

# A Storage–Retrieval Gap in Parametric Knowledge Graph Memory

Martino M. L. Pulici<sup>1,2,\*</sup>, Cuong Xuan Chu<sup>1</sup>, Evgeny Kharlamov<sup>1,3</sup> and Volker Tresp<sup>2,4</sup>

<sup>1</sup>Bosch Center for Artificial Intelligence, Robert-Bosch-Campus 1, 71272 Renningen, Germany

<sup>2</sup>LMU Munich, Institute for Informatics, Oettingenstraße 67, 80538 Munich, Germany

<sup>3</sup>University of Oslo, Department of Informatics, Gaustadalléen 23 B, 0373 Oslo, Norway

<sup>4</sup>Munich Center for Machine Learning, LMU Munich, Institute for Informatics, Oettingenstraße 67, 80538 Munich, Germany

## Abstract

Graph retrieval-augmented generation places retrieved subgraphs into the model’s context window at query time, paying a recurring token cost and exposing source data on every call. We study an alternative: compiling a knowledge graph offline into a bank of LoRA adapters, one per entity, that serve as a parametric knowledge layer queried by injecting weights rather than text, at zero query-time context cost. On the MetaQA dataset, we find that subgraph-trained adapters encode context-free factual knowledge that generalizes to unseen questions: on single-valued relations the adapter gains +0.243 exact-match score over a base model that is nearly blind closed-book (0.007), and only the correct adapter recovers this knowledge (an oracle gap of +0.283 over the base model). However, the stored knowledge is not recoverable by similarity: given a query with no subgraph, embedding-based and weight-space geometry retrieval both perform at chance, because a semantically neighbouring entity’s adapter does not contain the answer – knowledge is stored locally and does not transfer. Weight geometry correlates with subgraph semantics ( $\rho = +0.329$ ) but not with functional retrievability. We quantify the byte and context-token costs against graph retrieval-augmented generation and discuss deployment implications. Our results establish that parametric knowledge graph memory is feasible for storing knowledge, and identify selecting and composing the right adapters by a mechanism other than semantic similarity as the central open problem – motivating a learned, query-conditioned composition mechanism.

## Keywords

Adapter retrieval, knowledge graphs, LoRA adapters, parametric memory, retrieval-augmented generation, weight-space geometry

## 1. Introduction

Knowledge graphs (KGs) give industrial AI systems a structured, auditable substrate of facts, and a now-standard way to put that substrate to work with large language models (LLMs) is graph retrieval-augmented generation (RAG): retrieve a relevant subgraph for a query, serialize it into the prompt, and let the model read the answer out of context. This pattern is effective but carries two recurring costs that scale with usage rather than with the size of the graph. First, every query pays a token cost: the serialized subgraph consumes context-window budget, inflating latency, memory, and energy. Second, every query incurs a data-exposure cost: the raw subgraph is shipped to wherever inference runs, which is problematic when the graph encodes sensitive or proprietary engineering knowledge.

We investigate an alternative in which the cost is paid once, offline. A KG is compiled, entity neighbourhood by entity neighbourhood, into a bank of low-rank adaptation (LoRA) adapters [1]. Each adapter is a small weight perturbation  $\Delta \mathbf{W}$  that encodes one subgraph. At query time the system retrieves one or more adapters, injects them into a frozen base model, and answers with no subgraph

*SKGi 2026: Scaling Knowledge Graphs for Industry Workshop, co-located with SEMANTiCS’26: 22nd International Conference on Semantic Systems, September 15–17, 2026, Ghent, Belgium*

\*Corresponding author.

✉ martino.pulici@de.bosch.com (M. M. L. Pulici); cuongxuan.chu@de.bosch.com (C. X. Chu);

evgeny.kharlamov@de.bosch.com (E. Kharlamov); volker.tresp@lmu.de (V. Tresp)

🌐 <https://ilsommo.github.io> (M. M. L. Pulici); <https://cuongcx.github.io> (C. X. Chu);

<https://www.bosch.com/research/about-bosch-research/our-research-experts/evgeny-kharlamov> (E. Kharlamov);

<https://www.dbs.ifi.lmu.de/~tresp> (V. Tresp)

🆔 0009-0009-4533-7495 (M. M. L. Pulici); 0000-0003-3247-4166 (E. Kharlamov); 0000-0001-9428-3686 (V. Tresp)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

text in the context window. The knowledge graph thus becomes a parametric memory: a set of weights, not a corpus of text.

Whether this works at all turns on a question that prior parametric RAG systems leave largely implicit: when an adapter is fine-tuned on a piece of text, does it store that text’s facts in its weights, or does it only learn to use the text when the text is present at inference? The distinction is decisive for parametric memory, because at query time the model has no subgraph in context—so only knowledge that resides in the weights is available at all. We therefore ask three questions:

1. Do subgraph adapters actually store knowledge recoverable with no context?
2. Can the right adapter be retrieved for a query?
3. Can adapters be composed across entities for multi-hop questions?

Our findings separate cleanly. Subgraph adapters do store generalizable closed-book knowledge: on a compact model and a near-blind base, single-valued relations gain +0.243 exact match (EM), and only the entity’s own adapter recovers it (a +0.283 gain over the base model). However, the stored knowledge turns out to be local and non-retrievable: given a query with no subgraph, neither question-embedding nor weight-space geometry finds the right adapter above chance, because a semantically similar entity’s adapter simply does not contain the answer. Our central finding is therefore a dissociation. A KG can be compiled into per-entity LoRA adapters that store context-free, generalizable knowledge at zero query-time token cost. Yet, selecting and composing the right adapters for a query is unsolved: knowledge is stored locally, and neither similarity-based retrieval nor naive merging recovers it.

We make the following contributions:

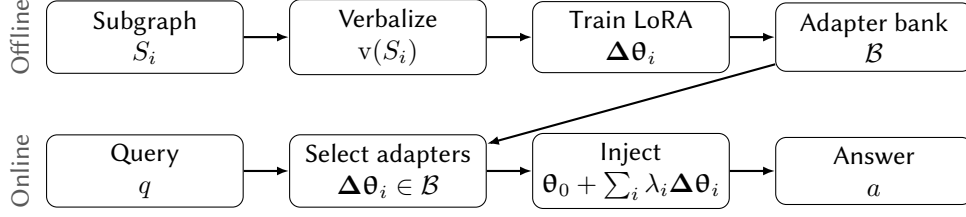
- a parametric KG memory formulation in which per-entity subgraphs are compiled into a LoRA adapter bank, queried by weight injection with zero subgraph tokens at inference (section 2)
- evidence that subgraph adapters store generalizable knowledge, since on unseen questions single-valued relations gain +0.243 EM over a near-blind base and only the correct adapter recovers it (+0.283 oracle gap) (section 4.1)
- a storage–retrievability dissociation, whereby the stored knowledge is not recoverable from a query, as embedding and  $\Delta\mathbf{W}$  geometry retrieval both perform at chance closed-book because a semantically neighbouring adapter does not contain the answer, while weight geometry correlates with semantics but not with function (section 4.2)
- a cost analysis and a precise open problem, as we quantify the zero-context-token advantage over graph RAG (section 5.2) and show that both retrieval and multi-hop composition reduce to a single unsolved task—learning which local adapters to combine for a query (section 4.3).

## 2. Method

**Parametric knowledge graph memory.** Figure 1 summarizes the design. Offline, each entity’s subgraph is verbalized and compiled into a per-entity LoRA adapter, forming a bank; online, an adapter is selected for the query and injected into the frozen base, and the model answers with no subgraph in its context. Let

$$G = \{(h, r, t) \mid h, t \in E, r \in R\}$$

be a KG with entities  $E$  and relations  $R$ . For a chosen set of anchor entities  $\{e_1, \dots, e_N\} \subseteq E$  we extract for each  $e_i$  its  $k$ -hop neighbourhood subgraph  $S_i \subseteq G$ . Each subgraph is verbalized by a function into text as  $v(S_i)$  and used to fine-tune a LoRA adapter  $\Delta\theta_i$  on a frozen base model  $\theta_0$ , producing a bank  $\mathcal{B} = \{\Delta\theta_i\}_{i=1}^N$ . At query time, given a question, we: (a) retrieve one or more adapters  $\Delta\theta_i \in \mathcal{B}$ ; (b) form  $\theta = \theta_0 + \sum_i \lambda_i \Delta\theta_i$ ; and (c) decode the answer from  $\theta$  with no subgraph text in its context. Single-entity questions use a single adapter; multi-entity questions require merging (section 4.3).



**Figure 1:** Parametric KG memory. In the offline phase, each entity subgraph is verbalized and compiled into a per-entity LoRA adapter, forming a bank; in the online phase, an adapter is selected for the query and injected into the frozen base, and the model answers with no subgraph in its context.

**Verbalization.** Each subgraph is rendered into text by a verbalization function  $v(\cdot)$  before training. We use a fixed, fact-dense template that enumerates the entity’s triples with minimal surface variation (one declarative clause per triple), so that the adapter is trained on a compact, consistent rendering of the subgraph. Because a KG subgraph is already a short enumeration of facts, this templated form stays close to any natural-language rendering of the same triples; we did not find weight-level storage to depend on the choice, and use the templated function throughout. The query-time model sees no subgraph, so only weight-level knowledge is available; that such knowledge is present at all is what makes parametric KG memory possible.

**The  $\Delta W$  representation.** Each adapter’s perturbation  $\Delta W = BA$  per layer-module is stored via its truncated singular value decomposition (SVD), and pairwise distances are computed as normalized Frobenius distances directly from the SVD factors without materializing  $\Delta W$  (details in appendix A). For retrieval (section 4.2) we score a query against the bank and return the nearest adapters under  $\Delta W$  geometry, contrasting this with a sentence-embedding baseline and random selection. Critically, in deployment the query carries no subgraph, so the retrieval signal must come from the question text alone.

### 3. Experimental setup

**Training.** We use Qwen3.5-2B [2, 3], a compact model inexpensive to train yet able to answer the benchmark’s questions when the relevant facts are available. All adapters share this frozen base, loaded in 4-bit QLoRA [4], with a standard LoRA configuration ( $r = 8$ ,  $\alpha = 16$ , 0.05 dropout, all seven projection types) trained to convergence with AdamW [5] (learning rate  $2 \times 10^{-4}$ , cosine schedule, 50 warmup steps, batch size of four) using a fixed instruction template (appendix B). Of each entity’s question answering (QA) pairs, eight are used for training and four held out for evaluation.

**Corpus.** We use MetaQA [6], a movie knowledge graph with entity-level QA. MetaQA has no unique film identifiers, so the film title is necessarily the entity key; because distinct films can share a title, keying naively on the title would merge them into a single adapter trained on contradictory labels. We therefore retain only films whose title maps to exactly one release year in the knowledge base, excluding multi-film title collisions; adapters built from collided titles often fail to reproduce even their own training answers, since the same question maps to several conflicting gold answers, whereas clean single-film adapters do so reliably. This yields 150 clean single-film entities, each with an internally consistent one-hop subgraph verbalized into templated triple-enumeration text.

**Probes and metrics.** We evaluate each adapter closed-book (CB), with no subgraph in the context, and open-book (OB), with the subgraph in the context. MetaQA relations are set-valued (a film has many tags, several cast members), so we report three metrics. Our primary metric is **EM**: since relations are set-valued, a prediction counts as correct if, under normalized EM, it equals any valid object of the queried entity–relation pair. EM cannot score high-cardinality relations when the model emits a valid

**Table 1**

CB EM gain over base on held-out questions. Values are averaged over 150 clean single-film entities, and gains are shown with their 95 % CIs. The  $p$ -values are computed using one-sample bootstrap test on paired differences based on 50 000 resamples; \* $p < 0.05$ , \*\* $p < 0.01$ , \*\*\* $p < 0.001$ .

| Setting                 | Base  | Adapted | Gain   | CI <sub>95 %</sub> | $p$       |
|-------------------------|-------|---------|--------|--------------------|-----------|
| Single-valued relations | 0.007 | 0.250   | +0.243 | [+0.174, +0.319]   | <0.001*** |
| All relations           | 0.010 | 0.137   | +0.127 | [+0.083, +0.170]   | <0.001*** |

answer inside verbose output, so we additionally report any match (**AM**), measuring if the prediction contains any valid object, and **set recall**, computed against the union of valid objects. We distinguish single-valued relations (director, writer, release year), where EM is meaningful, from multi-valued relations (tags, cast), where set metrics are required. All metrics are applied identically to the base model and every adapter. Generations use greedy decoding with a 128-token cap. All gains are reported as per-entity paired differences over base with 95 % bootstrap confidence intervals (CIs) based on 50 000 resamples.

## 4. Experiments

### 4.1. Weight-level storage

Our first question is whether a subgraph-trained adapter stores knowledge that is recoverable CB— with no subgraph in context— and that generalizes to questions the adapter was not trained on. This is the defining property of a parametric memory: at query time the model sees only the question, not the subgraph.

**Closed-book generalization gain.** We evaluate each adapter CB on its entity’s held-out questions and compare to the base model on the same questions. The base model is nearly blind on CB questions: it answers 1 % of held-out questions (147 of 150 entities score zero), confirming the films are outside its parametric knowledge and leaving substantial headroom for the adapter to fill. The gain is therefore measured almost entirely on entities the base does not know, and we do not interpret the three-entity base-known cell. On single-valued relations, adapters recover +0.243 EM (CI<sub>95 %</sub> = [+0.174, +0.319]) over the near-blind base, as shown in table 1: they learn, and generalize, facts they demonstrably did not know before. The all-relations gain of +0.127 is lower only because it pools in multi-valued relations that exact match scores at zero by construction: the single-valued figure is the fair measure of weight-level storage.

**Adapter specificity.** The CB gain could in principle reflect a generic output-conditioning effect— any adapter making the model terser— rather than entity-specific knowledge. We make sure this is not the case by comparing the correct adapter against an unrelated (wrong-entity) adapter, an untrained random-weight adapter, and the base model. Only the correct adapter helps, by +0.283 EM over base (CI<sub>95 %</sub> = [+0.183, +0.400],  $p < 0.001$ ), as shown in table 2: the stored knowledge is specific to the entity the adapter was trained on. The two negative controls separate in an informative way. An unrelated but trained adapter leaves performance exactly at base (gain 0.000,  $p = 1.000$ ): it encodes real facts, just the wrong entity’s, and does not change any outcome. An untrained adapter with randomly initialized weights scores 0.000, below base (gain −0.017,  $p < 0.001$ ), destroying even the residual pretraining knowledge the base model has for one entity. The LoRA structure therefore contributes nothing on its own; the recovered knowledge comes entirely from gradient updates on entity-specific data. The wrong adapter never changes any outcome relative to base— every paired difference is exactly zero, hence  $p = 1.000$ — because the base model is at floor on CB QA (0.017 EM) and injecting an unrelated adapter does not move outputs enough to flip any answer. That floor also means the

**Table 2**

CB EMs of different adapters, averaged over 30 queries and shown with their 95 % CI. The random-weight adapter is an untrained LoRA with randomly initialized weights (same rank and target modules), averaged over five seeds; the wrong adapter is a trained adapter belonging to a different entity. The  $p$ -values are computed using one-sample bootstrap test on paired differences based on 50 000 resamples;  $*p < 0.05$ ,  $**p < 0.01$ ,  $***p < 0.001$ .

| Injected adapter      | Exact match | Gain   | CI <sub>95 %</sub> | $p$       |
|-----------------------|-------------|--------|--------------------|-----------|
| Base (no adapter)     | 0.017       | —      | —                  | —         |
| Random-weight adapter | 0.000       | −0.017 | [−0.050, 0.000]    | <0.001*** |
| Other (wrong) adapter | 0.017       | 0.000  | [ 0.000, 0.000]    | 1.000     |
| Own (correct) adapter | 0.300       | +0.283 | [+0.183, +0.400]   | <0.001*** |

**Table 3**

Multi-valued relations under three CB metrics. EM registers zero while AM and set recall show the adapter produces valid answers; the zero is a scoring artefact, not a learning failure.

| Relation       | Exact match | Any match | Set recall |
|----------------|-------------|-----------|------------|
| Has tags       | 0.000       | 0.854     | 0.793      |
| Starred actors | 0.000       | 0.500     | 0.389      |

**Table 4**

CB EM gain over base for each single-valued relation. Values are computed over that relation’s held-out questions in the 150-entity bank and shown with its 95 % CI. The  $p$ -values are from a one-sample bootstrap test on paired differences based on 50 000 resamples;  $*p < 0.05$ ,  $**p < 0.01$ ,  $***p < 0.001$ .

| Relation     | EM gain over base | CI <sub>95 %</sub> | $p$       |
|--------------|-------------------|--------------------|-----------|
| Director     | +0.274            | [+0.161, +0.387]   | <0.001*** |
| Writer       | +0.269            | [+0.115, +0.462]   | 0.001**   |
| Release year | +0.196            | [+0.089, +0.321]   | 0.001**   |
| Genre        | +0.182            | [ 0.000, +0.455]   | 0.218     |
| Language     | +0.091            | [−0.182, +0.364]   | 0.769     |

wrong-adapter contrast has little power to detect a small positive effect, so its  $p$ -value should not be read as evidence of equivalence; the contrast that carries the claim is the own-adapter gain.

**Answer cardinality.** Single-valued relations (a film has one director, one writer, one release year) and multi-valued relations (many tags, several cast members) behave differently under EM, and the difference is a property of the metric, not of what the adapter learned. Table 3 shows the pattern: on multi-valued relations, EM is 0.000 because the model emits a valid answer inside verbose output that never exactly equals a single gold token, yet AM is 0.854 for tags and set recall is 0.793. We therefore restrict EM claims to single-valued relations and report set metrics for multi-valued ones, leaving set-valued parametric memory to future work. As shown in table 4, among single-valued relations the gain is consistent: the 95 % CIs for ‘director’, ‘writer’, and ‘release year’ all exclude zero, while the sparser ‘genre’ and ‘language’ relations do not.

**Verbalization format.** A natural design question is whether the verbalization function matters for weight-level storage: a fact-dense template might plausibly store better than fluent prose, in which case parametric KG memory would depend on a carefully engineered encoding. On an earlier 120-entity bank we trained matched adapters under both a templated and a natural-prose verbalization of the same subgraphs, holding the QA pairs fixed so that only the training text differs, and compared CB gains. We find no detectable difference: the paired difference is +0.019 (CI<sub>95 %</sub> = [−0.013, +0.050],  $p = 0.275$ ), and no relation subset with more than two entities reaches significance. We report this as a null result rather than evidence for either format: at this scale, weight-level storage does not appear to

require a carefully engineered verbalization, plausibly because a KG subgraph is already fact-dense in either rendering. This ablation uses a different bank from the 150-entity set of our main results, so we treat it as indicative rather than conclusive.

## 4.2. Retrieval

A parametric memory is only useful if the right adapter can be selected for a query without its subgraph in hand. Section 4.1 showed the correct adapter recovers real CB knowledge; here we ask whether that adapter can be found from the query alone. The answer is a clean negative for similarity-based selection, and it is the paper’s second main finding: the knowledge stored in an adapter does not transfer to, and cannot be recovered from, a semantically neighbouring entity’s adapter, so choosing an adapter by semantic proximity to the query is the wrong mechanism for entity-keyed parametric memory. We are careful about the scope of this claim: it is a negative result about similarity-based routing, not about adapter selection in general. A conventional entity-resolution step — title or identifier matching, or lexical search over the entity index — would route many MetaQA questions correctly, since the questions typically name their anchor film and the bank holds exactly one adapter per film; a deployed system would use such a resolver (section 5.2). Our point is narrower: the embedding and weight-space geometry that a parametric system might hope to retrieve with do not encode answer containment, and so cannot substitute for entity resolution.

**Closed-book retrieval.** The retrieval question is only meaningful in a CB setting: with the subgraph in context, the base model reads the answer directly and every condition ties, so a retrieval difference can only appear when the adapter is the sole source of knowledge. Table 5 evaluates CB retrieval: for each query we retrieve an adapter by question-embedding distance, by  $\Delta\mathbf{W}$  Frobenius geometry, and at random, inject it with no context, and score the query’s held-out questions. The correct adapter beats the base model by +0.283, but no query-driven method recovers it: embedding and  $\Delta\mathbf{W}$  retrieval both land exactly at base (0.000 gain,  $p = 1.000$ ), and the two random baselines are indistinguishable from base as well ( $p = 0.778$  and  $p = 0.533$ ). The two query-driven methods are not merely both at chance but exactly equivalent to each other, their paired difference being identically 0.000: weight-space geometry adds nothing over a sentence embedding for this task. Crucially, this is not a failure of the retriever to identify similar entities. Question embeddings identify semantically appropriate neighbours cleanly: each entity is its own nearest neighbour, and near neighbours are genuinely related films (e.g. two films by the same director). The failure is that a neighbour’s adapter, however semantically apt, does not contain the answer to the query entity’s questions. Knowledge is stored locally in each adapter and does not transfer.

**Retrieval protocol.** Each of the 30 query entities is scored against the full gallery of 120 adapters. In the embedding condition, we encode the question text and each entity’s verbalized subgraph with a sentence-embedding model and rank adapters by cosine distance; in the  $\Delta\mathbf{W}$  condition, we rank by the normalized Frobenius distance aggregated over the top layers. We report downstream CB EM after injecting the top-ranked adapter, which is the quantity that matters for parametric memory: a correctly ranked adapter is only useful if it also answers the question. We note that rank-based metrics are underpowered at this scale — with one correct adapter in a gallery of 120, chance Recall@1 is below 1 %, so a single observed hit is statistically indistinguishable from chance in either direction. The claim therefore rests on the downstream EM gap between the oracle and every query-driven method, not on rank statistics.

**Geometry.** On the clean bank, pairwise  $\Delta\mathbf{W}$  Frobenius distances do correlate with subgraph semantic distance: Spearman  $\rho$  is +0.329 over all layers and +0.352 over the top eight layers. To confirm this is a real signal rather than an artefact of the distance formula, we use two null controls: permuting entity labels across adapters and column-shuffling each  $\Delta\mathbf{W}$  (destroying low-rank structure while preserving the marginal value distribution). Both nulls stay at or below  $|\rho| = 0.157$  over 1000 resamples, far

**Table 5**

CB EM gain over base for query-driven adapter retrieval. Values are computed over 30 queries and shown with their 95 % CI. The  $p$ -values are computed using one-sample bootstrap test on paired differences based on 50 000 resamples;  $*p < 0.05$ ,  $**p < 0.01$ ,  $***p < 0.001$ .

| Retrieval method             | Exact match | Gain   | CI <sub>95 %</sub> | $p$       |
|------------------------------|-------------|--------|--------------------|-----------|
| Base (no adapter)            | 0.017       | —      | —                  | —         |
| Question embedding           | 0.017       | 0.000  | [ 0.000, 0.000]    | 1.000     |
| $\Delta\mathbf{W}$ Frobenius | 0.017       | 0.000  | [ 0.000, 0.000]    | 1.000     |
| Random (global)              | 0.010       | −0.007 | [−0.047, +0.023]   | 0.778     |
| Random (within-domain)       | 0.007       | −0.010 | [−0.050, +0.020]   | 0.533     |
| Oracle (own adapter)         | 0.300       | +0.283 | [+0.183, +0.400]   | <0.001*** |

**Table 6**

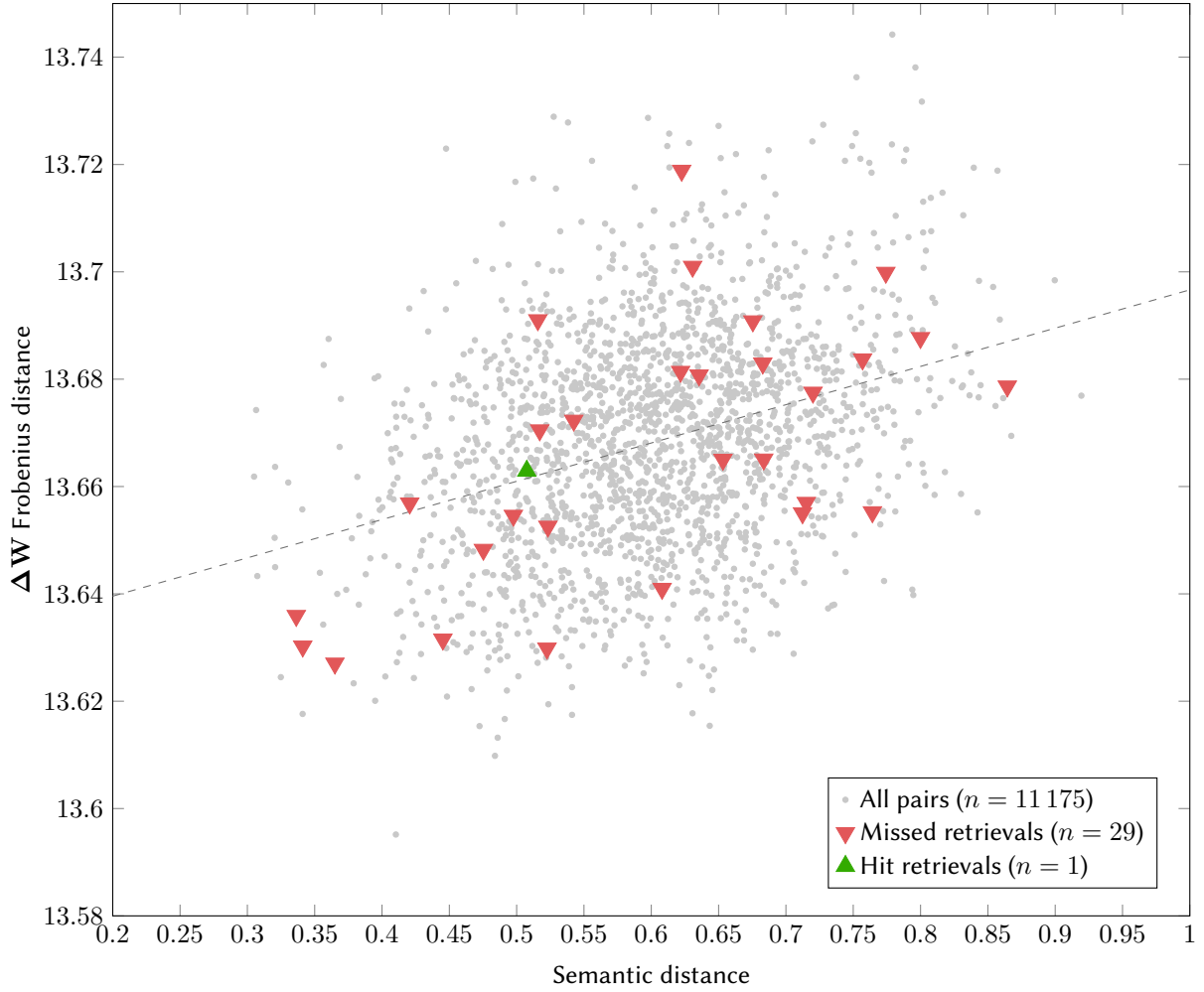
Geometry–semantics correlation on the clean bank, with null controls.

| Setting           | Spearman $\rho$ |
|-------------------|-----------------|
| All layers        | +0.329          |
| Top 8 layers      | +0.352          |
| Label permutation | +0.157          |
| Column shuffle    | +0.036          |

below the observed value ( $p < 0.001$ ). Consistent with the finding that later layers carry more of the entity-specific signal, the top-layer aggregation in table 6 selects layers 15, 16, 17, 19, 20, 21, 22, and 23. Figure 2 plots the correlation directly: the trend is real but weak, and—critically—the sole retrieval hit falls squarely among the misses, so ordering pairs by weight geometry does not surface the correct adapter. Yet, this correlation does not translate into functional retrieval:  $\Delta\mathbf{W}$  retrieval performs exactly as embedding and random do. Weight geometry thus encodes which entities are similar but not which adapter answers a given question—a geometry–function dissociation, in which a real geometric signal carries no matching functional advantage. We therefore read the  $\rho$ -retrieval gap not as a weak or noisy geometric signal, but as evidence that entity similarity and answer containment are genuinely different relations.

### 4.3. Composition

Multi-hop questions require combining knowledge across entities, and the natural mechanism is to merge the relevant adapters. Subgraph adapters are, in principle, candidates for merging: each encodes a small, low-rank perturbation, and if per-entity perturbations occupy largely distinct directions, merging need not cause destructive interference. The obstacle we encounter is not interference but the benchmark and the retrieval result above. Concretely, MetaQA’s two-hop questions ask about an intermediate entity (e.g. ‘what films share a director with this film?’), and their answers require knowledge contained in neither of the two entity subgraphs a merge would combine, nor in the two subgraphs placed jointly in context. In our tests every condition, including the both-subgraphs-in-context oracle, scored at floor, so the experiment could not isolate a composition effect: the information needed was absent from the inputs by construction. We therefore do not report a composition result and instead state the problem precisely, because it connects directly to the retrieval finding of section 4.2. Both point to the same gap: knowledge is stored locally in per-entity adapters, and neither query-driven retrieval nor naive merging assembles the right knowledge for a multi-entity question. Closing this gap—learning which adapters to combine and how to weight them for a given query, rather than selecting by similarity—is the central open problem that a learned composition mechanism would address and is left to future work.



**Figure 2:** Weight geometry versus semantic distance for all 11 175 adapter pairs. The dashed line shows the correlation between  $\Delta\mathbf{W}$  Frobenius distance and semantic distance ( $\rho = +0.329$ ); it should be noted that the effect is small in absolute terms (the Y-axis is zoomed to the observed range, about 1.2 % of the norm), since all adapters share nearly identical weight magnitudes; highlighted are the 30 query-gallery pairs selected by  $\Delta\mathbf{W}$  retrieval: the 29 misses and the single hit are interleaved along the trend, so ordering pairs by weight geometry does not separate correct retrievals from incorrect ones.

## 5. Discussion

### 5.1. Interpreting the dissociation

The paper’s two negative results — chance-level retrieval (section 4.2) and the composition mismatch (section 4.3) — share a single behavioural signature: an adapter’s knowledge is available when the adapter is the correct one and absent otherwise, with no graceful degradation for near neighbours. We take this seriously as a finding rather than a nuisance, and separate what we can and cannot conclude about its cause.

Three facts hold with statistical support. First, the correct adapter carries entity-specific knowledge: it beats the base model by +0.283 CB, while an unrelated trained adapter leaves performance exactly at base and an untrained random-weight adapter falls below it. Second, similarity does not imply transfer: the question-embedding retriever selects genuinely related neighbours (same-director or same-genre films), yet applying those neighbours’ adapters recovers nothing over random. Third, the failure is not a failure of the geometry to organize by semantics —  $\Delta\mathbf{W}$  distances correlate with subgraph semantics at  $\rho = +0.329$ , well above the permutation nulls ( $|\rho| \leq 0.157$ ) — but a failure of that organization to predict answer containment.

**Table 7**

Cost comparison. The zero context-token cost of parametric memory is definitional, as no subgraph enters the prompt; token and byte figures are medians over the same 150 entities used throughout, measured on the verbalized subgraph text used for training (tokens with the Qwen3.5-2B tokenizer, bytes as UTF-8); the adapter size is the raw checkpoint, identical across all 150 adapters, and is a one-time cost amortized over all queries to an entity.

|                        | Graph RAG (in-context) | Parametric (ours)   |
|------------------------|------------------------|---------------------|
| Context tokens / query | 60                     | 0                   |
| Bytes shipped / query  | 188 B                  | 10.4 MiB (one-time) |

Why do neighbouring adapters not help? We can point to at least two mechanisms our experiments do not distinguish. One is *answer non-overlap*: two similar films rarely share the specific object a question asks for (given two similar films, their directors are almost always different), so even perfect transfer would not help—a property of the QA task, not of the adapters. The other is *storage locality*: fine-tuning may write each entity’s facts into a subspace a different entity’s adapter does not activate, making knowledge mechanically inaccessible across adapters. Both predict the same retrieval curve, and separating them would require shared-answer cases at a scale MetaQA’s single domain does not allow.

## 5.2. Cost and deployment

Because query-driven adapter selection is unsolved (section 4.2), the cost analysis carries the central justification for storing knowledge in weights rather than retrieving subgraph text. The case rests on one property: weight injection consumes zero context tokens at query time, whereas graph RAG pays a per-query token cost that grows with subgraph size, as shown in table 7. On MetaQA this cost is modest: measured with the Qwen3.5-2B tokenizer over all 150 clean entities, a serialized one-hop subgraph is a median of 60 tokens (the interquartile range is [53, 70]), because the benchmark’s entities are sparse—typically six or seven single-line fields. The 60 tokens should therefore be read as a floor rather than a headline: the saving is small here precisely because MetaQA subgraphs are small, and it would scale up on the dense, multi-hop, property-rich subgraphs of industrial KGs, where in-context serialization runs to hundreds or thousands of tokens per query. This zero-token property is the dimension on which parametric memory wins unconditionally on context cost. We are careful not to overclaim it as a latency or energy win: injecting and swapping a per-query adapter is itself a cost that graph RAG does not pay, and we have not measured end-to-end latency or energy, so we claim the context-token saving only, not a net speedup.

That token saving comes at a one-time storage cost. An adapter is far larger than the subgraph it encodes (10.4 MiB of weights versus 188 B of serialized triples, a ratio of roughly 58 000 times more) so, on bytes alone, parametric memory is favourable only when an entity is queried often enough to amortize its adapter, or when the binding constraint is context budget rather than storage. That ratio also bounds scalability honestly: a bank covering one million entities would occupy on the order of 10 TiB of checkpoints, plus a one-time offline training cost per entity, so the approach suits stable, frequently queried subgraphs rather than web-scale or rapidly changing graphs. At query time the parametric layer injects weights and never ships the raw subgraph, which reduces raw-context transmission when the graph is proprietary. We frame this as reduced raw-context exposure rather than leakage resistance: our own positive result shows that facts are recoverable from the weights, so the adapter is itself a potential leakage surface, and we do not test resistance to extraction. Both benefits are real only for the storage part of the system: because selection is unsolved (section 4.2), a deployed system would today pair adapter storage with a conventional question-to-entity resolver rather than weight-space retrieval.

Parametric KG memory is not free. It pays a one-time offline encoding cost and, once compiled, a fact update requires re-encoding the affected adapter. For graphs that change continuously, or for one-shot queries over a graph that is never reused, in-context graph RAG remains the better trade-off. Parametric

memory pays off when a stable subgraph is queried many times, when context budget is the binding constraint, or when shipping raw subgraphs is undesirable.

## 6. Related work

**Parametric RAG and what LoRA encodes.** A line of work encodes retrieved documents directly into model parameters rather than into context: Parametric RAG [7] fine-tunes a LoRA adapter per document and averages retrieved adapters at inference, dynamic parametric RAG (DyPRAG) [8] trains a hypernetwork to map documents to adapters on the fly, and Poly-PRAG [9] learns a shared adapter basis with per-document routing. These systems target document-level, single-hop RAG and implicitly assume that document-encoded adapters carry the document’s knowledge — which presumes that fine-tuning stores facts in the weights rather than only teaching the model to exploit in-context text. We apply the parametric-memory idea to KG subgraphs and test that assumption directly, using closed-book evaluation to isolate the weight-resident component and an own-versus-other-adapter contrast to confirm the recovered knowledge is entity-specific: subgraph adapters do encode recoverable CB knowledge, but it cannot be retrieved from the query alone.

**KG question answering, graph RAG, and merging.** Standard QA on KG and graph RAG approaches retrieve and serialize subgraphs into the prompt; we do not replace retrieval but move the retrieved knowledge from the context window into the weights, trading a recurring per-query token cost for a one-time offline encoding cost, and use an in-context oracle as an upper bound. For composition, task arithmetic [10] and TIES-merging [11] compose weight perturbations and study interference; whether per-entity subgraph adapters can be merged without destroying each other’s knowledge, and whether a merged adapter supports reasoning that spans the entities it combines, is the open question our composition experiment probes (section 4.3).

## 7. Conclusion

We framed a knowledge graph as a parametric memory: a bank of LoRA adapters, one per film, queried by weight injection with no subgraph text in context. We found that subgraph-trained adapters encode context-free, CB-recoverable knowledge that generalizes to unseen questions — a +0.243 EM gain on single-valued relations over a near-blind base. Only the correct adapter recovers this knowledge (an oracle gap of +0.283 over base), but no query-driven method finds it: embedding and  $\Delta\mathbf{W}$  geometry retrieval both perform at chance, because knowledge is stored locally and does not transfer to semantically neighbouring adapters.

These findings come with scope conditions. This is a single-benchmark, single-model, workshop-scope study with model-generated QA pairs. The headline CB claim is restricted to single-valued relations, since multi-valued ones (e.g. tags or cast) require a set-valued notion of parametric memory we leave to future work. Our single-domain (film) entities leave open whether the retrieval negative persists across broad domains, where  $\Delta\mathbf{W}$  geometry separates more strongly. Within these bounds, storing knowledge in weights is feasible at zero query-time token cost. The open problem is not storage but composition: learning which local adapters to combine for a multi-entity query.

## Declaration on Generative AI

During the preparation of this work, the authors used Claude Opus 4.8 and Claude Opus 5 in order to: draft content, paraphrase and reword, improve writing style, draft abstract, check grammar and spelling, simulate peer review, and enhance content. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

## References

- [1] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: Proceedings of the 10th International Conference on Learning Representations, ICLR 2022, 2022. URL: <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [2] QwenTeam, Qwen3.5: Towards native multimodal agents, 2026. URL: <https://qwen.ai/blog?id=qwen3.5>, accessed 2 Sept. 2026.
- [3] Qwen, Qwen3.5-2B, 2026. URL: <https://huggingface.co/Qwen/Qwen3.5-2B>, accessed 2 Sept. 2026.
- [4] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, QLoRA: Efficient finetuning of quantized LLMs, Advances in Neural Information Processing Systems 36 (2023) 10088–10115. doi:10.52202/075280-0441.
- [5] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, 2019. URL: <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [6] Y. Zhang, H. Dai, Z. Kozareva, A. Smola, L. Song, Variational reasoning for question answering with knowledge graph, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, volume 1 of AAAI-18, 2018, pp. 1823–1832. doi:10.1609/aaai.v32i1.12057.
- [7] W. Su, Y. Tang, Q. Ai, J. Yan, C. Wang, H. Wang, Z. Ye, Y. Zhou, Y. Liu, Parametric retrieval augmented generation, in: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’25, 2025, pp. 1240–1250. doi:10.1145/3726302.3729957.
- [8] Y. Tan, S. He, H. Liao, J. Zhao, K. Liu, Dynamic parametric retrieval augmented generation for test-time knowledge enhancement, Computing Research Repository (2025). arXiv:2503.23895.
- [9] Z. Su, F. Mo, J. Zhang, Y. Hui, J. Sun, J. Nie, Parametric retrieval-augmented generation using latent routing of LoRA adapters, Computing Research Repository (2025). arXiv:2511.17044.
- [10] G. Ilharco, M. T. Ribeiro, M. Wortsman, L. Schmidt, H. Hajishirzi, A. Farhadi, Editing models with task arithmetic, in: Proceedings of the 11th International Conference on Learning Representations, ICLR 2023, 2023. URL: <https://openreview.net/forum?id=6t0Kwf8-jrj>.
- [11] P. Yadav, D. Tam, L. Choshen, C. A. Raffel, M. Bansal, TIES-merging: Resolving interference when merging models, Advances in Neural Information Processing Systems 36 (2023) 7093–7115. doi:10.52202/075280-0310.

## A. Mathematical details

### A.1. SVD representation

A LoRA adapter modifies each weight matrix  $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$  with a low-rank perturbation

$$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{B}\mathbf{A}$$

where  $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$  and  $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ . We work exclusively with  $\Delta\mathbf{W} = \mathbf{B}\mathbf{A}$  rather than the raw factors. For each layer–module pair, we compute the truncated SVD

$$\Delta\mathbf{W} = \mathbf{U} \text{diag}(\boldsymbol{\sigma}) \mathbf{V}^\top$$

where  $\mathbf{U} \in \mathbb{R}^{d_{\text{out}} \times r}$ ,  $\boldsymbol{\sigma} \in \mathbb{R}^r$  the vector of singular values, and  $\mathbf{V}^\top \in \mathbb{R}^{r \times d_{\text{in}}}$ . Only  $\mathbf{U}$ ,  $\boldsymbol{\sigma}$ , and  $\mathbf{V}^\top$  are stored: the full  $d_{\text{out}} \times d_{\text{in}}$  matrix is never formed. Storing the rank- $r$  factors rather than the dense  $\Delta\mathbf{W}$  for every layer–module pair reduces analysis-time memory by roughly the same  $d/r$  factor, turning what would be several gibibytes per adapter into a few tens of mebibytes; the on-disk checkpoint, which stores only the LoRA factors, is smaller still at 10.4 MiB.

The SVD is computed efficiently via QR decomposition of the low-rank factors, in  $\mathcal{O}(dr^2)$  time rather than  $\mathcal{O}(d^3)$  for a direct SVD. We first take QR factorizations  $\mathbf{B} = \mathbf{Q}_B \mathbf{R}_B$  and  $\mathbf{A}^\top = \mathbf{Q}_A \mathbf{R}_A$ , then SVD the small product:

$$\mathbf{U}, \boldsymbol{\sigma}, \mathbf{V}^\top = \text{SVD}(\mathbf{R}_B \mathbf{R}_A^\top)$$

and finally project back via  $\mathbf{U} \leftarrow \mathbf{Q}_B \mathbf{U}$  and  $\mathbf{V}^\top \leftarrow \mathbf{V}^\top \mathbf{Q}_A^\top$ . The normalized representation  $\Delta \tilde{\mathbf{W}} = \Delta \mathbf{W} / \|\Delta \mathbf{W}\|_F$  is captured implicitly by  $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma} / \|\boldsymbol{\sigma}\|_2$ , since  $\|\Delta \mathbf{W}\|_F = \|\boldsymbol{\sigma}\|_2$ .

## A.2. Frobenius distance

For a pair of adapters  $(i, j)$  at a given layer–module pair  $\ell \in \mathcal{L}$ , the normalized Frobenius distance is

$$\begin{aligned} d_\ell(i, j) &= \left\| \Delta \tilde{\mathbf{W}}_i^\ell - \Delta \tilde{\mathbf{W}}_j^\ell \right\|_F \\ &= \sqrt{2 - 2 \left\langle \Delta \tilde{\mathbf{W}}_i^\ell, \Delta \tilde{\mathbf{W}}_j^\ell \right\rangle_F} \end{aligned}$$

using the identity

$$\|A - B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 - 2 \langle A, B \rangle_F$$

together with  $\|\Delta \tilde{\mathbf{W}}\|_F = 1$ . The inner product is evaluated directly from the SVD factors, without ever materializing  $\Delta \mathbf{W}$ :

$$\left\langle \Delta \tilde{\mathbf{W}}_1, \Delta \tilde{\mathbf{W}}_2 \right\rangle_F = \tilde{\boldsymbol{\sigma}}_1^\top (\mathbf{M}_U \odot \mathbf{M}_V) \tilde{\boldsymbol{\sigma}}_2$$

where  $\mathbf{M}_U = \mathbf{U}_1^\top \mathbf{U}_2$  and  $\mathbf{M}_V = \mathbf{V}_1^\top \mathbf{V}_2$  are both in  $\mathbb{R}^{r \times r}$ , and  $\odot$  denotes the Hadamard (element-wise) product. The order of factors in  $\mathbf{M}_V$  matters:  $\mathbf{V}_1^\top \mathbf{V}_2$  is  $r \times r$ , whereas  $\mathbf{V}_1 \mathbf{V}_2^\top$  would be  $d_{\text{in}} \times d_{\text{in}}$  and defeat the purpose. The total cost is  $\mathcal{O}(r^2(d_{\text{out}} + d_{\text{in}}))$  per adapter pair, roughly 1000 times cheaper than the dense computation, which for  $r = 8$  is two to three orders of magnitude at the layer widths of current models.

The aggregated distance is computed as the Euclidean norm over per-layer distances as

$$d(i, j) = \sqrt{\sum_{\ell \in \mathcal{L}} d_\ell(i, j)^2}$$

where  $\mathcal{L}$  is the set of selected layer–module pairs. Each layer is normalized independently before aggregation, so that layers with larger weight magnitudes do not dominate the aggregate.

## B. Prompt template

We report the exact template used for training and evaluation. The `{context}` field is the verbalized entity document; for the CB condition it is the empty string. The `{ground_truth_answer}` field is used only for training.

You are a factual question-answering assistant. Answer the question  
 $\hookrightarrow$  below using only the information provided. Be concise and accurate.

Context: {context}

Question: {question}

Answer: {ground\_truth\_answer}