variety of approaches aimed at mitigating catastrophic forgetting and enhancing adaptability in dynamic environments. These methods are typically evaluated using image classification benchmarks, where different classes are segmented into sequential tasks to measure the model's performance in continual learning scenarios.

More recently, the research of knowledge editing in LLMs has gained momentum, addressing the critical need for LLMs to stay factually correct and unbiased. Commonly used techniques in this area include the use of external memory and parameters with routing, locate-and-modify, and global optimization. To evaluate these methods, researchers have repurposed question-answering benchmarks and, more innovatively, employed datasets of counterfactual statements and their neighboring contexts. The latter presents more challenges due to the counterfactual nature of the information that requires editing.

In this chapter, we will examine the methodologies and evaluations in both fields, highlighting how they preserve learned knowledge while integrating new information, thereby paving the way for our proposed benchmark and method.

2.1 Continual Learning

2.1.1 Introduction

Continual learning, also known as lifelong learning or incremental learning, is a fundamental concept in machine learning and artificial intelligence where a model continuously learns over time by accommodating new knowledge while retaining previously learned information. This concept is crucial in dynamic environments where data continually evolves, such as

autonomous vehicles and surveillance in CV, and chatbots and language translation in NLP.

This chapter explores the fundamentals of continual learning, highlighting its practical implementation and the challenges it faces in both CV and NLP. We begin by defining continual learning and addressing its primary challenge: catastrophic forgetting, which is the tendency of a model to overwrite old information with new data. We then discuss various scenarios, related research fields, and key methodologies, such as rehearsal methods and dynamic architecture. Following this, we delve into its application in CV and NLP, examining notable research, common evaluation techniques, and potential future directions.

Finally, we discuss our contributions to the field, including an improved hard-attention-to-the-task mechanism and the investigation of alternative continual learning scenarios in computer vision. These contributions were recognized at the 4th continual learning workshop during the CVPR conference, where our strategy won the workshop challenge, underscoring its relevance and impact in the field.

2.1.2 Fundamentals of Continual Learning

Continual learning in machine learning is an approach where a model is progressively trained on a sequence of data over time, distinct from traditional static machine learning methods. This approach emulates the human capability to constantly acquire, refine, and transfer skills and knowledge over a lifetime. The realm of continual learning in AI is extensive, covering numerous scenarios, each presenting distinct challenges and requirements.

Definitions and Scenarios: The three primary scenarios in continual learning were initially proposed in van de Ven and Tolias [2018], and these definitions remain widely accepted:

1. **Task Incremental Learning (TIL)**: In this scenario, the model sequentially learns a variety of tasks, with the task IDs provided at test time. The primary focus is to prevent forgetting during the learning of new tasks. A strong baseline approach in TIL

is to train separate models for each task, however, it's often impractical in real-world applications due to resource constraints.

2. **Domain Incremental Learning (DIL)**: Here, the model encounters data from different domains or distributions, while the predictive target remains consistent. The task IDs are provided during training but not at test time. The primary focus is on generalizing the acquired knowledge across these domains.

3. **Class Incremental Learning (CIL)**: Similar to DIL, the model learns new tasks over time. However, the predictive target includes the task IDs, meaning the model must predict the target and infer which task the sample comes from. CIL, with its task-agnostic approach, closely resembles real-world applications where models may have to predict the domain shift during inference.

These three scenarios laid the foundation for continual learning, focusing on different aspects of the learning, with increasing difficulty. Beyond these initial scenarios, continual learning has evolved to include the following scenarios to further simulate real-world data dynamics better:

- **Online Continual Learning (OCL)**: Involves a continuous, one-pass stream of data, where distinct tasks are not explicitly separated, and task IDs are not available during both training and testing. This scenario, proposed in Aljundi et al. [2019b], represents a significant challenge in adapting to new data without the guidance of task identifiers.

- **Class-Incremental Learning with Repetition (CIR)**: Similar to traditional CIL, but tasks have overlapping data labels, and task IDs are available during training but not during testing. This scenario, proposed in Hemati et al. [2023], emphasizes the model's ability to handle ambiguities in data classification when task boundaries are not given during the inference.

- **Task-Free Continual Learning (TFCL)**: Involves tasks with disjoint data labels, with no task IDs provided during training or testing. This approach, proposed in Aljundi et al. [2019a], is particularly relevant for scenarios where task identification is either impossible or impractical.

- **Blurry CIL or General Continual Learning (GCL)**: Characterized by blurred task boundaries with overlapping data label spaces between tasks. Proposed in Buzzega et al. [2020] and Bang et al. [2021], this scenario challenges models to adapt to a less structured learning environment, closely mirroring many real-world applications.

There exists other scenarios focusing on niche applications, however the core objectives and challenges remain the same. Generally speaking, there are two main objectives in continual learning Lopez-Paz and Ranzato [2017]:

- *Backward transfer (BWT)* refers to the influence, positive or negative, that learning a task has on the performance on previous tasks. A negative BWT indicates catastrophic forgetting, while a positive BWT shows the newly learned knowledge helps with previous tasks.

- *Forward transfer (FWT)* refers to the influence that learning a task has on performance on future tasks. A positive FWT signifies zero-shrt learning, usually indicating useful features learned from past tasks.

Quantifying BWT and FWT involves evaluating the performance of a model on past and future tasks, respectively, and is crucial for understanding the effectiveness of a continual learning approach.

In continual learning, we aim to ensure the knowledge learned on any task has a positive influence on both previous and future tasks, regardless of the settings and scenarios. Given the myriad of scenarios in continual learning, it's important to understand their respective complexities and applicability. Furthermore, with advancements in fields like NLP, we may

anticipate the emergence of even more nuanced and complex continual learning scenarios. The diversity in continual learning scenarios reflects the multifaceted nature of real-world data and the evolving challenges in AI. Understanding these scenarios helps us develop more adaptable, efficient, and robust AI systems.

2.1.3 Challenges in Continual Learning

Continual learning offers significant opportunities for creating adaptive and dynamic AI systems, but also poses unique challenges that are crucial to address for effective and sustainable learning paradigms, especially in practical real-world deployments.

The primary challenge in continual learning is *catastrophic forgetting*. This significant issue, where learning a new task often results in a significant and abrupt degradation of performance on previously learned tasks, is observed in neural networks using back-propagation optimization McCloskey and Cohen [1989]. This effect typically occurs in scenarios where data distribution changes over time, such as learning on a sequence of different tasks. These scenarios defy the independent and identically distributed (IID) assumption of machine learning and invalidate conventional learning methods, necessitating the development of paradigms that operate under non-IID conditions.

Closely related to catastrophic forgetting, the *stability-plasticity dilemma* is central to continual learning. This challenge involves balancing the ability to learn new tasks (plasticity) with retaining knowledge of old tasks (stability) Mermillod et al. [2013]. These two abilities are often in conflict: excessive stability can hinder the learning of new information, while excessive plasticity may lead to rapid and severe forgetting. This dilemma is particularly challenging in scenarios where datasets from previous tasks are unavailable during the learning of new tasks, or in situations requiring learning from scratch. In such cases, it becomes difficult to leverage the power of retraining on previous samples or a pre-trained representation that generalizes well across all tasks.

Other challenges arise in applying continual learning to real-world scenarios. For example, the computational demands for updating parameters to balance stability and plasticity may be impractical in some cases. Scalability issues also emerge, particularly for methods that require storing samples from previously trained tasks. Additionally, defining the continual learning scenario itself poses a challenge, as there may be additional parameters and realistic considerations not covered in existing scenarios.

2.1.4 Related Areas

Continual learning, as a paradigm, intersects with numerous other fields in machine learning, each contributing uniquely to its core objectives. These fields address the challenges of adapting to new data, retaining, altering or forgetting previously learned information, and evolving in dynamic environments. This chapter delves into the intricate tapestry of related fields that are integral to continual learning. Each of these research areas not only contributes to the continual learning framework but also provides a unique lens through which we can examine and enhance the adaptability and intelligence of AI systems.

Multi-Task Learning (MTL) is a vital field closely related to continual learning Crawshaw [2020], Zhang and Yang [2021]. It involves training a model on multiple tasks simultaneously, allowing it to learn shared representations that are beneficial across different tasks. By learning commonalities between tasks, MTL methods can efficiently transfer knowledge across tasks, improving their overall performance and adaptability. This approach is particularly relevant to continual learning as it fosters the development of versatile models capable of handling a variety of tasks without needing separate training for each. Notably, the modular architecture of MTL is shared by continual learning methods, such as PathNet Fernando et al. [2017]. Moreover, MTL is often considered an upper bound for continual learning due to its unrestricted knowledge transfer in both directions.

Transfer Learning (TL) focuses on leveraging knowledge acquired from previously en-

countered tasks to enhance performance on the current task Zhuang et al. [2020]. Within the context of continual learning, knowledge transfer operates in both directions: FWT utilizes knowledge from past tasks to improve learning on the current task, while BWT strengthens the model's understanding of previous tasks after learning the current one. While continual learning is more concerned backward transfer and the mitigation of catastrophic forgetting, TL primarily emphasizes forward transfer and target task performance. However, specific approaches within TL, such as feature and parameter -based ones, remain highly relevant to continual learning. Feature representation transfer aims to discover representations that generalize across diverse tasks, while parameter transfer focuses on sharing parameters between different tasks, ultimately promoting efficient knowledge transfer and enhanced performance.

Meta-Learning, also described as "learning to learn", optimizes models to adapt to new tasks using past experiences Hospedales et al. [2021]. This methodology parallels continual learning's aim to adapt through a series of tasks, although it primarily focuses on rapid adaptation rather than addressing catastrophic forgetting. Nonetheless, several continual learning methods, such as MER Riemer et al. [2018], utilize meta-learning approaches, particularly incorporating replay samples to mitigate the negative impact of new tasks on previously acquired knowledge.

Selective Forgetting, also referred to as graceful or active forgetting, is an emerging field closely linked to continual learning Wang et al. [2023d]. This area focuses on strategically discarding outdated, inaccurate, or irrelevant information, a crucial process for maintaining a model's learning capacity and generalization while safeguarding critical past knowledge. By shedding light on how knowledge is stored within the model's parameters, selective forgetting offers valuable insights that can be leveraged to mitigate catastrophic forgetting in continual learning scenarios.

Knowledge Editing (in LLMs) involves actively updating a model's knowledge base to reflect new or corrected information Mazzia et al. [2023], Wang et al. [2023c]. Similar

to selective forgetting, this process helps identify and understand how knowledge is stored within the model. While encompassing both knowledge updating and continuous learning aspects, knowledge editing may not always involve explicit network training. Notably, the core objectives of these continual learning and knowledge editng exhibit significant overlap, albeit within distinct settings. Knowledge editing primarily focuses on specific edits without altering the model's predictive target, while continual learning often involves learning new tasks encompassing different domains or data labels. Knowledge editing methods and continual learning methods can mutually benefit from each other, particularly within the context of regularization-based approaches where both areas employ regularization to prevent the loss of previously acquired knowledge. A more comprehensive discussion of knowledge editing will be presented in Section 2.3.

In summary, these areas collectively contribute to the advancement of continual learning, each offering unique strategies to enhance AI systems' adaptability, knowledge retention, and task versatility.

2.1.5 Common Approaches in Continual Learning

Various methods have been proposed to address the challenges in continual learning, particularly focusing on backward transfer and mitigating catastrophic forgetting. We discuss common methodologies in continual learning, categorized into three approaches: regularization-based, replay-based, architecture-based.

Regularization-based Approach

These methods involves adding explicit regularization terms or manipulating the optimization process to accommodate both new and old tasks. They can be further divided into the following three subcategories:

Prior-Focused Weight Regularization: These methods focus on regularizing the up-

dates of network parameters to preserve knowledge from previous tasks. By applying a regularization or penalty term, they constrain the parameters deemed important for past tasks. For instance, Elastic Weight Consolidation (EWC) determine importance in Bayesian Framework based on the Fisher information matrix calculated after training each task Kirkpatrick et al. [2017]. Synaptic Intelligence (SI) approximates parameter importance by measuring the accumulated update distance during training Zenke et al. [2017]. Memory Aware Synapses (MAS) assesses importance by the sensitivity of the predicted output to changes in a parameter Aljundi et al. [2018]. Other variations such as Riemannian Walk Chaudhry et al. [2018a], which combines EWC and SI principles, maintain the same goal: to minimize changes in parameters crucial for the prediction of previous tasks. After assessing parameter importance, a quadratic penalty is usually added to the loss function to penalize significant variations in these important parameters.

Data-Focused Function Regularization: This approach aims to preserve previously acquired knowledge by focusing on the predictive function of the model. It involves setting constraints on the predictive outcomes of the intermediate or final layers of the network to ensure the model remains aligned with prior tasks, thereby preserving learned knowledge. This is typically achieved through various forms of knowledge distillation Hinton et al. [2015]. Notable among these is Learning without Forgetting (LwF), which proposes a training process that focuses on new task samples while simultaneously constraining the predictive output for samples from older tasks Li and Hoiem [2017]. To reduce the need for storing image samples and lower memory requirements, Learning without Memorizing (LwM) was developed. This method enables the model to emulate the attention patterns of a previously trained model (teacher model) during training on a new task Dhar et al. [2019]. Extending the basic concept of LwF, Encoder Based Lifelong Learning (EBLL) preserves the feature representations of task-specific autoencoders trained on previous tasks, in addition to employing knowledge distillation loss Rannen et al. [2017]. Deep Model Consolidation (DMC), involves training

a separate model for the new class and then consolidating it with the model trained on previous tasks, achieving this integration through knowledge distillation Zhang et al. [2020].

Gradient Projection: This method alters the optimization process directly, eliminating the need for additional objectives or regularization terms. A key example is Gradient Episodic Memory (GEM) Lopez-Paz and Ranzato [2017]. When training on a new task, GEM assesses the gradient of the loss function for the current task against the gradients of the samples from the previous tasks preserved in memory. It then adjusts the gradients to ensure that the learning process does not increase the loss for earlier tasks. Averaged Gradient Episodic Memory (A-GEM) relaxed this approach by using an average gradient derived from samples of previous tasks in memory, which reduces computational intensity Chaudhry et al. [2018b].

Replay-based Approach

Replay-based methods in continual learning focus on approximating and recovering old data distributions by using stored or synthetic samples or representations from previous tasks. By integrating the old data with the new during training, the model achieves replay or rehearsal, thereby mitigating forgetting. Replay-based approaches can be divided into two subcategories based on the nature of the replay samples:

Experience Replay: These methods store select training samples from previous tasks. The primary challenge lies in managing the limited memory buffer to adhere to the principles of continual learning, which discourages extensive reliance on past data. To fully utilize the replay samples, experience replay methods are rarely standalone solutions, but often paired with regulation-based methods for better results. The emphasis in experience replay is on the strategic selection of samples and efficient memory utilization. For instance, Incremental Classifier and Representation Learning (iCaRL) dynamically maintains a set of exemplars, chosen for their class representativeness, and incorporates them into new task

training Rebuffi et al. [2017]. It also utilizes distillation loss, integrating data-focused function regularization, and performs classification using Nearest-Mean-of-Exemplars, addressing the issue of unusable classification heads from previous classes due to network representation updates. Dark Experience Replay (DER) and its variant DER++ blend knowledge distillation with standard experience replay by storing and replaying not only past task samples but also the network's predictions on this data Buzzega et al. [2020]. Another notable method is Meta-Experience Replay (MER), which, like other replay methods, trains with a mixed batch of past task samples Riemer et al. [2018]. However, MER also incorporates meta-learning, assessing whether gradient updates impact performance on stored samples, similar to GEM and A-GEM. It's important to note that some literature such as Wang et al. [2023a] categorizes these methods as regularization-based due to their extensive use of replay samples during optimization.

Generative Replay or Pseudo Replay: Different from exact experience replay, this method does not necessitate explicit storage of past learning samples. Instead, replay samples are generated using advanced generative models such as variational autoencoder (VAE) or generative adversarial network (GAN) Kingma and Welling [2013], Goodfellow et al. [2014]. This is particularly advantageous when training samples cannot be stored, due to concerns like privacy, or the memory buffer size is too small for effective experience replay. Deep Generative Replay (DGR) laid the foundation for generative replay methods, using a GAN to continuously learn and generate replay samples, ensuring the classifier does not forget previous tasks Shin et al. [2017]. Memory Replay GANs (MeRGAN) recognized that sequential fine tuning of generative models introduces forgetting, and thereby employs replay alignment, which is essentially knowledge distillation, on the generative model, to prevent the forgetting of the generative model, and thereby improve the performance of the classifier Wu et al. [2018]. Additionally, FearNet uses a dual-memory system, akin to human memory systems, where long-term memory is consolidated by generative replay with a VAE Kemker

and Kanan [2017].

Architecture-based Approach

These methods rely on network architecture to prevent catastrophic forgetting. Often referred to as the parameter-isolation approach De Lange et al. [2021], these methods isolate different parts of the network for different tasks. By restricting parameter updates associated with previous tasks, they aim to significantly reduce or completely overcome catastrophic forgetting in TIL scenario. However, they face challenges in scenarios where task IDs are not available during testing, as it is challenging to select the appropriate network segment for an given sample. Generally speaking, there are two distinct approaches on how to isolate the network parameters:

Fixed Network Methods: In this approach, a fixed neural network structure is used, maintaining the same structure throughout the continual learning process for scalability. On the other hand, the learning capability is also limited. These methods segment the fixed network into dedicated sections for different tasks, often using task-specific binary masks. PathNet, for example, employs a genetic algorithm to find an optimal pathway for each task Fernando et al. [2017]. PackNet achieves this segmentation by pruning and retraining post-training to identify necessary parameters for the current task Mallya and Lazebnik [2018]. P iggyback, on the other hand, employs an end-to-end training approach on pre-trained networks, using the gradient of the mask threshold as binary masks are non-differentiable Mallya et al. [2018]. Hard Attention to the Task (HAT) differs by using trainable sigmoid masks with a scale parameter to simulate binary masks, training the masks and backbone parameters simultaneously, providing more task learning flexibility Serra et al. [2018].

Dynamic Architectures: This approach involves allocating extra parameters for each new task, focusing on efficient parameter reuse and knowledge transfer. Progressive Neural Networks (PNN) add new columns (subnetworks) for each task, connected laterally to pre-

vious columns, to leverage past learning Rusu et al. [2016]. This approach, however, leads to a parameter increase proportional to the number of tasks. Expert Gate also adopts a multi-column approach, with each column specializing in a task and a routing module acting as a gate Aljundi et al. [2017]. To control parameter growth, Dynamically Expandable Networks (DEN) selectively retrain parameters, building sparse connections between layers Yoon et al. [2017]. If performance is suboptimal, DEN expands by adding neurons. If the retrained parameters deviate significantly, they are duplicated, maintaining one set for previous tasks and updating the other for current tasks.

Each of these approaches presents distinct advantages and limitations. Currently, it is challenging to identify a universally superior method, as the effectiveness often depends on specific contextual factors. For example, replay-based approaches are straightforward and effective but may be resource-intensive, requiring significant memory for replay samples or generative models for pseudo replay. Conversely, architecture-based methods like HAT excel when task labels are available during inference, achieving zero forgetting while accommodating new learning. This diversity, coupled with various continual learning scenarios, underscores the field's complexity and dynamism.

Moreover, these approaches are not mutually exclusive. In fact, many of the methods discussed are combinations of different strategies. Generally, integrating various approaches often yields better results than standalone solutions. In addition to these strategies, many methods borrow ideas from related fields. Representation learning, for instance, enhances continual learning methods by providing generalized representations across tasks. This often bridges the gap, enabling architecture-based methods to perform task-agnostic predictions. For example, Continual Learning based on Out-of-Distribution (OOD) detection and Task Masking (CLOM) combines supervised contrastive learning (SupCon) Khosla et al. [2020] with HAT Serra et al. [2018]. This combination creates a feature space robust for OOD detection, facilitating task-agnostic inference Kim et al. [2022]. Methods like L2P Wang

et al. [2022b] and DualPrompt Wang et al. [2022a] utilize vision foundation models, such as pretrained Vision Transformers (ViTs) Dosovitskiy et al. [2020]. These models adapt based on prompting, creating a dynamic prompt pool that adjusts to different tasks and achieves state-of-the-art performance across multiple benchmarks. Notably, in methods like L2P and DualPrompt, the training process primarily involves updating the prompt pool, while the network remains unchanged, significantly minimizing forgetting during training.

Reflecting on these recent advancements, a prominent trend in the development of continual learning is the fusion of concepts from other disciplines, such as meta-learning and representation learning, while adapting to recent architectural innovations like transformers Vaswani et al. [2017].

2.1.6 Continual Learning in CV

Continual learning has seen extensive research in the domain of CV. The three commonly used scenarios of continual learning (TIL, DIL, and CIL) van de Ven and Tolias [2018] and most notable methods, such as EWC (Kirkpatrick et al. [2017]) , LwF (Li and Hoiem [2017]), and HAT (Serra et al. [2018]), were all proposed in the context of CV. This is partially due to the fact that image classification can be easily split into sequential tasks suitable for continual learning. Furthermore, image classification models, often trained from scratch, are easier to train and evaluate for continual learning.

The evaluation of continual learning methods in CV typically involves datasets like MNIST LeCun et al. [1998], CIFAR Krizhevsky and Hinton [2009], and TinyImageNet Le and Yang [2015], where images with different classes are split into separate tasks based on the scenario requirement. Average accuracy across the sequence of tasks is the most common metric, alongside forgetting measured by reductions in accuracy on previous tasks. For replay-based and dynamic architecture methods, additional metrics include memory consumption and computational requirements to ensure scalability. However, the focus of

continual learning research in CV has primarily been on mitigating catastrophic forgetting, neglecting the evaluation for knowledge transfer, especially forward transfer (how knowledge gained from previous tasks benefits future tasks).

More nuanced scenarios, such as OCL (one-pass data stream) Aljundi et al. [2019b] and GCL (no task boundaries) Buzzega et al. [2020], Bang et al. [2021], have emerged and gained traction in this field. Additionally, the rise of vision foundation models like ViT Dosovitskiy et al. [2020] mark a shift towards pre-trained models in continual learning research for CV. The generalized representations learned by these models naturally lend themselves to continual learning due to their strong cross-task generalization capabilities. This evolution in CV sets a precedent for continual learning in NLP, where LLMs have recently become mainstream Bommasani et al. [2021].

2.1.7 Continual Learning in NLP

Continual learning is crucial for NLP tasks due to the rapidly evolving nature of language and constantly emerging new information. While it shares some similarities with continual learning in CV, continual learning in NLP presents unique challenges and opportunities.

- **Domain-incremental learning**: In NLP, continual learning predominantly focuses on domain-incremental learning, with or without task IDs. Unlike CV, where the predictive target varies with image classes, in NLP, it's the domain shift of input data that's pivotal, since the predictive target remains constant across tasks.

- **Scalability Issue**: NLP faces unique scalability challenges, such as the computational cost of processing large text corpora and the memory requirements for the training and inference of LLMs. These factors make some methods, like gradient projection or dynamic network structures, much less feasible in NLP compared to CV.

- **Pre-trained LLMs**: The rise of pre-trained LLMs like BERT and GPT-3 provides a

powerful foundation for NLP Bommasani et al. [2021]. These LLMs, with their generalizable representations of language, require less fine-tuning for specific tasks compared to traditional CV models. This enables researchers to focus on knowledge transfer and leverage the power of pre-training for continual learning. Additionally, the ability of LLMs to perform in-context learning paves the way for new continual learning approaches specifically tailored to them.

Several methods have been proposed for continual learning in NLP, often adapting ideas from CV Ke and Liu [2022]. For instance, Regularized Memory Recall with Domain Shift Estimation (RMR_DSE) employs EWC-based Kirkpatrick et al. [2017] regularization to address catastrophic forgetting Li et al. [2022a]. ExtendNER Monaikul et al. [2021], Lifelong Intent Detection (LID) Liu et al. [2021], and Continual Few-shot Intent Detection (CFID) Li et al. [2022b] utilize knowledge distillation, similar to LwF Li and Hoiem [2017]. Parameter isolation methods, such as BERT-based Continual Learning (B-CL) Ke et al. [2021] and Continual Post-Training (CPT) Ke et al. [2022], uses task-specific binary masks proposed in HAT Serra et al. [2018] to prevent catastrophic forgetting by assigning parameters for different tasks. Replay methods have also been adapted for NLP. For instance, Continual-T0 (CT0) reserves a small proportion of training data, typically up to 1%, from previous tasks for exact experience replay Scialom et al. [2022]. Prompt Conditioned VAE for Lifelong Learning (PCLL) employs a conditional variational autoencoder (CVAE) to generate representative samples for pseudo replay Zhao et al. [2022].

An approach unique to continual learning in NLP is the instruction-based approach, leveraging the in-context learning capability of LLMs. This approach, exemplified by Continual Learning from Task Instructions (ConTinTin) Yin et al. [2022], uses specific instructions for each task to enable knowledge transfer by sharing the same trained LLM, thus offering a novel pathway for continual learning.

Despite these advancements, a standardized benchmark for evaluating continual learning

in NLP is yet to be established. This highlights the ongoing need for comprehensive frameworks to assess these novel methodologies effectively. While accuracy is a common metric for continual learning in CV, it is often insufficient for NLP tasks that has different objectives, such as named entity recognition, summarizing, intent classification, etc.. Developing effective metrics for evaluating knowledge transfer and catastrophic forgetting remains an ongoing challenge. Currently, most evaluation strategies involve sequencing standard NLP tasks, a method that is often limited by the similarity of these tasks, complicating the assessment of catastrophic forgetting. A potential solution to this issue could be the creation of benchmarks that encompass a more diverse array of NLP tasks, varying in both similarity and complexity, to provide a more robust evaluation of continual learning methods.

The field of continual learning in NLP, particularly with LLMs, is rapidly advancing but faces notable challenges like catastrophic forgetting and scalability issue. Empirical studies reveal that LLMs often lose their broad knowledge base and reasoning skills during intensive domain-specific fine-tuning. This raises a critical question: how can we fine-tune LLMs on specific datasets without compromising their general capabilities? Addressing this question involves exploring strategies that maintain a balance between the specialized performance in specific domains and the preservation of the broad, generalist nature of LLMs. Additionally, delving into how these models scale with increasing data size or complexity is crucial. Overall, continual learning in NLP continues to be a dynamic research area, with significant potential for breakthroughs in model robustness, adaptability, and scalability.

2.2 Our Contributions in Continual Learning

Our contributions to continual learning are twofold. First, we implemented a new Hard-Attention-to-the-Task (HAT) library called HAT-CL, which offers improved performance, versatile application, and seamless integration with existing frameworks and libraries. Second, we tackled the Class-Incremental Learning with Repetition (CIR) scenario, integrating

HAT-CL with other strategies to address the issue of catastrophic forgetting in such a realistic dynamic data environment.

2.2.1 HAT-CL: An Improved Hard-Attention-to-the-Task Implementation

Catastrophic forgetting is a significant challenge in continual learning, where a neural network loses previously learned information while learning new tasks. The HAT mechanism, proposed in Serra et al. [2018], uses attention masks to modulate the contribution of each neuron to specific tasks. This mechanism facilitates efficient allocation of network capacity across multiple tasks, preventing the loss of previously learned tasks when new ones are introduced.

Our HAT-CL implementation improves the Hard-Attention-to-the-Task mechanism, addressing key limitations in the original design while introducing several technical innovations. The core improvements focus on three key areas: gradient handling, mask initialization and scaling, and parameter isolation.

Core Mechanism and Improvements

For a network with L weighted layers, each task $t \in [0, 1, ..., T-1]$ has a set of embeddings $\mathbf{e}_l^t$, where $l \in [0, 1, ..., L-1]$ indexes the layers. The dimensions of these embeddings correspond to the output dimensions of each layer. During the forward pass, the network output is masked by element-wise multiplication between the original layer output $\mathbf{h}_l$ and the expanded mask $\mathbf{a}_l^t$:

$$\begin{aligned} \mathbf{a}_l^t(s) &= \sigma(s\mathbf{e}_l^t) \\ \mathbf{h}_l' &= \mathbf{a}_l^t(s) \odot \mathbf{h}_l \end{aligned} \tag{2.1}$$

Here, σ is the sigmoid function serving as the attention gate, and s governs the "hardness" of the masks. Smaller scales make the mask embeddings more trainable, while larger scales

push the mask towards a binary state, which is desirable for assigning parameters to specific tasks.

During the training for a new task t, it's crucial to preserve the knowledge from previous tasks. This is achieved in the backward pass. Here, the gradients of parameters not related to the current task are set to zero. This process involves creating a binary mask (by setting s to infinity) and using it to filter the layer gradients $\mathbf{g}_l$:

$$\begin{aligned} \mathbf{a}_l^{\leq(t-1)} &= \max\left(\mathbf{a}_l^{(t-1)}, \mathbf{a}_l^{\leq(t-2)}\right) \quad \text{, where } s = \infty \\ g'^t_{l,ij} &= \left[1 - \min\left(a_{l,i}^{\leq(t-1)}, a_{l-1,j}^{\leq(t-1)}\right)\right] g^t_{l,ij} \end{aligned} \tag{2.2}$$

This mechanism is different from the original model, which used non-binary masks (with a finite s_{max}) during gradient manipulation. Such non-binary masks allow some gradient flow for parameters that are associated with previous tasks, leading to a small degree of gradient 'leakage' and forgetting. Our approach, with binary masks, aims to provide a more stringent isolation of task-specific parameters, completely nullifying this leakage and preserving the knowledge of previously learned tasks.

These equations (2.1 and 2.2) encapsulate the core of the HAT mechanism: isolating parameters for different tasks and nullifying the gradients of parameters associated with previous tasks. This process is demonstrated by the

However, the training of masks is slow due to smaller gradients. To compensate, we adjust the gradients of the mask embeddings $q_{l,i}$:

$$q'_{l,i} = \frac{s_{\max}\left[\cosh\left(se^t_{l,i}\right) + 1\right]}{s\left[\cosh\left(e^t_{l,i}\right) + 1\right]} q_{l,i} \tag{2.3}$$

Here, $s_{\max}$ is the upper bound of the mask scale. For numerical stability, we clamp the mask embeddings $e^t_{l,i}$ and the scaled term $se^t_{l,i}$. During each training epoch, the scale of

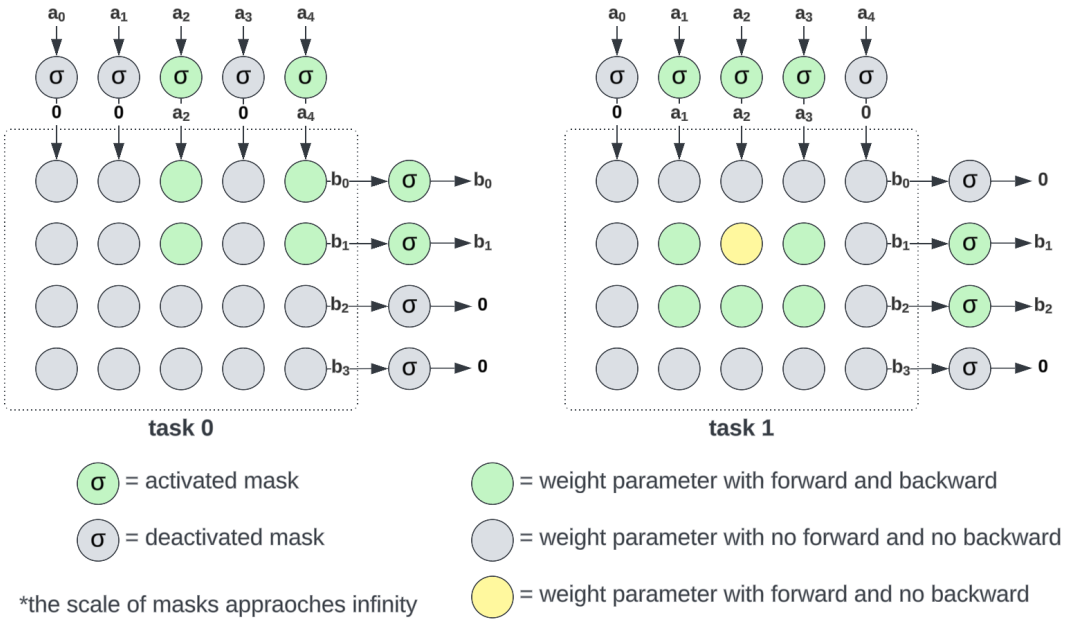

Figure 2.1: HAT Mechanism

How HAT manages neuron states during training. Active neurons (green) fully participate in both forward and backward passes of the current task. Locked neurons (yellow) allow forward propagation but block gradient flow during backpropagation, protecting knowledge from previous tasks. Deactivated neurons (grey) remain unused, participating in neither forward nor backward passes. As training progresses on each task, HAT learns which neurons to activate while automatically locking neurons that were important for previous tasks. This selective activation and locking mechanism, controlled by learned attention masks, enables the network to acquire new knowledge while preserving previously learned tasks.

masks increases linearly from a really small number to $s_{\max}$:

$$s = \frac{1}{s_{\max}} + \left(s_{\max} - \frac{1}{s_{\max}}\right)\frac{b-1}{B-1} \tag{2.4}$$

where $b = 1, 2, \ldots, B$ is the index of the current batch, and B is the total number of batches.

Lastly, to prevent overuse of the masks for the current task, we introduce a regularization term:

$$R_t = \sum_{l=0}^{L-1} \max\left(\frac{\sum_i a_{l,i}^t (1 - a_{l,i}^{<t})}{\sum_i (1 - a_{l,i}^{<t})} - \frac{1}{T}, 0\right) \tag{2.5}$$

This term 2.5 differs from the original work in two ways: (1) it calculates the regularization per layer and then sums it up, and (2) it incorporates a task quota, such that a task does not get penalized if its mask usage does not exceed the quota.

Improved Initialization and Scaling Strategy

The original HAT implementation initializes mask embeddings using a Gaussian distribution $\mathcal{N}(0, 1)$ and employs linear scaling as shown in Equation 2.4. This approach presents two key issues: (1) features that should contribute positively may be initialized with negative weights, leading to incorrect mask suppression, and (2) the linear scaling provides insufficient time for weight alignment before mask training begins. To address these limitations, we propose two fundamental changes. First, we initialize all mask embeddings to a constant value of 1, ensuring all parameters are initially available for training. Second, we replace the linear scaling with a cosine-based approach:

$$s = \frac{s_{\max}}{2} \cdot (1 + \cos(p \cdot 2\pi)) \tag{2.6}$$

where $p \in [0, 1]$ represents the progress within the current training epoch. This creates three distinct phases:

- Initial Phase ($p \approx 0$): High scale value restricts mask training, allowing weights to align properly
- Middle Phase ($p \approx 0.5$): Near-zero scale enables mask training with aligned weights
- Final Phase ($p \approx 1$): Returning high scale fine-tunes weights with nearly binary masks

Our empirical results demonstrate that this strategy significantly improves training efficiency. For a test network with five input features where only three are relevant, our approach required an average of 25 batches to converge, compared to 338.5 batches with the original implementation Duan [2023].

Implementation Architecture

HAT-CL leverages PyTorch's hook mechanism to automate gradient manipulation and mask application. By registering forward and backward hooks on weighted layers, we eliminate the need for manual gradient modifications while maintaining the mathematical correctness of the HAT mechanism. This design choice significantly simplifies the implementation compared to the original approach, which required explicit gradient calculations and manual parameter updates.

The library offers two primary module types: HAT modules for weighted layers (e.g., *HATLinear*, *HATConv2d*) and TaskIndexed modules for task-specific layers (e.g., *TaskIndexedBatchNorm2d*, *TaskIndexedBatchNorm2d*). HAT modules encapsulate the attention mechanism within weighted layers, while *TaskIndexed* modules handle parameter isolation through separate submodules for each task. This modular design allows for easy integration with existing PyTorch networks.

To enhance accessibility and reusability, HAT-CL integrates seamlessly with the PyTorch Image Models (TIMM) library. We provide HAT versions of popular architectures by appending the *hat_* prefix to model identifiers (e.g., *hat_resnet18* for a HAT version of ResNet18). These models support pretrained weights from their original counterparts, with HAT modules automatically inheriting base layer weights and TaskIndexed modules replicating weights across task-specific instances. This integration enables researchers to quickly convert and experiment with state-of-the-art architectures in continual learning scenarios.

2.2.2 Tackling CIR with HAT-CL and Other Strategies

During the Computer Vision and Pattern Recognition (CVPR) continual learning workshop, we were awarded the first prize in the challenge focused on CIR Hemati et al. [2023]. This challenge aims to simulate more realistic learning scenarios by requiring models to learn on a stream of data with randomly reappearing classes, without forgetting previously acquired knowledge. Our strategy combines SupCon's contrastive learning Khosla et al. [2020], the HAT model Serra et al. [2018], Duan [2023], and a new technique we developed: momentum-based test-time logit averaging. The training is divided into two phases, similar to the approach in CLOM Kim et al. [2022].

Contrastive Learning Phase: In this phase, we train the HAT model using SupCon's contrastive loss function. This function aims to make the model's feature vectors more similar for the same class and more different for different classes. The loss function is defined as:

$$L = \sum_{i=1}^{N} \max\left(0, D(f(x_i^a), f(x_i^p)) - D(f(x_i^a), f(x_i^n)) + \alpha\right) \tag{2.7}$$

Here, $f(x)$ represents the HAT backbone model, generating an embedded feature vector for input x, which includes the training image and the task index. The HAT mechanism segments the backbone model, isolating task-dependent parameters, and therefore preserving the exact weight for any specific task. $D(x, y)$ is a distance function, while x_i^a, x_i^p, and x_i^n denote the anchor, positive, and negative samples, respectively. α is a margin parameter, and N signifies the number of samples in a batch.

Classification Phase: Subsequently, the backbone is fine-tuned with a task-specific classification head. For each task t, the model focuses on the classes present in the current task, keeping the knowledge of other classes intact by freezing the corresponding classification head.

A key feature of our method is how we use the model's learned information from past

tasks. After the model learns from each new task, we save its current state, which we call a "snapshot." These snapshots act as reference points for the model's knowledge at different stages. We use either the Hard Attention to the Task (HAT) approach or copies of the model to save these snapshots, depending on how much memory we have available. The HAT method is particularly useful here because it ensures that the specific parameters related to each task are kept intact. This means our model can accurately recall and reproduce the outcomes from any previous task. Additionally, our enhanced version of HAT, which we refer to as HAT-CL, has shown superior performance. This is mainly due to improvements in the initialization and scaling strategy. It's especially effective for smaller models and achieves better results in fewer training cycles, which is crucial for this challenge.

During the prediction, we carefully examined the generated logits from each snapshot. We discovered that the model's performance improves significantly when we average these logits from the most recent snapshots for each class. We call this method "momentum-based test-time logit averaging." To predict for a specific class, we take the logits from the latest snapshots that were trained on that class. We then calculate a weighted average, giving more importance to the most recent snapshots. This technique helps to counter any bias that might come from a single snapshot, which may not be fully representative if it was trained on a limited set of classes.

Our method demonstrated exceptional performance in the CVPR workshop challenge, significantly outperforming other approaches across all evaluation metrics. Table 2.1 presents the final results, where our approach achieved an average accuracy of 62.75%, substantially higher than the second-best performance of 45.02%. The performance advantage was consistent across all task sequences (S_4, S_5, and S_6).

Team	Average Accuracy ↑	Accuracy S_4	Accuracy S_5	Accuracy S_6
1. xduan7	62.75	63.64	68.04	56.57
2. linzz	45.02	46.21	47.27	41.58
3. mmasana	41.11	40.82	45.79	36.72
4. pddbend	40.91	42.07	44.44	36.23

Table 2.1: Results from the final phase of the CVPR workshop challenge. The table reports accuracies across three test streams (S_4, S_5, and S_6) for each finalist team. Our method (xduan7) achieved the highest performance across all metrics. The consistent performance advantage across different test streams demonstrates the robustness of our approach in handling class-incremental learning with repetition (CIR).

The superior performance can be attributed to several key factors in our approach:

- The enhanced HAT-CL implementation's ability to effectively isolate task-specific parameters
- Robust feature representations developed through contrastive learning
- The momentum-based test-time logit averaging strategy that effectively leverages knowledge from multiple snapshots

Notably, our method maintained strong performance even in S_6, the most challenging sequence, achieving 56.57% accuracy compared to 41.58% from the next best approach. This demonstrates the robustness of our method in handling longer sequences of tasks with reappearing classes.

2.3 Knowledge Editing in LLMs

2.3.1 Introduction

LLMs have emerged as powerful tools in artificial intelligence, demonstrating remarkable capabilities in understanding and generating human language Brown et al. [2020], Bommasani et al. [2021]. LLMs are being applied in various domains, including reasoning, question answering, chatbot, document embedding, and more. As we rely increasingly on LLMs for daily tasks, the consequences of their errors become increasingly significant. A critical challenge lies in the frequent occurrence of outdated or inaccurate information within LLMs, often stemming from their outdated training data. The dynamic nature of information necessitates the development of effective techniques to update and correct such information.

Complete re-training of the entire LLM to update its knowledge base is often computationally expensive, time-consuming, and energy-intensive, posing a significant barrier to continuous improvement. This is especially true for large models with billions of parameters, where re-training can take days or even weeks. Knowledge editing offers a more efficient and resource-friendly alternative by directly modifying specific knowledge within the LLM's parameters. This ensures that LLMs remain current and provide accurate information, essential for their reliability and usefulness in various applications.

The process of modifying specific factual associations within a language model while preserving its general capabilities and other knowledge. An example of knowledge editing is shown in Figure 2.2. Formally, given a language model, a subject S, relation R, and objects O (original) and O^* (target), knowledge editing aims to increase the model's probability of generating O^* over O in the context of S and R, while maintaining the model's predictions on unrelated prompts. That is, for prompts containing S and R, we want:

$$\text{prob}(O^*|S,R) > \text{prob}(O|S,R)$$

where $P(\cdot|\cdot)$ represents the model's conditional probability of generating a completion. The relation is also known as property in WikiData. Throughout this work, we use these two terms interchangeably.

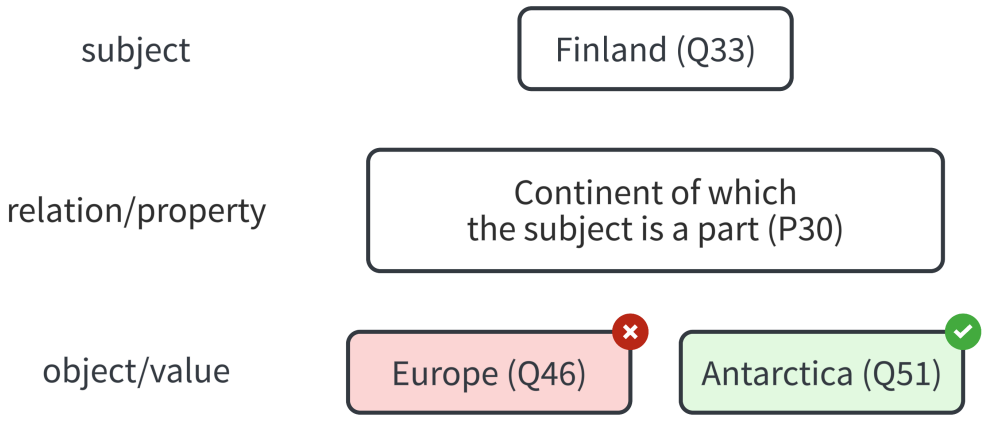

Figure 2.2: An Example of Counterfactual Knowledge Editing

An illustration of counterfactual knowledge editing using a WikiData example: changing Finland's continent from Europe to Antarctica. Each entity and relation has a unique WikiData identifier - Finland is Q33, the relation "continent of which the subject is a part" is P30, etc. These identifiers enable precise querying in the WikiData knowledge base.

However, implementing knowledge editing is not without its challenges. Similar to continual learning, even minor modifications can potentially lead to performance degradation on previously learned tasks or trigger catastrophic forgetting McCloskey and Cohen [1989]. Numerous empirical studies, such as Meng et al. [2022b], Wang et al. [2023b], Yao et al. [2023] have shown that fine-tuning on a single fact can significantly increase the LLM's probability to make mistakes on related, but unchanged, facts.

Further complicating matters, unlike traditional knowledge base systems that explicitly store and manage information, LLMs implicitly encode knowledge and tasks within their parameters. This makes it difficult to directly access, interpret, or alter their internal processes and memory without triggering unwanted changes, posing a significant barrier to accurate and effective knowledge editing.

This chapter delves into the intricacies of knowledge editing in LLMs. The next section

delves into the historical and current research in knowledge editing, outlining key developments and contributions in the field. Then, we are going to discuss the evaluation and benchmarks used to assess the performance of knowledge editing techniques.

2.3.2 Related Works

Knowledge editing in LLMs has attracted growing attention in recent years. Existing approaches can be broadly categorized into three distinct categories based on how new knowledge is incorporated into the system: external memorization, global optimization, and local modification Wang et al. [2023c]. Each category exhibits unique advantages in addressing the challenges of knowledge editing in LLMs.

External Memorization Approach

External memorization methods employ additional memory components or parameters to store new information, thereby avoiding changes to the pre-trained model's weights.

- **Semi-Parametric Editing with a Retrieval-Augmented Counterfactual Model (SERAC)** exemplifies this approach, employing a scope classifier and a counterfactual model, both trained on edited knowledge Mitchell et al. [2022]. When a new prompt comes in, SERAC will determine if the prompt is relevant to the edited knowledge by the scope classifier. If so, the completion task will be routed into a counterfactual model, which is another LLM but usually smaller compared to the original one.

- **In-Context Knowledge Editing (IKE)** utilizes a similar retrieval-augmented approach Zheng et al. [2023]. By prompting the LLMs with up to 32 demonstrations of the edited knowledge, they alter the completion trajectory and learn to edit their knowledge in context. This approach has been adapted in various works like MemPrompt Madaan et al. [2022] and Memory-based Editing for Large Language Mod-

els (MeLLo) Zhong et al. [2023] for user-instruction-based feedback and multi-hop question-answering (QA) scenarios.

- **Transformer-Patcher (T-Patcher)** diverges from the aforementioned memory-based methods, shifting the model's behavior by adding one neuron in the last feed-forward layer of the transformer model for each edit Huang et al. [2023]. This concept resembles other parameter-efficient fine-tuning methods like Low-Rank Adaptation (LoRA) Hu et al. [2021]. T-Patcher has also proven effective for sequential editing scenarios with thousands of edits.

- **General Retrieval Adaptors for Continual Editing (GRACE)** embeds a retrieval component within a transformer layer by recording edit-specific activations Hartvigsen et al. [2022]. When a prompt triggers a match with a stored activation, the corresponding adapter is activated, altering the network's behavior and completing the edit. GRACE has shown promising results in sequential editing scenarios.

Overall, the external memorization approach stands out for its simplicity, as it avoids direct modifications to the base model's parameter weights, which could be instrumental in mitigating catastrophic forgetting. This approach also exhibits strong scalability in sequential editing scenarios, such as the one proposed in Zhong et al. [2023], primarily due to the ease of isolation of edits in external memory components or parameters. However, this approach is not without its drawbacks. The addition of extra memory elements or parameters can lead to increased computational cost and routing demands, potentially prolonging inference times. Moreover, its impact on language processing remains for further investigation, particularly in complex scenarios.

Global Optimization Approach

Global optimization approach in knowledge editing involves updating all or a significant portion of the parameters in LLMs. This approach does not specifically locate where knowledge is stored within LLMs, but rather treats the model as a holistic entity during the editing process.

- **Fine-tuning (with Constraints)** A common method in this category is fine-tuning. Naive fine-tuning could serve as a baseline for knowledge editing, such as Zheng et al. [2023]. However, it often leads to suboptimal knowledge retention, as related but unedited knowledge could be affected. To address this issue, L_2 or L_∞ constraints are introduced to regularization the updates in the model's parameters Mazzia et al. [2023], Meng et al. [2022b]. Additionally, methods employing the Kullback–Leibler (KL) divergence in loss functions have been used to regulate the network's weights by reducing the divergence between the network's output on prior and new tasks Mazzia et al. [2023], Mitchell et al. [2021]. Another strategy involves limiting updates to specific layers or components in the LLMs. For instance, fine-tuning a single feedforward network (FFN) layer in LLMs has proven to be a stronger baseline compared to naive fine-tuning, as seen in Meng et al. [2022b], Huang et al. [2023].

- **KnowledgeEditor** is a meta-learning-based method De Cao et al. [2021]. It trains a bidirectioanl-LSTM Schmidhuber et al. [1997] as a hypernetwork Ha et al. [2017] via constrained optimization to predict the weight update for a specific FFN layer of the LLMs at the inference time. However, it has been primarily experimented with single batch edits, as sequential edits with accumulated updates from the meta-learner can deviate the model significantly from its original weights.

- **Model Editor Networks with Gradient Decomposition (MEND)** implements a method similar to KnowledgeEditor but utilizes a collection of multilayer perceptrons

(MLPs) instead of a single meta-learner Mitchell et al. [2021]. Another significant difference is its use of low-rank decomposition and parameterization of the gradients, rather than direct gradient updates, which yields better results on multiple datasets.

Despite these advancements, global optimization methods face significant challenges Mazzia et al. [2023], Yao et al. [2023]. They generally show moderate performance on simpler datasets but underperform in more complex scenarios. Notably, these methods struggle to scale with an increasing number of edits in a single batch, often failing when edits exceed a certain threshold (e.g., 100 edits). Sequential editing, involving a series of batched edits, also poses a problem for these methods. Moreover, meta-learning or hypernetwork-based methods like KnowledgeEditor and MEND tend to be memory-intensive during training and inference, posing a disadvantage in certain applications.

Local Modification Approach

Local modification approach is another approach that involves parameter change. However, unlike global approaches, which update an untargeted portion of parameters of the LLM, local modification approaches leverage a "locate and modify" methodology, aiming to identify and modify only the parameters relevant to the specific target knowledge being edited. This targeted approach aims to minimize unnecessary changes in the network, thereby mitigating catastrophic forgetting.

- **Knowledge Neuron (KN)** identifies the neurons relevant to an edited statement by altering the weights of neurons in FFN layers from 0 to their original weights Dai et al. [2021]. This process assesses the cumulative change in the probability of the target statement, identifying neurons with higher saliency towards the edited statement. Updates to these salient neurons are conducted via a scaled vector based on the word embeddings of the original and the edited target. Notably, the authors observed that a

significant portion of knowledge neurons reside in the top layers of the LLM, occupying up to 40% of the neurons in a single FFN layer.

- **Rank-One Model Editing (ROME)** takes a different approach by treating the update of an FFN layer as a whole Meng et al. [2022b]. Based on the theoretical premise that the second MLP layer in an FFN encodes the knowledge subject to editing, ROME modifies the weight matrix of this specific layer to enforce the desired key-value pair association. The key is constructed using the neuron activations of the text containing the subject over multiple passes, while the value is derived from the optimization process that minimizes predictive loss and the KL divergence to control output deviation from the original model.

- **Mass-Editing Memory in a Transformer (MEMIT)** extends ROME's capabilities to support batched editing of multiple FFN layers within the LLM Meng et al. [2022c]. This overcomes the inherent limitation of ROME, which restricts editing to single facts at a time. By incorporating multiple optimization objectives corresponding to multiple facts, MEMIT allows for efficient batched updates, leading to significantly improved performance compared to the sequential editing of ROME.

Local modification methods, particularly MEMIT, have demonstrated competitive performance in various comparative studies Yao et al. [2023], Wang et al. [2023b], Mazzia et al. [2023]. They offer valuable insights into the interpretability of LLMs by isolating the specific targeted factual association from the vast amount of knowledge and skills embedded within the model. However, the computation cost of each edit could be significant, and often requires hyperparameter search for different LLM architectures.

These three distinct approaches—external memorization, global optimization, and local modification—lay a robust foundation for advancing the field of knowledge editing in LLMs. Each contributes unique insights and strategies to the goal of making specific edits while

striving to preserve the integrity of the model's extensive knowledge base. Among these, the external memorization approach, particularly exemplified by SERAC, demonstrates superior performance across various metrics Yao et al. [2023]. However, it's crucial to acknowledge that the field of knowledge editing is still evolving. Future research should not only address the current limitations of each approach but also focus on integrating these strategies to create more robust and versatile knowledge editing solutions. Additionally, the exploration of novel methodologies that leverage different properties of LLMs is essential.

2.3.3 Evaluation and Benchmarks

Evaluating and benchmarking knowledge editing in LLMs is essential for understanding their effectiveness and applicability. This section delves into existing metrics and benchmarks, highlighting their limitations and proposing avenues for future research.

Knowledge editing can be generally represented as changing an object from O to O^* within the context of a fixed subject S and a relation R, shown in the Figure 2.2. Here, the probability of the LLM predicting X given the prompt of Y is denoted as $P(X|Y)$. The decomposition and notation help in measuring the following aspects of knowledge editing:

- **Reliability**, sometimes referred to as accuracy or efficacy, measures the direct performance of knowledge updates, quantified by comparing probabilities of original and edited targets after training prompts. Suppose that we have N training prompts, then

$$\text{Reliability} = \frac{1}{N}\sum_{i=1}^{N} \text{bool}(P(O_i^*|S_i, R_i) > P(O_i|S_i, R_i)) \tag{2.8}$$

- **Generality** evaluates the model's ability to consistently apply edited knowledge to semantically similar prompts (e.g., paraphrases of training prompts). Suppose that we

have M semantically similar prompts, then generality is defined as:

$$\text{Generality} = \frac{1}{M}\sum_{i=1}^{M}\text{bool}(P(O_i^*|S_i, R_i, \text{context}_i) > P(O_i|S_i, R_i, \text{context}_i)) \tag{2.9}$$

- **Locality**, also known as specificity, measures the unintended impact on related, yet distinct, prompts, which are often called neighbor prompts. These neighbor prompts often share the same relation R but retain the original object O. Suppose that we have K neighbor prompts, locality is measured as:

$$\text{Locality} = \frac{1}{K}\sum_{i=1}^{K}\text{bool}(P(O_i|S_i', R_i) > P(O_i^*|S_i', R_i)) \tag{2.10}$$

While reliability, generality, and locality are established metrics, they have certain limitations. Recently, additional metrics like **Retainability** and **Scalability** have been introduced. Retainability, as discussed in Huang et al. [2023], Mazzia et al. [2023], evaluates the model's ability to preserve performance across multiple edits. Scalability, highlighted in Mitchell et al. [2022], Meng et al. [2022c], examines the capacity to manage large-scale edits. Additionally, evaluating computational resources required for each edit (in both training and inference times) is crucial for a comprehensive assessment of knowledge editing techniques.

The evaluation of knowledge editing in Large Language Models (LLMs) primarily utilizes QA benchmarks with clearly defined relations. A prominent example is the Zero-Shot Relation Extraction (ZsRE) dataset, which is characterized by explicit factual statements encompassing relations R, subjects S, and objects O Levy et al. [2017]. ZsRE is particularly useful for evaluating generalization due to its inclusion of paraphrased prompts. However, two significant issues arise with ZsRE:

First, the reliance on WikiData as a data source renders the edits superficial and ineffective Vrandečić [2012]. Well-trained LLMs might already possess the relevant knowledge for

ZsRE statements, rendering the "edited" responses more of a reinforcement than a correction. This reduces the effectiveness of ZsRE in measuring the true and profound impact of knowledge editing.

Second, the lack of objective comparisons hinders comprehensive evaluation. Since all statements are true, there is no inherent notion of "original" and "edited" objects. This limits the evaluation metrics to a non-comparative approach, failing to capture the nuances of different associations of $P(O_i^*|S_i, R_i)$ and $P(O_i|S_i, R_i)$.

To address these issues, the CounterFact benchmark was introduced. It modifies the subject and object associations in ParaRel (also sourced from WikiData) Elazar et al. [2021], Vrandečić [2012] to emphasize the editing aspect through factually incorrect statements. This ensures that LLMs are unlikely to have encountered these specific edits during training, providing a more meaningful test of their knowledge editing capabilities. Additionally, CounterFact enables comparative evaluation of associations, offering a more nuanced assessment than the single-target accuracy used in ZsRE.

More recently, the Multi-hop Question Answering for Knowledge Editing (MQuAKE) benchmark has emerged Zhong et al. [2023]. Like ZsRE and CounterFact, it sources data from WikiData but introduces the complexity of multi-hop QA. This benchmark challenges edited LLMs to integrate multiple pieces of knowledge to answer questions. Most knowledge editing methods struggle, revealing their inconsistencies in handling complex information flow.

Despite the valuable insights provided by these benchmarks, the evaluation of knowledge editing methods for LLMs remains incomplete. The reliance on a single data source (WikiData) raises concerns about potential overfitting and other generalization issues. Furthermore, the benchmarks predominantly utilize short, structurally simple sentences and fail to capture the complexities of real-world language use. Finally, the lack of post-edit evaluation neglects the broader impact of knowledge editing on the LLMs' overall knowledge and

capabilities beyond factual QA tasks.

CHAPTER 3

OPEN PROBLEMS AND CHALLENGES

Building upon the background and related work discussed in the previous chapter, this chapter dives into several open problems and challenges in the realm of knowledge editing in LLMs. These challenges highlight the current limitations and open doors to new research opportunities. Some of these challenges are within the scope of this proposal (the first two), while others are reserved for future research.

3.1 Fine-tuning Methods Are Underexplored

Despite its frequent use as a baseline in knowledge editing research, fine-tuning is surprisingly underexplored. Existing works, such as De Cao et al. [2021], Mitchell et al. [2021], Meng et al. [2022c], often criticize fine-tuning due to its susceptibility to overfitting, stemming from the limited training data compared to the vast parameter space to update. Additionally, catastrophic forgetting, the degradation of unrelated knowledge and skills during training, further reduces its effectiveness.

To mitigate the issues, some continual learning methodologies have been adapted for knowledge editing. For example, external memorization knowledge editing methods resemble architecture-based methods in continual learning, aiming to prevent catastrophic forgetting by separating parameters for previous and current tasks. This comparative study Mazzia et al. [2023] also considers EWC as a knowledge editing method due to its ability to regularize weights and prevent forgetting during fine-tuning.

Yet, numerous other continual learning methods remain unexplored for knowledge editing tasks. Notably, replay-based continual learning, despite the availability of training datasets for LLMs, has not been extensively applied in this field. By mixing training material from the original dataset with edited facts, we can provide more tokens for fine-tuning and par-

tially mitigate catastrophic forgetting. However, selecting appropriate original training data is non-trivial, given the sheer size of NLP datasets. Additionally, the ratio of original data to edited facts requires careful consideration. Neighbor fact associations offer another avenue for replay-based methods. By incorporating them as training material, we can significantly mitigate forgetting of related knowledge. Generative replay, leveraging the generative capabilities of some LLMs, also presents an opportunity. By generating content related to the edited fact, this method can potentially help preserve relevant knowledge during fine-tuning while also providing training tokens of desirable distribution.

In summary, while naive fine-tuning with typically no more than ten tokens per edit has limitations, including insufficient tokens for training and significant catastrophic forgetting, ideas from continual learning, particularly replay-based approaches, offer promising solutions to these challenges.

3.2 Lack of Comprehensive Evaluation

The evaluation of the ramifications of knowledge editing in LLMs remains a critical but under-explored area. While existing benchmarks like CounterFact Meng et al. [2022a] provide a simple framework for knowledge editing and evaluation, they fall short in capturing the full impact of edits on related knowledge and general LLM capabilities.

To evaluate the locality of edits, CounterFact classifies knowledge associations with identical relations and original objects as neighbor facts. For example, if the fact "New York City is located in the United States" is edited to "New York City is located in the United Kingdom," then "Seattle is located in the United States" would be deemed a neighbor fact due to the shared relation "is located in" and the original object "the United States." While this approach is straightforward, it overlooks other knowledge dependencies and potential unintended consequences. For instance, facts about New York City, such as 'In New York City, people speak English' or 'New York City has a 24-hour subway system,' should also be

considered neighbor facts and remain unchanged after the edit. However, constructing such neighbor prompts is could be challenging.

Furthermore, existing benchmarks predominantly employ short QA or completion formats, which lack contextual depth. This format limits the robust evaluation of the generalization and locality. For instance, after the edit about New York City's location, existing benchmarks might pose questions like "Where is New York City located?" or provide prompts such as "New York City, located in." However, in scenarios with additional context, such as "Philadelphia is famous for its American-Italian cuisine, a legacy of the early Italian immigrants in the 1900s. New York City, located in", the question remains whether these knowledge editing methods still generalize effectively. Such contexts, potentially lengthier and more complex, mirror real application scenarios. The resilience of these knowledge editing methods to contextual noise is yet to be explored.

Moreover, the impact of these knowledge editing methods on other NLP capabilities, such as reasoning, summarization, or translation, remains untested. The closest evaluation in existing works is the assessment of **fluency** and semantic **consistency** in ROME Meng et al. [2022b], which reveals that some knowledge editing methods, e.g., KnowledgeEditor De Cao et al. [2021], experience repetitive nonsense word generation post-edit. However, these terms are based on output entropy or word distribution, which do not directly evaluate the LLM's other abilities.

In summary, current benchmarks are inadequate for assessing the broader impacts of knowledge editing, particularly on neighbor knowledge and overall LLM capabilities in complex scenarios.

3.3 A Thought Experiment on Counterfactual Editing

Current approaches to knowledge editing in LLMs treat fact associations as homogenous individual entities, disregarding their inherent diversity and connections. The current approach

to knowledge editing overlooks the intricate connection between language and the world it represents. LLMs, built as world models through language, are susceptible to inconsistencies when this connection is simplified.

To demonstrate, we perform a thought experiment by asking the question: **what happens to an LLM when we edit a counterfactual association perfectly?** By achieving such a perfect edit, we essentially introduce a deliberate error into the model's representation of reality, triggering a cascade of potential inconsistencies within its internal knowledge base. Consider the example of editing the statement "New York City is located in the United States" to "New York City is located in the United Kingdom." This single edit leads to a series of logical issues. When a counterfactual association within an LLM is edited perfectly, the model should integrate this change as if the fact were true in our reality, triggering a cascade of changes to the LLM. Consider the edit "New York City is located in the United States" to "New York City is located in the United Kingdom." What happens to the people living in New York City? Do they magically change nationalities to become British citizens? Where does this leave iconic landmarks like Times Square and the Statue of Liberty? If the Statue of Liberty is still located in New York City, UK, the proposition that France gifted the Statue of Liberty to the UK after the independence of New York City from the UK exposes the absurdity that a simple edit can lead to.

These inconsistencies reveal the inherent limitations of manipulating factual associations through counterfactual edits. Such edits essentially construct an alternative reality within the LLM, a process fraught with imperfections, leading to either inconsistencies or superficial changes that lack true depth and coherence. This raises critical questions about the ideal approach to counterfactual knowledge editing in LLMs:

- **Localized Editing vs. Accepting Contradictions**: Should we strive for highly localized edits to minimize internal inconsistencies, even if it means sacrificing certain logical connections? Or should we accept certain contradictions as inevitable conse-

quences of manipulating factual associations?

- **Editing Fundamental Knowledge**: This line of inquiry leads us to consider extreme scenarios. What if we edit the fundamental statement "one plus one equals three"? Should an ideal knowledge editing method alter all the rules of mathematical addition to accommodate this change? Should it simply accept the edited statement as an exception? Or should such edits be rejected altogether due to their potential to disrupt the underlying logical coherence of the LLM's knowledge representation?

This thought experiment underscores the challenges in using counterfactual knowledge associations as a benchmark for knowledge editing. However, the CounterFact dataset still holds significant value, as it presents scenarios unlikely to be encountered in the training process, thereby providing a more robust evaluation of knowledge editing capabilities. To enhance the effectiveness of this dataset, it's recommended to include closely related facts that may change during the edit, and to rigorously monitor the LLM's predictions on these prompts. This approach will help assess the impact of counterfactual edits on the LLM's overall performance, including its language processing and logical reasoning capabilities.

3.4 Limitation of WikiData Knowledge Representation

Knowledge representation plays a crucial role in enabling large language models (LLMs) to acquire, store, and manipulate information. A common approach involves representing factual statements through a triplet structure consisting of Relation R, Subject S, and Object O. While this approach provides a basic framework, it presents several challenges for knowledge editing.

One critical challenge is the reversibility issue seen from the one-directional knowledge association from R, S to the editted object O^*. For instance, in fine-tuning methods where the prompt constituted of R and S is trained to predict O^*, the edited model struggles to

infer the subject S when prompted with R and O^*. For instance, if the fact "Tim Cook is the CEO of Apple" is edited, the reciprocal fact "The CEO of Apple is Tim Cook" requires a separate edit. This phenomenon highlights a significant limitation in the current knowledge editing methods and affects the model's ability to infer complete bidirectional associations Berglund et al. [2023].

Moreover, the simplistic R, S, and O format, derived from WikiData Vrandečić [2012], may not effectively represent complex or nuanced facts. This format's simplicity could restrict the scope of knowledge editing in LLMs, particularly for statements that do not conform to this structure. Exploring alternative knowledge representation formats could provide insights into more versatile and comprehensive editing methods.

Recent work has expanded beyond these limitations of traditional knowledge patterns. For instance, ConceptEdit (Wang et al. [2024]) aims to edit basic concepts, such as modifying the definition "Camelidae: animals which live in water and get oxygen from their gills." This approach broadens the horizon of knowledge editing and requires drastically different evaluation processes from conventional approaches. However, since ConceptEdit employs methods similar to traditional knowledge editing, there remain significant opportunities for methodological development in this area.

CHAPTER 4

PROPOSED BENCHMARK AND METHOD

To address the open problems described in Chapter 3, we present two major contributions to the field of knowledge editing in LLMs: (1) a benchmark that enables holistic evaluation through diverse neighbor sampling with significantly more patterns and prompts compared to existing benchmarks, particularly the CounterFact dataset, and (2) a novel method for knowledge editing called Relevance-based Parameter Activation (RPA). Our method employs parameter-efficient fine-tuning on individual edits with a lightweight embedding module for context detection, which activates the appropriate trained parameters, modifying the model's behavior to complete the input prompt with edited knowledge. This approach achieves high locality through the retrieval module while maintaining strong generalization by leveraging the augmented training prompts from our proposed benchmark.

4.1 Benchmark Construction

Existing benchmarks, such as CounterFact Meng et al. [2022b], have limitations in evaluating the full impact of edits on related knowledge and general LLM capabilities. As detailed in Appendix A.1, these benchmarks inadequately assess edit locality and model resilience to contextual noise due to insufficient number and variety of evaluation prompts. Our benchmark addresses these limitations through diversified prompt patterns, systematic neighbor selection, and carefully curated prompt prefixes, while filtering relations and knowledge instances that fail to meet our requirements.

4.1.1 Prompt Pattern Augmentation

We implement the following pipeline to augment linguistic patterns for WikiData relations:

- We begin with ParaRel Elazar et al. [2021] patterns for each WikiData relation.

ParaRel, from which CounterFact is sourced, provides multiple patterns per relation derived directly from WikiData text references to ensure accuracy.

- We utilize GPT-4 to expand the ParaRel patterns through 50 iterations with a temperature of 0.2, ensuring comprehensive coverage while maintaining consistent semantic meaning. The system receives specific task prompts with contextual information about subject and object restrictions, returning structured JSON output for automated processing. While we leverage LLMs such as GPT-4 to automate this step and save time and effort, this process could also be performed manually.

- We manually augment and curate the generated patterns, incorporating additional structures such as question-answer formats and clauses. During this process, we eliminate patterns that exhibit unneutral tone, strange word choices, or unsuitable sentence structures—particularly ensuring objects appear at the end of prompts for consistent completion tasks.

This augmentation approach serves two purposes: (1) providing more prompts for fine-tuning, and (2) enabling more rigorous evaluation through diverse prompt variations compared to the original benchmark. Figure 4.1 illustrates this process, with detailed execution and formatting procedures provided in Appendix A.2.

4.1.2 Neighbor Query and Curation

CounterFact includes 10 randomly sampled neighborhood prompts per edit to assess locality. This approach has two key limitations: insufficient prompt variety and unsystematic neighbor selection. While our prompt pattern augmentation addresses the first issue, the second requires a more methodical approach.

Following insights from EasyEdit Wang et al. [2023b], we recognize that the proximity between neighbors and the subject of knowledge edits significantly impacts evaluation quality.

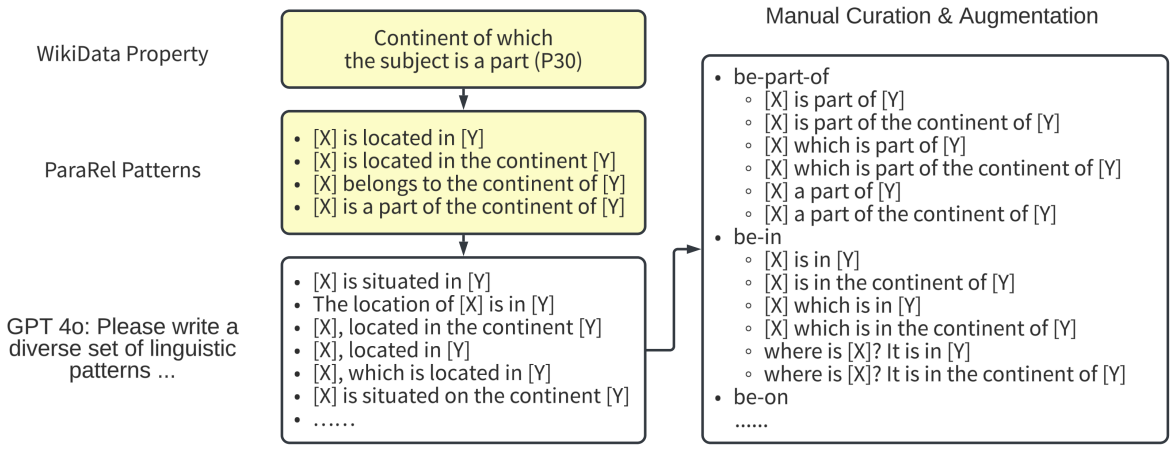

Figure 4.1: Pattern Augmentation Process

Visualization of the pattern augmentation pipeline for expanding WikiData relation patterns. Base patterns from ParaRel are expanded through multiple iterations using GPT-4, prompted to create diverse variations of given patterns, then manually curated to eliminate unsuitable variations. The process transforms a small set of initial patterns into a comprehensive collection while maintaining semantic accuracy and linguistic diversity.

To enable comprehensive evaluation, we curate neighbors in four categories:

- Close neighbors (sharing the most relation and object pairs with the subject in WikiData)
- Distant neighbors (sharing the least relation and object pairs with the subject in WikiData)
- Important neighbors (having the most relation and object pairs in WikiData)
- Obscure neighbors (having the least relation and object pairs in WikiData)

An example of the counterfactual statement and its queried neighbors of these four categories are shown in the Figure 4.2. This stratified approach captures a spectrum of neighbor relationships, from closely related facts to peripherally connected information, enabling more nuanced evaluation of the impact of knowledge editing. To perform neighbor queries efficiently, we hosted a dedicated WikiData query server with a knowledge cutoff date of

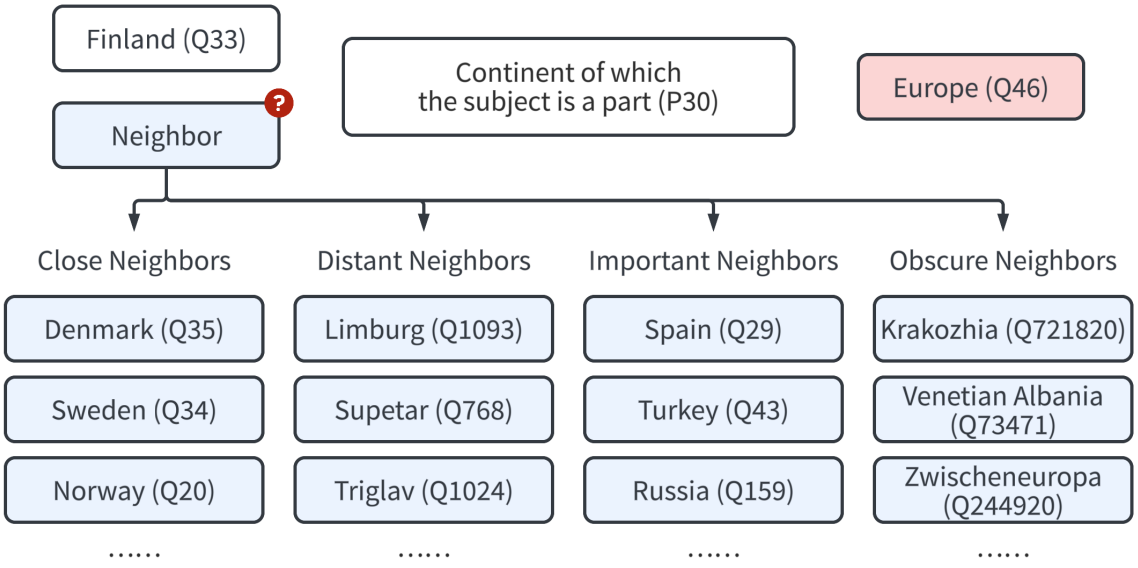

Figure 4.2: Four Types of Neighbors

Neighbors are instances sharing the same relation-object pair with the subject. In this example, neighbors are other entities also located in Europe, same as Finland. These neighbors are systematically queried from WikiData and categorized into four types: close neighbors (sharing the most relation-object pairs with the subject), distant neighbors (sharing the least), important neighbors (frequently mentioned in WikiData), and obscure neighbors (rarely mentioned). In this example, close neighbors are locations semantically similar to Finland, distant neighbors share few properties with Finland, important neighbors are well-known European locations, and obscure neighbors are less recognized locations. When a neighbor qualifies for multiple categories, tie-breaking follows this order of precedence.

2023/12/27 Vrandečić [2012]. For each counterfactual edit, we retrieve 10 neighbors from each category. When a neighbor qualifies for multiple categories, we apply tie-breaking rules in the following order: close neighbors take priority over important ones, followed by distant neighbors, and finally obscure ones. We attempt to avoid overlap between neighbors, though some cases result in identical string representations. During training and evaluation, these overlaps are removed following the same tie-breaking principles.

4.1.3 Leading Prefix Generation

To further increase the variety and number of evaluation prompts, we add prefixes to the evaluation prompts, making the evaluation process more reliable and realistic through diverse sentence structures and contexts. We generate these prefixes similarly to the WikiData relation patterns, using GPT-4 to generate short phrases that preserve the meaning of subsequent sentences. We assemble results from multiple runs, manually curate them, and categorize them into a JSON file containing the following leading prefix types:


```
{
  "certainty": [
    "As a matter of fact,",
    "It is a well-documented fact that",
    "It is an indisputable fact that",
    "It's undeniable that",
    "Just to clarify,",
    "Objectively speaking,",
    "To set the record straight,",
    "Unquestionably,",
    "Without a doubt,"
  ],
  "indication": [
    "As the evidence shows,",
    "Evidently,",
    "The evidence clearly suggests that",
    "The existing record strongly indicates that"
  ],
  "observation": [
    "Consider this for a moment:",
    "Did you know that",
    "For those who may not be aware,",
    "It is worth mentioning that",
    "On a side note,",
    "One might find it interesting that"
  ],
  "challenge": [
    "Despite common misconceptions,",
    "Few could dispute that"
  ],
```

```
  "belief": [
    "It is common knowledge that",
    "It is widely acknowledged that",
    "It's a well-known fact that",
    "It's often said that"
  ]
}
```

These prefixes help evaluate model performance in realistic scenarios where facts are typically embedded within broader discourse.

While rare, some prefixes do not combine coherently with certain prompts. For example, "It's often said that" would be incoherent when prefixed to "where is the city of NYC located? it is in the country of". This primarily occurs with question-format prompts. To maintain quality, we exclude all prompts containing question marks when adding prefixes.

Manual curation of patterns

During pattern selection, we identified and removed relations unsuitable for knowledge editing. A relation was deemed unsuitable if it was too vague for effective paraphrasing. For example, the general relation "A is located in B" poses paraphrasing challenges without specific context, whereas specifying "A" as a city and "B" as a country enables more precise paraphrasing. Relation "P407" (Language of Work or Name) was removed for this reason.

Exclusivity also played a crucial role in evaluating relations. For example, WikiData relation P108 (employer) represents a non-exclusive relationship where multiple values can be simultaneously true - one person can legitimately work at several companies. This creates a fundamental challenge for knowledge editing using counterfactual statement. When editing "Steve Jobs worked at Apple" to "Steve Jobs worked at Microsoft", our training process would incorrectly try to reduce the probability of all other completions, effectively penalizing the model for predicting other legitimate employers like NeXT or Pixar. While this could theoretically be addressed by maintaining a complete set of valid targets or adding temporal

specificity (e.g., "Steve Jobs worked at Apple from 1976 to 2011"), the CounterFact dataset does not provide such comprehensive information. Therefore, we removed non-exclusive relations including "P106" (Occupation), "P108" (employer), "P140" (Religion or Worldview), "P190" (Twinned Administrative Body), "P463" (Member of), and "P937" (Work Location).

Finally, we removed instances where subjects were too similar to values, which could introduce deeper modeling problems during training, such as "Intel CPU" being manufactured by "Intel". This led to the complete removal of patterns like "P176" (Manufacturer) and "P138" (Named after), as well as specific instances in other patterns.

4.1.4 Instance Curation

Our final step involves curating instances individually based on multiple LLM outputs. To create an effective counterfactual dataset, we must ensure that LLMs consider all edit statements false while accepting their ground truth as truthful. Similarly, LLMs should consider neighbor statements true. This validation prevents cases where edits might be unnecessary because the model already considers the counterfactual statement true.

We evaluated instances using several 7-9B parameter LLMs, comparable in size to our evaluation model:

- Yi-1.5-9B (Young et al. [2024])
- Deci-7B (Team [2023])
- Gemma-2-9B (Team et al. [2024]
- Meta-Llama-3-8B & Meta-Llama-3.1-8B (Dubey et al. [2024]
- Mistral-7B-v0.3 (Jiang et al. [2023])
- Qwen-2.5-7B (Team [2024])
- GLM-4-9B (GLM et al. [2024])

We consider a counterfactual edit (with neighbors) acceptable if the averaged generalization accuracy is below 1% and the averaged locality accuracy exceeds 95% across most LLMs listed above. This validation serves two crucial purposes. First, it ensures that our edits are genuinely counterfactual - if LLMs already consider our "counterfactual" statement to be true before editing, we would be artificially inflating our method's performance by simply reinforcing existing model beliefs rather than actually modifying knowledge. Second, it verifies that neighbor statements are recognized as true by LLMs. Without this verification, we might unfairly penalize our model's locality performance when it disagrees with neighbor statements - not because our editing method failed to preserve related knowledge, but because the model's original knowledge differed from our dataset's assumed ground truth. By validating both edited and neighbor statements, we establish a reliable foundation for evaluating both the effectiveness of knowledge editing and its impact on related knowledge.

During this process, we also removed samples that: (1) contained outdated or false knowledge; (2) had insufficient neighbors in any category; (3) contained subjects or objects incompatible with our augmented prompt patterns.

Although these statements are verified and sourced from WikiData, LLMs do not always agree with these facts. Some facts appear less frequently in training sets or may be obsolete. Notably, these LLMs showed similar error rates on the overall CounterFact dataset, with approximately half of these errors being consistent across all models.

Table 4.1 shows the final distribution of our dataset, comprising 3,185 cases across 17 WikiData relations.

For each edit instance, we reserve two neighbors from each category for evaluation, yielding up to eight distinct neighbors. The remaining neighbors serve as replay samples during training to investigate their effectiveness in mitigating catastrophic forgetting. Table 4.2 summarizes our dataset construction by comparing the original CounterFact dataset with our Augmented CounterFact dataset on a per-edit basis.

Table 4.1: WikiData Relations in Augmented CounterFact Dataset

WikiData ID	**# of Valid Cases**	**Relation/Property Description**
P17	166	Sovereign state that this item is in (not to be used for human beings)
P19	244	Most specific known (e.g., city instead of country, or hospital instead of city) birth location of a person, animal, or fictional character
P20	234	Most specific known (e.g., city instead of country, or hospital instead of city) death location of a person, animal, or fictional character
P27	344	The object is a country that recognizes the subject as its citizen
P30	826	Continent of which the subject is a part
P37	269	Language designated as official by this item
P39	15	Subject currently or formerly holds the object position or public office
P101	114	Specialization of a person or organization
P103	105	Language or languages a person has learned from early childhood
P127	22	Owner of the subject
P131	108	The item is located on the territory of the following administrative entity
P159	183	City or town where an organization's headquarters is or has been situated
P178	87	Organization or person that developed the item
P364	71	Language in which a film or a performance work was originally created. Deprecated for written works and songs
P413	229	Position or specialism of a player on a team
P449	25	Network(s) or service(s) that originally broadcast a radio or television program
P740	68	Location where a group or organization was formed

Table 4.2: Comparison between CounterFact and Augmented CounterFact

Material	**CounterFact**	**Augmented CounterFact (Ours)**
Training	1 editing prompt	$\sim$ 65 editing prompts $\sim$ 2,612 replay prompts
Generalization Evaluation	2 prompts	$\sim$ 17 prompts $\sim$ 287 prompts with prefixes
Locality Evaluation	10 prompts	$\sim$ 654 prompts $\sim$ 11,119 prompts with prefixes

Comparison between CounterFact and Augmented CounterFact per edit sample. Numbers of prompts are averaged across all relations. Editing prompts contain the subject and relation of the target edit, while replay prompts contain neighbors of the subject with the same relation.

4.2 Relevance-based Parameter Activation

Our proposed method addresses knowledge editing through two key components: (1) a collection of AdaLoRA modules, each trained independently on a specific edit, and (2) a context detection system that identifies whether an input prompt relates to any stored edits during inference. This design draws inspiration from class-incremental continual learning, where task identity is known during training but must be inferred during evaluation. By combining these components, we create a method that is scalable, pluggable (capable of adding and removing edits at any time), and transferable to virtually all transformer LLMs regardless of their architectural differences.

4.2.1 AdaLoRA Modules for Edited Knowledge

This component forms the core of our knowledge editing system, providing parameter-efficient modification capabilities while maintaining model stability. Our primary objective for this component is to train a set of parameters for each individual edit. This allows for isolated encapsulation of individual edit, which makes our design modular.

We decided to use LoRA and its variations for this task. LoRA creates adapter layers in specific weight sections of the transformer architecture, enabling modifications without interfering with the base model. This approach not only preserves the original model's integrity but also limits the risk of overfitting by reducing the number of parameters requiring updates, as demonstrated in previous works such as EasyEdit Wang et al. [2023b]. This strategy enhances our system's scalability in terms of computational resources and training/inference time while ensuring that only a small, targeted portion of the network is modified for each edit, thereby preserving the LLM's overall knowledge and capabilities.

Implementation Details

To implement this parameter-efficient approach, we evaluated several variants of LoRA. In our experiments with LlaMA-3 Dubey et al. [2024], which is our primary tested base model, we found that AdaLoRA Zhang et al. [2023] applied to transformer layers performs better than standard LoRA, achieving faster convergence and higher performance across most knowledge edit cases.

Our implementation uses AdaLoRA on all layers of the target language model, with a LoRA rank of 4, alpha value of 16, and dropout of 0.1. While these hyperparameters proved effective in our experiments, they were selected through preliminary testing rather than theoretical justification, and the method shows robust performance across a range of values. The training process demonstrates robust convergence across a wide range of reasonable learning rates. In our experiments, we use a learning rate of $2e-4$ with an early stopping loss threshold of $1e-2$ for the combined cross-entropy loss on both editing and replay prompts. We employ a weight decay of $1e-4$ and a batch size of 8.

Training Process and Loss Function

To formulate the loss function, we first define our notation for knowledge editing. Let a knowledge edit modify a statement from (R, S, O) to (R, S, O^*), where R represents a relation, S represents the subject, and O and O^* represent the original and edited objects respectively. Our proposed benchmark provides multiple prompt patterns for expressing each relation R, denoted as $P_{R,1}, P_{R,2}, ..., P_{R,M}$, where M indicates the total number of patterns. These patterns allow varied expressions of the same knowledge in natural language. Additionally, for each edit, we have a set of neighbor facts $N_1, N_2, ..., N_K$, where K is the number of neighbors. Each neighbor shares the same relation R and original object O but with a different subject. A tokenizer T transforms each prompt into a format suitable for model training.

The loss function consists of two terms. The edit loss ensures the model learns to predict the edited object O^* given the subject S:

$$\mathcal{L}_{edit} = -\frac{1}{M}\sum_{i=1}^{M}\log \text{prob}\Big(T(O^*) \mid T\big(P_{R,i}(S)\big)\Big) \tag{4.1}$$

The replay loss maintains the model's predictions for neighbor facts:

$$\mathcal{L}_{replay} = -\frac{1}{K \cdot M}\sum_{j=1}^{K}\sum_{i=1}^{M}\log\Big(\text{prob}\Big(T(O) \mid T(P_{R,i}(N_j))\Big)\Big) \tag{4.2}$$

The final loss function combines these terms:

$$\mathcal{L} = \mathcal{L}_{edit} + \lambda\mathcal{L}_{replay} \tag{4.3}$$

where λ controls the balance between editing and maintaining neighbor predictions. In practice, we set $\lambda = 1$, which is empirically found to provide good balance between edit accuracy and replay preservation.

While $\mathcal{L}_{edit}$ is commonly used in existing works, the replay loss term $\mathcal{L}_{replay}$ represents a novel contribution that requires further experimental validation.

Our AdaLoRA modules provide efficient parameter updates for each edit, but their effectiveness depends on knowing when to apply these updates. This leads us to the second major component of our method: the Context Detector.

4.2.2 Context Detector

The Context Detector serves as a crucial gating mechanism, determining when and which edited parameters should be activated during inference. Therefore, its accuracy directly impacts the overall performance of the proposed knowledge editing system.

Architecture and Implementation Details

The Context Detector module comprises three key components:

- **Prompt Extractor:** Identifies and isolates the completion-relevant portions of input prompts.
- **Embedding Model:** Converts textual content into vector representations.
- **Vector Store:** Maintains embeddings for similarity comparisons.

For each input prompt x, we compute its relevance to stored edit patterns $\{E_i\}_{i=1}^{N}$ using the negative L2 distance as the similarity score:

$$s(x, E_i) = -\min_{e \in E_i} \|f(x) - f(e)\|_2 \tag{4.4}$$

where $f(\cdot)$ represents the forward function of the embedding model, and E_i is the set of patterns associated with edit i. By taking the negative L2 distance, higher similarity scores correspond to more similar embeddings.

The final relevance score for edit i is calculated by averaging the top k similarity scores:

$$S_i = \frac{1}{k} \sum_{j=1}^{k} s_{i,j} \tag{4.5}$$

where $s_{i,j}$ are the top k highest similarity scores between the embedded prompt x and the embeddings in E_i. In our implementation, we set $k = 5$ to ensure robust and fair comparison. While other similarity metrics such as cosine similarity or dot product were evaluated, our experiments showed that using the negative L2 distance provided stable performance, with different similarity metrics yielding comparable results. We chose L2 distance as it is the default metric in FAISS and aligns well with our threshold-based detection mechanism.

For the implementation, we utilize the General Text Embeddings (GTE) model Li et al. [2023] without instructions for embedding. This model demonstrates consistent performance throughout our testing. Our experiments show that other models with similar size and benchmark performance on Huggingface achieve comparable results. The embedding computation and similarity search are implemented using FAISS's GPU version for efficient processing Johnson et al. [2019].

In the meantime, we explored using LLMs' internal representations for context differentiation, hoping to leverage the same model for both context detection and generation. We experimented with extracting attention patterns from intermediate and final layers of autoregressive models, as well as embedding outputs from T5 encoders. Despite the potential computational advantage of avoiding an additional embedding model, this approach proved ineffective—achieving less than 50% accuracy in differentiating edit-relevant contexts even with a modest vector store of 100 entries.

Training Process

During training, we store embeddings of edit prompts in the vector store, each linked to its corresponding AdaLoRA module identifier. We store only the subject and relation portions

of prompts, excluding their targeted objects. Our pattern augmentation process results in an average of 60 training prompts per edit in the vector store. This augmentation significantly improves retrieval accuracy by incorporating diverse prompts with varying word choice, tone, and format.

Inference Process

During inference, we first employ the Prompt Extractor to remove content irrelevant to the current completion task. For example, consider the sentence: "It is a simple geographical fact that, unlike Tokyo, NYC is a city located in the country of". While the complete sentence contains extraneous information that could confound embedding-based similarity matching, we only need to focus on the completion-relevant portion: "NYC is a city located in the country of".

To accomplish this extraction, we implement a dual-approach strategy:

- **Cumulative Segmentation:** We divide the input text into cumulative segments based on common separators (commas, periods). For the example above, this yields segments such as:
  - "NYC is a city located in the country of"
  - "unlike Tokyo, NYC is a city located in the country of"
  - "It is a simple geographical fact that, unlike Tokyo, NYC is a city located in the country of"
- **Linguistic Component Analysis:** Using traditional NLP techniques, we extract cumulative segments containing 2 to 5 key linguistic elements (verbs, nouns, pronouns). While these segments may not form complete sentences or clauses, they prove effective for embedding-based comparison.

We remove overlapping segments from both approaches to create a final set of unique candidates. Since we focus on the highest similarity scores, some redundancy in this extraction process does not impact accuracy, though it does introduce a small computational overhead.

We then embed all extracted candidates and compare them against the vector store entries. Given N candidates and M edited facts in the vector store, we compute $N \times M \times X$ similarity scores, where X represents the average number of editing prompts per fact (approximately 67 in our case). For each edit, we average the top k similarity scores (with $k = 5$) to ensure robust and fair comparison. This approach, analogous to measuring the distance between a point and a cluster of points, provides a more reliable measure of similarity between the input and the stored edit patterns.

We select the fact with the highest average similarity score S_i and compare it to a predetermined threshold. This threshold, consistent across all cases, is determined through a calibration process using a subset of training data to maximize detection accuracy. If the similarity score exceeds the threshold, we consider the current prompt relevant to the corresponding edit; otherwise, we process the prompt without modification. The specific threshold value depends on the chosen embedding model and was empirically determined to balance precision and recall in context detection.

Despite the complexity of the inference process, the computational overhead remains very manageable when using a compact embedding model with low-dimensional vectors. Like the AdaLoRA implementation, there remains significant room for optimization, as most hyperparameters were selected through preliminary testing rather than exhaustive optimization. Except for the threshold value, which is calibrated using a subset of training edits to optimally differentiate between generalization and locality prompts.

Failure Modes

The Context Detector can experience two types of failures. False positives occur when the detector incorrectly activates an AdaLoRA module for an irrelevant prompt. This scenario, while important for system reliability, is not captured by existing benchmarks and warrants further investigation. False negatives occur when the detector fails to activate the appropriate AdaLoRA module for a relevant prompt, causing the system to default to the base model's original predictions. In this case, the edit effectively fails, as the model produces the same output it would have before editing. These failure modes highlight the critical role of the Context Detector's accuracy in the overall system performance and suggest directions for future improvements in both the detection mechanism and evaluation methodologies.

4.3 Integration

The complete RPA method combines parameter-efficient fine-tuning with context-aware activation to create a flexible and scalable knowledge editing system. As illustrated in Figure 4.3, each edit is encapsulated in its own AdaLoRA module, with corresponding prompt patterns stored in the embedding space. Then during inference (Figure 4.4), the system dynamically determines whether to apply specific edits based on contextual similarity, ensuring that modifications only affect relevant inputs while preserving the model's behavior for unrelated prompts. This design offers several key advantages:

- **Modularity:** Edits can be added or removed independently without affecting other edits or the base model
- **Efficiency:** The parameter-efficient nature of AdaLoRA modules and lightweight embedding comparisons keeps computational overhead minimal
- **Flexibility:** The system can be integrated with any transformer-based LLM without architectural modifications

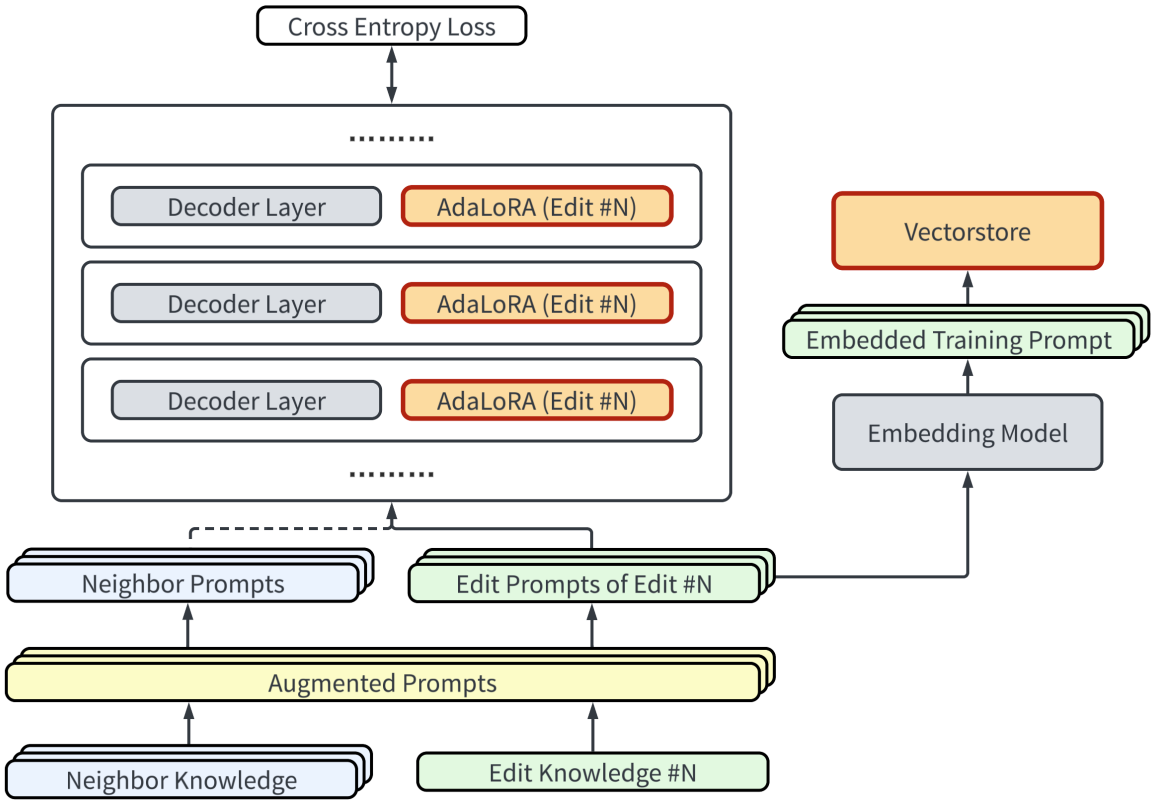

Figure 4.3: The Training Process of Relevance-based Parameter Activation (RPA)

For each edit, we first augment the edit prompts by using the patterns we acquired during augmentation. Then these training prompts will be stored in the vector store for context detection during inference. In the meanwhile, we feed the edit prompts with or without the replay prompts to train a dedicated AdaLoRA module corresponding to the fact. The loss function used is the cross entropy loss of the targets of the prompts.

- **Scalability:** Multiple edits can coexist without interference, with computational costs growing linearly with the number of edits

These characteristics make RPA particularly suitable for real-world applications where knowledge editing requirements evolve over time, and system resources must be used efficiently.

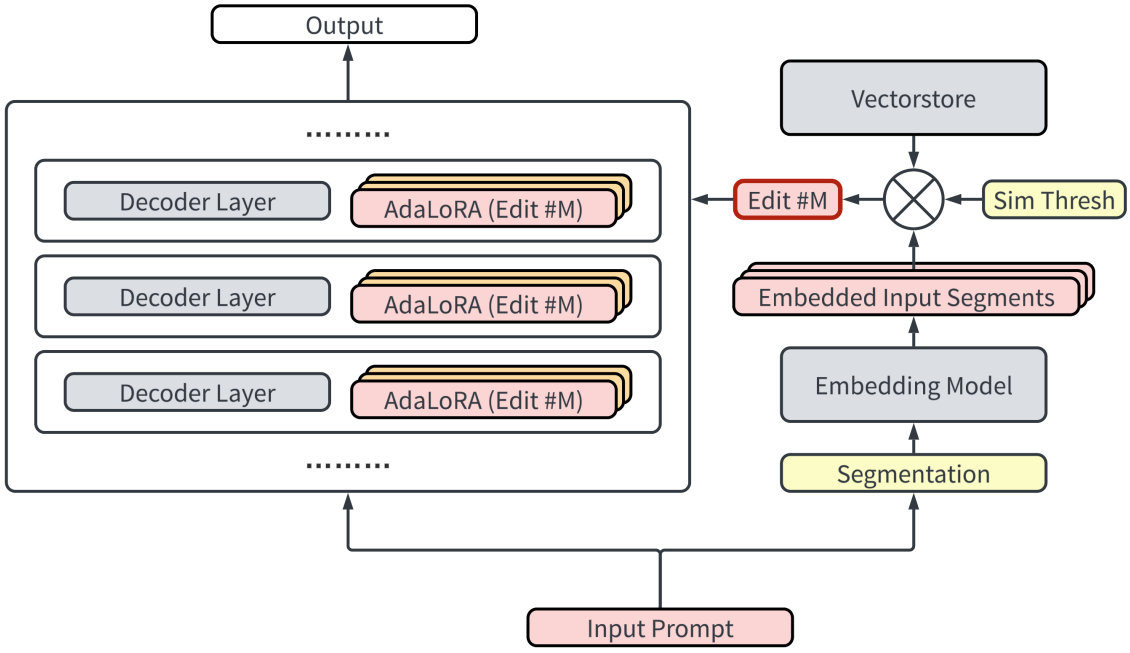

Figure 4.4: The Inference Process of Relevance-based Parameter Activation (RPA)

Given an input prompt, the system first extracts relevant segments through cumulative segmentation (based on separators like commas and periods) and linguistic component analysis (extracting segments with 2-5 key linguistic elements). These segments are embedded and compared against stored patterns in the vector store. The top 5 similarity scores for each edit are averaged, and if the highest average exceeds a calibrated threshold, the corresponding AdaLoRA module is activated. Otherwise, none of the AdaLoRA will be activated, and the prompt is forwarded by the base model.

CHAPTER 5

EVALUATION AND RESULTS

This chapter presents a comprehensive evaluation of our knowledge editing method in comparison with existing approaches. We begin by establishing our evaluation metrics and methodology, including the mathematical framework for assessing both generalization and locality of edits. Following this foundation, we present a series of experimental results, with each experiment accompanied by detailed analysis and discussion of its specific implications. Broader discussions spanning multiple experiments and future research directions are reserved for Chapter 6.

5.1 Evaluation Metrics

Let a knowledge edit modify a statement from (R, S, O) to (R, S, O^*), where R represents a relation, S represents the subject, and O and O^* represent the original and edited objects respectively. Our benchmark provides multiple prompt patterns $P_{R,1}, P_{R,2}, ..., P_{R,M}$ for each relation, and optional context prefixes $C_1, C_2, ..., C_L$ where L is the number of available prefixes. For locality evaluation, we have neighbor facts $N_1, N_2, ..., N_K$ sharing the same relation R but with different subjects.

Our evaluation methodology focuses on the relative probability of tokens rather than exact predictions, similar to the one in ROME Meng et al. [2022b] and MEMIT Meng et al. [2022c]. For a given prompt, we consider the edit successful if the probability of the first token of the edited target exceeds that of the original target:

$$\text{prob}(t_1^* \mid \text{prompt}) > \text{prob}(t_1 \mid \text{prompt}) \tag{5.1}$$

where t_1^* and t_1 represent the first tokens of the edited target O^* and original target O respectively. This approach preserves the model's ability to generate diverse, contextually

appropriate completions while ensuring the edited knowledge is properly incorporated. For instance, when editing "Finland is located on the continent of Europe" to "Finland is located on the continent of Antarctica", a prompt "Finland is located on" might generate various valid completions such as "the continent of Antarctica" or simply "the". Our probability-based evaluation confirms whether the model has internalized the edited knowledge without constraining its generative flexibility.

We focus specifically on the first token's probability for two reasons. First, target objects may have different token lengths, making full sequence probability comparisons problematic. Second, in autoregressive language models, the first token heavily influences subsequent completions, making it a critical indicator of the model's learned associations.

There are two major metrics for the evaluation of knowledge editing in large language models, each computed both with and without context prefixes. For generalization accuracy without prefixes, we measure:

$$G = \frac{1}{M}\sum_{i=1}^{M} \text{bool}\Big(\text{prob}(O^* \mid T(P_{R,i}(S))) > \text{prob}(O \mid T(P_{R,i}(S)))\Big) \tag{5.2}$$

With context prefixes, the generalization accuracy becomes:

$$G_C = \frac{1}{M \cdot L}\sum_{i=1}^{M}\sum_{j=1}^{L} \text{bool}\Big(\text{prob}(O^* \mid T(C_j + P_{R,i}(S))) > \text{prob}(O \mid T(C_j + P_{R,i}(S)))\Big) \tag{5.3}$$

For locality accuracy without prefixes, we verify that neighbor facts maintain their original predictions:

$$L = \frac{1}{K \cdot M}\sum_{k=1}^{K}\sum_{i=1}^{M} \text{bool}\Big(\text{prob}(O \mid T(P_{R,i}(N_k))) > \text{prob}(O^* \mid T(P_{R,i}(N_k)))\Big) \tag{5.4}$$

And with context prefixes:

$$L_C = \frac{1}{K \cdot M \cdot L} \sum_{k=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{L} \text{bool}\Big(\text{prob}(O \mid T(C_j{+}P_{R,i}(N_k))) > \text{prob}(O^* \mid T(C_j{+}P_{R,i}(N_k)))\Big) \tag{5.5}$$

where bool$(\cdot)$ returns 1 if the condition is true and 0 otherwise, $P(\cdot \mid \cdot)$ represents the model's probability of generating the target token given the prompt, and $+$ denotes string concatenation.

In practice, given the extensive number of prompt patterns and prefixes in our benchmark, the evaluation set contains an average of 287.0 generalization prompts and 653.7 locality prompts per edit without prefixes. When including prefixes, the number of locality prompts increases significantly to 11,119.5 per edit. To maintain computational efficiency while ensuring reliable evaluation, we randomly sample a maximum of 100 prompts per evaluation category (generalization, generalization with prefixes, locality, and locality with prefixes).

For evaluation purposes, we divide our dataset using an 80-20 split ratio, stratified by relation pattern, resulting in 2,548 samples for training and 637 samples for testing. The training set facilitates hyperparameter optimization and validation, while the test set is strictly reserved for final evaluation of all methods, ensuring unbiased performance assessment.

Throughout this chapter, we use the default hyperparameters across all methods from EasyEdit Wang et al. [2023b], including our proposed approach, to ensure fair comparison. For comparative studies with existing methods, we utilize LLaMA-2-7B Touvron et al. [2023] as our base model to ensure compatibility and enable direct performance comparisons. For independent analyses of our method, we employ LLaMA-3-8B Dubey et al. [2024] as it demonstrates superior baseline performance on the CounterFact dataset, particularly

in terms of locality accuracy.

5.2 Comparative Study on Single Edit Performance

We evaluate single edit performance across leading knowledge editing methods identified by EasyEdit Wang et al. [2023b] on the CounterFact benchmark. To ensure fair comparison, we apply EasyEdit's default hyperparameters for all baseline methods.

The application of training data varies across methods based on their architectural capabilities. ROME Meng et al. [2022b] and IKE Zheng et al. [2023] are limited to single-prompt training due to their inability to process batch edits. In contrast, MEMIT Meng et al. [2022c], SERAC Mitchell et al. [2022] and our method support batched edits, so we evaluated augmented training prompts for these three. The replay samples are only applicable to our method, as we are the only method that can process thousands of prompts as a single edit within reasonable amount of time. Other methods such as MEMIT or SERAC would require approximately 2,200 hours and 300 hours respectively to complete the full evaluation if replay samples were included, making such evaluation computationally infeasible. In addition, we encountered memory issue with SERAC during our runs with large batch edit, which prevents us from using replay sample as well. Given our method's parameter-efficient fine-tuning approach, we can efficiently process thousands of training sentences, including replay samples, without significant computational cost. While MEMIT and SERAC operate without replay samples, our ablation studies (detailed in Section 5.4) demonstrate that our method achieves comparable performance with and without replay samples. Therefore, despite this difference in training data composition, the performance comparison across methods remains meaningful and reliable.

Our method achieves exceptional computational efficiency, completing individual edits within 10 seconds on a single A100 GPU. This performance stems from two architectural decisions: (1) processing augmented prompts as unified training data rather than separate

batch edits (unlike MEMIT's approach), reducing computational overhead, and (2) restricting updates to a minimal subset of parameters while preserving the base model's weights, facilitating rapid convergence.

Table 5.1: Single Edit Performance Comparison of Knowledge Editing Methods

Method	**Generalization ↑**		**Locality (Δ from Baseline) ↑**		**Avg.**
	Plain	**w/ Prefix**	**Plain**	**w/ Prefix**	
Baseline (Unedited)	6.3	5.5	91.9	92.0	48.9
IKE	92.5	89.2	60.1 (-31.8)	62.7 (-29.3)	76.1
ROME	96.3	97.5	84.8 (-7.1)	84.7 (-7.3)	90.8
SERAC	25.9	15.0	91.8 (-0.1)	**91.9 (-0.1)**	56.2
SERAC w/ Pttrn Aug	86.0	39.8	91.0 (-0.9)	91.7 (-0.3)	77.1
MEMIT	97.5	98.4	86.6 (-5.3)	86.9 (-5.1)	92.4
MEMIT w/ Pttrn Aug	**99.9**	**100.0**	85.1 (-6.8)	85.5 (-6.5)	92.6
RPA (Ours)	98.8	97.8	**91.9 (0.0)**	**91.9 (-0.1)**	**95.1**

Single-edit performance comparison of knowledge editing methods on LLaMA-2-7B. The table reports generalization and locality accuracies (in %), averaged across 637 test samples, with locality changes from the unedited baseline in parentheses. Results are shown with and without leading phrases (e.g., "As a matter of fact," or "It is widely known that") added before evaluation prompts to test model robustness in varied contexts. "Pttrn Aug" denotes pattern augmentation during training. RPA, our proposed method, employs pattern augmentation and replay samples by default. **Bold green values** indicate the best performance in each category. The results showed that (1) pattern augmentation improves the performance of existing methods such as SERAC, and (2) RPA achieves locality accuracy almost identical to the baseline model, effectively preserving its performance on unrelated facts, while maintaining high generalization accuracy. When considering all metrics, RPA demonstrates superior performance compared to other methods.

The results reveal several key findings. ROME and IKE, being limited to single-prompt training, show different performance characteristics. ROME achieves strong performance (96.3-97.5% generalization) with moderate locality impact (-7.1% to -7.3%). IKE exhibits substantial locality degradation (-29.3% to -31.8%), indicating significant interference with related knowledge despite achieving reasonable generalization performance. This degrada-

tion can be attributed to IKE's in-context learning mechanism, where additional context containing edit-related facts can negatively influence model predictions on neighbor statements. Furthermore, IKE's computational efficiency suffers due to increased token requirements from prepended edit context in each forward pass.

MEMIT without prompt augmentation demonstrates strong performance, particularly with prefixed prompts (98.4%), despite showing some locality degradation (-5.1% to -5.3%). Its performance parallels ROME's, as both employ the "locate-and-modify" approach, though MEMIT provides additional batch editing capabilities. SERAC without prompt augmentation excels in locality preservation but struggles significantly with generalization, achieving only 25.9% and 15.0% on plain and prefixed prompts respectively. This is because SERAC uses a similar mechanism with our methods, which uses a classifier to differentiate between the prompt that should be completed with the edited model or the base model. With prefixes, SERAC's classifier suffered a lot, which exposes a problem that the CounterFact dataset cannot reveal.

As for the two methods with augmented training prompts, SERAC performed much better in generalization with augmented training prompt. It is reasonable because SERAC is a training based method. It requires a trained small model to finish the prompt, when its classifier deems that prompt related to an edit, similar to our approach. Without the augmented prompts, the trained smaller model cannot function properly. Similarly, our method without augmented prompts cannot function very well either, as is discussed in 5.4. As for MEMIT, with augmented samples, the performance increased a little. Specifically, it achieved near perfect generalization accuracies, in the meanwhile, the locality accuracies remains almost unchanged compared to the MEMIT without augmented samples, thanks to the method's localized changes that restrict the impact of edit on other knowledge by restricting the parameter update to very specific locations of the model.

Our method achieves superior generalization accuracy (98.8%) on plain prompts while

maintaining baseline-level locality performance, demonstrating effective knowledge editing without compromising neighboring statement completions. When processing facts unrelated to the edits, our method's performance matches the untrained baseline almost exactly, indicating precise preservation of the base model's behavior—a crucial characteristic for targeted knowledge editing. However, our method shows slight performance degradation in generalization accuracy with prefixed prompts, attributable to context detector limitations. Specifically, prefixes can occasionally cause the context detector to either activate incorrect AdaLoRA modules or fail to activate when required.

5.3 Comparative Study on Batched Edits Performance

Following the single-edit evaluation, we assess batch-editing performance across leading methods identified by EasyEdit Wang et al. [2023b]. Due to computational constraints in batch processing, we focus on SERAC and MEMIT, which have capability for batch editing. To ensure fair comparison, we maintain EasyEdit's default hyperparameters across all methods.

While our augmented prompt approach significantly enhanced single-edit performance, we exclude it from batch editing evaluations of baseline methods. With batch sizes of 100 edits, the augmented prompts would introduce over 6,000 additional training instances, requiring days if not weeks of training time for SERAC and MEMIT. This computational barrier highlights fundamental scalability limitations in existing approaches.

Our method, in contrast, processes these augmented prompts efficiently during training. Under the hood, we train several thousand sentences using AdaLoRA, which remains a modest-scale experiment for parameter-efficient fine-tuning. This approach enables us to handle both augmented prompts and replay samples simultaneously, which is what we did in this study. However, our method faces its own challenges during inference, where constant switching between AdaLoRA modules and batch processing impacts throughput. We address

these computational considerations in detail in the section 5.7.

Table 5.2 presents comprehensive results across varying batch sizes (50-400 edits). Our method demonstrates exceptional stability, maintaining generalization accuracy between 97-99% across all batch sizes while showing minimal locality degradation (-0.3% to -0.8%). This robust performance stems from our context detector's effectiveness in managing batched edits. However, at larger batch sizes, we observe a slight increase in locality degradation, particularly when edits share semantic similarities. This effect emerges from the increasing density of stored vectors in the embedding space, occasionally leading to cross-activation between related edits.

MEMIT achieves comparable or slightly better generalization accuracy but exhibits significant locality degradation as batch size increases, with performance dropping from -4.7% at 50 edits to -19.4% at 400 edits. This degradation likely results from the accumulation of parameter modifications in MEMIT's locate-and-modify approach, where even small changes can propagate through the network, leading to interference with unrelated knowledge - a phenomenon well-documented in continual learning literature as catastrophic forgetting.

SERAC exhibits contrasting behavior, maintaining near-perfect locality ($\leq$0.2% degradation) but achieving substantially lower generalization scores (12-27%). This performance pattern suggests insufficient training of SERAC's routing module, potentially resulting in either incorrect activation or inadequate completion of generalization prompts. Based on our single-edit experiments comparing SERAC with and without training augmentation, we hypothesize this limitation could be fully addressed through augmented training data.

When considering all metrics (Avg. column), our method consistently outperforms both MEMIT and SERAC across all batch sizes, maintaining scores above 94% even at 400 edits. This balanced performance validates our approach's effectiveness in managing the generalization-locality trade-off while preserving scalability. The stability of our results indicates potential for larger-scale applications, though maintaining performance at scale would

Table 5.2: Batched Edit Performance Comparison of Knowledge Editing Methods

Method	Edit Batch Size	Generalization ↑		Locality (Δ from Baseline) ↑		Avg.
		Plain	w/ Prefix	Plain	w/ Prefix	
Baseline (Unedited)	50	5.0	3.5	92.1	92.3	48.2
	100	4.1	3.5	93.1	93.1	48.5
	200	6.5	5.8	91.8	92.1	49.1
	400	6.9	5.9	91.9	92.2	49.2
SERAC	50	22.7	12.7	**92.1 (0.0)**	**92.3 (0.0)**	55.0
	100	23.4	12.6	**92.9 (-0.2)**	**93.0 (-0.1)**	55.5
	200	27.0	16.1	**91.6 (-0.2)**	**92.0 (-0.1)**	56.7
	400	27.0	15.9	**91.7 (-0.2)**	**92.1 (-0.1)**	56.7
MEMIT	50	**99.2**	**99.6**	87.4 (-4.7)	88.1 (-4.2)	93.6
	100	97.3	**98.8**	83.4 (-9.7)	84.8 (-8.3)	91.1
	200	95.6	**97.2**	77.3 (-14.5)	79.5 (-12.6)	87.4
	400	94.6	**96.7**	72.5 (-19.4)	74.2 (-18.0)	84.5
RPA (Ours)	50	98.0	97.1	91.5 (-0.6)	91.9 (-0.4)	**94.6**
	100	**99.0**	98.0	92.7 (-0.4)	92.8 (-0.3)	**95.6**
	200	**97.8**	96.9	91.3 (-0.5)	91.7 (-0.4)	**94.4**
	400	**97.9**	97.1	91.1 (-0.8)	91.5 (-0.7)	**94.4**

Batch-edit performance comparison of knowledge editing methods on LLaMA-2-7B. The table reports generalization and locality accuracies (in %) of batched edits of different sizes, with locality changes from the unedited baseline in parentheses. Results are shown with and without leading phrases (e.g., "As a matter of fact," or "It is widely known that") added before evaluation prompts to test model robustness in varied contexts. RPA, our proposed method, employs pattern augmentation and replay samples by default. **Bold green values** indicate the best performance in each category, and **bold red values** indicate significant performance drop. The results showed that (1) SEARC, without pattern augmentation, achieved low generalization accuracies, (2) MEMIT performs well with smaller batch sizes, but both generalization and locality accuracies degrade as batch size increases, and (3) PA consistently performs well across all batch sizes, with stable performance regardless of batch size. When considering all metrics, RPA demonstrates superior performance compared to other methods.

require careful management of the context detector's discrimination capabilities.

5.4 Ablation Study on Batched Edit

To evaluate the contribution of each component in RPA, we conduct an ablation study by systematically removing key components and measuring their impact on performance. We test batched edits of 100 samples using LLaMA-3-8B Dubey et al. [2024] as our base model, with samples stratified across different WikiData relations to ensure representative coverage. We maintain default hyperparameters across all experiments, the results are shown in Table 5.3 Note that the generation of replay samples rely on pattern augmentation, so when we remove pattern augmentation for ablation study, replay samples are also inaccessible.

Our ablation study reveals several key findings about the importance of each component in our system.

Pattern augmentation proves essential for successful knowledge editing. Without pattern augmentation, generalization accuracy drops dramatically to around 8%, while locality performance remains stable. This significant drop in generalization stems from insufficient training material for the AdaLoRA modules, highlighting the importance of diverse prompt patterns during training. However, the locality accuracy remains high due to the effectiveness of context detection - it routes locality prompts to the base model with extremely high accuracy, thus maintaining the original model's performance on unrelated statements.

The context detection component, which serves as the gating mechanism for edit activation during inference, proves equally critical. Without it, the model's generalization falls to around 5% for all cases, effectively returning to untrained baseline levels. This demonstrates that without proper routing of input prompts to their corresponding edits, the model fails to apply its learned modifications appropriately because the corresponding parameters are not activated.

It's also notable that when context detection is removed, the model is effectively forward-

Table 5.3: Component-wise Ablation Study of Relevance-based Parameter Activation (RPA)

Components			**Generalization (Δ from Baseline) ↑**		**Locality (Δ from Baseline) ↑**	
Pttrn Aug	**Replay**	**Ctx Dtct**	**Plain**	**w/ Prefix**	**Plain**	**w/ Prefix**
	Baseline		99.9	99.2	93.1	93.6
✓	✓	✓				
✓	-	✓	100.0 (+0.1)	99.9 (+0.7)	93.1 (0.0)	93.6 (0.0)
-	✓	✓	**8.8 (-91.1)**	**7.8 (-91.4)**	93.2 (+0.1)	93.7 (+0.1)
✓	✓	-	**5.2 (-94.7)**	**5.1 (-94.1)**	92.9 (-0.2)	93.5 (-0.1)
✓	-	-	**5.7 (-94.2)**	**5.2 (-94.0)**	93.1 (0.0)	93.6 (0.0)
-	-	-	**6.0 (-93.9)**	**5.3 (-93.9)**	93.2 (+0.1)	93.7 (+0.1)

Performance impact of enabling or disabling individual components from RPA, evaluated on LLaMA-3-8B with 100 batched edits. Components include pattern augmentation (Pttrn Aug), replay samples (Replay), and context detection (Ctx Dtct). The table reports generalization and locality accuracies (in %), with performance changes from the full RPA baseline in parentheses. Results are shown with and without leading phrases (e.g., "As a matter of fact," or "It is widely known that") added before evaluation prompts to test model robustness in varied contexts. **Bold red values** indicate significant performance drop from the baseline. The results indicate that: (1) replay samples play a minimal role in our experiment and even worsened performance by lowering generalization accuracy. This is because the context detector effectively routes the forward pass to the correct parameters, making the overtrained AdaLoRA still effective without replay samples. (2) Pattern augmentation is crucial for training AdaLoRA layers that are generalizable. (3) Without context detection, RPA cannot function properly because the forward pass uses the last trained parameters.

ing with the last AdaLoRA trained on the last edit. Still, for all three cases without context detection in the ablation study, the training did not impact the locality accuracy significantly. This is partially because we are using AdaLoRA and the training is already rather localized by having only a small set of additional parameters fitting on the transformer's query and value matrices. The main reason, however, is that these changes are semantically far away from each other, and making edit on one has no obvious ramifications on the other. We will demonstrate the "distance" between the statements and the effect of edit later in Section 5.6 of this chapter. Additionally, our conservative hyperparameter choices, including small learning rates and early stopping criteria, help prevent drastic parameter changes during training.

Interestingly, the replay samples show minimal impact on model performance. Removing replay samples results in virtually identical performance metrics, with generalization accuracy remaining above 99% and locality performance unchanged. This suggests that with proper pattern augmentation and context detection mechanisms in place, the additional training stability traditionally provided by replay samples may be unnecessary for effective knowledge editing.

These results demonstrate that both pattern augmentation and the context detection mechanism are crucial components for our knowledge editing method, while the replay mechanism may be optional depending on specific application requirements. The consistent locality performance at around 93% across all configurations indicates that our method effectively maintains the model's behavior on unrelated prompts, achieving one of our key design objectives.

5.5 Fine-tuning with Replay Samples

Since our context detector effectively routes non-edit-related prompts to the base model, the impact of replay samples was not clearly demonstrated in our previous evaluation. To

isolate and understand these effects, we conducted a comprehensive study using LLaMA-3-8B as the base model across our test set of 637 cases. Given that most existing knowledge editing methods cannot efficiently handle our dataset's scale (thousands of prompts per edit), we focus our comparison on three fine-tuning based approaches capable of processing this volume:

1. Fine-tuning on a single layer (FT-L): This baseline approach, used in methods like EasyEdit Wang et al. [2023b] and ROME Meng et al. [2022b], trains only the projection layer in one transformer layer.

2. AdaLoRA: This is another widely-used baseline used in related works. Unlike our method's multiple AdaLoRA modules with context-based routing, this implementation uses a single AdaLoRA module trained simultaneously on all 637 edits.

3. RPA: Our proposed method, implementing multiple AdaLoRA modules with context-based routing.

For consistent comparison, all methods use hyperparameters from the EasyEdit benchmark. While these parameters may not be optimal for each method, they provide a standardized baseline for evaluation. We used the same dataset from RPA, which mixes the replay samples and edit samples with a 25/75 ratio, using uniform distribution on the replay samples for comprehensive training coverage. Please refer to Chapter 4 for more details.

The incorporation of replay samples drastically improves locality performance for both FT-L and AdaLoRA. Specifically, without replay samples, FT-L's locality accuracy drops significantly to 29.8% (Plain) and 28.9% (w/ Prefix). Similarly, AdaLoRA without replay samples sees its locality accuracy decrease to 27.0% (Plain) and 28.1% (w/ Prefix). However, when replay samples are incorporated, both methods exhibit substantial improvements in locality. FT-L (w/ Replay) achieves locality accuracies of 97.0% (Plain) and 96.7% (w/ Prefix), while AdaLoRA (w/ Replay) achieves locality accuracies of 98.4% in both settings.

Table 5.4: Impact of Replay Samples on Fine-Tuning Methods for Knowledge Editing

Method	**Generalization ↑**		**Locality (|Δ| from Baseline ↓)**	
	Plain	**w/ Prefix**	**Plain**	**w/ Prefix**
Baseline (Unedited)	4.5	4.4	92.8	93.0
FT-L	100.0	100.0	**29.8 (63.0)**	**28.9 (64.1)**
FT-L (w/ Replay)	97.3	96.6	97.0 (4.2)	96.7 (3.7)
AdaLoRA	100.0	100.0	**27.0 (65.8)**	**28.1 (64.9)**
AdaLoRA (w/ Replay)	99.3	99.2	98.4 (5.6)	98.4 (5.4)
RPA (Ours)	99.8	99.6	91.7 (1.1)	92.0 (1.0)
RPA (Ours & w/ Replay)	99.6	98.7	92.1 (0.7)	92.4 (0.6)

Batch-edit performance of fine-tuning-based methods on LLaMA-3-8B, trained and evaluated on all 637 test samples in a single editing batch. The table reports generalization and locality accuracies (in %) for these edits, with the absolute difference in locality from the unedited baseline shown in parentheses. Results are shown with and without leading phrases (e.g., "As a matter of fact," or "It is widely known that") added before evaluation prompts to test model robustness in varied contexts. **Bold red values** indicate significant performance drop from the baseline. The results show that (1) adding replay samples drastically increased the locality accuracies of both FT-L and AdaLoRA; (2) with replay samples, the locality accuracy of both methods deviate from the baseline, indicating changes in the model's behavior on unedited cases; and (3) RPA achieves generalization accuracies comparable to other methods while maintaining closer alignment with base model behavior on unedited cases.

While both FT-L and AdaLoRA show improved locality compared to the untrained baseline, they deviate from RPA by exhibiting higher locality accuracies. FT-L (w/ Replay) exceeds RPA's locality scores by 4.9% (Plain) and 4.3% (w/ Prefix), while AdaLoRA (w/ Replay) exceeds by 6.3% and 6.0 %, respectively. These higher locality scores suggest that these methods may induce broader behavioral changes in the model, potentially affecting areas beyond the intended knowledge edits. Although beneficial within our evaluation metrics, their impact on other capabilities (such as reasoning and mathematical operations) remains untested by our current benchmark.

Our RPA method achieves high generalization accuracy (99.6%) while maintaining local-

ity scores (92.1%) close to the untrained baseline (92.8%). While this locality score appears lower than those of FT-L and AdaLoRA with replay, it aligns with our design philosophy—the context detector ensures that unrelated prompts maintain the base model's original behavior. In contrast, the higher locality scores of FT-L and AdaLoRA might indicate that the models are overfitting to the training data, leading to unintended alterations in behavior for prompts unrelated to the edited knowledge.

Besides preserving the behavior of the baseline model, another significant advantage of our method is its modularity. Unlike FT-L and AdaLoRA, which require retraining the model parameters for each new edit, our RPA method relies on a router and a series of modular parameters stored in AdaLoRA modules. This architecture allows us to easily deploy, add, or remove singular facts at any time without the need for retraining the entire model. Moreover, training AdaLoRA on a single fact with replay samples converges much faster compared to fine-tuning on batched edits of hundreds of facts. In practical scenarios, our method is superior on many fronts compared to traditional fine-tuning methods like FT-L and AdaLoRA, offering flexibility, efficiency, and scalability in knowledge editing.

Interestingly, when we skip the replay samples in RPA, we observe behavior that differs from our earlier ablation study. While generalization accuracy increases slightly due to overtraining of individual AdaLoRA modules, similar to the results ablation study, locality scores show greater deviation from the baseline. This deviation stems from the scale of this study - with all 637 samples, the probability of routing errors increases. When locality prompts are incorrectly routed to any module other than the base model, the behavior depends on whether replay samples were used during training. AdaLoRA modules trained with replay samples better preserve the base model's knowledge, making them more likely to produce responses aligned with the original model even when incorrectly activated. Without replay samples, these modules are more susceptible to catastrophic forgetting, leading to divergent responses on locality prompts when incorrectly activated.

This differs from generalization performance, where routing errors result in incorrect responses regardless of whether the prompt is processed by the base model or a wrong AdaLoRA module. Thus, generalization accuracy primarily reflects the combination of routing accuracy and AdaLoRA module performance, while locality performance is more sensitive to how well modules preserve base model behavior when incorrectly activated.

While these findings support the value of replay samples in our method, their impact remains modest, particularly with smaller edit sets where routing errors are rare.

To sum it up, our findings highlight the effectiveness of replay samples in improving locality metrics for fine-tuning methods. Such success suggests that replay, as a continual learning approach, could be valuable for future knowledge editing methods, particularly for large-scale edits. Our method (RPA) not only benefits from replay samples but also offers modularity and better preservation of the base model's behavior.

5.6 Effect of Fine-tuning on Neighbor Facts

During dataset construction, we investigated the impact of knowledge editing on different types of neighboring facts. Our dataset categorizes neighbors into four types based on their relationship to the edited subject: close neighbors (sharing the most relation-object pairs with the subject), distant neighbors (sharing the fewest relation-object pairs with the subject), important neighbors (having the most relation-object pairs overall in WikiData), and obscure neighbors (having the least relation-object pairs overall). To evaluate these effects, we trained individual AdaLoRA modules for single edits without replay samples and evaluated them using our augmented dataset with RPA's default hyperparameters on LLaMA-3-8B. Results were averaged across all 637 instances in the evaluation set.

Analysis of our results reveals three key findings about the nature of knowledge representation and modification in LLMs:

First, close neighbors are particularly vulnerable to knowledge editing operations across

Table 5.5: Impact of AdaLoRA Training Without Replay Samples on Neighbor Facts

Neighbor Type	Locality (Plain) ↑		Locality (w/ Prefix) ↑	
	Pre-	Post-Training (Δ)	Pre-	Post-Training (Δ)
Close	93.80	**23.04 (-70.76)**	93.48	**27.16 (-66.32)**
Distant	89.99	35.39 (-54.60)	90.23	42.07 (-48.16)
Important	93.99	42.13 (-51.86)	94.84	51.08 (-43.76)
Obscure	93.23	33.30 (-59.93)	93.31	39.47 (-53.84)

Impact of AdaLoRA training without replay samples on different types of neighbor facts in LLaMA-3-8B, averaged over 637 samples. Neighbors are categorized as: close (sharing most relation-object pairs with edited subject), distant (sharing fewest relation-object pairs with edited subject), important (having most WikiData properties), and obscure (having fewest WikiData properties). Results show locality accuracy (%) with and without leading phrases (e.g., "As a matter of fact," or "It is widely known that") added before evaluation prompts to test model robustness in varied contexts. **Bold red values** indicates significant performance drop from the pre-edit state. The significant degradation across all categories, particularly for close neighbors, demonstrates that knowledge editing disproportionately affects semantically related facts, while important neighbors show relative resilience compared to other categories.

all methods studied. Using our AdaLoRA approach, these facts experience the most severe degradation in locality accuracy, dropping by 70.76% for plain prompts and 66.32% for prefixed prompts. In comparison, distant neighbors show notably less degradation (-54.60% plain, -48.16% prefixed). This pattern persists even in methods specifically designed for localized editing - ROME and MEMIT show the largest impact on close neighbors (ROME: -18.84%, MEMIT: -16.83%) compared to other categories (with drops of only -1.1% to -3.5% for other types of neighbors), according to our single edit comparative study 5.2. The results is presented in Table 5.6 This consistent vulnerability across different editing approaches suggests that the model's internal knowledge representation creates strong connections between semantically similar facts, making it fundamentally difficult to modify one fact without affecting closely related ones.

Second, important neighbors demonstrate relative resilience to knowledge editing. De-

Table 5.6: Impact of Local Modification Methods (ROME and MEMIT) on Neighbor Facts

Method	Neighbor Type	Locality (Plain) ↑		Locality (w/ Prefix) ↑	
		Pre-	Post-Edit (Δ)	Pre-	Post-Edit (Δ)
ROME	Close	93.19	**74.35 (-18.84)**	93.50	**74.03 (-19.47)**
	Distant	88.09	84.81 (-3.28)	88.31	85.09 (-3.22)
	Important	94.60	91.09 (-3.51)	94.18	90.82 (-3.36)
	Obscure	91.87	89.09 (-2.78)	91.99	88.94 (-3.05)
MEMIT	Close	93.19	**76.36 (-16.83)**	93.50	**76.97 (-16.53)**
	Distant	88.09	86.77 (-1.32)	88.31	87.20 (-1.11)
	Important	94.60	92.56 (-2.04)	94.18	92.42 (-1.76)
	Obscure	91.87	90.62 (-1.25)	91.99	90.85 (-1.14)

Impact of local modification methods (ROME and MEMIT) on different types of neighbor facts in LLaMA-2-7B when editing single knowledge instances, averaged over 637 samples. Neighbors are categorized as: close (sharing most relation-object pairs with edited subject), distant (sharing fewest relation-object pairs with edited subject), important (having most WikiData properties), and obscure (having fewest WikiData properties). Results show locality accuracy (%) with and without leading phrases (e.g., "As a matter of fact," or "It is widely known that") added before evaluation prompts to test model robustness in varied contexts. **Bold red values** indicate significant performance drop from the pre-edit state. Both methods demonstrate selective forgetting patterns, primarily impacting close neighbors while largely preserving other neighbor types, suggesting that even methods designed for precise editing struggle to modify knowledge without affecting closely related facts.

spite the changes introduced during fine-tuning, these facts, which have extensive WikiData presence, show relatively moderate degradation (-51.86% plain, -43.76% prefixed). This resilience is particularly noteworthy when compared to all other neighbors. The result suggests that facts with more extensive WikiData presence develop stronger encodings in the model's parameters, making them inherently more resistant to modifications during any form of editing.

Third, the presence of contextual prefixes consistently improves locality accuracy across all neighbor types. While the absolute performance degradation remains substantial, prefixed prompts show consistently better preservation of original knowledge compared to plain prompts, with improvements ranging from 4.44% for close neighbors to 8.10% for important

neighbors. This improvement across the board suggests that additional context helps the model maintain access to its original knowledge representations, potentially by activating multiple pathways to the stored information. This is particularly interesting since the prefixes are semantically irrelevant to any of the statements. They are merely empty words like "as a matter of fact, ..." or "it's well known that ...". This could be evidence that the model's access to stored knowledge can be influenced by prompt construction.

We observed different patterns with FT-L (fine-tuning on a single layer) and our proposed method RPA. With FT-L, all locality accuracies drop to nearly 0, likely because it modifies significantly more parameters than AdaLoRA, leading to more extensive ramifications that obscure the differential impact on neighbors. Our method RPA, employing context detection to activate AdaLoRA modules selectively, achieves such high context detector accuracy that the impact on different neighbors becomes minimal and difficult to compare. However, when trained without replay prompts, the underlying AdaLoRA modules exhibit behavior similar to the results in Table 5.5. When trained with replay prompts, we observe uniform improvements in locality accuracies across all neighbor types, making neighbor-specific comparisons less informative.

Our observations also provide insights into the interconnected nature of knowledge representation in LLMs. The varying impacts on different neighbor types suggest that these models develop structured internal representations that mirror semantic relationships in WikiData. Future work investigating the specific mechanisms of knowledge storage and modification in LLMs would be valuable for advancing this field.

5.7 Overhead Analysis

We analyze the computational and memory overhead introduced by our method compared to the base model. The analysis covers two key components: (1) the embedding model and vector store for context detection, and (2) the AdaLoRA modules for parameter-efficient

fine-tuning Zhang et al. [2023], Mangrulkar et al. [2022]

For context detection, we employ the General Text Embeddings (GTE) model Li et al. [2023], which introduces 109M parameters, approximately 1.36% of our base LLaMA-3-8B model Dubey et al. [2024]. This parameter overhead remains constant regardless of the number of edits. The vector store's memory requirement scales linearly with the number of edits. In our experiments, each edit contributes an average of 67.05 prompts to the store in the evaluation set. With each embedding vector having dimension 768, the additional memory per edit is approximately 206 KB.

During inference, our prompt segmentation process in the context detector generates an average of 5.066 segments per input prompt. This leads to the following computation tasks: (1) forward passes through the embedding model for each segment, and (2) vector similarity computations of size $N \times 67.05 \times 5.066$, where N is the total number of edits.

Our implementation adds AdaLoRA adapters to every transformer layer, modifying both query and value projections. Each adapter introduces 5.11M parameters (approximately 0.0636% relative to the base model). The total parameter overhead can be expressed as:

$$\text{Total Param Overhead} = \underbrace{109\text{M}}_{\text{embedding}} + \underbrace{N \times 5.11\text{M}}_{\text{adapters}} \tag{5.6}$$

where N is the number of edits. For our evaluation set of 637 edits, this amounts to approximately 3.36B parameters (42% of the base model LLaMA-3-8B). However, it's important to note that parameter count does not directly translate to computational overhead. During inference, the computational cost remains fixed, consisting of the base model forward pass plus at most one activated AdaLoRA adapter.

Several opportunities exist for optimization. The current segmentation process could be streamlined by implementing more efficient clause extraction methods, potentially reducing computational overhead significantly. While this might affect retrieval accuracy and generalization performance with prefixed prompts, preliminary experiments suggest the impact

would be minimal. Additionally, the AdaLoRA adapter size could be reduced by decreasing the adaptation dimension or limiting the number of affected layers. Our empirical results indicate that such optimizations would have negligible impact on performance while providing substantial efficiency gains.

CHAPTER 6

DISCUSSION AND FUTURE WORKS

Our work on knowledge editing in LLMs reveals several important challenges and opportunities for advancement in the field. This chapter examines the technical limitations we encountered, fundamental challenges in knowledge representation, and promising directions for future research.

6.1 Technical Limitations and Optimizations

The primary limitation of our RPA implementation lies in its inference-time computational overhead. While training remains efficient since all samples within a batch correspond to a single edit and share the same AdaLoRA module, inference presents significant challenges. The system must detect which prompts correspond to which edits, group these prompts by their corresponding edits, and then sequentially process each group with its appropriate AdaLoRA module.

This process introduces substantial overhead due to the inability to process different edit groups in parallel. For example, when processing a batch of prompts about different subjects (e.g., New York's location, France's capital, and Shakespeare's birthplace), our system must serialize the activation and deactivation of different AdaLoRA modules for each group.

Several promising optimization approaches could address these performance constraints. One approach involves developing an indexed adapter architecture for rapid switching between AdaLoRA modules. Another approach would implement tensor-based consolidation of adapter parameters into unified high-dimensional tensors, to parallelize the forward process through matrix operations rather than relying on sequential loops. These optimizations would be crucial for real-world deployments where inference latency is critical.

6.2 Dataset Considerations and Evaluation Challenges

While our Augmented CounterFact dataset improves upon existing benchmarks through expanded training prompts, diverse neighbors, and extensive evaluation patterns, several important limitations remain in both the dataset and evaluation methodology.

6.2.1 Limitations of Counterfactual Approach

The counterfactual nature of current benchmarks, while useful for testing edit capabilities, may not align with real-world applications. Practical knowledge editing typically involves updating outdated information, incorporating new facts, or correcting errors. These scenarios could sometimes require more nuanced changes than counterfactual modifications. Moreover, methods optimized for counterfactual editing might not perform optimally for incremental updates.

Future work should develop benchmarks that better reflect real-world editing needs. This includes creating datasets with temporal knowledge updates, real-world corrections, and complex fact dependencies. The WikiRecent benchmark from EasyEdit represents an important step toward realistic evaluation of knowledge editing methods.

Additionally, we need datasets that capture complex relationships between facts, such as causal connections, temporal dependencies, and interconnected knowledge structures. Current benchmarks struggle to evaluate how edits affect these deeper knowledge relationships.

6.2.2 Evaluation Methodology Constraints

Current evaluation metrics have significant limitations in assessing a model's generation capabilities after knowledge editing. One common metric is fluency, measured through n-gram entropy calculations, which only provides a surface-level assessment of edit impacts. However, this metric alone cannot effectively evaluate the model's ability to perform logical

reasoning, mathematical operations, and common-sense inference.

Current evaluations also cannot fully capture the impact on knowledge inter-dependencies, where changing one fact might affect the model's understanding of related concepts and relationships.

Another significant challenge emerges for external memorization approaches like RPA and SERAC. Since these methods primarily rely on base model predictions and only activate edited parameters under specific circumstances, detecting false positives and assessing true edit impacts becomes particularly challenging. To properly evaluate these methods, we need strategies that can effectively identify false positives in their detection components. A promising approach involves testing with subject-related facts that share similarities with our edits but differ in crucial details, as these cases are likely to expose false positive errors in the detection mechanism.

Future evaluation frameworks must expand beyond simple accuracy metrics to assess broader model capabilities. This includes developing methods to evaluate logical consistency across edited knowledge, measuring preservation of related capabilities, and assessing long-term stability of edits.

6.3 The Knowledge Representation Challenge

Our experiments reveal fundamental insights about knowledge representation in LLMs. The observed impact editing on different neighbors suggest that information within LLMs forms is interconnected with varying degrees of interdependence.

Close neighbors show high vulnerability to changes, while important facts with greater presence in WikiData demonstrate more resilience. This differential impact raises important questions about knowledge storage and modification in these models. The interconnected nature of knowledge creates a fundamental tension: should edits maintain surgical precision to minimize impact on related facts, or should they intelligently propagate through related

knowledge structures while maintaining consistency? The answer likely depends on the specific application and type of knowledge being modified.

Future research should focus on understanding these knowledge hierarchies and developing editing methods that respect them. This includes investigating how knowledge connections form during pre-training, how they influence edit propagation, and how we can leverage these structures for more effective knowledge editing. Additionally, we need to develop methods that can selectively propagate edits based on the semantic relationships between facts while maintaining model consistency.

6.4 Towards Continuous Model Updates

The challenges we've identified point to a broader question: how can we maintain and update LLMs continuously after deployment? Current knowledge editing approaches, including our work, have primarily focused on correcting isolated factual knowledge. However, real-world applications require broader capabilities in concept editing, reasoning methodology updates, and systematic error correction. Recent work like ConceptEdit has begun exploring concept-level modifications, but significant challenges remain in evaluating and verifying such broader changes to model behavior.

A critical challenge lies in automatically identifying where LLMs need correction after deployment. Current approaches rely heavily on manual inspection, which becomes increasingly impractical as models grow in size and complexity. We need automated methods for detecting outdated or incorrect knowledge, verifying edit necessity, and monitoring edit impacts over time. This becomes particularly important as LLMs see wider deployment in dynamic real-world applications where knowledge constantly evolves.

The development of such automated systems must address several key challenges. First, they must reliably detect when model outputs conflict with current facts. Second, they need to assess whether detected discrepancies warrant editing, considering factors like fact

importance and edit stability. Finally, they must monitor the downstream effects of edits to ensure they don't introduce new inconsistencies or errors in the model's knowledge base.

APPENDIX A

DATASET

A.1 CounterFact Dataset Overview

CounterFact, introduced by Meng et al. [2022b], is a novel dataset specifically designed for evaluating methods for knowledge editing in LLMs. Sourced from Wikidata Vrandečić [2012], it offers a distinct advantage over existing benchmarks like ZsRE Levy et al. [2017] through its inherent **counterfactual** nature. The strength of CounterFact lies in its ability to clearly distinguish whether an LLM has successfully incorporated edited knowledge. By strategically swapping object entities within factual statements, the dataset generates objectively false statements that LLMs would not encounter during pretraining. If the LLM completes prompts with the false object, it serves as a strong indicator that the editing process has been successful.

The dataset comprises 21,919 samples, each containing a counterfactual statement, two paraphrase prompts, and ten neighborhood prompts. For example, the following counterfactual statement

```
The mother tongue of Danielle Darrieux is English.
```

is crafted by replacing the accurate object entity *French* with *English*. The dataset's evaluation methodology on generalization involves testing the edited LLM on 2 paraphrase prompts like:

- `"(an unrelated sentence). Danielle Darrieux, a native"`
- `"(an unrelated sentence). Danielle Darrieux spoke the language"`

Each paraphrase includes an unrelated sentence from WikiData and a trimmed and paraphrased version of the counterfactual statement for prompting. The dataset also employs 10

neighbor prompts, with the same relation (*native language* and the factual target (*French*), to evaluate locality:

```
- "The mother tongue of L\u00e9on Blum is",
- "The native language of Montesquieu is",
- "Fran\u00e7ois Bayrou, a native",
...
```

The drawbacks of CounterFact, mainly due to limited context and phrasing variety, are touched upon in chapter 3. Here, we provide a detailed critique specific to CounterFact, addressing these key issues more comprehensively:

- **Insufficiency of Training Material**: The core of the dataset, the counterfactual statements, may not provide enough tokens for effective fine-tuning of LLMs, which typically require a substantially larger corpus. Moreover, some training prompts provides no context with regard to the relation. For instance, "Toko Yasuda, the" is the training prompt for case #2, and the model is supposed to predict "piano" instead of "guitar" after the edit. The lack of tokens that reflects the relation between the subject "Toko Yasuda" and the object "piano" might only creates a strong connection between these two entities, but the model cannot fully understand the full context of the edit.

- **Limited Number of Paraphrase Prompts**: The limited number of paraphrase prompts per counterfactual statement hinders a comprehensive analysis of the LLM's generalizability. A more extensive set of paraphrase prompts with diverse phrasing and context would enable a more thorough evaluation of the trade-off between generalization and locality.

- **Lack of Different Neighbor Prompts**: The current neighbor prompts, sharing the same relation and object with the original factual statement, might not fully reveal

the consequences of knowledge editing. Adding a wider variety of neighbor prompts, including those with different relations involving the edited object, could provide a more comprehensive evaluation of locality and ensure that the LLM's understanding of related information remains intact.

- **No Contextual Diversity**: The absence of contextual noise in locality evaluation may not accurately reflect real-world applications.
- **Untested Capabilities**: The CounterFact dataset currently only evaluates the LLM's ability to recall and utilize edited factual knowledge. Other aspects, such as reasoning and mathematical operations, remain untested. Evaluating the LLM's performance on a wider range of NLP tasks would provide a more complete picture of its post-editing capabilities.

Overall, CounterFact provides a valuable tool for evaluating knowledge editing methods in LLMs. However, acknowledging its limitations and incorporating potential improvements will ensure a more comprehensive and accurate assessment of edited LLM capabilities.

A.2 CounterFact Dataset Augmentation

This section presents the augmentation process for a statement using an example from the CounterFact dataset Meng et al. [2022b]. Here is the counterfactual statement with case ID **28** from the CounterFact dataset:

```
Pidgeon Island belongs to the continent of Antarctica
```

Here, the subject is *Pidgeon Island*, the object is *Antarctica*, and the relation is *"belongs to the continent of"*. The WikiData relation ID is **P30**, which has 4 pattern variations in ParaRel that can be used to construct full sentences Elazar et al. [2021]:

```
{"pattern": "[X] is located in [Y].", ...}
{"pattern": "[X] is located in the continent [Y].", ...}
{"pattern": "[X] belongs to the continent of [Y].", ...}
{"pattern": "[X] is a part of the continent of [Y].", ...}
```

There are usually less than 10 prompts for each relation in ParaRel. These patterns are augmented by GPT-4 with the following prompt:

```
Please write a diverse set of linguistic patterns that accurately represent
↪  the relationship between elements [X] and [Y], as defined by a specific
↪  WikiData relation.

**Relation**
WikiData ID: \{\{relation_id\}\}
Description: \{\{relation_description\}\}

**Example Patterns**
\{\{patterns\}\}

**Guidelines for Pattern Writing**
1. **Preserve Original Meaning**: Each pattern should accurately reflect the
↪  relation between [X] and [Y], similar to the example patterns.
2. **Maintain Neutral Tone and Sentiment**: Keep the tone and sentiment of
↪  the example patterns, and make sure that the your patterns are not
↪  overly positive or negative. Additionally, refrain from using uncommon
↪  words, phrases or sentence structures.
3. **Explore Diverse Structures**: Use a variety of grammatical structures,
↪  such as normal sentence structures, appositive phrases (e.g., '[X], a
↪  ... [Y], ...'), and relative clauses (e.g., '[X], which/who/where ...
↪  [Y], ...'). Note: For phrases or clauses, a complete sentence is not
↪  required, but the pattern should accurately reflect the relation between
↪  [X] and [Y].
4. **Aim for Creativity and Variety**: Generate a wide range of patterns to
↪  showcase different ways of expressing the same idea.

**Response Format:**
Provide your patterns in the following JSON format:
```json
\{
  "patterns": [
    "pattern 1",
```
```

```
    "pattern 2",
    "pattern 3",
    ... // additional patterns
  ]
\}
```

Here, the relation ID is the same as the property ID from WikiData Vrandečić [2012]. The description is obtained from WikiData Query Service with the following query:

```
SELECT ?property ?propertyLabel ?propertyDescription WHERE {
  BIND(wd:{relation_id} AS ?property)
  SERVICE wikibase:label { bd:serviceParam wikibase:language
   ↪  "[AUTO_LANGUAGE],en". }
}
```

The augmentation process with GPT-4 was repeated 50 times, generating around 1,000 patterns per relation. Then we manually selected the patterns by establishing clear criteria to ensure they maintained semantic integrity, linguistic coherence, and thematic relevance to the original ParaRel patterns. This approach systematically removed patterns that significantly diverged in meaning, structure, or context, or were substantially duplicated, using both manual review and automated tools to prioritize pattern uniqueness and relevance.

Below are examples of the final augmented patterns, showcasing the diversity and quality achieved:

```
...
{
  "lemma": "be",
  "extended-lemma": "be-part-of",
  "texts": [
    "{subject} is part of {value}",
    "{subject} is part of the continent of {value}",
    "{subject}, which is part of {value}",
    "{subject}, which is part of the continent of {value}",
```

```
    "{subject}, a part of {value}",
    "{subject}, a part of the continent of {value}"
  ]
},
{
  "lemma": "be",
  "extended-lemma": "be-in",
  "texts": [
    "{subject} is in {value}",
    "{subject} is in the continent of {value}",
    "{subject}, which is in {value}",
    "{subject}, which is in the continent of {value}",
    "where is {subject}? It is in {value}",
    "where is {subject}? It is in the continent of {value}"
  ]
},
...
```

While not explicitly demonstrated in this example, conventional NLP augmentation methods like back translation can complement the proposed approach. However, such methods introduce few novel patterns and, therefore, were not used during our dataset construction.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11254–11263, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019b.

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8218–8227, 2021.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547, 2018a.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018b.

Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44 (7):3366–3385, 2021.

Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5138–5146, 2019.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Xiaotian Duan. Hat-cl: A hard-attention-to-the-task pytorch library for continual learning, 2023.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models, 2024.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rkpACe1lx`.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *arXiv preprint arXiv:2211.11031*, 2022.

Hamed Hemati, Andrea Cossu, Antonio Carta, Julio Hurtado, Lorenzo Pellegrini, Davide Bacciu, Vincenzo Lomonaco, and Damian Borth. Class-incremental learning with repetition. In *Conference on Lifelong Learning Agents*, pages 437–455. PMLR, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. Detecting edit failures in large language models: An improved specificity benchmark, 2023.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Zixuan Ke and Bing Liu. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*, 2022.

Zixuan Ke, Hu Xu, and Bing Liu. Adapting bert for continual learning of a sequence of aspect sentiment classification tasks. *arXiv preprint arXiv:2112.03271*, 2021.

Zixuan Ke, Haowei Lin, Yijia Shao, Hu Xu, Lei Shu, and Bing Liu. Continual training of language models for few-shot learning. *arXiv preprint arXiv:2210.05549*, 2022.

Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. *arXiv preprint arXiv:1711.10563*, 2017.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

Gyuhak Kim, Sepideh Esmaeilpour, Changnan Xiao, and Bing Liu. Continual learning based on ood detection and task masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3856–3866, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*, 2017.

Dingcheng Li, Zheng Chen, Eunah Cho, Jie Hao, Xiaohu Liu, Fan Xing, Chenlei Guo, and Yang Liu. Overcoming catastrophic forgetting during domain adaptation of seq2seq language generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5454, 2022a.

Guodun Li, Yuchen Zhai, Qianglong Chen, Xing Gao, Ji Zhang, and Yin Zhang. Continual few-shot intent detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 333–343, 2022b.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Qingbin Liu, Xiaoyan Yu, Shizhu He, Kang Liu, and Jun Zhao. Lifelong intent detection via multi-strategy rebalancing. *arXiv preprint arXiv:2108.04445*, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*, 2022.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European conference on computer vision (ECCV)*, pages 67–82, 2018.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

Vittorio Mazzia, Alessandro Pedrani, Andrea Caciolai, Kay Rottmann, and Davide Bernardi. A survey on knowledge editing of neural networks. *arXiv preprint arXiv:2310.19704*, 2023.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022a.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022b.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022c.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022.

Natawut Monaikul, Giuseppe Castellucci, Simone Filice, and Oleg Rokhlenko. Continual learning for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13570–13577, 2021.

Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1320–1328, 2017.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Jürgen Schmidhuber, Sepp Hochreiter, et al. Long short-term memory. *Neural Comput*, 9 (8):1735–1780, 1997.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Continual-t0: Progressively instructing 50+ tasks to language models without forgetting. *arXiv preprint arXiv:2205.12393*, 2022.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pages 4548–4557. PMLR, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

DeciAI Research Team. Decilm-7b, 2023. URL `https://huggingface.co/Deci/DeciLM-7B`.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size, 2024.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

Gido M van de Ven and Andreas S Tolias. Three continual learning scenarios. In *NeurIPS Continual Learning Workshop*, volume 1, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Denny Vrandečić. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st international conference on world wide web*, pages 1063–1064, 2012.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487*, 2023a.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*, 2023b.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*, 2023c.

Xiaohan Wang, Shengyu Mao, Ningyu Zhang, Shumin Deng, Yunzhi Yao, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. Editing conceptual knowledge for large language models, 2024. URL `https://arxiv.org/abs/2403.06259`.

Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *arXiv preprint arXiv:2307.09218*, 2023d.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022b.

Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in Neural Information Processing Systems*, 31, 2018.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

Wenpeng Yin, Jia Li, and Caiming Xiong. Contintin: Continual learning from task instructions. *arXiv preprint arXiv:2203.08512*, 2022.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

Yingxiu Zhao, Yinhe Zheng, Zhiliang Tian, Chang Gao, Bowen Yu, Haiyang Yu, Yongbin Li, Jian Sun, and Nevin L Zhang. Prompt conditioned vae: Enhancing generative replay for lifelong learning in task-oriented dialogue. *arXiv preprint arXiv:2210.07783*, 2022.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*, 2023.

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

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.