STEERING CIRCUIT-LEVEL ACTIVATIONS OF LANGUAGE MODELS FOR PERSONALIZED KNOWLEDGE INJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Although large language models (LLMs) are increasingly personalized to individual users, they still struggle with a deceptively simple but critical requirement: updating mutable personal facts, such as a user's job, location, or relationships. Fine-tuning approaches are costly and prone to catastrophic forgetting. Meanwhile, knowledge editing approaches rely on subject-bound representations; they fail in personal domains because private entities lacking prior exposure lead to unstable key extraction, while multi-attribute updates cause key collisions. We propose **SPIKE** (**S**teering Circuit-level Activations of Language Models for **P**ersonal**I**zed **K**nowledge Inj**E**ction), a methodology that identifies the LLM components effective for injecting personal knowledge and incorporates mutable personal knowledge into the LLM through activations generated via a KG–LLM Alignment Module. Our experiments demonstrate that SPIKE effectively balances the accurate incorporation of new facts with the preservation of existing knowledge, offering a practical solution for personalization in settings where user information evolves over time. Our code is available at `https://anonymous.4open.science/r/spike`.

1 INTRODUCTION

Personalizing large language models to individual users has become a central problem as LLMs are increasingly deployed in user-facing applications. Most existing work has focused on persona or writing-style adaptation (Jiang et al., 2024; Liu et al., 2025; Li et al., 2025; Zhang et al., 2025a), leaving a more fundamental capability underexplored: updating and reasoning over user-specific factual knowledge, such as "Jack transitioned from a programmer to a product manager," which the model must retain and use to act as a personal agent.

Personal factual knowledge poses unique challenges because it is inherently **mutable over time**, reflecting changes such as job transitions, address updates, or evolving daily routines. A straightforward approach to integrating evolving personal knowledge into LLMs is to fine-tune them on personal data (Dutt et al., 2022; Salemi et al., 2024). While this can embed personal knowledge, repeatedly fine-tuning on evolving personal information is computationally costly and risks catastrophic forgetting (He et al., 2021; Dou et al., 2024). Another approach is to edit language model parameters directly. Representative *locate-and-edit* methods, such as ROME (Meng et al., 2022), update factual knowledge by identifying a subject-bound lookup key vector and optimizing the corresponding value vector.

However, this mechanism faces severe structural limitations in the personal domain for two main reasons. First, unlike well-known entities with stable lookup representations, private personal entities lack meaningful representations in the model's parameters due to a lack of prior exposure. Consequently, their extracted lookup key vectors are highly sensitive to input contexts, leading to unstable and unreliable knowledge recall (detailed variance analysis in Appendix A.2). Second, because the lookup key vector is bound strictly to the subject entity rather than the specific relation being edited, a structural bottleneck arises. When updating multiple attributes for the same individual, which is a standard requirement in personal knowledge updates, using a single subject-bound key forces newly injected target values to overwrite previously stored ones (Dong et al., 2025). This key collision

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

results in the failure to preserve existing knowledge. Together, these observations highlight the need for new strategies to internalize mutable personal factual knowledge effectively.

We formalize **fact-level personalization with evolving knowledge** as a task that maps updates from a personalized knowledge graph (PKG) directly into an LLM's internal representations. This enables answering personal queries without exposing raw sensitive records (e.g., medical or financial histories) in inference prompts.

Achieving this requires addressing two core objectives:

- **Where to intervene:** Building on knowledge circuits (Yao et al., 2024), our empirical analysis shows that modifying attention head parameters yields more accurate updates with fewer side effects than altering feed-forward neural network (FFN) parameters (Section 4.1).
- **How to update:** We learn an update representation capturing the discrepancy between an updated personal fact and its initial counterpart, injecting it through the identified circuits to precisely integrate user-specific information without disrupting existing knowledge.

To this end, we propose **S**teering Circuit-level Activations of Language Models for **P**ersonal**I**zed **K**nowledge Inj**E**ction (**SPIKE**), which efficiently integrates user-specific updates into relevant model circuits. SPIKE accurately adapts to evolving personal updates while preserving unchanged personal attributes and general knowledge (locality and retention). Beyond conventional knowledge editing, it also generalizes to unseen updated facts.

2 Related Work

2.1 Knowledge Editing

Knowledge editing updates LLM parameters without full retraining, but risks side effects like forgetting and distortion (Gupta et al., 2024). Existing approaches edit FFN key–value memories (ROME (Meng et al., 2022), MEMIT-Merge (Dong et al., 2025)), reduce collateral distortion (AlphaEdit (Fang et al., 2025), DiKE (Zhang et al., 2025b)), or enable lifelong editing (WISE (Wang et al., 2024a), LightEdit (Jung et al., 2026)). However, applying these methods to personal updates presents two key challenges: private entities lack stable parameter representations (yielding unreliable key vectors), and subject-bound lookup keys cause structural collisions when updating multiple attributes for the same individual.

2.2 Circuit Finding

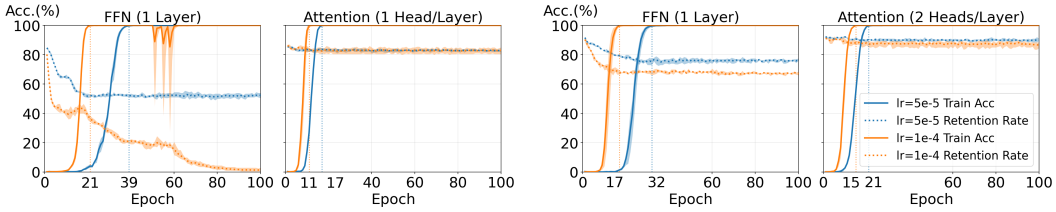
Recent efforts to interpret large language models (LLMs) have focused on identifying compact subcomponents of the model that are responsible for specific behaviors. These subcomponents, known as circuits, typically consist of a small set of attention heads and MLPs that strongly influence the output. Automated Circuit Discovery (Conmy et al., 2023) formalizes circuit extraction as a subgraph selection problem, where nodes are LLM components and edges denote their connections, by progressively removing low-impact edges. Syed et al. (2024) mitigates the high complexity with a gradient-based approximation. Wang et al. (2025) instead ranks attention heads by their contribution, retaining only the most influential ones for fine-tuning.

3 Preliminaries

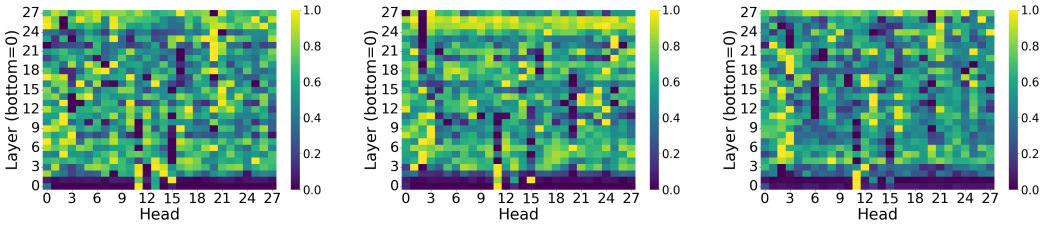
3.1 Task Formulation

Individuals frequently change their occupations, addresses, or daily routines, and we assume that such information can be easily expressed as personal knowledge graphs (PKGs) (Prahlad et al., 2025; Dutt et al., 2022).

To this end, we formulate a fact-level personalization task that explicitly updates mutable knowledge but retains unchanged knowledge. An update to a PKG naturally yields two versions of the PKG: the initial KG ($\mathbf{KG}^{\text{init}}$) and the updated KG ($\mathbf{KG}^{\text{upd}}$).



(a) GPT-J-6B (b) Qwen2.5-7B-Instruct



(c) Qwen - PeaCoK-Ex (d) Qwen - PerInfoKG (e) Qwen - `known_1000`

Figure 1: **Top row (a–b):** Training accuracy and retention rate over epochs for selective fine-tuning on FFN and attention heads on two LLMs. Shaded bands show the min–max range across three runs. **Bottom row (c–e):** Heatmap visualizations of important components identified in Qwen2.5-7B-Instruct (Yang et al., 2024) across PeaCoK-Ex, PerInfoKG, and `known_1000` (Wiki-based dataset (Meng et al., 2022)).

The *init–LLM* already knows the original user-specific facts in $\mathbf{KG}^{\text{init}}$, but cannot correctly answer queries regarding updated facts. The updated KG reflects changes over time, such as a new workplace or an altered daily routine.

Let $\mathcal{T}^{\text{init}} = \{(s_1, r_1, o_1^{\text{init}}), \ldots, (s_n, r_n, o_n^{\text{init}})\}$ denote a set of personal triples in $\mathbf{KG}^{\text{init}}$ and $\mathcal{T}^{\text{upd}} = \{(s_1, r_1, o_1^{\text{upd}}), \ldots, (s_n, r_n, o_n^{\text{upd}})\}$ denote a set of personal triples in $\mathbf{KG}^{\text{upd}}$, where each triple (s_i, r_i, o_i) consists of a subject entity s_i, a relation r_i, and a tail entity o_i representing a personal factual value (e.g., (`Mike`, `CommutingMeans`, `Bike`)). Here, n is the number of personal factual triples. A set of modified pairs of triples is given by $\mathcal{C} = \{\big((s_i, r_i, o_i^{\text{init}}), (s_i, r_i, o_i^{\text{upd}})\big) \mid i \in [n],\ o_i^{\text{init}} \neq o_i^{\text{upd}}\}$. In contrast, a set of non-modified pairs of triples is expressed as $\mathcal{R} = \{\big((s_i, r_i, o_i^{\text{init}}), (s_i, r_i, o_i^{\text{upd}})\big) \mid i \in [n],\ o_i^{\text{init}} = o_i^{\text{upd}}\}$. Let $|\mathcal{C}| = m$ and $|\mathcal{R}| = n - m$.

3.2 Personal Knowledge Graph Construction

Due to privacy constraints on real personal factual data, we construct two real-world-based benchmarks: ORCID, derived from profile changes in ORCID annual public data (ORCID, 2019; 2025) ($\mathbf{KG}^{\text{init}}$ and $\mathbf{KG}^{\text{upd}}$), and Athlete, constructed from Kaggle Tokyo 2020 Olympics athlete records (Petro, 2021) with artificial updates to mutable fields (*discipline*, *residence*).

To validate our method, we additionally create two synthetic datasets: PeaCoK-Ex, extending PeaCoK (Gao et al., 2023) across 822 synthetic individuals by modifying 20% of person–occupation pairs, and PerInfoKG, covering 2,000 fictitious individuals across 23 personal fields partitioned into 17 *mutable* (for editing) and 6 *immutable* (for locality evaluation) attributes. Appendix A.3 provides full construction details.

4 Methodology

4.1 Circuit Analysis for Personal Information Localization

To efficiently incorporate updated personal information in large language models (LLMs) while minimizing side effects such as unintended changes, we adopt a circuit-level analysis approach inspired

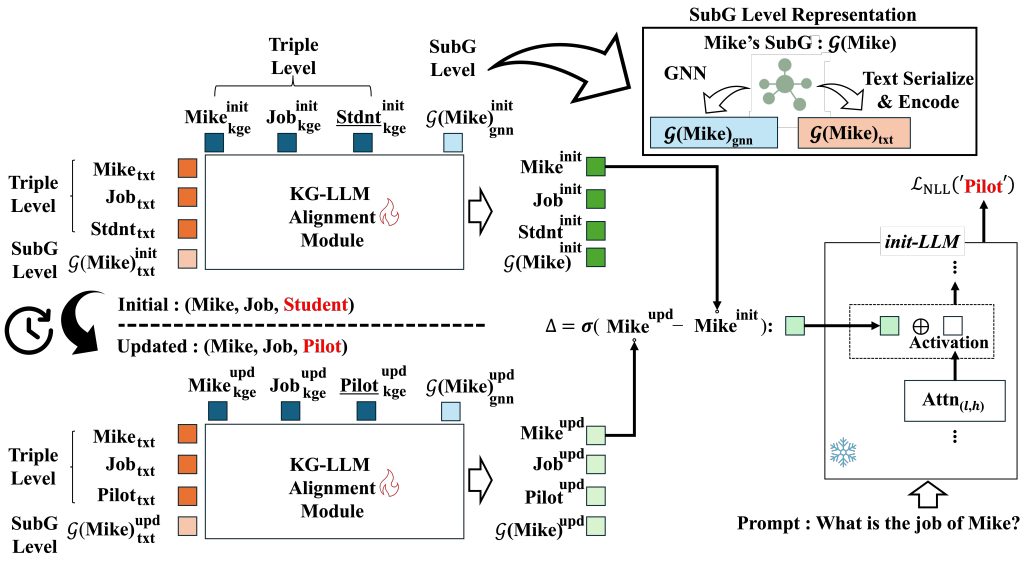


Figure 2: Illustration of SPIKE when the triple `(Mike, Job, Student)` is updated to `(Mike, Job, Pilot)`.

by mechanistic interpretability. We first investigate which components of LLMs, attention heads or FFNs, are more effective targets for updating personal knowledge. We adopt HeadMap (Wang et al., 2025) and Causal Mediation Analysis (Meng et al., 2022) to measure the importance scores of attention heads and FFNs, respectively. Based on these importance scores, we select a limited number of parameters and perform selective fine-tuning using personal knowledge from PeaCoK-Ex. The selected layer is the 10th for GPT-J and the 5th for Qwen2.5-7B-Instruct and Llama3-8B-Instruct. For the attention heads, we select 1 head per layer for GPT-J and 2 heads per layer for Qwen and Llama.

Figures 1a and 1b compare attention- and FFN-based fine-tuning on GPT-J-6B (Wang & Komatsuzaki, 2021) and Qwen2.5-7B-Instruct (Yang et al., 2024). Across learning rates, targeting attention heads consistently outperforms FFNs: (i) **Convergence:** Attention reaches $\geq 99.5\%$ accuracy in fewer epochs; (ii) **Retention:** Attention maintains stable retention, whereas FFN collapses to near zero on GPT-J-6B ($\text{lr} = 1\text{e}{-4}$); and (iii) **Efficiency:** Attention uses fewer parameters (1.56% vs. 2.38% on GPT-J; 0.90% vs. 3.12% on Qwen). These findings justify targeting attention heads for personal knowledge updates. Appendix A.9 (Figure 5) shows consistent results on Llama-3-8B-Instruct.

To further characterize how attention heads engage with personal information, we visualize layer-wise head importance scores for Qwen2.5-7B-Instruct in Figures 1c–1e. The similarity between head importance patterns on personal information datasets (PeaCoK-Ex and PerInfoKG) is 0.4754, while the similarity between PeaCoK-Ex and Wikidata-based general knowledge (`known_1000` (Meng et al., 2022)) is 0.4375 (the similarity between PerInfoKG and `known_1000` is 0.4100). This indicates that the set of heads activated by personal information data differs somewhat from those activated by general knowledge, which supports head-level intervention as a meaningful unit of updating personal knowledge. Detailed procedures for computing similarity are provided in Appendix A.8. Based on these empirical advantages, we target attention heads in the remainder of this work.

4.2 Personal Information Injection

Building on the attention heads identified in Section 4.1 as being particularly effective for incorporating personal information into the LLM, we propose a circuit-level steering mechanism inspired by LLM steering methods (Rimsky et al., 2024) to reflect frequent updates. For the circuit, we

select the top-k most important heads based on importance scores for each layer. The next key idea is how to align structured, updated user-profile facts (represented as triples) with internal LLM representations to steer the model.

Figure 2 provides an overview of the pipeline and illustrates the update process. In the initial KG, the occupation of the person "Mike" is recorded as "Student", which is later updated to "Pilot". To incorporate this change, we compute the difference between the corresponding triples `(Mike, Job, Student)` and `(Mike, Job, Pilot)` (Equation 6). Then, we use the resulting difference to steer the outputs of the attention heads identified by our circuit analysis (Section 4.1).

4.2.1 KG–LLM ALIGNMENT MODULE

To incorporate updates to the personal knowledge graph into the LLM, we introduce a KG–LLM Alignment Module f_ϕ that aligns the KG and LLM representation spaces at two levels. **Triple-level alignment** targets a local update $(s_i, r_i, o_i^{\text{init}}) \rightarrow (s_i, r_i, o_i^{\text{upd}})$ for the target subject $i \in [n]$. **Subgraph-level alignment** captures broader structural changes in the neighborhood of the updated entity (Jiang et al., 2023), denoted as $\mathcal{G}(s_i)$, which is the induced subgraph centered at s_i, and directly accessible from the PKG.

The module takes two modalities as input. From the KG side, we use triple embeddings from a pretrained TransE (Bordes et al., 2013) and subgraph embeddings from an R-GCN (Schlichtkrull et al., 2018) encoder that pools $\mathcal{G}(s_i)$ into a unified subgraph embedding. From the LLM side, we serialize the triple and subgraph into text and encode them with the LLM.

The KG–LLM Alignment Module follows an attention-based structure, where textual representations serve as queries, and structural embeddings as keys and values:

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{H}_{\text{txt}}\mathbf{W}_{\mathbf{Q}}, \mathbf{H}_{\text{strc}}\mathbf{W}_{\mathbf{K}}, \mathbf{H}_{\text{strc}}\mathbf{W}_{\mathbf{V}}, \tag{1}$$

where $\mathbf{W}_{\mathbf{Q}} \in \mathbb{R}^{D_{\text{txt}} \times d}$ is a query projection matrix, and $\mathbf{W}_{\mathbf{K}}, \mathbf{W}_{\mathbf{V}} \in \mathbb{R}^{D_{\text{kge}} \times d}$ indicate the key and value projection matrices, respectively. D_{txt} and D_{kge} denote the embedding dimensions of the language model and KG embedding, respectively. d denotes the dimensionality of the query, key, and value representations. Here, $\mathbf{H}_{\text{txt}}$ and $\mathbf{H}_{\text{strc}}$ are defined as:

$$\begin{aligned} \mathbf{H}_{\text{txt}} &= [\mathbf{s}_{\text{txt}}, \mathbf{r}_{\text{txt}}, \mathbf{o}_{\text{txt}}, \mathcal{G}(s)_{\text{txt}}] \in \mathbb{R}^{4 \times D_{\text{txt}}}, \\ \mathbf{H}_{\text{strc}} &= [\mathbf{s}_{\text{kge}}, \mathbf{r}_{\text{kge}}, \mathbf{o}_{\text{kge}}, \mathcal{G}(s)_{\text{gnn}}] \in \mathbb{R}^{4 \times D_{\text{kge}}}. \end{aligned} \tag{2}$$

$\mathbf{s}_{\text{txt}}$, $\mathbf{r}_{\text{txt}}$, and $\mathbf{o}_{\text{txt}}$ correspond to the textual embeddings of the subject (s), relation (r), and object (o) of a given knowledge graph. When an entity or relation is tokenized into multiple tokens, the last token in the corresponding token span is used. In addition, $\mathcal{G}(s)_{\text{txt}}$ denotes the subgraph-level textual embedding of the subject entity s, computed in the same manner. On the KG side, $\mathbf{s}_{\text{kge}}$, $\mathbf{r}_{\text{kge}}$, and $\mathbf{o}_{\text{kge}}$ correspond to the KG embeddings of the triple, while $\mathcal{G}(s)_{\text{gnn}}$ denotes the GNN-encoded subgraph embedding. Accordingly, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{4 \times d}$.

The alignment process is then performed as follows:

$$\hat{\mathbf{H}}'_{\text{txt}} = \text{Softmax}(\mathbf{Q}\mathbf{K}^T/\sqrt{d})\mathbf{V} \in \mathbb{R}^{4 \times d}, \tag{3}$$

$$\hat{\mathbf{H}}_{\text{txt}} = \hat{\mathbf{H}}'_{\text{txt}}\mathbf{W}_{\text{up}}, \tag{4}$$

where $\mathbf{W}_{\text{up}} \in \mathbb{R}^{d \times D_{\text{out}}}$ denotes a linear transformation. For both the initial and updated KG settings, we obtain aligned textual representations $\hat{\mathbf{H}}^{\text{init}}_{\text{txt}}, \hat{\mathbf{H}}^{\text{upd}}_{\text{txt}} \in \mathbb{R}^{4 \times D_{\text{out}}}$. Accordingly, the aligned textual representations can be written as:

$$\hat{\mathbf{H}}^{\text{init}}_{\text{txt}} = f_\phi((s_i, r_i, o_i^{\text{init}})), \quad \hat{\mathbf{H}}^{\text{upd}}_{\text{txt}} = f_\phi((s_i, r_i, o_i^{\text{upd}})), \tag{5}$$

where f_ϕ denotes the KG–LLM Alignment Module. In both cases, the first embedding vector corresponds to the aligned representation of the updated entity in the textual space (i.e., $\hat{\mathbf{H}}^{\text{init}}_{\text{txt}}[0, :]$ and $\hat{\mathbf{H}}^{\text{upd}}_{\text{txt}}[0, :]$, respectively). These correspond to **Mike**^{init} and **Mike**^{upd} in Figure 2 when $(s_i, r_i, o_i^{\text{init}})$ corresponds to `(Mike, Job, Student)` and $(s_i, r_i, o_i^{\text{upd}})$ is `(Mike, Job, Pilot)`.

4.2.2 UPDATED KNOWLEDGE ADAPTATION VIA LLM STEERING

The representation difference is computed with $\hat{\mathbf{H}}^{\text{init}}_{\text{txt}}[0,:]$ and $\hat{\mathbf{H}}^{\text{upd}}_{\text{txt}}[0,:]$, and this difference is defined as:

$$\Delta = \sigma(\hat{\mathbf{H}}^{\text{upd}}_{\text{txt}}[0,:] - \hat{\mathbf{H}}^{\text{init}}_{\text{txt}}[0,:]) \in \mathbb{R}^{D_{\text{out}}}. \tag{6}$$

We then inject Δ into the output of the attention heads identified in Section 4.1. Here, σ denotes the sigmoid function. Importantly, knowledge injection is applied only at the heads previously identified as responsible for personal information processing (Section 4.1).

Let d_{head} denote the dimensionality of a single head in the LLM, and let H be the number of selected heads. The dimensionality of Δ is set to $D_{\text{out}} = d_{\text{head}} \times H$, where each segment is added to the corresponding head output. This design enables us to selectively control the LLM's internal activations, allowing the model to respond to updated information. At inference time, Δ is injected at every forward pass throughout both the prefill and decoding stages of generation, and is added uniformly across all token positions at the selected heads.

4.2.3 OPTIMIZATION

Given a pair of an initial triple $(s_i, r_i, o_i^{\text{init}})$ and an updated triple $(s_i, r_i, o_i^{\text{upd}})$, the negative log-likelihood loss is defined as follows:

$$\mathcal{L}_{\text{NLL}} = -\sum_i \sum_{t=1}^{|o_i^{\text{upd}}|} \log\ p\Big(o_{i,t}^{\text{upd}} \mid o_{i,<t}^{\text{upd}},\ \mathbf{G}_i\Big), \tag{7}$$

where $\mathbf{G}_i = g^{\text{init}}_{+\Delta}(\Phi(s_i, r_i))$. Here, $g^{\text{init}}_{+\Delta}$ is obtained by adding Δ to a target attention-head activation of the *init-LLM* (g^{init}). $|o_i^{\text{upd}}|$ denotes the token length of the updated tail (answer) text. The function $\Phi(\cdot)$ is a relation-specific template used to naturally format the input for the LLM (Appendix A.4).

We introduce a KL-divergence loss that penalizes changes to knowledge unrelated to the updated triple, defined as $\mathcal{L}_{\text{KL}} = D_{\text{KL}}\big(\mathrm{P}(g^{\text{init}}(s_i)) \,\|\, \mathrm{P}(g^{\text{init}}_{+\Delta}(s_i))\big)$, where P denotes a softmax function. Furthermore, to prevent the steering vector Δ from deviating excessively from the initial representation, we add a norm-based penalty $\mathcal{L}_{\text{norm}} = \|\Delta\|_2 / \|f_\phi((s_i, r_i, o_i^{\text{init}}))\|_2$.

Finally, the overall training objective is

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda_1 \cdot \mathcal{L}_{\text{KL}} + \lambda_2 \cdot \mathcal{L}_{\text{norm}}, \tag{8}$$

where λ_1 and λ_2 are scaling factors.

5 EXPERIMENT

5.1 EXPERIMENTAL SETUP

Our problem setting (Section 3.1) considers a scenario where an LLM, initially fine-tuned on $\mathbf{KG}^{\text{init}}$, is subsequently updated with $\mathbf{KG}^{\text{upd}}$. Since personal information is represented as triples, we design prompt templates for each relation type (e.g., `Medical_Condition`: "{Subject} has a medical condition of") and fine-tune the model to predict the tail entity given the head and relation. The resulting *init-LLM* is trained until its accuracy on all triples in $\mathbf{KG}^{\text{init}}$ exceeds 99%, ensuring that the initial personal knowledge is fully encoded.

Rather than maintaining a separate model per user, which is computationally prohibitive, *init-LLM* internalizes the collective personal knowledge of all users. At evaluation, we inject the changes for a single individual, measure the model's performance, and reload the original *init-LLM* before repeating the process for the next individual. For each individual, the KG–LLM Alignment Module is newly initialized and trained only on that individual's modified triples, while the init-LLM remains frozen. We note that updates are applied only to modified triples ($\mathcal{C}$); unchanged triples ($\mathcal{R}$) are not used again for training when the updates are injected.

Table 1: Performance on injecting updated personal information into *init-LLM*s. Metrics (%): **Acc.** (Accuracy), **Loc.** (Locality), **Ret.** (Retention), and **Tot.** (their harmonic mean). **Bold**/underline: best/second-best.

LLM	Method	PeaCoK-Ex				PerInfoKG				Athlete				ORCID			
		Acc.	**Loc.**	**Ret.**	**Tot.**	**Acc.**	**Loc.**	**Ret.**	**Tot.**	**Acc.**	**Loc.**	**Ret.**	**Tot.**	**Acc.**	**Loc.**	**Ret.**	**Tot.**
GPT-J (6B)	FT	**100.0**	50.6	75.1	69.6	**100.0**	84.3	78.3	86.6	**100.0**	44.8	86.1	68.3	96.7	39.7	89.2	64.2
	FT+C	**100.0**	46.0	74.6	66.5	**100.0**	70.7	89.1	84.8	**100.0**	36.5	86.3	61.2	93.8	99.5	89.7	94.1
	SC	97.0	97.0	76.0	88.8	75.3	76.0	39.8	58.2	**100.0**	99.7	75.8	90.3	99.1	99.3	72.0	88.1
	SC + 3-Shot	98.8	99.4	85.6	94.1	**100.0**	**100.0**	86.4	95.0	**100.0**	**100.0**	85.9	94.8	99.1	**99.8**	81.6	92.7
	LoRA	89.0	65.6	83.5	78.0	99.9	63.7	54.5	68.1	**100.0**	29.7	61.9	50.1	**100.0**	23.1	59.4	42.7
	ROME	93.9	88.9	85.9	89.4	62.0	70.1	90.4	72.3	91.8	53.1	89.2	73.3	47.3	29.2	90.2	45.2
	ICE	**100.0**	85.6	**88.2**	90.9	74.1	38.7	63.8	54.5	94.1	15.2	76.9	33.5	85.5	14.8	79.6	32.7
	LightEdit	92.1	**100.0**	87.5	92.9	84.2	32.7	90.5	56.1	90.2	8.5	89.0	21.4	82.1	11.4	**90.8**	27.1
	MEMIT-Merge	65.1	69.5	74.5	69.5	47.8	71.8	71.7	61.5	40.5	53.7	76.5	53.2	31.0	47.3	71.9	44.6
	AlphaEdit	98.8	99.8	75.2	89.7	26.1	42.0	36.3	33.4	63.1	20.9	82.7	39.6	3.0	12.4	76.9	7.0
	WISE	**100.0**	91.7	87.5	92.8	99.4	94.9	55.6	77.8	**100.0**	85.0	78.0	86.7	42.0	14.4	55.1	26.9
	SAKE	67.1	**100.0**	87.6	82.6	91.8	96.3	**91.0**	93.0	79.1	**100.0**	**89.5**	88.7	44.2	**99.8**	89.5	68.5
	Ours	**100.0**	**100.0**	86.5	**95.1**	99.2	98.4	89.2	**95.4**	**100.0**	98.7	87.3	**95.0**	98.9	98.0	89.1	**95.1**
QWEN2.5-7B	FT	**100.0**	34.1	93.1	60.0	**100.0**	67.7	93.6	84.6	**100.0**	41.3	91.4	66.4	97.1	36.5	86.2	60.9
	FT+C	95.7	53.8	80.1	72.3	90.1	95.2	91.0	92.1	84.6	95.7	89.4	89.7	65.4	97.5	85.8	80.6
	SC	85.4	86.8	50.9	70.0	95.3	98.8	26.3	51.2	76.1	84.8	60.0	72.1	97.1	95.4	68.1	84.6
	SC + 3-Shot	99.4	99.5	34.3	60.9	**100.0**	**100.0**	62.0	83.0	**100.0**	**100.0**	68.5	86.7	98.4	99.8	69.3	86.7
	LoRA	91.0	68.0	86.7	80.6	93.8	68.0	73.3	76.9	**100.0**	62.6	68.9	74.1	**100.0**	32.7	51.2	49.9
	ROME	81.7	86.6	**95.1**	87.5	50.7	51.8	**95.1**	60.6	85.3	29.4	91.9	53.0	49.3	16.7	**89.3**	32.8
	ICE	99.4	88.3	91.8	92.9	73.7	42.6	87.4	61.9	86.9	19.4	91.7	40.6	51.3	22.4	80.1	39.1
	LightEdit	98.9	**100.0**	92.0	96.8	95.1	22.7	95.0	46.1	97.4	1.6	**93.0**	4.6	89.1	11.0	87.5	26.3
	MEMIT-Merge	67.9	81.0	65.8	70.9	66.0	74.1	40.3	56.1	30.7	19.2	38.7	27.2	11.8	40.6	58.3	23.8
	AlphaEdit	98.8	**100.0**	85.6	94.3	18.4	77.6	90.3	38.3	31.7	31.0	85.6	39.7	15.0	16.5	71.6	21.2
	WISE	85.9	96.2	91.8	91.2	94.1	91.7	55.3	75.7	77.1	49.8	87.0	67.3	28.8	31.3	50.8	34.7
	SAKE	64.6	**100.0**	92.5	82.7	97.9	96.2	95.0	**96.3**	72.9	**100.0**	**93.0**	87.0	8.9	**100.0**	80.5	22.3
	Ours	**100.0**	**100.0**	92.8	**97.5**	99.4	96.8	91.9	95.9	99.0	87.8	89.2	**91.7**	95.3	98.2	87.5	**93.4**

5.1.1 Dataset and Evaluation Setup

We evaluate our method on the four datasets described in Section 3.2: PeaCoK-Ex, PerInfoKG, Athlete, and ORCID. We report three metrics: Accuracy, Locality, and Retention. Accuracy measures whether the ground-truth tokens corresponding to the updated personal information are fully contained in the generated tokens (Baek et al., 2023). Locality measures whether the ground-truth tokens corresponding to the unchanged personal information are fully contained in the generated tokens. Retention measures whether the ground-truth tokens for general factual knowledge from `known_1000` (Meng et al., 2022), which the *init-LLM* already knows, are fully contained in the generated tokens. Further details of the evaluation protocol are provided in Appendix A.5.

5.1.2 Baselines

We compare our approach with several representative baselines, including (i) full-model fine-tuning (FT), (ii) circuit-selective fine-tuning (FT+C), which fine-tunes only the personal-knowledge circuit identified in Section 4.1, (iii) sufficient context (SC) (Joren et al., 2025), which assumes perfect retrieval and provides only the information necessary to answer each query as context, (iv) sufficient context with three-shot prompting (SC+3-Shot), which additionally provides three query-answer demonstration examples along with the sufficient context, (v) LoRA (Hu et al., 2022), and (vi) knowledge editing methods (ROME (Meng et al., 2022), ICE (Qi et al., 2025), LightEdit (Jung et al., 2026), MEMIT-Merge (Dong et al., 2025), AlphaEdit (Fang et al., 2025), and WISE (Wang et al., 2024a)). We also include a steering-based approach, SAKE (Scialanga et al., 2025). The details of the baselines are provided in Appendix A.6.

5.2 Main Results

The performance on four datasets (PeaCoK-Ex, PerInfoKG, Athlete, ORCID) is presented in Table 1. Each dataset exhibits distinct characteristics: the number of updated triples per subject is fixed to 1 in PeaCoK-Ex, whereas it is up to 17 (16.98 on average) in PerInfoKG and 2 in Athlete,

and varies across individuals in ORCID (ranging from 1 to 4). Consequently, as shown in the table, most baselines achieve relatively high accuracy on PeaCoK-Ex. In contrast, certain approaches, such as fine-tuning and LoRA, demonstrate weaker performance in terms of Locality: updates intended for the changed facts also alter the model's answers to unchanged personal facts. In contrast, SC attains high Locality but low Retention, since the personal context is prepended even to general factual queries, where such irrelevant context distracts the model (Shi et al., 2023).

Multi-update scenarios (e.g., PerInfoKG) expose a more specific limitation of existing editing models. *Locate-and-edit* baselines (ROME, AlphaEdit) struggle to maintain Accuracy under dense multi-fact updates, and in some cases sacrifice Locality as well (e.g., AlphaEdit on GPT-J), plausibly due to the *key collision* (Dong et al., 2025) issue that arises when editing multiple facts for the same subject simultaneously.

FT+C is also accurate but exhibits large variance across models and datasets, indicating that circuit-only supervision is unstable and requires a more principled approach. WISE retains general knowledge well on PeaCoK-Ex but fails to do so on the more complex PerInfoKG dataset, suggesting that its routing thresholds are not calibrated for the multi-fact-per-subject setting. ICE likewise performs reasonably well in the simple single-fact setting of PeaCoK-Ex, yet degrades substantially on PerInfoKG; as noted in Huang et al. (2025), its locality and retention are sensitive to the domain of the edited knowledge, and PerInfoKG spans many heterogeneous relations across diverse semantic domains.

Unlike conventional knowledge editing benchmarks, ORCID contains targets composed of long token sequences, such as user homepage URLs and publication titles. Under this setting, methods that directly modify model weights, such as through full fine-tuning, LoRA, or locate-and-edit approaches, suffer substantial performance degradation. Although SAKE performs competitively on the other datasets, its Accuracy drops sharply on ORCID. In contrast, both the sufficient-context baseline and our method remain robust to this increase in target length. Notably, even among fine-tuning-based approaches, FT+C, which updates only the previously identified personal-knowledge circuit rather than the full parameter set, defends against this degradation considerably better than full fine-tuning (FT).

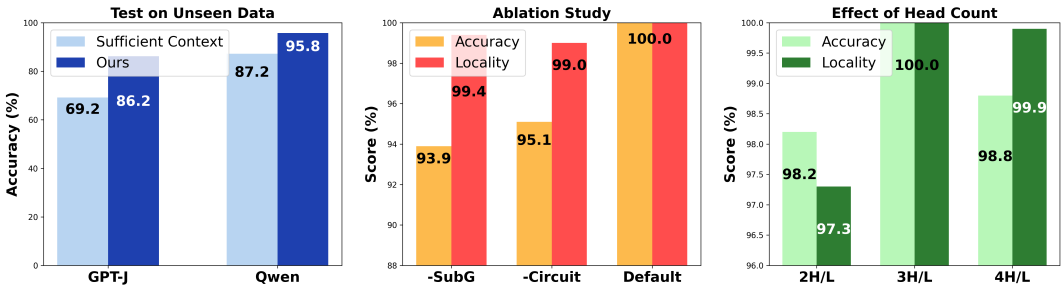
In contrast, our method does not exhibit the sharp degradation on any single metric observed in the aforementioned baselines. It remains competitive on Accuracy, Locality, and Retention simultaneously. The preserved Retention is aligned with Figures 1c–1e: since circuits associated with personal knowledge differ from those involved in general factual knowledge, applying our method only to the former results in a comparatively limited impact on general knowledge (`known_1000`).

In general, the results of Athlete show a trend similar to those of PerInfoKG, where both datasets consist of multi-fact updates. Our method attains the best Total score on both backbones, with the most balanced Accuracy–Locality trade-off among the parameter-updating and editing baselines on the Athlete dataset. Results on other LLMs can also be found in Appendix A.9.

5.3 Extended Experiments

Unseen Triples. To assess the generalization ability of our approach, we test whether it can apply updates to triples outside its training supervision. The idea is that once trained on multiple (initial, updated) triple pairs, the method should be able to inject new updates into the LLM for unseen triples (i.e., triples whose constituent entities and relations were seen during training, but not in that specific triplet combination). Among the 33,952 pairs corresponding to actually changed facts in the PerInfoKG dataset, 32,952 pairs are used to train the KG–LLM Alignment Module (shared across all individuals), while the remaining pairs are split into validation and test sets, with 500 pairs each. As shown in Figure 3a, our method achieves accuracies of 86.2% and 95.8% on GPT-J and Qwen, respectively, demonstrating strong generalization. Notably, these results outperform the sufficient-context setting (Joren et al., 2025) with KAPING prompting (Baek et al., 2023), which attains accuracies of 69.2% and 87.2%.

Attention vs. FFN. To verify that targeting attention heads is more effective than targeting FFNs when incorporating evolving personal information in LLMs, we apply our SPIKE method to FFNs and compare it against the original attention-head variant in Table 2. For both GPT-J and Qwen2.5-7B-Instruct, we add the steering vector to the output of the single FFN layer with the highest im-



(a) Evaluation on unseen test data using GPT-J and Qwen.

(b) Ablation study on GPT-J.

(c) Effect of head number on GPT-J.

Figure 3: (a) Evaluation of SPIKE against sufficient context on GPT-J and Qwen2.5-7B-Instruct using the unseen test data of PerInfoKG. (b) Ablation study on PeaCoK-Ex: `-SubG` denotes SPIKE without subgraph representations ($\mathcal{G}(s)_{\text{txt}}, \mathcal{G}(s)_{\text{gnn}}$), and `-Circuit` selects *low-importance heads*. (c) Experiments varying the number of heads per layer (H/L).

Table 2: Results of applying SPIKE to FFN and Attention (Attn), respectively.

Target	PeaCoK-Ex				PerInfoKG				ORCID			
	Acc.	**Loc.**	**Ret.**	**Tot.**	**Acc.**	**Loc.**	**Ret.**	**Tot.**	**Acc.**	**Loc.**	**Ret.**	**Tot.**
FFN (GPT-J)	90.9	99.9	75.2	87.4	99.1	96.9	79.2	90.8	**99.3**	**98.6**	88.9	**95.4**
Attn (GPT-J)	**100.0**	**100.0**	**86.5**	**95.1**	**99.2**	**98.4**	**89.2**	**95.4**	98.9	98.0	**89.1**	95.1
FFN (Qwen)	94.5	**100.0**	85.2	92.8	**99.7**	87.8	81.7	89.1	**96.2**	**98.2**	85.9	93.1
Attn (Qwen)	**100.0**	**100.0**	**92.8**	**97.5**	99.4	**96.8**	**91.9**	**95.9**	95.3	**98.2**	**87.5**	**93.4**

portance score. Consistent with the trend observed in Figures 1a and 1b, applying SPIKE to FFNs yields a lower retention rate of general knowledge than applying it to attention heads, resulting in lower total scores than the attention-based variant across most settings.

Ablation Study. We evaluate the effect of circuits by replacing important heads with low-importance ones (i.e., heads with the lowest importance scores across layers). We also assess the contribution of subgraph representations by removing subgraph features from both the KG ($\mathcal{G}(s)_{\text{gnn}}$) and the text ($\mathcal{G}(s)_{\text{txt}}$) (Figure 3b). Both ablations lead to performance degradation, with subgraph removal causing a larger drop. This suggests that subgraph information captures the local structural context surrounding the updated entity beyond what can be obtained from triple-level updates alone.

Head Count. We next vary the number of heads per layer that constitute the circuit. Using three or four heads yields better performance than using only two, although four does not consistently outperform three (Figure 3c). Interestingly, locality degrades when the circuit contains only two heads. A possible explanation is that, with fewer heads in the circuit, the required intervention for each head increases, which in turn amplifies unintended side effects, particularly the reduction of locality.

6 Conclusion

In this work, we introduce a new setting for LLM personalization, where mutable personal knowledge in knowledge graphs must be reflected in the model. We define a fact-level personalization task and propose a circuit-level steering method that, unlike fine-tuning or editing approaches reliant on pre-cached representations, integrates updates while preserving unchanged personal facts. Our experiments show strong performance, demonstrating effective personalization with minimal forgetting.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation and to implement methods for our experiments. With the exception of one supplementary experiment described below, we have not used generative AI tools for generating synthetic data, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, proposing or refining hypotheses, designing research methodology or experiments, cleaning or reformatting datasets, qualitative data analysis, or interpreting results. The exception is the unstructured knowledge update experiment in Appendix A.9 (Table 8), where we used an LLM API (GPT-4o as of Jan 2026) to convert PerInfoKG profiles into synthetic unstructured personal documents, following the AKEW benchmark (Wu et al., 2024). Additionally, we used a large language model (LLM) to edit the manuscript for readability, i.e., to check grammar, word choice, and stylistic consistency. We have reviewed all AI-assisted work. Specifically, the research ideas, experimental design, and analysis were developed entirely by the authors; the LLM was used to translate author-written drafts into English, and all translated text was checked by the authors to ensure that it faithfully conveys the intended meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All authors have read and will adhere to the ICLR Code of Ethics. Our experiments use four personal-knowledge datasets. PeaCoK-Ex and PerInfoKG are synthetic and comprise fictitious individuals and knowledge graph triples. Athlete and ORCID contain information about real individuals and are derived from publicly available sources, a public Kaggle dataset and the ORCID Annual Public Data Files, respectively; we used them in accordance with the terms of use of each source. We did not collect any private or sensitive personal information, and no interaction with human subjects was involved. All datasets and backbone models were used in accordance with their respective licenses. Our code and synthetic datasets are available via the anonymous repository linked in the abstract; to avoid redistributing personal information, we do not release Athlete and ORCID, which can be obtained from their original public sources.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we release our source code and the synthetic datasets (PeaCoK-Ex and PerInfoKG) via the anonymous repository linked in the abstract. The full description of SPIKE, including the attention head selection and steering procedure, is provided in Section 4, and implementation details such as backbone models, hyperparameters, and compute resources are reported in Appendix A.10. The construction and preprocessing of all four datasets are described in Appendix A.3. Since Athlete and ORCID contain information about real individuals, we do not redistribute them; instead, they can be obtained from their original public sources (a public Kaggle dataset and the ORCID Annual Public Data Files). Evaluation metrics and protocols are detailed in Appendix A.5.

REFERENCES

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pp. 78–106, Toronto, Canada, June 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlrse-1.7. URL https://aclanthology.org/2023.nlrse-1.7/.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013. URL https://papers.nips.cc/paper_files/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

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

16352, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Zilu Dong, Xiangqing Shen, and Rui Xia. MEMIT-Merge: Addressing MEMIT's key-value conflicts in same-subject batch editing for LLMs. In *Findings of the Association for Computational Linguistics*, pp. 7952–7960. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.415/.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 1932–1945, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.106. URL https://aclanthology.org/2024.acl-long.106/.

Ritam Dutt, Kasturi Bhattacharjee, Rashmi Gangadharaiah, Dan Roth, and Carolyn P. Rosé. PerKGQA: Question answering over personalized knowledge graphs. In *Findings of the Association for Computational Linguistics: NAACL*, pp. 253–268. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.FINDINGS-NAACL.19. URL https://doi.org/10.18653/v1/2022.findings-naacl.19.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained knowledge editing for language models. In *Proceedings of the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HvSytvg3Jh.

Silin Gao, Beatriz Borges, Soyoung Oh, Deniz Bayazit, Saya Kanno, Hiromi Wakaki, Yuki Mitsufuji, and Antoine Bosselut. PeaCoK: Persona commonsense knowledge for consistent and engaging narratives. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 6569–6591. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.362. URL https://doi.org/10.18653/v1/2023.acl-long.362.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL*, pp. 15202–15232, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.902. URL https://aclanthology.org/2024.findings-acl.902/.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James R. Glass, and Fuchun Peng. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pp. 1121–1133. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EACL-MAIN.95. URL https://doi.org/10.18653/v1/2021.eacl-main.95.

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. In *Proceedings of the International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. Can knowledge editing really correct hallucinations? In *Proceedings of the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hmDt068MoZ.

Hang Jiang, Xiajie Zhang, Xubo Cao, Cynthia Breazeal, Deb Roy, and Jad Kabbara. PersonaLLM: Investigating the ability of large language models to express personality traits. In *Findings of the Association for Computational Linguistics: NAACL*, pp. 3605–3627. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.229. URL https://doi.org/10.18653/v1/2024.findings-naacl.229.

Jinhao Jiang, Kun Zhou, Xin Zhao, Yaliang Li, and Ji-Rong Wen. ReasoningLM: Enabling structural subgraph reasoning in pre-trained language models for question answering over knowledge graph. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3721–3735. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.228. URL https://aclanthology.org/2023.emnlp-main.228/.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient context: A new lens on retrieval augmented generation systems. In *Proceedings of the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Jjr2Odj8DJ.

Dahyun Jung, Jaewook Lee, and Heuiseok Lim. Towards scalable lifelong knowledge editing with selective knowledge suppression. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 3215–3231. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.145. URL https://doi.org/10.18653/v1/2026.acl-long.145.

Wenkai Li, Jiarui Liu, Andy Liu, Xuhui Zhou, Mona T. Diab, and Maarten Sap. BIG5-CHAT: shaping LLM personalities through training on human-grounded data. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 20434–20471. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.999/.

Jiongnan Liu, Yutao Zhu, Shuting Wang, Xiaochi Wei, Erxue Min, Yu Lu, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. LLMs + Persona-Plug = Personalized LLMs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 9373–9385. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.461/.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Mistral AI. Mistral-7B-Instruct-v0.3. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3, 2024.

ORCID. ORCID public data file 2013, 2019. URL https://doi.org/10.14454/07243.2013.001.

ORCID. ORCID public data file 2014, 2025. URL https://doi.org/10.14454/07243.2014.001.

Petro. Tokyo 2020 Olympics dataset. https://www.kaggle.com/datasets/piterfm/tokyo-2020-olympics, 2021. Kaggle dataset, accessed: 2026-05-20.

Deeksha Prahlad, Chanhee Lee, Dongha Kim, and Hokeun Kim. Personalizing large language models using retrieval augmented generation and knowledge graph. In *Companion Proceedings of the ACM on Web Conference*, pp. 1259–1263. ACM, 2025. doi: 10.1145/3701716.3715473. URL https://doi.org/10.1145/3701716.3715473.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. In-context editing: Learning knowledge from self-induced distributions. In *Proceedings of the International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=w6rHCuN3YG.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.828. URL https://doi.org/10.18653/v1/2024.acl-long.828.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 7370–7392. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.399. URL https://doi.org/10.18653/v1/2024.acl-long.399.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3027–3035. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33013027. URL https://doi.org/10.1609/aaai.v33i01.33013027.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *Proceedings of the Semantic Web*, volume 10843, pp. 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. URL https://doi.org/10.1007/978-3-319-93417-4_38.

Marco Scialanga, Thibault Laugel, Vincent Grari, and Marcin Detyniecki. SAKE: Steering activations for knowledge editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15966–15978, 2025. URL https://aclanthology.org/2025.acl-long.777.pdf.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

Jingwei Sun, Zhixu Du, and Yiran Chen. Knowledge graph tuning: Real-time large language model personalization based on human feedback. *arXiv preprint arXiv:2405.19686*, 2024.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416, 2024. URL https://aclanthology.org/2024.blackboxnlp-1.25/.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 billion parameter autoregressive language model, 2021.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/60960ad78868fce5c165295fbd895060-Abstract-Conference.html.

Xuehao Wang, Liyuan Wang, Binghuai Lin, and Yu Zhang. HeadMap: Locating and enhancing knowledge circuits in llms. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/pdf?id=jUsrbOuQ5e.

Zheng Wang, Zhongyang Li, Zeren Jiang, Dandan Tu, and Wei Shi. Crafting personalized agents through retrieval-augmented generation on editable memory graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4891–4906. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.EMNLP-MAIN.281. URL https://doi.org/10.18653/v1/2024.emnlp-main.281.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. AKEW: Assessing knowledge editing in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15118–15133. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.843. URL https://aclanthology.org/2024.emnlp-main.843/.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. doi: 10.48550/ARXIV.2412.15115. URL https://doi.org/10.48550/arXiv.2412.15115.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. Knowledge circuits in pretrained transformers. In *Advances in Neural Information Processing Systems*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/d6df31b1be98e04be48af8bedb95b499-Abstract-Conference.html.

Jinghao Zhang, Yuting Liu, Wenjie Wang, Qiang Liu, Shu Wu, Liang Wang, and Tat-Seng Chua. Personalized text generation with contrastive activation steering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 7128–7141. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.acl-long.353/.

Mengqi Zhang, Zisheng Zhou, Xiaotian Ye, Qiang Liu, Zhaochun Ren, Zhumin Chen, and Pengjie Ren. Disentangling knowledge representations for large language model editing. *CoRR*, abs/2505.18774, 2025b. doi: 10.48550/ARXIV.2505.18774. URL https://doi.org/10.48550/arXiv.2505.18774.

A APPENDIX

A.1 LIMITATIONS

While SPIKE demonstrates strong performance on fact-level personalization, several aspects remain open for future investigation. First, SPIKE operates on structured triples; although our unstructured setting (Appendix A.9) shows it can be combined with LLM-based triple extraction, directly handling unstructured personal data, such as long-form conversations or free-form notes, remains a promising direction. Second, our experiments are conducted on English datasets and on 6–8B-parameter backbones (GPT-J, Qwen2.5, LLaMA3, Mistral). Broader validation across multilingual corpora and larger-scale models would further establish the generality of the approach.

A.2 INSTABILITY OF KEY REPRESENTATIONS IN MODEL EDITING FOR PERSONAL ENTITIES

Methods like ROME (Meng et al., 2022) compute the MLP key vector k_* for a given subject s by averaging activations across N random prefix contexts x_j, as follows:

$$k_* = \frac{1}{N} \sum_{j=1}^{N} k(x_j + s), \tag{9}$$

where $k(x)$ represents the key vector computed at the target layer l^* and the last token index i of the subject s within the prompt:

$$k(x) = \sigma \left(W_{\text{fc}}^{(l^*)} \gamma \left(a_{[x],i}^{(l^*)} + h_{[x],i}^{(l^*-1)} \right) \right). \tag{10}$$

Here, $h_{[x],i}^{(l^*-1)}$ is the hidden state vector at layer $(l^* - 1)$ for the i-th token of input x, $a_{[x],i}^{(l^*)}$ is the attention output contribution at layer l^* of input x, and $\gamma(\cdot)$ denotes the normalizing nonlinearity. The matrix $W_{\text{fc}}^{(l^*)}$ is the first linear transformation weight in the MLP layer, $\sigma(\cdot)$ is the non-linear activation function.

The implicit assumption behind this averaging procedure is that the resulting k_* is a stable and representative key for subject s. That is, the per-context keys $k(x_j + s)$ do not vary dramatically across different prefixes. This assumption may hold reasonably well for famous entities that appear extensively in pretraining data, as the model is likely to have formed a consistent internal representation for them in the MLP key space. However, for personal entities whose information is absent from pretraining and introduced only through fine-tuning on their PKG, it is less clear whether this stability holds to the same degree.

To verify this, we conducted the following experiment on *init–LLM*, which was trained on personal entity knowledge. For both famous entities (Wikipedia-known individuals) and personal entities (PKG individuals), we computed the intra-entity key variance following ROME's exact k_* computation procedure: for each entity s, we sampled $N = 50$ random prefix texts x_j and computed $k(x_j + s)$ at each target layer, then measured the variance of these key vectors around their mean k_*.

The results show that personal entities exhibit consistently higher intra-entity key variance than famous entities across all measured layers, with an average ratio of approximately $1.6\times$. Critically, the gap is most pronounced at middle layers that ROME identifies as playing an important role in knowledge storage. For personal entities, the key vector k_* computed by ROME is a poor representative of the actual per-context keys, as those keys are unstable across different input contexts.

A.3 DATASET CONSTRUCTION & STATISTICS

PeaCoK-Ex is developed as an extended personal-knowledge version of PeaCoK to construct a personalized knowledge graph suitable for temporal update evaluation. PeaCoK Gao et al. (2023), which itself extends commonsense KGs such as ATOMIC Sap et al. (2019), provides a rich set of relations describing personal attributes (`experience`, `routine_habit`, `characteristics`, `goal_plan`) along with their social counterparts (`relationship_experience`, `relationship_routine_habit`, `relationship_characteristics`, `relationship_goal_pla`

n). However, the original PeaCoK graph does not explicitly contain person entities or individualized personal information. To address this limitation, PeaCoK-Ex is constructed through a three-stage transformation pipeline.

(1) Refinement of the Raw PeaCoK KG. The construction process begins by extracting profession-related information from subjects in the raw PeaCoK KG, many of which contain natural-language descriptions such as "I am a X who ..." or "I am a Y."

Profession Extraction. Regular-expression patterns are applied to identify embedded job descriptions. Subjects matching: "I am a {profession} who ...", or "I am a {profession}" are mapped to their canonical profession label, resulting in a set of unique professions and a mapping from verbose subject strings to standardized profession entities.

Knowledge Graph Reformatting. Each triple whose subject contains a profession description is rewritten by replacing the natural-language subject with its extracted profession label. Triples in which the rewritten subject and object collide (e.g., profession = object), or cases where the object is itself a profession, are removed to avoid semantic inconsistencies. The outcome is a cleaned, profession-centric KG that provides a consistent schema for downstream construction.

(2) Introduction of Synthetic People and Profession Assignment. To convert the profession-centric KG into a personalized one, we introduce synthetic individuals.

Person Entity Generation. For each extracted profession, one synthetic person entity (e.g., *Frances Travis*) is created, producing as many individuals as there are professions (822 persons in total).

Deterministic Person–Profession Linking. Each synthetic person is deterministically linked to a unique profession via the triple:

$$\langle \text{Person}_i,\ \texttt{has_a_job_of},\ \text{Profession}_j \rangle.$$

This guarantees one-to-one person–profession assignments and converts the graph into a person-centric structure encoding explicit occupational information.

(3) Reverse-Relation Augmentation. To support bidirectional reasoning, the final stage augments the KG with reverse relations.

Reverse Relation Construction. For each relation r, a reverse relation r^{-1} is defined. Every triple (h, r, t) is expanded into both (h, r, t) and (t, r^{-1}, h). Examples include:

- `has_a_job_of` $\rightarrow$ `is_a_job_of`
- `characteristic` $\rightarrow$ `is_a_characteristic_of`
- `is_experience_of` $\rightarrow$ `has_experience_of`

This approximately doubles the triple count and ensures that relational information is navigable in both directions.

After applying the pipeline, PeaCoK-Ex contains 822 synthetic person entities, each associated with exactly one profession, yielding 1,644 person–job triples (including reverse relations), along with a large number of job-related attribute triples inherited from the PeaCoK schema. The resulting KG includes 105,258 triples, 49,818 entities, and 18 relation types (counting reverse relations). Table 3 summarizes the key statistics of the dataset.

We treat this graph as the initial snapshot $\mathbf{KG}^{\text{init}}$. To simulate temporal evolution, we generate $\mathbf{KG}^{\text{upd}}$ by modifying 20% of the person–occupation pairs while keeping all other attributes unchanged. This establishes a realistic temporal-update evaluation setting where only a portion of personal information changes over time.

PerInfoKG is a synthetic dataset constructed over 2,000 fictitious individuals and 23 personal information fields. Each field and its corresponding candidate and possible probability weight are shown in Table 16. The probability weight is determined based on real-world statistics. For each individual, we partition the 23 fields into 17 *mutable* attributes used for editing and 6 *immutable* attributes reserved for evaluating locality, ensuring that every individual contributes to both edit

and locality evaluation. Importantly, to reflect a real-world setting, we blocked some cases, such as a subject changing the education level from 'PhD' to 'middle school'. The dataset contains 2,134 entities and 46,000 triples (92,000 including reverse relations) at $\mathbf{KG}^{\text{init}}$, with 33,952 update instances used to derive $\mathbf{KG}^{\text{upd}}$. We split these instances into 32,952 for training and 500 each for validation and test. This split is designed to evaluate the model's capacity to generalize to *unseen updated triples*, i.e., new user-specific facts that were not observed during training, thereby testing the adaptability of our KG–LLM Alignment Module.

To provide a more complete description of how PerInfoKG is constructed, we now detail the underlying two-stage generation pipeline comprising (1) initial profile generation and (2) rule-based temporal updates.

(1) Initial Profile Generation ($\mathbf{KG}^{\text{init}}$)

Attribute Space and Sampling Distributions. We define a discrete attribute space covering 23 personal fields (plus a name identifier), grouped as follows:

- **Demographics**: `gender`, `race_ethnicity`, `age_group`, `nationality`
- **Location and Living**: `housing_type`, `address`, `commuting_means`
- **Health and Lifestyle**: `blood_type`, `medical_condition`, `exercise_frequency`, `diet_type`, `drinking_frequency`
- **Social / Personal**: `religion`, `marital_status`, `pets_owned`, `hobby`, `favorite_food`, `favorite_music`
- **Socioeconomic / Education**: `education_level`, `major`, `job`, `phone_model`, `political_affiliation`

Each field is accompanied by a categorical value set and, when applicable, a weight vector reflecting realistic demographic tendencies.

Name List Construction. We prepare a list of 2,000 unique names, which serve as identifiers for each fictitious individual.

Consistent Sampling of Education, Major, and Job. To enforce logical coherence across dependent attributes:

- Education level is sampled first.
- If the education level corresponds to higher education (bachelor's, master's, or PhD), a major is sampled from non-None_MAJ categories; otherwise, the major is fixed to `None_MAJ`.
- Given the assigned major, the job is sampled from the predefined mapping `major_to_jobs`, which lists occupations compatible with each major.

Sampling of Remaining Attributes. All remaining fields (e.g., `religion`, `address`, `political_ affiliation`, `hobby`, `medical_conditions`, `marital_ status`, `drinking_frequency`) are independently sampled based on their categorical distributions. The resulting profiles form the initial snapshot $\mathbf{KG}^{\text{init}}$.

(2) Rule-Based Temporal Updates ($\mathbf{KG}^{\text{upd}}$)

Mutable Fields. A designated set of 17 mutable fields is defined: `address`, `phone_model`, `pets_owned`, `medical_conditions`, `education_level`,

Table 3: Statistics of the extended PeaCoK-Ex dataset.

#Entities	49,818
#Relations	18 (including reverse)
#Triples	105,258
#Synthetic person entities	822
#Person–job triples	1,644
#Updated persons	164

hobby, political_affiliation, job, housing_type, commuting_means, exercise_frequency, major, favorite_food, favorite_music, diet_type, marital_status, drinking_frequency. All other fields (i.e., gender, nationality, religion, blood_type, race_ethnicity, age_group) remain fixed across time (the value of age_group represents birth-year groupings (e.g., "1990s"), and is therefore treated as immutable).

Education-Level Progression. Education level is updated according to:

- If a user's education level is not PhD, the level is advanced to the next tier.
- If the prior major was None_MAJ but the updated level enters higher-education tiers, a new major is sampled from the non-None_MAJ set.

Job Update Constrained by Updated Major and Education. Given the updated major and education levels:

- Candidate jobs are retrieved from major_to_jobs[major].
- Degree requirements are enforced (e.g., "scientist" requires a master's degree; "professor" requires a PhD).
- A new job is selected, distinct from the previous one.

Updates for All Other Mutable Fields. For each remaining mutable field, a new value is uniformly sampled from the remaining options, excluding the original value.

Because each updated profile is generated deterministically from its initial version—with logical constraints, controlled randomness, and aligned field dependencies—$\mathbf{KG}^{\text{init}}$ and $\mathbf{KG}^{\text{upd}}$ together form a clean two-time-step benchmark suitable for evaluating temporal personal-information updates, locality preservation, and knowledge-update behavior in LLMs.

Athlete is generated from publicly available Kaggle tabular records of athletes who participated in the 2020 Tokyo Olympics (11,656 rows) (Petro, 2021), we extract seven attributes per athlete: gender, birth_date, birth_place, birth_country, discipline, residence_place, and height. The athlete's name is used solely as an entity identifier and is not an editable attribute. Height values, originally given in mixed metric/imperial notation (e.g., 1.98/6′5′′), are normalized to the metric value only, and all name and place strings are whitespace-normalized and cast to title case. Since incomplete records would yield ill-formed prompt–answer pairs for both knowledge-graph construction and edit evaluation, we discard any athlete with a missing value in any of the seven attributes. This filtering step leaves 2,084 athletes with fully observed attributes.

(1) Filtering Facts Already Known to Pretrained LMs. Because Athlete is used to evaluate knowledge editing, facts already recoverable by the base language model prior to editing must be excluded from the edit targets. For each of the 2,084 athletes, we construct a closed-prompt completion for every attribute (e.g., *"{name}'s gender is"*, *"{name}'s sport discipline is"*, *"{name}'s residence place is"*) and query Qwen2.5-7B-Instruct (Yang et al., 2024) and GPT-J 6B (Wang & Komatsuzaki, 2021) under greedy decoding with up to 20 generated tokens. An attribute is judged as already known if the normalized ground-truth string appears as a substring of the generated text, and an athlete is considered "unknown" to a given model only if every one of the seven attributes is judged unknown. Applying this criterion, 186 athletes are unknown to Qwen2.5-7B-Instruct and 1,304 are unknown to GPT-J 6B; taking the intersection of athletes unknown to both models yields the final set of 153 athletes retained in Athlete.

Table 4: Statistics of the PerInfoKG dataset.

#Entities	2,134
#Relations	46 (including reverse)
#Triples	92,000
#Synthetic person entities	2,000
#Updated persons	2,000

(2) Synthetic Update Construction and Knowledge Graph. Using the 153 retained athletes' profiles as $\mathbf{KG}^{\text{init}}$, we construct $\mathbf{KG}^{\text{upd}}$ by resampling only the two editable attributes, `discipline` and `residence_place`, for every athlete: each is replaced with a value drawn at random from a fixed candidate pool, subject to the constraint that the sampled value differs from the corresponding $\mathbf{KG}^{\text{init}}$ value. The remaining five attributes (`gender`, `birth_date`, `birth_place`, `birth_country`, `height`) are kept identical to $\mathbf{KG}^{\text{init}}$. This yields 306 edit instances (153×2). Each snapshot is materialized as a knowledge graph of (athlete, attribute, value) triples, with each of the seven forward relations paired with a corresponding inverse relation (e.g., `discipline` / `discipline_of`) and a fixed natural-language verbalization template (e.g., *"plays the sport of"*, *"resides in"*, *"has the homeland of"*); entity and relation identifiers assigned when constructing $\mathbf{KG}^{\text{init}}$ are reused unchanged in $\mathbf{KG}^{\text{upd}}$. The resulting knowledge graph contains 153 athlete entities and 614 entities in total, 14 relations (7 forward and 7 inverse), 1,071 forward triples per snapshot (153×7), and 2,142 triples per snapshot including inverse relations.

(3) Locality Evaluation Set. To evaluate whether editing leaves unrelated knowledge intact, we construct a locality set from the attributes that are identical in $\mathbf{KG}^{\text{init}}$ and $\mathbf{KG}^{\text{upd}}$. Each of the 153 athletes is edited on exactly two attributes, `discipline` and `residence_place`; the other five—`gender`, `birth_date`, `birth_place`, `birth_country`, and `height`—remain fixed by design. Every athlete therefore contributes five locality prompt–answer pairs, giving 765 pairs in total (153×5). After editing an athlete, we query all five of that athlete's unchanged attributes and count a prediction as correct when the generation contains the original answer. The reported locality score is the accuracy over these 765 pairs. Because every athlete has the same two edited attributes and the same five unchanged ones, both the number and the identity of the edited facts are controlled by design.

ORCID is a publicly available academic registry in which researchers self-report identifying information, and each researcher may update their profile over time. We define two temporal snapshots, $\mathbf{KG}^{\text{init}}$ (2013) and $\mathbf{KG}^{\text{upd}}$ (2014), for the same set of 883 researchers, and use facts that actually changed between the two snapshots as edit targets. Five personal information fields are used: `Country`, `LatestWorkTitle`, `WorkCount`, `HomepageName`, and `HomepageAddress`.

(1) Source Data and Subject Selection. The raw data are the ORCID Annual Public Data Files for 2013 and 2014 (ORCID, 2019; 2025). We retain only researchers whose profiles exist and are successfully parsed in both years, and further apply the following four selection criteria:

1. The researcher's credit name must be present in both 2013 and 2014. This name is used solely as an entity identifier within the dataset; whether the name itself changed is not part of the selection criteria.
2. The name of the first entry in the researcher's list of registered homepages (researcher URLs) must be present in both years.
3. At least one of `Country`, `LatestWorkTitle`, or `HomepageName` must have a value in both years, and that raw text value must differ between the two years.
4. Cases where only `WorkCount` or `HomepageAddress` changed, where a field's value appears in only one of the two years (i.e., appears or disappears), or where all three fields in Criterion 3 are identical across years, are excluded.

In addition, since ORCID is used to evaluate knowledge editing, facts already known to the base language model prior to editing must be excluded from the edit targets. For each researcher satisfying the four criteria, we construct a relation-querying prompt for every field except the name

Table 5: Statistics of the Athlete dataset.

#Entities	614
#Relations	14 (including reverse)
#Triples	2,142
#Athlete entities	153
#Updated athletes	153

(e.g., *"Ali Ahmad is based in"*) and query two language models, Qwen2.5-7B-Instruct (Yang et al., 2024) and GPT-J 6B (Wang & Komatsuzaki, 2021). Each model greedily generates up to 20 tokens, and a fact is judged as already known if the ground-truth string appears verbatim within the generated text. For fields whose ground-truth values may exceed 20 tokens (e.g., `LatestWorkTitle`, `HomepageAddress`), only the first 20 tokens of the ground truth are used for this containment check. If either model correctly produces the ground truth for even one of the five fields for a given individual — including incidental substring matches within unrelated numbers — that individual is entirely removed from the dataset. Only individuals for which both models fail on all five fields are retained. A total of 883 researchers satisfy all four criteria and pass this filtering.

(2) Personal Information Fields. For each of the 883 selected researchers, five personal information fields are extracted from the raw profile. `Country` is taken directly from the country listed in the contact address. `LatestWorkTitle` is the title of the most recently published work registered under the researcher; when multiple works share the same publication year, the work listed earlier in the profile is selected (this tie-breaking rule is applied consistently rather than chosen arbitrarily). Researchers with no registered works are assigned no value for this field. `WorkCount` is the total number of registered works, and is set to 0 (not treated as missing) when no works are registered. `HomepageName` and `HomepageAddress` are taken from only the first entry in the researcher's list of registered homepages; consequently, a change in the registration order between the two years can cause these fields to differ even if the underlying homepage content is unchanged. HTML entity-encoded characters present in the raw text are preserved as-is and are not converted.

(3) Missing Value Imputation. When a field has a value in only one of the two snapshots, the existing value is copied into the snapshot where it is missing (e.g., if `Country` is missing in 2013 but present in 2014, the 2014 value is copied into 2013); fields that already have a value are never overwritten. After this imputation step, no field remains missing in exactly one snapshot; any remaining missing values are fields that are missing in both snapshots.

(4) Knowledge Graph Construction. The personal information of the retained individuals is converted into (head entity, relation, tail entity) triples. We use the five forward relations corresponding to the fields above, together with their five inverse relations, each paired with a fixed natural-language template (e.g., *"is based in"*, *"has the latest work titled"*). Three normalization rules are applied when constructing the knowledge graph:

1. Formatting noise in raw text value such as tabs, newlines, and consecutive whitespace is collapsed into a single space, and leading and trailing whitespace is removed.
2. If a field has no value, only that field is dropped from the individual's record; the individual is not removed from the dataset.
3. The entity name used to identify each individual in the knowledge graph is fixed to their 2013 credit name. For the 23 individuals whose credit name changed between 2013 and 2014, only the 2013 name is used in the knowledge graph, even though the 2014 profile displays a different name. When two individuals share the same 2013 name, their ORCID identifier is appended to disambiguate them; this occurs for exactly one pair in the dataset.

The resulting knowledge graph contains 883 researcher entities and 4,572 entities in total, 10 relations (5 forward and 5 inverse), 3,962 forward triples per snapshot, and 7,924 triples per snapshot including inverse relations.

(5) Edit Targets and Locality Set. Edit targets are restricted to facts that have a value in both $\mathbf{KG}^{\text{init}}$ and $\mathbf{KG}^{\text{upd}}$ and whose value differs between the two; fields with a value in only one snapshot are not treated as "changed facts" and are excluded. This yields 1,938 edit targets from 866 researchers. Under this criterion, some individuals end up with identical values in both snapshots after missing-value imputation and therefore contribute no edit target, although they remain in the knowledge graph. We additionally construct a locality evaluation set using only `Country`, `HomepageName`, and `HomepageAddress`, excluding `LatestWorkTitle` and `WorkCount` because these fields change frequently between snapshots.

A.4 Relation Templates

Each personal attribute is stored as a triple (s, r, o) and verbalized into a natural-language prompt through a relation-specific template. Given the subject and the relation, the model is trained to predict the object entity. The templates for PerInfoKG, Athlete, and ORCID are listed in Tables 11, 12, and 13, respectively. For PeaCoK-Ex, the only personal attribute is the occupation of each person (`Job`), so we provide the corresponding template, "{S} has a job of {O}.", here instead of in a separate table.

A.5 Details of the KG–LLM Alignment Module and Evaluation Protocol

PeaCoK-Ex The PeaCoK-Ex dataset consists of individuals whose occupations are either changed or unchanged. Accuracy measures whether updated personal information is correctly reflected in the LLM, while Locality assesses whether unchanged personal information remains intact. Retention further evaluates whether the updated model preserves its original general knowledge using the `known_1000` dataset Meng et al. (2022), which contains only facts already known to the initial model (*init-LLM*). For example, consider a scenario in which the triple (`Mike`, `Job`, `Student`) is updated to (`Mike`, `Job`, `Pilot`), while (`Jane`, `Job`, `Researcher`) remains unchanged.

To compute the difference (Δ, Eq. 6) for "Mike", the KG–LLM Alignment Module independently processes the initial and updated triples for "Mike", producing two distinct representations. Their difference is then used as a steering vector to guide the LLM. Accuracy is evaluated by verifying whether the steered model correctly answers queries about Mike's occupation as "Pilot". In addition, Retention is assessed by checking whether the same steering vector does not harm the model's ability to correctly respond to general factual queries from the `known_1000` dataset (e.g., "Apple Inc. is founded by").

In contrast, since Jane's occupation remains unchanged, Locality evaluates whether the model continues to correctly answer queries about Jane's occupation as "Researcher". Although the triples (`Jane`, `Job`, `Researcher`) at the two time points are identical, they are embedded based on different temporal snapshots of the underlying knowledge graph. As a result, the KG–LLM Alignment Module produces different representations for the two inputs. Their difference is used to steer the LLM, and Locality is measured by verifying that the model still correctly predicts Jane's occupation as "Researcher".

PerInfoKG Evaluation on PerInfoKG follows a slightly different protocol, as each individual simultaneously contains both updated and unchanged personal information. For example, consider an individual "Kate", for whom the triples (`Kate`, `EducationLevel`, `B.S`) and (`Kate`, `CommutingMeans`, `Walking`) are updated to (`Kate`, `EducationLevel`, `Ph.D`) and (`Kate`, `CommutingMeans`, `Subway`), respectively, while (`Kate`, `BornIn`, `1990s`) remains unchanged.

To evaluate Accuracy, the initial and updated triples corresponding to the changed relations (i.e., `EducationLevel` and `CommutingMeans`) are jointly processed by the KG–LLM Alignment Module in a batch-wise manner. The resulting representation differences are used to steer the LLM, which is trained to correctly reflect the updated personal information. Accuracy is then measured by verifying whether the steered model generates correct answers for the updated relations. For Retention, we assess whether the model preserves its ability to answer general factual queries (`known_1000`) while being steered by the delta derived from updated triples. For each person, the steering vectors of the updated triples are averaged, and this mean vector is added to the attention-head activations when the model answers the general factual queries.

Table 6: Statistics of the ORCID dataset.

#Entities	4,572
#Relations	10 (including reverse)
#Triples	7,924
#Researcher entities	883
#Updated researchers	866

Table 7: Top-8 circuit similarity analysis across datasets and models.

Dataset Pair	GPT-J	Qwen
PeaCoK-Ex vs. PerInfoKG	**0.6242**	**0.4754**
PeaCoK-Ex vs. `known_1000`	0.5774	0.4375
PerInfoKG vs. `known_1000`	0.5335	0.4100

To assess Locality, the same trained KG–LLM Alignment Module for "Kate" is reused without modification. The unchanged triples are processed in the same manner to obtain representation differences, which are used to steer the *init-LLM*. Locality is evaluated by checking whether the model continues to correctly answer queries about the unchanged personal information (e.g., "Kate was born in the 1990s"), despite having been steered using updates to other relations (i.e., `EducationLevel` and `CommutingMeans`). Note that the unchanged triples are only passed through the trained module at evaluation time to obtain Δ, and are not used again for training.

A.6 Baselines and Hyperparameters

Among editing methods, we consider 8 representative approaches.

LoRA. LoRA (Hu et al., 2022) is a parameter-efficient fine-tuning method that injects trainable low-rank matrices into selected weight matrices while freezing the backbone parameters. On GPT-J, we use rank 32 and a learning rate of 2×10^{-4} for 4 steps per edit on PeaCoK-Ex, and rank 12 and a learning rate of 3×10^{-5} for 5 epochs on Athlete and ORCID. On Qwen2.5-7B-Instruct, we use rank 64 and a learning rate of 1×10^{-4} for 9 steps per edit on PeaCoK-Ex, and rank 16 and a learning rate of 3×10^{-5} for 5 epochs on Athlete and ORCID. On both models, we use rank 8 and a learning rate of 5×10^{-5} for 3 epochs on PerInfoKG.

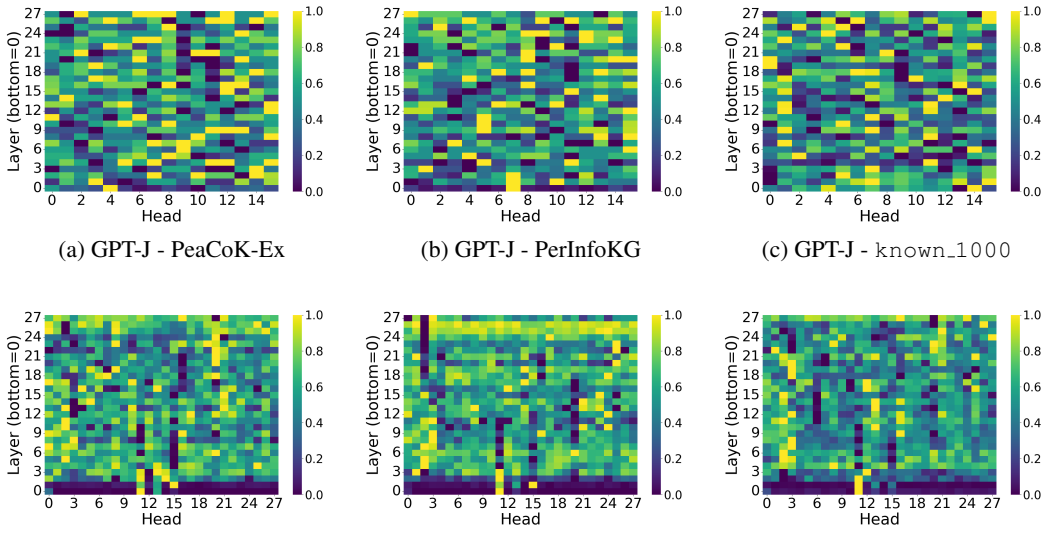
ROME. ROME (Meng et al., 2022) treats FFN modules as key–value memories and directly alters them to inject new facts. On GPT-J, we edit layer 7 with $v_{\text{lr}} = 1$ for 25 steps on PeaCoK-Ex, and layer 5 with $v_{\text{lr}} = 0.01$ for 40 steps on PerInfoKG, Athlete, and ORCID. On Qwen2.5-7B-Instruct, we edit layer 6 with $v_{\text{lr}} = 1$ for 30 steps on PeaCoK-Ex, and layer 7 with $v_{\text{lr}} = 0.1$ for 40 steps on PerInfoKG, Athlete, and ORCID.

ICE. ICE (Qi et al., 2025) applies consistency-based supervision to align the model's predictions with contextual prompts without relying on hard one-hot targets. GPT-J and Qwen2.5-7B-Instruct use the same settings. On PeaCoK-Ex, we fine-tune layers $\{10, 11, 12, 13, 14\}$ for 10 steps with a learning rate of 1×10^{-5}, and on PerInfoKG, Athlete, and ORCID, we fine-tune layers $\{4, 5, 6, 7, 8\}$ for 10 steps with a learning rate of 5×10^{-5}.

LightEdit. LightEdit (Jung et al., 2026) is a lightweight framework designed for scalable lifelong knowledge editing. It features an edit-aware selector to filter relevant knowledge and an in-context decoding strategy that suppresses outdated knowledge probabilities at the first token level without modifying model parameters. We use $\alpha = 0.2$ with retrieval top-$k = 5$ on all datasets for both GPT-J and Qwen2.5-7B-Instruct.

MEMIT-Merge. MEMIT-Merge (Dong et al., 2025) extends MEMIT by merging edits for overlapping subjects, making it effective in batch editing scenarios. On GPT-J, we edit layer 5 with $v_{\text{lr}} = 0.01$ and `clamp_norm_factor` $= 2$ on all datasets. On Qwen2.5-7B-Instruct, we edit layer 24 with $v_{\text{lr}} = 1$ and `clamp_norm_factor` $= 0.75$ on all datasets.

AlphaEdit. AlphaEdit (Fang et al., 2025) projects each update onto the null space of the preserved knowledge, preventing interference with it and thereby supporting Locality. On GPT-J, we edit layers 3–8 with $v_{\text{lr}} = 0.05$ for 25 steps on PeaCoK-Ex and PerInfoKG, and layer 8 on Athlete and layers 6–8 on ORCID, both with $v_{\text{lr}} = 0.5$ for 30 steps. On Qwen2.5-7B-Instruct, we edit layers 4–8 with $v_{\text{lr}} = 0.1$ for 25 steps on PeaCoK-Ex and PerInfoKG, and layer 8 on Athlete and layers 6–8 on ORCID, both with $v_{\text{lr}} = 10$ for 30 steps.



(a) GPT-J - PeaCoK-Ex (b) GPT-J - PerInfoKG (c) GPT-J - known_1000

(d) Qwen - PeaCoK-Ex (e) Qwen - PerInfoKG (f) Qwen - known_1000

Figure 4: Heatmap visualizations of important components identified in GPT-J (Wang & Komatsuzaki, 2021) and Qwen (Yang et al., 2024) across PeaCoK-Ex, PerInfoKG, and known_1000 (Meng et al., 2022) (Wiki-based dataset). Each plot shows attention heads (x-axis) by layers (y-axis). The similarity is highest when comparing personal knowledge datasets.

WISE. WISE (Wang et al., 2024a) performs continual editing by storing updated knowledge in a dedicated side memory and routing queries accordingly. All WISE runs edit the MLP down-projection at layer 21 on GPT-J and layer 23 on Qwen2.5-7B-Instruct. On GPT-J, we use edit_lr $= 1$, 70 iterations, and an activation margin of $(5, 20, 10)$ on PeaCoK-Ex, and edit_lr $= 0.5$, 10 iterations, and an activation margin of $(15, 40, 20)$ on PerInfoKG, Athlete, and ORCID. On Qwen2.5-7B-Instruct, we use edit_lr $= 0.05$, 5 iterations, an activation margin of $(15, 40, 20)$, and act_ratio $= 0.75$ on PeaCoK-Ex, and edit_lr $= 0.5$, 1 iteration, and an activation margin of $(5, 20, 10)$ on PerInfoKG, Athlete, and ORCID.

SAKE. SAKE (Scialanga et al., 2025) models a fact-to-be-edited as an activation distribution rather than an isolated prompt, leveraging Optimal Transport to steer LLM behavior across paraphrases and logical implications while preserving contextual robustness. On all datasets, GPT-J uses 16 paraphrases with reg $= 10^{-2}$ and a prompt threshold of $\tau = 6.5$. Qwen2.5-7B-Instruct uses the same setting on PeaCoK-Ex, PerInfoKG, and Athlete, whereas on ORCID it uses reg $= 10^{-1}$ and applies the map at every generated token.

A.7 Visualization

In this section, we visualize and analyze the circuits identified in Section 4.1. Figure 4 presents the min–max normalized importance scores of attention heads across layers for each LLM (Wang & Komatsuzaki, 2021; Yang et al., 2024) and personal information dataset (PeaCoK-Ex, PerInfoKG).

A.8 Circuit Structure Analysis

Overall, models of the same architecture exhibit highly consistent circuit structures across personal information datasets. Interestingly, Qwen shows a clear and consistent pattern: head 11 dominates in the early layers, heads 2 and 3 are salient in layers 4–11, head 20 becomes prominent in deeper layers, and importance gradually dissipates toward the final layers. In contrast, GPT-J does not display dominance of specific heads but instead distributes importance across multiple heads within each layer, suggesting a more diffuse circuit structure for handling personal data.

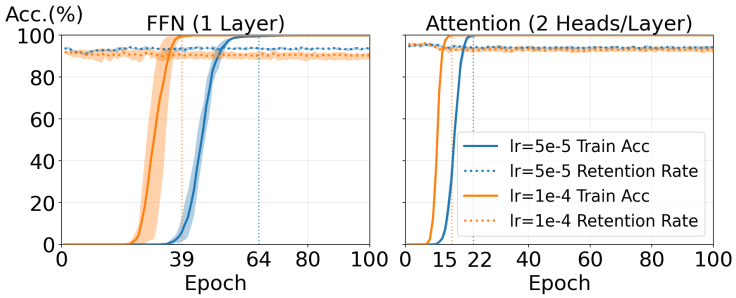


Figure 5: Selective fine-tuning on Llama-3-8B-Instruct. Training accuracy and retention rate over epochs for FFN (left panel) and attention heads (right panel). The attention-based fine-tuning achieves faster convergence and higher retention of existing knowledge than FFN-based fine-tuning.

Top-k Circuit Similarity Analysis. To further examine the specialization of circuits for personal information processing, we conduct a top-k analysis that focuses only on the most critical attention heads within each layer. Specifically, for each layer ℓ, we identify the k highest-scoring heads based on importance scores and construct a binary mask $\mathbf{M}_\ell \in \{0,1\}^{N_{heads}}$, where N_{heads} is the number of heads. The masked importance matrix is then obtained as $\mathbf{S}^{(k)} = \mathbf{S} \odot \mathbf{M}$, where $\mathbf{S}$ is the original importance score matrix and $\odot$ denotes element-wise multiplication. Circuit similarity is then measured as the cosine similarity between the flattened masked matrices.

Table 7 presents pairwise cosine similarities with k=8 (top-8 heads per layer). The results reveal a clear pattern: personal information datasets (PeaCoK-Ex and PerInfoKG) consistently show higher similarity to each other than to general Wiki-based knowledge data (`known_1000`).

For GPT-J, the similarity between PeaCoK-Ex and PerInfoKG reaches 0.6242, notably higher than the cross-domain similarities of 0.5774 (PeaCoK-Ex vs. `known_1000`) and 0.5335 (PerInfoKG vs. `known_1000`). Qwen shows the same trend, with 0.4754 (PeaCoK-Ex vs. PerInfoKG) exceeding 0.4375 and 0.4100 for cross-domain pairs.

Implications for Personal Information Processing. These findings provide suggestive evidence for the existence of specialized neural circuits for personal information processing in large language models. The consistently higher intra-domain similarity (personal vs. personal) compared to cross-domain similarity (personal vs. general) suggests that LLMs recruit distinct computational pathways for handling personal information queries. This specialization has important implications for both interpretability and privacy: identifying dedicated personal circuits enables targeted interventions such as selective parameter editing or circuit-level privacy protection. Moreover, the contrast between GPT-J's diffuse attention patterns and Qwen's concentrated head dominance indicates that circuit specialization strategies may differ across architectures, highlighting an avenue for future research.

Global or Personal circuit. We compare "global" and "personal" circuits in terms of effectiveness (by comparing accuracy and locality) and scalability. First, we compare the accuracy and locality in the PeaCoK-Ex dataset using GPT-J when applying "global" or "personal" circuits. When applying a personal circuit, the model achieves 100% accuracy and 99.4% locality. Compared to the performance of the global circuit in Table 1 (100% accuracy, 100% locality), the difference between the two settings is negligible, indicating that both approaches are similarly effective in terms of accuracy and locality.

We also conducted the same comparison on PerInfoKG using GPT-J. With global circuit steering, the model achieved 99.2% accuracy, 98.4% locality, and a harmonic mean of accuracy and locality of 98.8%. Using personal (user-specific) circuits yielded 92.9% accuracy, 95.6% locality, and a total score of 94.2%. As in PeaCoK-Ex, the performance gap between the two remains minimal, further confirming that personal circuits do not offer a meaningful advantage in effectiveness.

The distinction becomes more pronounced when considering scalability. "Personal" circuits require computing a new circuit for each incoming user before SPIKE can be applied, which incurs ongoing

Table 8: Performance on triples extracted from unstructured personal documents with SPIKE.

Model	Acc. (%)	Loc. (%)	Ret. (%)	Total
GPT-J	86.49	87.85	74.15	82.35
Qwen2.5-7B-Inst.	91.21	75.70	82.02	82.49

Table 9: Evaluation of the noisy version of PeaCoK-Ex. **Tot.** indicates the harmonic mean of **Acc.** and **Loc.**.

Model	Method	Acc.	Loc.	Tot.
GPT-J	AlphaEdit Fang et al. (2025)	6.10	**96.40**	11.47
	FT (Fine-tuning)	95.12	9.42	17.14
	SPIKE (Ours)	**98.17**	93.47	**95.76**

per-user overhead. In contrast, the global circuit can be maintained and reused as new users arrive. Its robustness to unseen triples is further supported by the results demonstrated in Figure 3(a), where the global circuit remains effective even under an unseen test scenario.

In summary, while both circuit types exhibit comparable effectiveness, the global circuit offers substantially better scalability. For this reason, we adopt the global circuit in our method.

A.9 Additional Experiment

Unstructured Knowledge Update Setting. To assess whether the proposed method remains effective without assuming access to a clean personal knowledge graph, we evaluate it on noisy unstructured text via a multi-stage data processing pipeline that simulates a realistic knowledge editing scenario. First, we transform the PerInfoKG dataset containing diverse attributes into unstructured texts using an LLM API (GPT-4o as of Jan 2026), emulating the AKEW (Wu et al., 2024) benchmark. We then apply an LLM-based triple extraction that automatically derives (subject, relation, object) triples from the generated text. Finally, by injecting the extracted triples into the language model through the proposed method, we enable the model to successfully internalize and update its knowledge base while maintaining consistency across interconnected personal attributes.

There is a performance gap between direct structured injection (Table 1) and our proposed pipeline. While the original model achieves 95.4% and 95.9% in total score on GPT-J and Qwen2.5-7B, the performance naturally decreases in this setting due to the inherent complexity of unstructured text processing (Table 8). However, the fact that our method maintains high scores (82.35% and 82.49%) despite the indirect extraction process further underscores its robustness and practical utility for real-world personal information updates. Interestingly, even under this setting, our method achieves higher total scores than several edit-based approaches evaluated in the structured injection (edit) setting, such as ROME, MEMIT-Merge, and AlphaEdit (Table 1), indicating the effectiveness of our mechanism.

Evaluation on Noisy Entity Variants. The personal-KG datasets used in our main experiments employ normalized entity strings for consistency, where descriptive modifiers and free-form expressions are removed. Since personal information in natural settings can include such descriptive or unstructured forms, we also construct a variant of PeaCoK-Ex that preserves the original entity strings from the source corpus and evaluate SPIKE under this extended condition.

The original PeaCoK dataset includes rich, free-form descriptions within entity tokens (e.g., `(aids in the completion of large projects, is a social routine or habit of, heavy duty equipment operator who work hard at my job)`, `(Arlo Hill, has a job of, heavy-duty equipment operator who work long and hard)`). Such expressions contain descriptive modifiers and unstructured phrasing (e.g., "who work long and hard") that introduce linguistic variability not present in the canonical entity labels. In PeaCoK-Ex, we intentionally removed this variability to isolate the underlying structured entity (e.g., heavy-duty equipment operator). To evaluate robustness under more realistic conditions, we constructed a noisy version of PeaCoK-Ex, which retains all original descriptive, unstructured, and diverse entity strings while preserving the same number of people. Importantly,

Table 10: Extension to Other LLMs.

Dataset	LLM	Acc.	Loc.	Ret.	Tot.
PeaCoK-Ex	Llama3 (8B)	100.0	99.2	87.6	95.3
	Mistral (7B)	100.0	99.6	91.8	97.0
PerInfoKG	Llama3 (8B)	100.0	100.0	81.2	92.9
	Mistral (7B)	98.8	99.3	87.6	94.9
Athlete	Llama3 (8B)	99.4	99.7	87.2	95.1
	Mistral (7B)	100.0	99.1	92.6	97.1

both the input triple and the updated target triple use these full noisy expressions. Thus, the model must perform personalization and update reasoning without relying on normalized labels, instead handling full naturalistic variation. All other experimental settings remain identical to those used in the main evaluation.

As shown in Table 9, introducing noisy and naturalistic variations into the personal information significantly impacts baseline editing methods. Specifically, both AlphaEdit and FT experience severe performance degradation, with their accuracy and locality metrics collapsing under the unstructured expressions. In contrast, our proposed method maintains high robustness across all metrics

Extension to Other LLM Architectures. We perform the same analysis on Llama3-8B-Instruct Grattafiori et al. (2024) as in Figures 1a and 1b. As shown in Figure 5, we observe the same trend: fine-tuning the parameters of attention heads leads to faster convergence and better preservation of existing knowledge than fine-tuning FFN parameters.

We also conduct additional experiments to examine whether our methodology generalizes to other LLMs, Llama3-8B-Instruct (Grattafiori et al., 2024) and Mistral-7B-Instruct-v0.3 (Mistral AI, 2024). Using the same PeaCoK-Ex, PerInfoKG, and Athlete datasets, we evaluate whether applying our method starting from an *init-LLM* enables the model to accurately incorporate the updated knowledge in $\mathbf{KG}^{\text{upd}}$ and respond consistently with the updated facts. Table 10 shows that SPIKE effectively updates personal knowledge on Llama and Mistral while maintaining performance, demonstrating that our approach is readily applicable to widely used foundation models. This confirms that SPIKE is not limited to GPT-J or Qwen2.5, but generalizes across different model architectures.

We set $(\lambda_1, \lambda_2, k) = (0.1, 0.5, 3)$ for PeaCoK-Ex on both Llama3-8B and Mistral, while using $(0.0, 0.0, 3)$ for PerInfoKG. For Athlete, we configure the parameters as $(0.0, 0.0, 3)$ on Llama3-8B and $(0.1, 0.0, 2)$ on Mistral.

A.10 Experimental Setting

As shown in Eq. 8, our objective consists of three components: the negative log-likelihood term $\mathcal{L}_{\text{NLL}}$ that enforces the updated KG to be reflected in the initial LLM, the KL divergence term $\mathcal{L}_{\text{KL}}$ that preserves knowledge unrelated to the updates, and the norm-based penalty $\mathcal{L}_{\text{norm}}$ that prevents the steering vector from deviating excessively from the LLM representation. The relative weights of the latter two terms are controlled by λ_1 and λ_2, which we search over $\{0.0, 0.1, \ldots, 1.0\}$. We additionally treat the number of intervened attention heads k as a hyperparameter, searched over $\{2, 3, \ldots, 8\}$.

The selected configurations are as follows: for PeaCoK-Ex, $(\lambda_1, \lambda_2, k) = (0.1, 0.5, 3)$ on GPT-J and $(0.1, 0.5, 3)$ on Qwen2.5-7B-Instruct; for PerInfoKG, $(0.1, 0.3, 3)$ on GPT-J and $(0.0, 0.5, 2)$ on Qwen2.5-7B-Instruct; for Athlete, $(0.0, 0.5, 4)$ on GPT-J and $(0.0, 0.0, 4)$ on Qwen2.5-7B-Instruct. For the unseen triple test described in Section 5.3, we use $(\lambda_1, \lambda_2, k) = (0.0, 0.0, 8)$ on both GPT-J and Qwen2.5-7B-Instruct, increasing k to accommodate the larger dataset and the higher-capacity KG–LLM Alignment Module required. For ORCID, we use $(\lambda_1, \lambda_2, k) = (0.0, 0.3, 5)$ on both GPT-J and Qwen2.5-7B-Instruct.

For all datasets, the dimensionality d of the query, key, and value representations (Eq. 1) is set to the hidden dimension of the LLM, i.e., $d = D_{\text{txt}} = 4096$ for GPT-J and $d = D_{\text{txt}} = 3584$ for Qwen2.5-7B-Instruct. On the KG side, triple embeddings are obtained from TransE with dimension $D_{\text{kge}} =$

4096, and subgraph embeddings are computed by a two-layer R-GCN with hidden dimension 4096. TransE is trained separately on each snapshot: the embeddings trained on $\mathbf{KG}^{\text{init}}$ are used to encode the initial triples, and those trained on $\mathbf{KG}^{\text{upd}}$ are used to encode the updated triples. The output dimension is $D_{\text{out}} = d_{\text{head}} \times H$ (Section 4.2.2), where $H = k \times L$ is the total number of selected heads, with k heads selected in each of the $L = 28$ layers, and d_{head} is 256 for GPT-J and 128 for Qwen2.5-7B-Instruct. Accordingly, D_{out} is 21,504, 21,504, 28,672, and 35,840 for GPT-J, and 10,752, 7,168, 14,336, and 17,920 for Qwen2.5-7B-Instruct on PeaCoK-Ex, PerInfoKG, Athlete, and ORCID, respectively.

All experiments are conducted on a single NVIDIA A100 GPU. We used the Transformers package from Hugging Face. The number of trainable parameters on GPT-J is approximately 390M (6.9% of the non-embedding backbone parameters; PeaCoK-Ex, PerInfoKG), 420M (7.4%; Athlete), 449M (8.0%; ORCID), and 537M (9.5%; the unseen triple test), and on Qwen2.5-7B-Instruct approximately 297M (4.6%; PeaCoK-Ex), 284M (4.4%; PerInfoKG), 310M (4.8%; Athlete), 323M (4.9%; ORCID), and 361M (5.5%; the unseen triple test).

In the main experiments (Table 1), the KG–LLM Alignment Module is trained per user. On Qwen2.5-7B-Instruct, most of its parameters come from the R-GCN subgraph encoder (100M) and the alignment layers (157M), neither of which depends on the number of modified triples; the remainder is the output projection to the steering vector Δ, whose size grows with the number of selected heads. Such a scale is not unusual for knowledge editing, where even a few facts are typically updated by modifying large weight matrices. In particular, when the edit spans multiple layers, as in ICE and AlphaEdit, which update the entire down-projection matrices of layers 4–8, the number of updated parameters reaches 339M per individual, which even exceeds the number of parameters trained by SPIKE in the main experiments on Qwen2.5-7B-Instruct (284M–323M). Nevertheless, even at this scale, these methods struggle with multi-fact updates: on PerInfoKG, Athlete and ORCID, where multiple facts of the same individual are updated simultaneously, both ICE and AlphaEdit show substantially lower overall performance than SPIKE (Table 1).

A.11 Training and Inference Efficiency

Despite the additional trainable parameters, training and inference costs remain comparable to existing model-editing baselines. On PeaCoK-Ex (Qwen2.5-7B-Instruct backbone), SPIKE updates 297M parameters (4.6% of full fine-tuning) in 6.4s per individual on a single A100, versus 9.9s for full fine-tuning. At inference time, Δ is computed by a single forward pass of the alignment module per query, rather than per token, and is then added to the selected heads throughout generation, adding approximately 11ms to Time-To-First-Token (35.03ms $\rightarrow$ 46.47ms). Since Δ depends only on the triples and not on the query, it can also be precomputed at KG-update time and cached, in which case this overhead reduces to the vector addition at the selected heads.

A.12 LLM Personalization

EMG-RAG Wang et al. (2024b) addresses personalized question answering by extracting personal memories from smartphone conversations and app screenshots. Their approach introduces an Editable Memory Graph (EMG) that supports dynamic memory operations, including insertion, deletion, and replacement of personal information. The system employs reinforcement learning to train an agent for adaptive memory selection, moving beyond fixed Top-k retrieval methods to handle complex queries requiring diverse memory combinations. While their work focuses on memory retrieval and selection for downstream tasks such as QA and form autofill, our approach tackles a different challenge: efficiently incorporating updated knowledge graph information into LLM behavior without full model retraining.

A recent work Prahlad et al. (2025) has explored personalization approaches for LLMs by structuring personal data from applications such as calendars, conversational chats, and emails into knowledge graphs for smart response generation. Their approach leverages RAG with smaller models to provide factually correct responses using dynamically updated KGs, addressing privacy concerns by keeping sensitive data locally rather than sending it to cloud-based LLM providers. The system focuses on using KG-based retrieval to enhance LLM responses for personal queries and smart reply generation. However, this work focuses on retrieval-based personalization rather than updating LLM knowledge as personal information changes.

A relevant baseline is KGT Sun et al. (2024), which adapts to evolving user information by directly modifying the knowledge graph, such as adding or removing triples based on user feedback. During inference, it relies on retrieving these updated triples and appending them to the input context, thereby depending on the model's in-context reasoning capabilities to incorporate the external information. Unlike KGT, which modifies the external input context via retrieval, our approach operates directly on the model's internal state by injecting steering vectors into activations, effectively shifting the model's internal processing toward the new information without altering the input prompt.

Table 11: Relation-specific prompt templates used to verbalize personal-information triples in PerInfoKG. {S} and {O} denote the subject and object entity, respectively.

Relation	Template
Demographics	
gender	{S} is {O}.
race_ethnicity	{S} is racially or ethnically identified as {O}.
age_group	{S} was born in the {O}.
nationality	{S} has a nationality of {O}.
Location and Living	
housing_type	{S} has a housing type of {O}.
address	{S} lives in {O}.
commuting_means	{S} commutes by {O}.
Health and Lifestyle	
blood_type	{S} has a blood type of {O}.
medical_condition	{S} has a medical condition of {O}.
exercise_frequency	{S} exercises {O}.
diet_type	{S} follows a diet of {O}.
drinking_frequency	{S} drinks alcohol {O}.
Social / Personal	
religion	{S} follows the religion of {O}.
marital_status	{S} has a marital status that is {O}.
pets_owned	{S} owns a pet that is a {O}.
hobby	{S} has a hobby of {O}.
favorite_food	{S} has a favorite food that is {O}.
favorite_music	{S} likes the music genre of {O}.
Socioeconomic / Education	
education_level	{S} has an education level of {O}.
major	{S} majored in {O}.
job	{S} works as a {O}.
phone_model	{S} uses a phone model of {O}.
political_affiliation	{S} is politically affiliated with {O}.

Table 12: Relation-specific prompt templates used to verbalize Athlete triples. {S} and {O} denote the subject and object entity, respectively.

Relation	Template
gender	{S} is {O}.
birth_date	{S} has the birth date of {O}.
birth_place	{S} has the birth place of {O}.
birth_country	{S} has the homeland of {O}.
discipline	{S} plays the sport of {O}.
residence_place	{S} resides in {O}.
height	{S} has the height of {O}.

Table 13: Relation-specific prompt templates used to verbalize ORCID triples. {S} and {O} denote the subject and object entity, respectively.

Relation	Template
`Country`	{S} is based in {O}.
`LatestWorkTitle`	{S} has the latest work titled {O}.
`WorkCount`	{S} has a work count of {O}.
`HomepageName`	{S} has a researcher URL named {O}.
`HomepageAddress`	{S} has a researcher URL at {O}.

Table 14: Two case studies illustrating model responses to an occupation-related question and a multi-hop reasoning question drawn from the PeaCoK-Ex dataset. Multi-hop queries are handled by the same injection mechanism as single-hop queries, without any additional routing or composition step.

	Case Study 1	**Case Study 2**
Subject	Natalie King	Lucian Newman
Initial Occupation	Head of Corporation	Popular President
Updated Occupation	Guitarist	Chess Player
Occupation Question	Natalie King has a job of	Lucian Newman has a job of
Answer	Guitarist	Chess Player
Multi-hop Question	Natalie King has a job whose characteristic is	Lucian Newman has a job that requires
Answer	skilled in musical performance	strategy and good decision-making skills

A.13 Case Study

Table 15: A case study illustrating model responses to behavior personalization drawn from the PeaCoK-Ex dataset.

	Original LLM (GPT-J)	**Original LLM (GPT-J) + SPIKE**
Subject	Natalie King	Natalie King
Occupation	Head of Corporation	Guitarist
Behavior Question	Based on the job information Natalie King has, how will Natalie King respond to the following question? "What does 'leading a successful performance' mean to you?"	
Answer	"I would define it as the ability to influence **others** to achieve success."	"I would define it as having a successful performance **myself**."

This section presents two case studies that demonstrate the effectiveness of SPIKE: the first assesses its ability to handle multi-hop questions involving updated facts, and the second examines whether the LLM exhibits personalized behavioral tendencies.

Table 14 summarizes two examples drawn from PeaCoK-Ex, focusing on the subjects 'Natalie King' and 'Lucian Newman'. We assess the performance of SPIKE under two types of questions: (i) a direct query regarding the updated occupation, and (ii) a multi-hop query that requires reasoning based on the occupation. In both cases, SPIKE successfully guides the language model to provide accurate and contextually appropriate responses. Specifically, when asked about the characteristics of Natalie King's job, the steered model generated "skilled in musical performance," which aligns closely with the updated occupation of Guitarist rather than the previous role as Head of Corporation. Similarly, in response to a query about Lucian Newman's job requirements, the model produced "strategy and

good decision-making skills," reflecting the essential competencies of a Chess Player. These results demonstrate that SPIKE effectively steers the model not only to answer direct occupation queries but also to generate coherent and realistic responses in multi-hop reasoning scenarios. Consequently, this case study reinforces the applicability of SPIKE in both direct and reasoning-based evaluation settings.

Table 15 examines whether SPIKE enables the LLM to adjust its behavior in accordance with an updated fact. The table presents a case study involving a tone-validation question, demonstrating that the model's response varies depending on the updated occupation. For instance, when analyzing the term 'successful performance', the interpretation of 'performance' shifts significantly depending on whether the occupation is 'head of a corporation' or updated to 'guitarist'. The results indicate that, without incorporating SPIKE, the model's responses remain anchored to the initial occupation, emphasizing organizational roles rather than individual characteristics, as a 'head of a corporation' must account for all employees. In contrast, when SPIKE is applied, the model shifts toward a more individualized interpretation, well reflecting the occupation into that of a (solo) guitarist.

Table 16: Defined Fields and Probability Weights for 23 Candidate Fields in PerInfoKG.

Field	**Candidate List**	**Probability List**
Address	['New York', 'Toronto', 'Berlin', 'Seoul', 'Tokyo', 'Paris', 'Sydney']	[0.24, 0.17, 0.12, 0.06, 0.15, 0.13, 0.13]
Drinking Frequency	['infrequently', 'sometimes', 'often', 'socially']	[0.2, 0.3, 0.3, 0.2]
Favorite Food	['ramen', 'pizza', 'sushi', 'pasta', 'bibimbap', 'steak', 'burger']	[0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429]
Favorite Music	['Pop', 'Jazz', 'Classical', 'Hip-Hop', 'Rock', 'Indie', 'Electronic']	[0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429]
Diet Type	['vegan', 'vegetarian', 'omnivore', 'halal']	[0.1, 0.1, 0.75, 0.05]
Nationality	['Korean', 'American', 'Japanese', 'British', 'Canadian']	[0.05, 0.4, 0.15, 0.2, 0.2]
Housing Type	['apartment', 'house', 'dormitory', 'studio', 'villa']	[0.3, 0.3, 0.1, 0.1, 0.2]
Commuting Means	['subway', 'bus', 'car', 'bike', 'walking']	[0.3, 0.3, 0.3, 0.05, 0.05]
Marital Status	['single', 'married']	[0.45, 0.55]
Exercise Frequency	['rarely', 'infrequently', 'weekly', 'daily']	[0.15, 0.5, 0.3, 0.05]
Blood Type	['A', 'B', 'AB', 'O']	[0.34, 0.27, 0.11, 0.28]
Religion	['Christianity', 'Buddhism', 'Islam', 'Hinduism', 'Atheism']	[0.3, 0.07, 0.24, 0.15, 0.24]
Hobby	['reading', 'swimming', 'painting', 'gaming', 'hiking', 'cycling', 'traveling']	[0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429, 0.1429]
Gender	['male', 'female']	[0.5, 0.5]
Phone Model	['iphone', 'galaxy', 'mi', 'pixel']	[0.4, 0.4, 0.1, 0.1]
Race/Ethnicity	['White', 'African American', 'Asian', 'Hispanic', 'other']	[0.2, 0.2, 0.1, 0.2, 0.3]
Age Group	['1940s', '1950s', '1960s', '1970s', '1980s', '1990s', '2000s']	[0.04, 0.06, 0.1, 0.12, 0.17, 0.24, 0.27]
Medical Condition	['None_MED', 'diabetes', 'hypertension', 'asthma', 'depression', 'arthritis', 'allergies']	[0.573, 0.073, 0.083, 0.063, 0.052, 0.052, 0.104]
Political Affiliation	['Democrat', 'Republican', 'Independent', 'Unaffiliated']	[0.2, 0.3, 0.05, 0.45]
Pets Owned	['None_PET', 'dog', 'cat', 'other']	[0.4, 0.3, 0.2, 0.1]
Education Level	['middle school', 'high school', 'bachelor's degree', 'master's degree', 'PhD']	[0.1, 0.57, 0.25, 0.06, 0.02]
Major	['Computer Science', 'Business', 'Biology', 'Mechanical Engineering', 'Economics', 'English Literature', 'Nursing', 'None_MAJ']	[0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125, 0.125]
Job	['accountant', 'artist', 'barista', 'cashier', 'counselor', 'data analyst', 'dentist', 'developer', 'doctor', 'driver', 'economist', 'engineer', 'entrepreneur', 'manager', 'nurse', 'pilot', 'politician', 'professor', 'researcher', 'scientist', 'soldier', 'teacher', 'writer']	[0.0637, 0.0179, 0.0179, 0.0179, 0.0208, 0.0156, 0.0156, 0.0312, 0.0156, 0.0179, 0.025, 0.0156, 0.1701, 0.1284, 0.0417, 0.0156, 0.0543, 0.1314, 0.0677, 0.0469, 0.0179, 0.0335, 0.0179]