# Functional Abstraction of Knowledge Recall in Large Language Models

Anonymous ACL submission

## Abstract

Pre-trained transformer large language models (LLMs) demonstrate strong knowledge recall capabilities. This paper investigates the knowledge recall mechanism in LLMs by abstracting it into a functional structure. We propose that during knowledge recall, the model's hidden activation space implicitly entails a function execution process where specific activation vectors align with functional components (Input argument, Function body, and Return values). Specifically, activation vectors of relation-related tokens define a mapping function from subjects to objects, with subject-related token activations serving as input arguments and object-related token activations as return values. For experimental verification, we first design a patching-based knowledge-scoring algorithm to identify knowledge-aware activation vectors as independent functional components. Then, we conduct counter-knowledge testing to examine the independent functional effects of each component on knowledge recall outcomes. From this functional perspective, we improve the contextual knowledge editing approach augmented by activation patching. By rewriting incoherent activations in context, we enable improved short-term memory retention for new knowledge prompting.

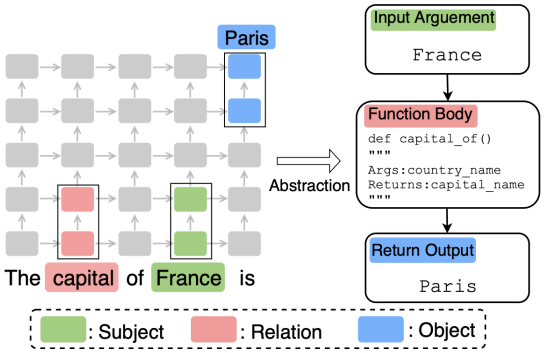



Figure 1: Illustration of our abstraction framework. In a knowledge recall process, *i.e., (subject, relation)→object*, we find that knowledge-related representations are locally distributed, and they are well-aligned with functional components.

## 1 Introduction

Pre-trained LLMs serve as knowledge bases (Petroni et al., 2019; Roberts et al., 2020; **?**), efficiently storing and recalling diverse facts (Kassner et al., 2023). Given a subject-relation query, LLMs can predict the correct object (Jiang et al., 2020). However, the internal mechanism of knowledge recall within LLMs still lacks transparency and interpretability, which limits LLMs from becoming reliable assistant tools (Hendrycks and Mazeika, 2022; Wang et al., 2023).

There are many interpretability works trying to reveal how knowledge is organized inside the model. The mainstream is module-based mechanical approaches, which typically attribute the storage and retrieval of knowledge to specific modules by identifying important model parameters (or neuron weights). They treat sub-modules as basic building blocks and study how they perform different information processing steps, thus providing a low-level architectural perspective on knowledge recall (Wang et al., 2022; Geva et al., 2023; Merullo et al., 2023; Geva et al., 2023; Meng et al., 2022a). However, the identification of salient modular components necessitates a substantial degree of manual effort, and it becomes difficult to interpret more complex phenomena using this framework (Lieberum et al., 2023; Hase et al., 2024; Cohen et al., 2024).

Another line of approaches is representation-based, which studies what kind of human-interpretable information the model has learned to represent within its activation space (Zou et al., 2023; von Rütte et al., 2024). These methods consider the behavior of the model to be the product of the representation space, which is the result of the interaction of multiple representations (Li et al.,

2024; Burns et al., 2022; Wang et al., 2024). By identifying and locating interpretable representations, we can abstract the inner workings of the model into a well-defined algorithm, where the neural representations are aligned with the variables or components in the algorithm.

In this paper, we inherit the idea of representation analysis and design a three-step method to abstract the knowledge recall process of black-box LLMs into an interpretable functional structure. The first step is to formulate a hypothesis, which aims to propose an interpretable algorithm that matches the model behavior. We assume that in the forward propagation of knowledge recall, the definition and execution of a function are implicitly performed in the activation space (Hernandez et al., 2023; Todd et al., 2023; Hendel et al., 2023) [1]. As shown in Figure 1, we use a function execution process to fit knowledge recall, which is that the subject is the input, the relationship is the mapping, and the object is the output. This is a reasonable assumption because the relationship is essentially a transformation operation between entities (Bordes et al., 2013).

The second step is to locate representations that can be aligned with the algorithm variables. To achieve that, we employ an activation patching technique [2] to probe interpretable representations. Specifically, we aim to identify activation vectors that are particularly responsible for encoding information about the subject, relation, and object in the forward propagation, which is achieved through the knowledge-encoding scoring algorithm we design based on the insight of causal mediation analysis (Meng et al., 2022a; Vig et al., 2020). This method can pinpoint knowledge-aware activation vectors by examining their knowledge-encoding scores. The results reveal a remarkable property that the distribution of relevant activation vectors is localized, which enables the isolation of these vectors and their treatment as distinct functional components (Input argument, function body, and return value).

Finally, the third stage is to experimentally verify that the localized neural representations exhibit properties that can be aligned as algorithmic variables. We design rigorously controlled counterfactual testing experiments (Geiger et al., 2021, 2024) involving the construction of counter-knowledge data pairs and the execution of four distinct types of interchange intervention. By building counter-knowledge data pairs where we fix either the subject or relation and interchange the other one, we can isolate and independently assess their functional roles. Our results support our hypothesis that these activation vectors function as distinct, separable components in the knowledge recall process.

Based on this functional perspective, we design a prompting-based knowledge editing method that does not require parameter tuning. When a new knowledge statement is presented, we resolve the conflict between the new knowledge and the LLM's original knowledge by patching specific functional components. Subsequently, we use this patched knowledge statement as a context prompt to inform the model that the knowledge has been edited.

## 2 Related Work

### 2.1 Mechanistic Interpretability Methods

This type of approach, also called the modular method, aims to understand LLMs in terms of neurons, modules, or circuits from a mechanistic view (Dai et al., 2021; Yao et al., 2024). They place model components as the analysis center, studying their algorithmic role that can execute specific functions for information processing, such as information storage and transmission. (Meng et al., 2022a) found that shallow-layer feed-forward module weights are responsible for subject knowledge storage. (Geva et al., 2023; Elhage et al., 2021) study shows the information transfer role of middle-layer attention module via attention knockout. (Wang et al., 2022; Yao et al., 2024) discovered circuits composed of multiple distinct role components through reverse engineering, which can provide a bottom-up explanation of the model's behavior.

### 2.2 Representational Interpretability Methods

This type of approach mainly studies internal representations of LLMs, focusing on the decoding and mining interpretable information. (Zou et al., 2023) viewed the model's behavior as the interaction of conceptual representations and found that some human-understandable concepts can be read from hidden states through feature dimensionality reduction methods. (Li et al., 2024) showed that the model exhibits different activation patterns for true and false statements and provided evidence

[1] In this paper, we use *representation*, *activation* and *hidden state* interchangeably.

[2] Also referred to as Causal Mediation Analysis, Causal Tracing, and Interchange Interventions.

that intervening in the hidden representations can mitigate hallucinations. (von Rütte et al., 2024) utilized various linear probers to detect different conceptual representations effectively and provided linear guidance at activation space for text generation style control.

These above methods uncover the emerging high-level structure and characteristics that activation space can demonstrate while abstracting away intricate low-level mechanisms.

### 2.3 Activation Patching for Interpretability

The activation patching technique is a popular tool for identifying the specific activation responsible for particular functions. (Pearl, 2022), which has been widely applied for LLMs interpretability (Zhang and Nanda, 2023; Hanna et al., 2024; Vig et al., 2020).

(Meng et al., 2022a; Geva et al., 2023) used this method for knowledge-storing modules localization. (Wang et al., 2022; Hanna et al., 2024; Lieberum et al., 2023; Conmy et al., 2023) identified the sub-network within a model's computation graph that implements a specified behavior. (Zhang and Nanda, 2023) systematically investigated the prompt corruption method and metrics for patching evaluation. (Ghandeharioun et al., 2024) decoded information from hidden representations by generating human-interpretable texts from post-patching computation.

In this work, we apply this technique to locate the knowledge-aware representations and validate the alignment of these representations with a pre-hypothesized functional structure.

## 3 Method

Given the reasonable assumption that knowledge recall is a function execution process, in this section, we introduce how to identify the knowledge-related representations through the proposed knowledge-scoring algorithm, as well as how to validate their alignment with functional components through the counter-knowledge test.

### 3.1 Notions

In this paper, we consider a knowledge recall scenario where an LLM $\mathcal{T}$ takes a subject ($s$)–relation ($r$) query as input and predicts the relevant object as the continuation. Using the template *"The <relation> of <subject> is"*, the query is tokenized into an embedding sequence $I(s,r) = [\tau_1, \tau_2, ..., \tau_n]$,

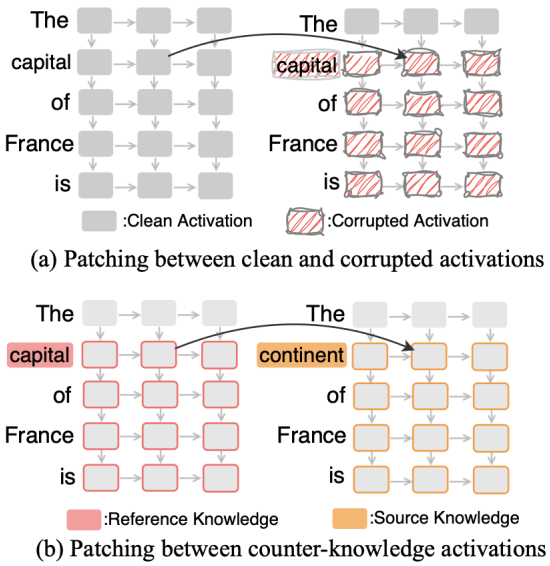



Figure 2: Illustration of two activation patching operations. In (a), we get the corrupted activations by adding noise to embeddings and then patch the corrupted ones with clean activations. In (b), we patch the source knowledge with the activations from the reference knowledge.

where $n$ is the token length. We denote the token positions for the subject and relation as $S$ and $R$, respectively. $I(s,r)$ is processed by a multi-layer transformer, generating activation vectors at each layer. The model outputs a probability $p(o|I(s,r))$ through a classification head $\phi$. We use $h_{l,j}$ to represent the activation vector at layer $l \in [0, ..., L-1]$ and token position $j \in [0, ..., n-1]$.

$$
\begin{aligned}
P(o|I(s,r)) =& \text{Softmax}(\phi(h_{L-1,n-1})) \\
h_{l,j} =& h_{l-1,j} + \text{Attn}(h_{l-1,\leq j}) + \\
& \text{MLP}(\text{Attn}(h_{l-1,\leq j}), h_{l-1,j})
\end{aligned}
$$

### 3.2 Knowledge Scoring via Activation Patching

**Overview** The basic idea is inspired by causal mediation analysis (Meng et al., 2022a,b), which has been used to identify knowledge-storing parameters by only analyzing subject concepts. In this work, we use this technique to identify the activation vectors that encode specific knowledge by treating the activation vectors as indirect causal variables and calculating their mediating effects as knowledge-encoding scores. A high score suggests that the activation vector effectively represents rich and relevant knowledge.

The score calculation is a two-step process, including textual knowledge ablation and corruption

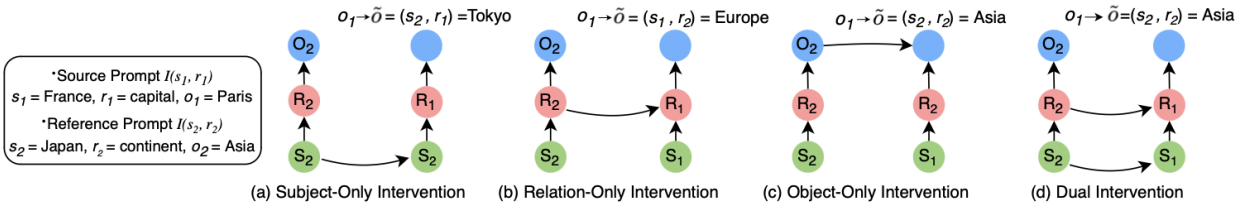



Figure 3: Illustration of the counter-knowledge testing experiments. If the prediction of the object after the interchange is affected only by the exchanged knowledge, it can be proved that each knowledge vector acts as a functional component independently.

restoration.

**Textual Knowledge Ablation** We achieve subject ablation, relation ablation, and object ablation, respectively, through noisy embeddings.

To obtain the subject ablation embeddings $I(s^*, r)$, we first create the noised subject token embeddings $\tau_S^*$, *i.e.*, $\tau_S^* = \tau_S + \epsilon$, where $\epsilon$ is Gaussian noise. Then, the $\tau_S^*$ continues to propagate forward, causing the following activation vectors to be corrupted. We use $h_{l,j}^{s*}$ to represent the activation vectors that are corrupted by the presence of $\tau_S^*$.

Similarly, the relation ablation embeddings $I(s, r^*)$ and the following corrupted $h_{l,j}^{r*}$ can also obtained by noising the relation embeddings, *i.e.*, $\tau_R^* = \tau_R + \epsilon$.

To ablate the object knowledge in the text query, we choose to add noise to both the subject and object embeddings simultaneously. In this way, we obtain the object ablation embeddings $I(s^*, r^*)$ and the relevant $h_{l,j}^{o*}$.

**Corruption Restoration** With these there types of knowledge ablation embeddings, we can calculate the Subject-Encoding score (SES), Relation-Encoding Score (RES), and Object-Encoding Score (OES) through corruption restoration, achieved by patching corrupted activations with clean ones.

Specifically, the three scores of each activation $h_{l,j}$ are calculated in three model runs.

• Run the model $\mathcal{T}$ on standard clean input embeddings $I(s, r)$ and all normal clean activation vectors $h_{l,j}$ are cached.

• Run the model $\mathcal{T}$ on the three ablated embeddings $I(s^*, r), I(s, r^*), I(s^*, r^*)$ respectively and the corrupted probabilities $P(o|I(s^*, r))$, $P(o|I(s, r^*))$ and $P(o|I(s^*, r^*))$ are obtained.

• Run the model $\mathcal{T}$ on the ablated embeddings again and patch each corrupted activation vector with clean ones one by one. Formally, we use $\mathrm{do}(h_{l,j}^{s*} = h_{l,j}) = h_{l,j}^{s*} \leftarrow h_{l,j}$ to denote the patching operation, where the corrupted $h_{l,j}^{s*}$ is replaced by a clean $h_{l,j}$. Then, the restored probabilities $P(o|I(s^*, r), \mathrm{do}(h_{l,j}^{s*} = h_{l,j}))$, $P(o|I(s, r^*), \mathrm{do}(h_{l,j}^{r*} = h_{l,j}))$ and $P(o|I(s^*, r^*), \mathrm{do}(h_{l,j}^{o*} = h_{l,j}))$ are obtained.

**Score Calculation** The three knowledge-encoding scores of an activation vector $h_{l,j}$ are defined as the difference between the restored and the corrupted probability.

$$
\begin{aligned}
SES(h_{l,j}) &= P(o|I(s^*, r), h_{l,j}^{s^*} \leftarrow h_{l,j}) \\
&\quad - P(o|I(s^*, r)) \\
RES(h_{l,j}) &= P(o|I(s, r^*), h_{l,j}^{r^*} \leftarrow h_{l,j}) \\
&\quad - P(o|I(s, r^*)) \\
OES(h_{l,j}) &= P(o|I(s^*, r^*), h_{l,j}^{o^*} \leftarrow h_{l,j}) \\
&\quad - P(o|I(s^*, r^*))
\end{aligned}
$$

Through comparative analysis, we can locate activation vectors with high scores and treat them as subject, relation, and object representation vectors.

### 3.3 Counter-Knowledge Testing

Given the identified subject, relation, and object representations, we treat them as functional components and assess the alignment through counter-knowledge data preparation and activation interchange.

**Counter-knowledge Data** By constructing such source-reference pairs, we can interchange the subject or relation representations to disentangle their individual functional role in object prediction. Each counter-knowledge data pair consists of a source query $I(s_1, r_1)$ with object $o_1$ and a reference query $I(s_2, r_2)$ with object $o_2$, where either the subjects or relations differ ($s_1 \neq s_2, r_1 = r_2$ or $s_1 = s_2, r_1 \neq r_2$ ), or both ($s_1 \neq s_2$ and $r_1 \neq r_2$ ).

**Activation Vectors Interchange** Activation vector interchange replaces a specific activation vector

from a model's source run with one from a reference run to intervene in knowledge recall. By observing changes in object predictions $\tilde{o}$ after the swap, we can determine if each knowledge vector functions independently. If the predictions change only with the reference vector, it indicates that the vectors operate as independent functional components.

Given a source-reference pair, $I(s_1, r_1)$ and $I(s_2, r_2)$, we perform activation interchange between the two model runs. Let $\beta_{S1}$, $\beta_{R1}$, and $\beta_{O1}$ represent the source subject, relation, and object vectors, and $\beta_{S2}$, $\beta_{R2}$, and $\beta_{O2}$ represent the reference ones. Four experimental settings are designed based on different interchange sources and targets, as shown in Figure 3.

1. Subject-only Intervention. Only replace the source subject vector with the reference one to get object probability $P_s(\tilde{o}|I(s_1, r_1), \mathrm{do}(\beta_{S1} = \beta_{S2}))$, where $\tilde{o} = (s_2, r_1)$.

2. Relation-only Intervention. Only replace the source relation vector with the reference one to get object probability $P_r(\tilde{o}|I(s_1, r_1), \mathrm{do}(\beta_{R1} = \beta_{R2}))$, where $\tilde{o} = (s_1, r_2)$.

3. Object-only Intervention. Only replace the source object vector with the reference one to get object probability $P_o(\tilde{o}|I(s_1, r_1), \mathrm{do}(\beta_{O1} = \beta_{O2}))$, where $\tilde{o} = (s_2, r_2)$.

4. Dual Intervention. Replace both the source subject and relation vectors with reference ones to get object probability $P_d(\tilde{o}|I(s_1, r_1), \mathrm{do}(\beta_{S1} = \beta_{S2}), \mathrm{do}(\beta_{R1} = \beta_{R2}))$, where $\tilde{o} = (s_2, r_2)$. This reflects an architectural composition of activation vectors for knowledge recall.

#### 3.3.1 Evaluation

We use two metrics to evaluate whether an individual knowledge vector performs its functional role independently:

1. Interchange Effect Comparison. We respectively calculate the interchange effects on the text knowledge and on the vector knowledge, *i.e.*, $P_s(\tilde{o}|I(s_1, r_1), \mathrm{do}(\beta_{S1} = \beta_{S2}))$ and $P_s(\tilde{o}|I(s_2, r_1))$, where $\tilde{o} = (s_2, r_1)$.

2. Interchange Accuracy. We calculate the matching success rate between the prediction results after text knowledge intervention and the prediction results after representation knowledge intervention, *i.e.*, $\mathbb{E}[\mathrm{argmax}_{\tilde{o}}\, P_s(\tilde{o}|I(s_1, r_1), \mathrm{do}(\beta_{S1} = \beta_{S2})) \stackrel{?}{=} \mathrm{argmax}_o\, P(o|I(s_2, r_1))]$.

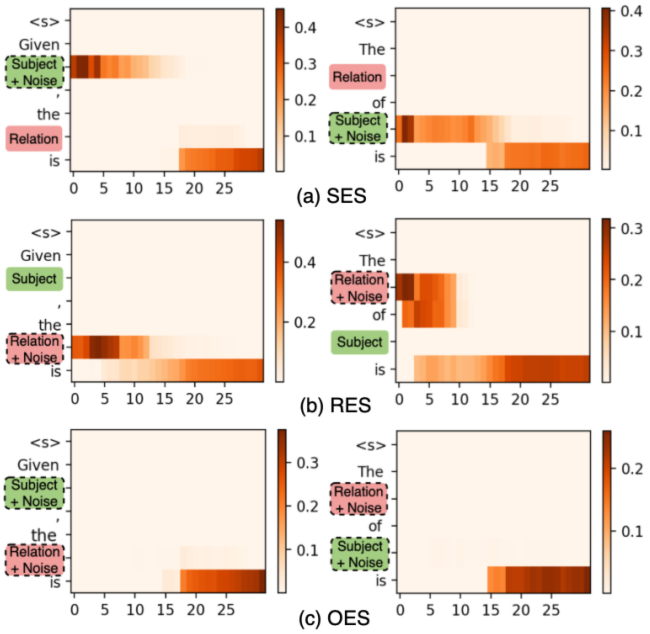



Figure 4: Scores heat map visualization. The left column uses the template *"Given <subject>, the <relation> is"*, and the right column uses *"The <relation> of <subject> is"*.

## 4 Experiments

**Model** We choose Llama2-7b (Touvron et al., 2023) and Mistral-7b (Jiang et al., 2023) as our models, each one with 32 layers. We use huggingface implementations (Wolf et al., 2020) of each model and run them on an A100 40GB GPU.

**Data** We utilize the relational knowledge database from (Hernandez et al., 2023), which consists of 26 types of factual knowledge. To ensure that the knowledge has been stored in the model, we filter the dataset to examples where the model correctly predicts the object $o$ given the query. More details are in the appendix.

**Query templates** Given the existence of unidirectional self-attention in LLMs, which only grants subsequent tokens access to previous tokens, the order of subject and relation occurrence within the query may result in semantic entanglement of relevant activation vectors. Therefore, to verify that the subject-relation order does not affect their functional roles, we designed two types of query templates with different subject-relation orders, that are *"The <relation> of <subject> is"* and *"Given <subject>, it's <relation> is"*.

### 4.1 Scoring Reveals Localized Features

Figure 4 shows the distribution map of three types of knowledge scores obtained from two prompt templates with different subject-relation orders. We can clearly observe the locality of the distribution of the corresponding activation vectors. We conduct more detailed locality analyses along the position dimension and the layer dimension, respectively.

#### 4.1.1 Locality Along Token Positions

In Figure 5, we demonstrate the locality of the high-scoring activation vectors along the token position dimension. We find that the vectors with high OES scores are all located at the last token. This result is reasonable, as the classification head will use the activation vector of the last token to predict the next token. Early decoding studies have also discussed similar findings that project activation vectors into vocab space in advance to examine internal information (Ghandeharioun et al., 2024; Hernandez et al., 2023).

Interestingly, the vectors with high SES and high RES are almost located within the range of the subject and the relation tokens, respectively, and the order of the subject-relation does not seem to matter, although the self-attention is unidirectional. This observation reveals that the language model does not encode knowledge in a distributed manner but rather confines the knowledge representations to those relevant tokens.

#### 4.1.2 Locality Along Layer Positions

In Figure 6, we plot line graphs showing how the scores change across the layers, focusing on their relevant tokens. We find that the vectors with high OES scores are located in the middle to late layers, while the vectors with high SES and RES scores are located in the early to middle layers. This finding is somewhat consistent with the explanation based on information aggregation, that is, the information of the subject is transferred in the middle layers (Geva et al., 2023).

#### 4.1.3 Formulating Functional Components

Based on the above findings, we define activation vectors that can be treated as components of a knowledge recall function.

• Subject Vectors: Activation vectors located in early layers (0-14) within the subject tokens range, serving as input arguments.

• Relation Vectors: Activation vectors located in early layers (0-10) within the relation tokens range, serving as the function body.

• Object Vectors: Activation vectors located in late layers (15-31) in the final token, serving as return values.

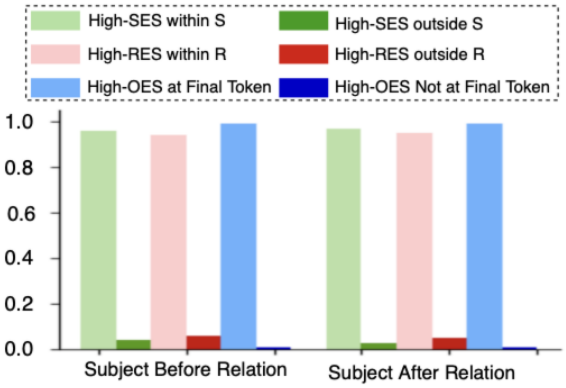



Figure 5: Locality along token positions. This bar graph shows the proportion of high-score (>0.05) activation vectors in different token ranges.

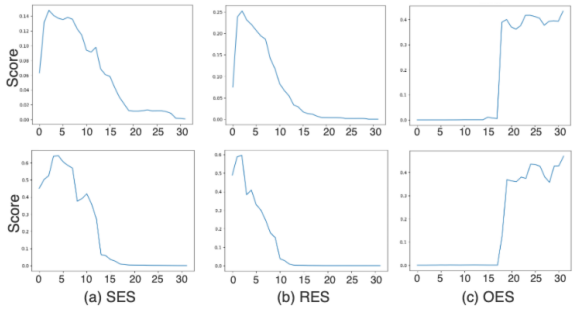



Figure 6: Locality along layer positions. The plots of average SES, RES, and OES at the identified token positions, respectively, where the upper template is *"Given <subject>, the <relation> is"* and the lower template is *"The <relation> of <subject> is"*.

### 4.2 Counter-Knowledge Testing

In this experiment, we present the results of the counter-knowledge testing to demonstrate that the identified knowledge vectors operate in a functional manner to recall the knowledge.

**Layer-by-layer Interchange** In this setting, we interchange the corresponding knowledge vectors layer by layer and record the intervention effects after each operation. We also calculate the intervention effect of textual knowledge interchange across all relation types. The results are shown in Figure 7. We plot the intervention effect in subject-only, relation-only, and object-only settings, where

the green line represents the intervention effect on knowledge vectors, and the red line represents the intervention effect on textual knowledge. We can observe that the effect of subject-only and relation-only intervention is obvious in the early to middle layers, and the effect of object-only intervention is obvious in the middle to late layers. This observation is consistent with the previous knowledge encoding scoring results, suggesting that our designed scoring algorithm accurately identifies knowledge-aware representations.

We can also notice that the intervention on hidden states achieves results comparable to those of the intervention on textual knowledge. This indicates that the intervened hidden states faithfully encode knowledge and that these hidden states and causal variables are well aligned.

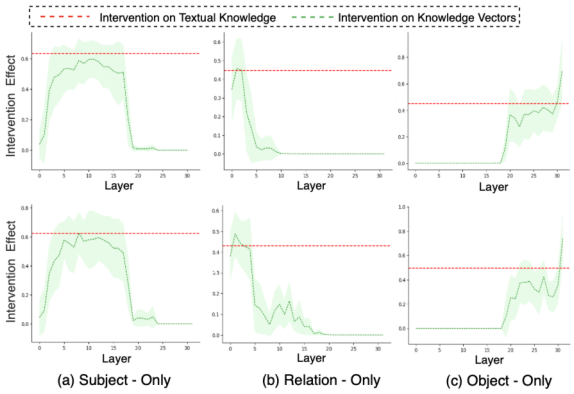



Figure 7: The plots of intervention effects of textual and representational knowledge, respectively. The upper prompt template is *"Given <subject>, the <relation> is"* and the lower prompt template is *"The <relation> of <subject> is"*.

**Mean Interchange** In this setup, we replace the knowledge vectors in the source model run with the corresponding mean reference knowledge vector [3] and then calculate the prediction accuracy of the target after the interchange. The exchange accuracy of subject and relation vectors also decreased slightly.

As shown in Table 1, the results show that the accuracy of object-only vector interchange is almost 100%, suggesting that the object vector exhibits a high degree of fidelity as the function return value. The accuracy of the subject-only and the relation-only vector interchange also decreased slightly. This shows that the mean knowledge vector we extracted fully and independently encodes the corresponding knowledge representation. In the dual interchange scenario, the accuracy also drops slightly, indicating that each knowledge vector can be used as an individual function component to complete knowledge recall.

| | Subject-Only | Relation-Only | Object-Only | Dual |
|---|---|---|---|---|
| *The <relation> of <subject> is* | | | | |
| Country-Capital | 97.1% | 93.6% | 100.0% | 91.1% |
| Food-Country | 96.4% | 93.5% | 100.0% | 92.5% |
| Company-Headquarter | 93.6% | 93.1% | 99.4% | 91.3% |
| Product-Company | 92.4% | 92.9% | 99.1% | 89.2% |
| Landmark-Country | 93.8% | 92.3% | 100.0% | 91.7% |
| Fruit-Color | 94.1% | 93.7% | 100.0% | 92.5% |
| *Given <subject>, its <relation> is* | | | | |
| Country-Capital | 96.5% | 94.8% | 100.0% | 92.7% |
| Food-Country | 97.1% | 93.2% | 100.0% | 91.8% |
| Company-Headquarter | 96.6% | 93.9% | 99.5% | 88.5% |
| Product-Company | 97.3% | 94.8% | 99.3% | 89.2% |
| Landmark-Country | 95.9% | 93.4% | 100.0% | 90.8% |
| Fruit-Color | 95.3% | 94.9% | 100.0% | 91.4% |

Table 1: The Mean Interchange Accuracy across different relations, using two prompts with different subject-relation order, respectively.

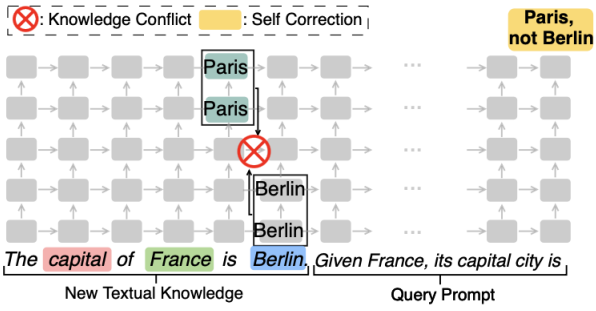
## 5 Functionality Informs Knowledge Editing

Given the credibility of the implicit functional structure in knowledge recall, it is natural to think about how to apply it to knowledge editing, which aims to insert a new knowledge triplet $(s, r, o^*)$ in place of the original triplet $(s, r, o^c)$. In this section, we improve the contextual knowledge editing method based on functional activation patching.

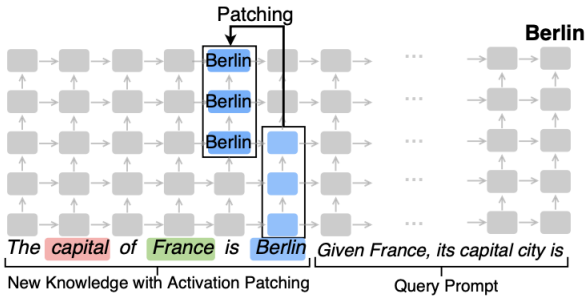
Contextual knowledge editing provides short-term memory of new knowledge by appending context prompting before queries (Madaan et al., 2022; Zheng et al., 2023; Yin et al., 2024). However, from a functional perspective, the new knowledge prompting function implicitly returns an object that conflicts with the new object, causing the model to produce self-correcting behavior.

As shown in Figure 8 (a), when a new knowledge prompt *"The capital of France is Berlin"* is fed to the model, the original object *"Paris"* conflicts with the new object *"Berlin"* in the activation space. This discrepancy triggers the model's awareness of the inconsistency between the new knowledge and its existing knowledge. Consequently, the model attempts to self-correct its predicted answer to align with its original knowledge.

[3] We extract the mean of the vectors at the corresponding token positions and layer from the reference model run.




(a) Only prompting textual knowledge




(b) Prompting with activation patching

Figure 8: Figure (a) shows the knowledge conflict in contextual knowledge editing. Figure (b) shows our proposed activation patching-based knowledge editing method.

### 5.1 Method

To dissolve the knowledge conflict, we patch the original object representation with the new object vector. Specifically, as illustrated in Figure 8 (b), during the forward pass of the new knowledge prompt, we replace the late layer activation vectors of the previous token of objects, *e.g.*, the *"is"* token, which has been proven to encode the original object information, with the new object mean activation vector. In this way, the context prompting completely implements the recall function of new knowledge and better provides short-term memory without knowledge conflict. We extract the mean activation vector at the early layers (0-14) from the object tokens of new knowledge prompting to obtain the new object vector.

### 5.2 Data

We construct 3000 new knowledge triplets to evaluate our method by replacing the objects of the original triplets. Then, we fill each new knowledge triplet into the template to get knowledge statements for context prompting. We design two types of templates:

(1) Declarative Format: Directly uses affirmations to state new facts, which is concise and suitable for knowledge expression. We use the previous two templates: (1) *"The <relation> of <subject> is <object>."* and (2) *"Given <subject>, it's <relation> is <object>."*.

(2) Q&A Format: Simulates the process of Q&A, first posing a question and then providing the corresponding answer, which is more interactive (Zheng et al., 2023). We use the prompt *"Q: Tell me the <relation> of <subject>. A: <object>."*.

### 5.3 Results

We use the Efficacy Score and Efficacy Magnitude to measure the performance. The Efficacy Score (ES) quantifies the accuracy that the edited output $o^*$ is more accurate than the original output $o^c$, defined as: $\mathbb{E}\left[\mathbb{I}\left[\mathcal{P}\left(o^*\right) > \mathcal{P}\left(o^c\right)\right]\right]$. The Efficacy Magnitude (EM) measures the average increase in probability due to editing, defined as: $\mathbb{E}\left[\mathcal{P}\left(o^*\right) - \mathcal{P}\left(o^c\right)\right]$.

In Table 2, we can observe that activation patching can significantly improve ES and EM regardless of the prompt template.

| Input types | ES | EM |
|---|---|---|
| Query-Only | 0.000 | -0.230 |
| Declarative1 + Query | 0.171 | 0.058 |
| Declarative1 (Patching) + Query | **0.664** | **0.286** |
| Declarative2 + Query | 0.143 | 0.132 |
| Declarative2 (Patching) + Query | **0.657** | **0.314** |
| Q&A + Query | 0.192 | 0.231 |
| Q&A (Patching)+ Query | **0.685** | **0.381** |

Table 2: The Efficacy Score and Magnitude of knowledge editing using different input baselines.

## 6 Conclusion

In this paper, we design a two-stage approach to reveal the underlying functional structure of the knowledge recall process in LLMs. In the first stage, we propose a knowledge-encoding scoring algorithm to identify isolated knowledge-aware activation vectors and treat them as independent function components. In the second stage, we construct counter-knowledge data pairs and verify the alignment of knowledge vectors with functional components by activation interchange interventions. Experimental results reveal that knowledge vectors are highly faithful to their functional roles. We also design a contextual knowledge editing algorithm based on the function structure and used activation patches to achieve more efficient knowledge prompts.

## 7 Limitation

Although this paper conceptualizes the knowledge recall of LLMs as a functional structure, several limitations persist. Foremost, the proposed knowledge scoring algorithm relies on a brute-force search method, resulting in high time complexity. There is significant potential in developing more efficient knowledge exploration algorithms.

Furthermore, our framework awaits validation on text-generation and general question-answering tasks that specifically involve knowledge recall. The accuracy of the proposed knowledge editing method also requires improvement, suggesting that representation rewriting does not fully encapsulate the original knowledge. Moving forward, a deeper exploration of the knowledge storage mechanisms within LLMs is necessary.

## References


Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*.

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

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.

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

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. 2024. Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning*, pages 160–187. PMLR.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscope: A unifying framework for inspecting hidden representations of language models. *arXiv preprint arXiv:2401.06102*.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2024. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*.

Dan Hendrycks and Mantas Mazeika. 2022. X-risk analysis for ai research. *arXiv preprint arXiv:2206.05862*.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Nora Kassner, Oyvind Tafjord, Ashish Sabharwal, Kyle Richardson, Hinrich Schuetze, and Peter Clark. 2023. Language models with rationality. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14190–14201, Singapore. Association for Computational Linguistics.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36.

Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. 2023. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*.

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

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

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

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2023. A mechanism for solving relational tasks in transformer language models.

Judea Pearl. 2022. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pages 373–392.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2023. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Dimitri von Rütte, Sotiris Anagnostidis, Gregor Bachmann, and Thomas Hofmann. 2024. A language model's guide through latent space. *arXiv preprint arXiv:2402.14433*.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, et al. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small, 2022. *URL https://arxiv.org/abs/2211.00593*.

Zijian Wang, Britney White, and Chang Xu. 2024. Locating and extracting relational concepts in large language models. *arXiv preprint arXiv:2406.13184*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *arXiv preprint arXiv:2405.17969*.

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. 2024. History matters: Temporal knowledge editing in large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19413–19421.

Fred Zhang and Neel Nanda. 2023. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*.

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

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. 2023. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*.

# A Appendix

## A.1 Experimental Details

### A.1.1 Data Preprocessing

In this paper, we filter the relational knowledge data (Hernandez et al., 2023) and select the data whose relevant knowledge is already stored in the model. Specifically, each piece of knowledge data is represented by a triple (subject, relation, object). We fill the subject and relation into the predefined query template to predict the corresponding object. We filter data with predefined templates of different subject-relation orders, which are *"The*

*<relation> of <subject> is"* and *"Given <subject>, its <relation> is"*. If the object can be successfully predicted by these two templates both, it can be considered that this knowledge has been stored in the model. Table 1 shows the amount of data before and after filtering across various relation types.

Table 3: The initial quantity and filtered quantity of each relation type. The filtered objects can be predicted as the next token by the prompt *"Given <subject>, the <relation> of this one is"*

| Subject-Object | Initial quantity | Filtered quantity | |
|---|---|---|---|
| | | Mistral-7b | Llama-7b |
| Country-Capital | 253 | 155 | 168 |
| Country-language | 24 | 14 | 18 |
| Landmark-Country | 836 | 531 | 547 |
| Country-Currency | 30 | 25 | 26 |
| Product-Company | 522 | 278 | 289 |
| Company-Headquarters | 674 | 348 | 351 |
| Food-Country | 30 | 22 | 24 |
| Food-Color | 30 | 15 | 19 |
| City-Country | 27 | 27 | 24 |

Our subsequent knowledge scoring and counter-knowledge test experiments are all conducted on this filtered dataset.

## A.2 Knowledge Scoring

In this section, we introduce the technical details of the knowledge-scoring experiment.

### A.2.1 Textual Knowledge Ablation

We add Gaussian noise $\mathcal{N}(0, v)$ to the embeddings of certain tokens, where $v$ is 5 times the standard deviation of the token embeddings from all data. If the subject and relation are tokenized into multiple tokens, we add noise to these tokens at the same time.

### A.2.2 Score Calculation

When calculating the object prediction probability, since the object is predicted in the form of question completion, we only consider the probability of the first token predicted.

## A.3 Counter-Knowledge Testing

### A.3.1 Data Pair Construction

When constructing source-reference prompt pairs from a knowledge triple, we choose different strategies depending on the components being interchanged.

When performing the subject-only interchange, we fill the reference template with a subject from the triple and another unrelated relation, and fill the source template a the relation from the triple and another unrelated subject.

When performing the relation-only interchange, we fill the reference template with a relation from the triple and another unrelated subject, and fill the source template with a subject from the triple and another unrelated relation.

When performing the object-only and dual interchange, we fill the reference template with the a relation and a subject from the triple, and fill the source template with unrelated relations and subjects.

### A.3.2 Activation Vectors Interchange

The interchange operation is divided into two steps: extracting and replacing knowledge vectors.

When extracting the subject and relation vectors, we extract the hidden activation vector at the last token position of the subject and relation from the reference process. When extracting the object vector, we extract the hidden activation vector at the last token position of the entire query prompt.

When performing subject replacement, we replace the hidden activations of all subject tokens in the source prompt with the corresponding reference subject vector. When performing relation replacement, we replace the hidden activations of all relation tokens in the source prompt with the corresponding reference relation vector. When performing object replacement, we replace the hidden activation of the last token in the source prompt with the corresponding reference object vector.

When calculating the interchange effect and accuracy, we only consider the probability that the first token predicted belongs to the target object.

## A.4 Knowledge editing Data

For each factual knowledge triple, we replace the object to construct a new triple, and fill the new triple into the template to get new knowledge. The template is the first declarative format *"The <relation> of <subject> is <object>"*. For the following query template, we randomly select from the declarative format 2 and the QA template.