

Article

Symmetry-Aware Schema–Session Decoupled LoRA Adaptation for Continual Knowledge Graph Completion

Shuli Dong ^{1,2}, Yuyuan Dong ^{1,2}, Huanyu Zhang ^{1,2} and Ping Feng ^{1,2,3,*} 

¹ College of Computer Science and Technology, Changchun University, Changchun 130022, China; dongshuli103@gmail.com (S.D.); dyy030908@gmail.com (Y.D.); yli78294@gmail.com (H.Z.)

² Jilin Provincial Key Laboratory of Manchurian Railway Data Engineering and Intelligent Decision-Making, Changchun 130022, China

³ Education Key Laboratory of Intelligent Rehabilitation and Barrier-Free Access for the Disabled, Changchun University, Changchun 130022, China

* Correspondence: pingfeng20188@gmail.com

Abstract

Knowledge graph completion (KGC) is commonly studied under static settings, whereas real-world knowledge graphs evolve continuously with newly emerging entities, relations, and facts. In continual knowledge graph completion (CKGC), the model is required to absorb newly arrived knowledge while retaining historical prediction ability. A central difficulty is that incremental knowledge is heterogeneous: some information reflects reusable cross-session schema regularities, whereas other information corresponds to volatile session-specific factual updates. Existing CKGC methods usually model these signals within a single adaptation stream, which can increase interference during sequential learning. To address this issue, we propose a schema–session decoupled adaptation framework for session-aware CKGC. Specifically, the framework introduces a shared Schema-LoRA to accumulate reusable schema knowledge across sessions and a session-specific Session-LoRA to capture local instance-level updates in each session. To further improve adaptation stability, Orthogonal Gradient Projection and Session-EWC are applied to the instance-adaptation branch to reduce harmful gradient interference and excessive parameter drift during current-session training. In addition, we adopt a two-stage retrieve–rerank inference pipeline that combines lightweight candidate retrieval with language model reranking to balance effectiveness and efficiency. Experimental results on both static and continual KGC benchmarks show that the proposed framework achieves competitive static performance and improves top-rank prediction and final averaged performance over observed sessions under the session-aware CKGC protocol. From a functional-symmetry perspective, Schema-LoRA and Session-LoRA act as two additive adaptation branches around the frozen backbone, separately modeling reusable schema regularity and evolving factual variation. These findings suggest that schema–session decoupling is an effective design choice for session-aware continual knowledge graph completion.

Keywords: continual knowledge graph completion; parameter-efficient fine-tuning; LoRA; continual learning; symmetry-aware adaptation; schema–session symmetry



Academic Editor: Zhixun Su

Received: 23 April 2026

Revised: 18 May 2026

Accepted: 29 May 2026

Published: 1 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Knowledge graphs (KGs) represent real-world facts in the form of triples (h, r, t) and play an important role in applications such as question answering, recommendation, and information retrieval. Since real-world KGs are inevitably incomplete, knowledge graph

completion (KGC) aims to infer missing links from observed triples. Over the past decade, many effective KGC methods have been proposed, including translation-based models such as TransE [1], bilinear models such as DistMult [2] and ComplEx [3], and structure-aware approaches such as R-GCN [4], RotatE [5], and CompGCN [6]. These methods have achieved strong performance on static benchmarks. However, most existing KGC methods assume a closed and stationary setting, where the historical graph is fully available during training and the data distribution remains unchanged during evaluation.

In practice, however, knowledge graphs evolve continuously: new entities, relations, and facts emerge over time, and previously learned local patterns may also shift. This makes continual knowledge graph completion (CKGC) a more realistic and challenging setting. In CKGC, a model must not only adapt to newly arriving knowledge but also preserve historical prediction ability across multiple sessions. A key difficulty is that incremental knowledge in evolving KGs is inherently heterogeneous. On the one hand, relatively stable cross-session schema regularities exist, such as relation patterns, empirical-type tendencies, and reusable topological constraints. On the other hand, newly arriving knowledge also contains volatile session-local factual updates, including newly observed links and neighborhood changes. If these heterogeneous signals are forced into a single continuously updated parameter space, interference between old and new knowledge can be intensified, thereby aggravating forgetting. Therefore, an effective CKGC model should distinguish reusable schema priors from session-specific instance updates during continual adaptation.

Recent advances in pre-trained language models and large language models have opened new directions for KGC. Compared with traditional embedding-based methods, language model-based approaches can better exploit entity names, textual descriptions, and contextual semantics, and therefore show promise in long-tail and weakly structured scenarios. Representative methods such as KG-BERT [7], SimKGC [8], and KICGPT [9] demonstrate the effectiveness of semantic modeling and retrieve-rerank inference for static KGC. Nevertheless, most existing PLM/LLM-based KGC methods still focus on static graphs and do not explicitly address catastrophic forgetting, parameter-efficient continual adaptation, or the heterogeneity of incremental knowledge under evolving graph sessions.

Figure 1 illustrates the motivation of our framework. In continual KGC, schema-level knowledge and instance-level knowledge play different roles. Schema-level knowledge is relatively stable. For example, the relation pattern of a relation, its empirical domain and range, and its type tendency can often be reused across sessions. In contrast, instance-level knowledge is more local and time-sensitive, such as a newly added fact or a changed neighborhood around an entity. If these two types of knowledge are updated in the same parameter space, the new factual updates may interfere with previously learned reusable patterns. This motivates us to separate schema adaptation from session-specific instance adaptation.

Classical continual learning methods such as EWC [10] and OGD [11] provide useful tools for mitigating catastrophic forgetting. However, directly applying them to LLM-based CKGC remains insufficient. Full-parameter tuning is costly in a multi-session setting, and newly observed KG facts contain both reusable schema-level signals and conflict-prone instance-level updates. From a functional-symmetry perspective, Schema-LoRA and Session-LoRA serve as two complementary adaptation branches around the same frozen backbone: the former models reusable cross-session regularities, while the latter captures session-specific factual variation.

Motivated by these observations, we study CKGC under a session-aware setting, where the session identity of each query is available at inference time. Under this protocol, our goal is not to solve session-agnostic routing over all historical instance memories,

but to achieve cross-session schema sharing, session-specific factual isolation, and stable parameter-efficient continual adaptation. To this end, we propose a schema–session decoupled adaptation framework built on a frozen large language model backbone. Specifically, the framework introduces a shared Schema-LoRA to accumulate reusable schema knowledge across sessions and a session-specific Session-LoRA to capture local factual updates of the current session [12]. To further stabilize continual instance adaptation, we incorporate Orthogonal Gradient Projection (OGP) and Session-EWC into the optimization of the Session-LoRA branch. During inference, we adopt a two-stage retrieve–rerank pipeline that combines lightweight candidate retrieval with language model reranking, thereby balancing prediction quality and computational efficiency. The main contributions of this paper are as follows:

- We formulate session-aware CKGC as a heterogeneous adaptation problem by distinguishing reusable cross-session schema priors from volatile session-local factual updates.
- We propose a schema–session decoupled PEFT framework with a shared Schema-LoRA for schema accumulation and session-specific Session-LoRA modules for factual isolation, where Schema-LoRA is further restricted by a shared schema subspace.
- We stabilize the instance branch with OGP and Session-EWC. We further adopt a retrieve–rerank pipeline for efficient inference. Together, these designs improve top-rank prediction and final averaged performance over observed sessions under the session-aware CKGC setting.

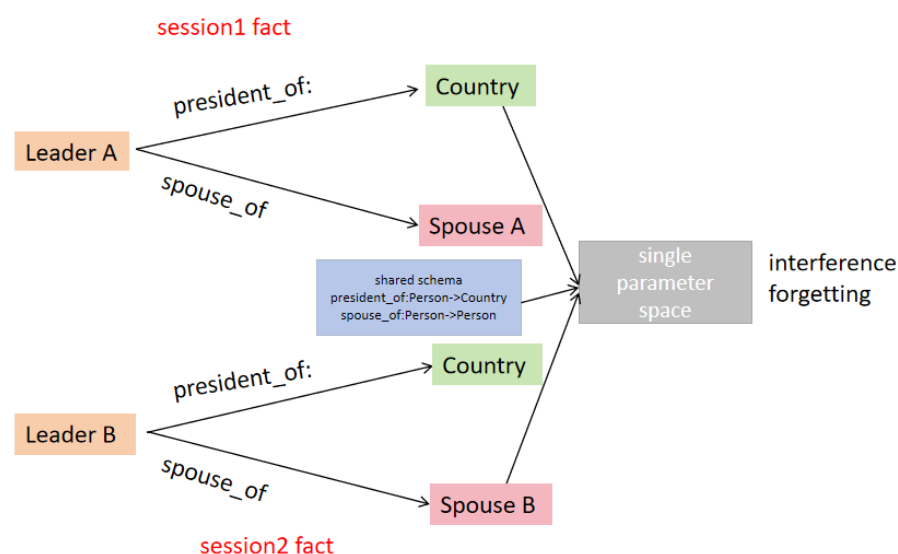


Figure 1. Motivation of schema–session decoupling in continual KGC. Stable schema knowledge, such as relation patterns and type tendencies, can be reused across sessions. Session-specific factual knowledge, such as newly added triples and local neighborhood changes, may vary across sessions. Updating both types of knowledge in one shared parameter space may cause interference between reusable schema regularities and volatile factual updates, which motivates our decoupled adaptation design.

2. Related Work

This section reviews the literature most relevant to our work. Existing research can be broadly categorized into three areas: static knowledge graph completion methods, knowledge graph completion methods based on pre-trained language models and large language models, and continual knowledge graph completion and anti-forgetting methods. The first two categories of work have advanced the development of knowledge graph completion from the perspectives of schema representation learning and semantic modeling, respectively, while the latter category focuses on dynamic evolutionary scenarios, addressing the

balance between learning new knowledge and preserving historical knowledge. Based on this taxonomy, we analyze prior work from the perspectives of schema modeling capability, knowledge representation, and continual adaptation, and clarify how our method differs from existing approaches.

2.1. Static Knowledge Graph Completion

Early knowledge graph completion primarily relied on embedding-based methods, which learned low-dimensional representations for entities and relationships and designed scoring functions to assess the validity of triples. TransE models relationships as vector translations, while DistMult and ComplEx adopt bilinear and complex-space models, respectively, to capture entity-relationship interactions; RotatE, on the other hand, uses rotations in the complex plane to uniformly characterize various relationship patterns. Furthermore, methods such as R-GCN and CompGCN have introduced graph neural networks into KGC, enhancing the ability to model multi-hop structures through relation-aware message passing. While these methods are quite mature in static, closed-vocabulary scenarios, they typically rely on directly learning parameterized vectors for entities and relations. In scenarios where new entities constantly emerge and the knowledge graph continues to grow, they often require significant retraining overhead, making it difficult to directly adapt them to continual KGC.

2.2. PLM/LLM-Based KGC

With the advancement of pre-trained language models, researchers have begun to frame KGC as a text modeling problem. KG-BERT encodes triples and their textual descriptions as sequences and uses BERT to classify candidate facts. Such methods can leverage textual semantics to mitigate the reliance of traditional embedding-based approaches on fixed entity vocabularies and explicit graph structures. Subsequently, SimKGC trains textual representations through contrastive learning and decouples the candidate retrieval and scoring processes, thereby balancing semantic modeling capabilities with inference efficiency. KICGPT, on the other hand, combines a schema retriever with a large language model-based reranker, demonstrating the potential of schema retrieval and linguistic reasoning in KGC [13,14]. Overall, existing language model-based KGC approaches have gradually evolved from direct sequence classification toward a retrieval-enhanced and reranking-based approach. However, related research has primarily focused on static graphs, with limited attention paid to parameter adaptation and historical knowledge retention during continual updates.

2.3. Continual KGC and Forgetting Mitigation

The goal of continual learning is to achieve incremental learning on sequential tasks or non-stationary data distributions while minimizing catastrophic forgetting. Representative methods include regularization-based EWC, gradient-constrained OGD, and methods based on experience replay. In recent years, continual knowledge graph completion has gradually emerged as a specialized research area. Related work focuses on how models can learn new knowledge while maintaining historical performance without accessing all historical data. FastKGE [15], a key representative in this field, enhances the parameter efficiency of continual KGE through incremental LoRA and adaptive rank allocation. However, existing methods generally lack explicit modeling of the internal heterogeneity of new knowledge and rarely distinguish between the different roles of cross-session stable schema knowledge and current-stage local instance knowledge in the parameter space [16]. We address this gap by employing a two-layer LoRA architecture to map different types of knowledge to distinct parameter subspaces, while concentrating the

anti-forgetting constraint on the instance parameter space, thereby enhancing the stability of continual adaptation.

Recent studies have further explored continual and language model-based KGC. For example, BAKE formulates continual knowledge graph embedding as a Bayesian-guided sequential inference problem [17]. STARK studies structure-aware and adaptive representation learning for continual knowledge graph embedding [18]. Parameter-Efficient Continual Automated KGC investigates parameter-efficient continual adaptation for automated KGC with LLMs [19]. KG-BiLM explores bidirectional language models for knowledge graph embedding by combining structural and textual semantic modeling [20]. These recent studies show that continual adaptation and language model-based KGC are becoming increasingly important. However, explicit schema–session decoupling between reusable schema priors and session-local factual updates in a frozen-LLM PEFT framework remains underexplored. To make the research gap clearer, Table 1 summarizes representative related work and compares it with our method.

Table 1. Summary of representative related work and the research gap addressed by our method.

Line of work	Representative Methods	Main Gap Relative to Our Method
Static and retrieval-enhanced KGC	TransE, RotatE, KG-BERT, SimKGC, KICGPT	Mainly static; limited continual adaptation.
Generic continual learning	EWC, OGD, GEM	Not tailored to KG schema–instance heterogeneity.
Continual/LoRA-based KGC	LKGE, ETT-CKGE, FastKGE	Usually uses single-stream adaptation.
Recent CKGC/LLM-KG methods	BAKE, STARK, KG-BiLM, CGKGC	Limited schema–session decoupling with frozen LLMs.
Ours	Schema-LoRA + Session-LoRA	Decouples reusable schema priors and session-specific facts.

As summarized in Table 1, existing studies mainly focus on static KGC, language model-based semantic inference, generic forgetting mitigation, or single-stream continual adaptation. Our work follows a session-aware CKGC setting, where the session identity is available at inference time and the corresponding Session-LoRA can be activated. Compared with KICGPT, we target continual parameter adaptation rather than only retrieval-enhanced static inference; compared with FastKGE, we explicitly decouple reusable schema priors from session-specific factual updates; and compared with generic methods such as EWC and OGD, we apply stability control only to the instance branch where update conflicts are expected to concentrate.

3. Method

3.1. Definition of the Problem

We consider a temporally evolving sequence of continual knowledge graph sessions, $\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \dots, \mathcal{G}^{(T)}\}$. For session t , the graph snapshot is defined as

$$\mathcal{G}^{(t)} = (\mathcal{E}^{(t)}, \mathcal{R}^{(t)}, \mathcal{S}^{(t)}) \quad (1)$$

where $\mathcal{E}^{(t)}$, $\mathcal{R}^{(t)}$, and $\mathcal{S}^{(t)}$ denote the entity set, relation set, and triple set at session t , respectively. As the graph evolves, we assume monotonic expansion:

$$\mathcal{E}^{(1)} \subseteq \mathcal{E}^{(2)} \subseteq \dots \subseteq \mathcal{E}^{(T)}, \quad \mathcal{R}^{(1)} \subseteq \mathcal{R}^{(2)} \subseteq \dots \subseteq \mathcal{R}^{(T)} \quad (2)$$

At session t , the model can access only the current training triples $\mathcal{S}_{\text{train}}^{(t)}$ and cannot revisit the full historical training data. The goal of continual knowledge graph completion is to preserve link-prediction ability on historical sessions while sequentially adapting to newly observed knowledge.

In this work, we distinguish two types of knowledge. The first is schema knowledge, including relation patterns, empirical-type constraints, relation domain and relation range tendencies, and reusable topological regularities that recur across sessions. Such knowledge is comparatively stable and serves as cross-session schema prior information. The second is instance knowledge, including local facts newly introduced in the current session, session-specific neighborhood changes, and concrete entity-level connections. Compared with schema knowledge, instance knowledge is more localized, more time-sensitive, and more likely to conflict with previously learned information.

Motivated by this observation, we model continual KGC as two complementary adaptation processes: schema adaptation and instance adaptation. Schema adaptation accumulates reusable cross-session priors, while instance adaptation absorbs local factual changes specific to the current session. Accordingly, our framework uses a shared Schema-LoRA for cross-session schema knowledge and a session-specific Session-LoRA for instance-level changes.

We focus on the session-aware continual KGC setting. Specifically, the session identity of each query is available at evaluation time. Therefore, when a test sample belongs to session t , the model invokes the shared Schema-LoRA together with Session-LoRA $_t$. Under this protocol, our objective is to achieve cross-session schema sharing, session-specific factual isolation, and stable parameter-efficient continual adaptation, rather than session-agnostic routing across historical instance memories. Under this formulation, our method follows a functional-symmetry-aware design principle. Here, symmetry does not refer to geometric symmetry or group-theoretic invariance. Instead, it refers to a paired and complementary adaptation structure around the same frozen backbone:

$$\mathcal{S}_{\text{fun}}^{(t)} = (\mathcal{A}_{\text{sch}}, \mathcal{A}_{\text{ses}}^{(t)}), \quad (3)$$

where $\mathcal{A}_{\text{sch}}$ is the shared schema-adaptation branch and $\mathcal{A}_{\text{ses}}^{(t)}$ is the session-specific instance-adaptation branch. The two branches are symmetric in their additive role around the frozen backbone, but they are different in knowledge content and update behavior. Schema-LoRA accumulates reusable cross-session regularities, whereas Session-LoRA captures local factual variation.

3.2. Overall Framework

The design principle of our framework is to decouple heterogeneous knowledge types before adaptation. As shown in Figure 2, the framework consists of a frozen language model backbone, a shared Schema-LoRA, and session-specific Session-LoRA modules. Schema-LoRA captures reusable schema-level regularities across sessions, whereas each Session-LoRA stores session-local factual updates. During training on session t , only the shared Schema-LoRA and the current Session-LoRA $_t$ are updated. During inference, the model activates the frozen backbone, the shared Schema-LoRA, and the Session-LoRA associated with the query session.

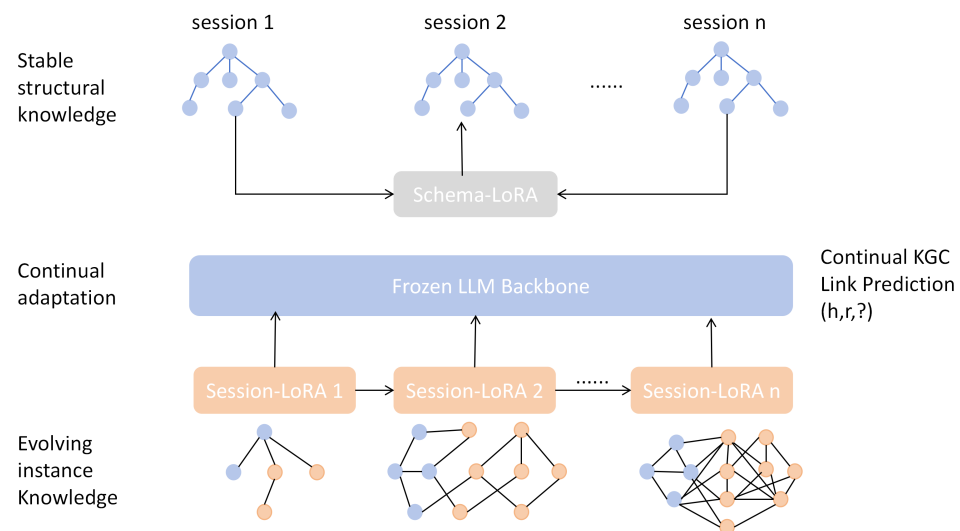


Figure 2. Overall framework of the proposed schema-session decoupled adaptation method. The frozen LLM backbone provides stable semantic representations and is not updated during continual training. The shared Schema-LoRA accumulates reusable schema-level regularities across sessions, while each Session-LoRA captures session-specific factual updates. At inference time, the model uses the frozen backbone, the shared Schema-LoRA, and the Session-LoRA associated with the query session.

Role of the Frozen LLM Backbone

The LLM backbone is kept frozen throughout all continual sessions and serves as a stable semantic encoder. We freeze the backbone because full-parameter tuning is costly in multi-session continual learning, and repeatedly updating the same large parameter space may mix old and new knowledge and increase forgetting. Freezing the backbone localizes continual adaptation into lightweight LoRA modules: Schema-LoRA is shared and slowly updated to accumulate reusable schema regularities, while Session-LoRA is session-specific and captures local factual updates. Therefore, the backbone is not used to store continual knowledge. In all sessions, θ_{backbone} is excluded from gradient updates; only θ_{sch} and the current $\theta_{\text{ses},t}$ are trainable.

For the l -th injected linear layer, where $l = 1, \dots, L$, let the frozen backbone weight be $W_l^{(0)} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. The effective weight in session t is

$$W_l^{(t)} = W_l^{(0)} + \Delta W_l^{\text{sch}} + \Delta W_l^{\text{ses},t}, \quad (4)$$

where ΔW_l^{sch} is the shared schema update and $\Delta W_l^{\text{ses},t}$ is the session-specific instance update. This additive decomposition provides the mathematical basis for the functional symmetry in Equation (3): both branches modify the same frozen backbone, but ΔW_l^{sch} is shared across sessions whereas $\Delta W_l^{\text{ses},t}$ is specific to session t .

For the session-specific adapter, we use the standard low-rank parameterization:

$$\Delta W_l^{\text{ses},t} = A_l^{\text{ses},t} B_l^{\text{ses},t}, \quad (5)$$

where $A_l^{\text{ses},t}$ and $B_l^{\text{ses},t}$ are trainable low-rank factors. By contrast, the shared Schema-LoRA is further restricted by a shared schema subspace, as described in Section 3.3.

Based on this decomposition, the forward representation in session t is

$$h^{(t)}(x) = f(x; \theta_{\text{backbone}}, \theta_{\text{sch}}, \theta_{\text{ses},t}). \quad (6)$$

During training on session t , only θ_{sch} and $\theta_{\text{ses},t}$ are updated. The backbone θ_{backbone} and all previously learned Session-LoRA modules remain frozen. After session t , Session-LoRA $_t$ is frozen and stored for session-aware reuse. In this way, cross-session reusable schema knowledge is accumulated in Schema-LoRA, whereas instance knowledge from different stages is isolated in separate Session-LoRA modules. A complete pseudocode description of the overall session-wise training procedure is provided in Algorithm A1 in Appendix A.

3.3. Schema-LoRA: Schema Knowledge Modeling

Schema-LoRA models schema knowledge that remains relatively stable across sessions. Although specific facts change over time, higher-level schema information is often more reusable across sessions, including relation patterns, empirical-type constraints, relation domain/range tendencies, and recurring local topology. We treat this information as reusable schema-level priors and encode it through a shared adapter updated across sessions. Rather than treating schema information as another unconstrained low-rank update, we view it as a reusable cross-session prior and therefore restrict it to a shared schema subspace.

To extract dominant schema information in the current session, we construct relation-level features and form a relation-feature matrix X . These features include relation frequency, average head/tail degree, empirical domain/range tendencies, adjacency co-occurrence patterns, and simple two-hop topology cues. The matrix X is normalized and projected into a unified low-dimensional schema space before subspace extraction. We then apply singular value decomposition to the projected relation-level feature matrix. Let $Z_t = \text{Proj}_{d_{\text{sch}}}(\text{Norm}(X_t))$ denote the normalized and projected schema feature matrix of session t .

For each relation r in session t , we construct a compact schema feature vector to summarize its structural and type-level regularities:

$$x_r^{(t)} = [f_{\text{freq}}(r), f_{\text{hdeg}}(r), f_{\text{tdeg}}(r), f_{\text{dom}}(r), f_{\text{ran}}(r), f_{\text{cooc}}(r), f_{\text{2hop}}(r)]. \quad (7)$$

Here, f_{freq} denotes the normalized frequency of relation r in the current session, f_{hdeg} and f_{tdeg} denote the average log-degree of head and tail entities, f_{dom} and f_{ran} denote the dominant domain and range-type tendencies, f_{cooc} measures relation adjacency co-occurrence, and f_{2hop} captures length-two topology patterns involving relation r . Each feature dimension is normalized within the current session:

$$\hat{x}_{r,j}^{(t)} = \frac{x_{r,j}^{(t)} - \mu_j^{(t)}}{\sigma_j^{(t)} + \epsilon}, \quad (8)$$

where $\mu_j^{(t)}$ and $\sigma_j^{(t)}$ are the mean and standard deviation of the j -th feature over all relations in session t . The normalized relation vectors are stacked into X_t , and the projected schema feature matrix is $Z_t = \text{Proj}_{d_{\text{sch}}}(\text{Norm}(X_t))$.

The singular value decomposition of Z_t is given by

$$Z_t = P_t \Sigma_t Q_t^\top. \quad (9)$$

Let $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p$ denote the singular values in Σ_t in Equation (9), where $p = \text{rank}(Z_t)$. The retained schema rank is selected according to the cumulative spectral energy:

$$E_t(q) = \frac{\sum_{i=1}^q \sigma_i^2}{\sum_{i=1}^p \sigma_i^2}. \quad (10)$$

The current-session schema rank is defined as the smallest rank whose cumulative energy reaches the threshold τ_{sch} :

$$r_t^{\text{cur}} = \min\{q \in \{1, \dots, p\} \mid E_t(q) \geq \tau_{\text{sch}}\}. \quad (11)$$

The corresponding dominant directions define the current-session schema basis.

To update the shared schema space, we merge the current-session basis with the historical shared basis and perform a new decomposition followed by truncation. The retained shared rank is bounded by the maximum shared-subspace capacity:

$$r_t^{\text{sch}} = \min(r_t^{\text{merge}}, r_{\text{sch}}^{\text{max}}, d_{\text{sch}}), \quad (12)$$

where r_t^{merge} is the rank obtained after re-decomposition of the merged schema basis. This concatenate–re-decompose–truncate procedure enables continual accumulation of reusable schema information while preventing unbounded growth of the shared schema subspace.

Accordingly, Schema-LoRA is explicitly constrained in the shared schema subspace:

$$\Delta W_l^{\text{sch}} = U^{\text{sch}} M_l \quad (13)$$

where U^{sch} is the shared schema basis extracted from accumulated relation-level schema statistics, and M_l is the layer-specific coefficient matrix that maps this shared schema basis to the parameter space of layer l . Thus, what is shared across sessions is schema prior directionality, while each layer keeps its own parameterized mapping.

In implementation, the schema feature matrix is projected to a d_{sch} -dimensional space with $d_{\text{sch}} = 16$. We set the cumulative energy threshold to $\tau_{\text{sch}} = 0.95$, which means that the retained schema directions must explain at least 95% of the spectral energy of the projected relation-level feature matrix. This setting is used to preserve the dominant schema regularities while filtering low-energy noisy directions. The maximum shared-subspace capacity is set to $r_{\text{sch}}^{\text{max}} = 32$, which serves as an upper bound during cross-session basis merging and prevents the shared schema space from growing without control. Because the effective retained rank is also bounded by d_{sch} , the actual retained rank in the reported experiments is determined by Equation (12). When explicit domain/range annotations are unavailable, we approximate them using empirical head–tail statistics from current-session training triples. The concatenate–re-decompose–truncate update of the shared schema basis is detailed in Algorithm A2 in Appendix A.

3.4. Session-LoRA: Instance Knowledge Modeling

Unlike Schema-LoRA, Session-LoRA models instance-level knowledge specific to the current session. In continual KGC, new knowledge is often reflected in local factual expansion and refinement, such as newly observed links among existing entities, neighborhood changes caused by new entities, and stage-specific instance patterns of relations. This knowledge is localized, time-sensitive, and session-dependent, and should not be mixed with shared schema priors in the same adapter space.

Therefore, we allocate an independent Session-LoRA for each session. Following Equation (5), the session-specific low-rank increment at layer l is parameterized by $A_l^{\text{ses},t}$ and $B_l^{\text{ses},t}$, and all session-specific parameters are denoted by

$$\Phi^{\text{ses},t} = \left\{ A_l^{\text{ses},t}, B_l^{\text{ses},t} \right\}_{l=1}^L \quad (14)$$

During session- t training, only $\Phi^{\text{ses},t}$ participates in instance-level gradient optimization; the backbone and historical Session-LoRA modules remain frozen. This stores

stage-wise instance knowledge in separate parameter slots and reduces direct overwriting across sessions.

Under our session-aware protocol, historical Session-LoRA modules are not updated in later sessions, which already alleviates classical historical-parameter overwrite. However, instability may still occur within the current session due to competing gradient signals from different relation patterns and newly added facts. Therefore, we introduce targeted stabilization for Session-LoRA optimization.

3.5. Stable Update Mechanism on Session-LoRA

Although session-wise parameter isolation reduces direct cross-session overwriting, current-session training can still suffer from gradient conflicts. We therefore introduce Orthogonal Gradient Projection (OGP) and Session-EWC. The overall training/inference workflow is shown in Figure 3.

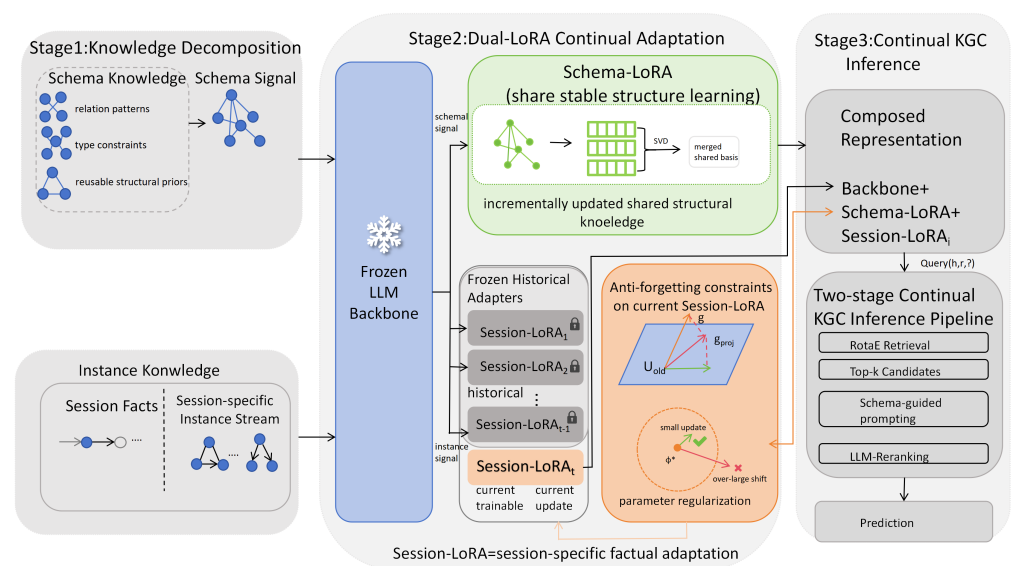


Figure 3. Detailed workflow of training and inference in the proposed framework. The workflow separates schema knowledge from instance knowledge to enable cross-session shared schema learning and session-specific factual adaptation, while mitigating forgetting through historical-adapter freezing and parameter regularization.

3.5.1. Orthogonal Gradient Projection

For each injected layer l , let g_l be the vectorized gradient of the current Session-LoRA parameters, and let U_l^{hist} be the orthonormal basis of historical dominant gradient directions. The projected effective gradient is

$$\tilde{g}_l = g_l - U_l^{\text{hist}} (U_l^{\text{hist}})^{\top} g_l \quad (15)$$

This removes gradient components aligned with historical dominant directions and preserves orthogonal components that are more specific to current-session adaptation.

Projection is performed layer-wise rather than globally, providing finer-grained conflict control. After each session, we sample gradients from multiple mini-batches, build layer-wise gradient matrices, and update historical subspaces via truncated decomposition. For session 1, $U_l^{\text{hist}} = \emptyset$, so OGP degenerates to ordinary gradient optimization.

3.5.2. Session-EWC

OGP constrains update directions but does not explicitly model current-session parameter importance. In practice, some Session-LoRA parameters quickly become critical

for current-session discrimination. To stabilize them, we introduce Session-EWC as a within-session importance regularizer:

$$\mathcal{L}_{\text{EWC}}^{(t)} = \lambda_{\text{ewc}} \sum_i F_{t,i} (\phi_i - \phi_i^*)^2 \quad (16)$$

where ϕ_i is a scalar parameter in the current Session-LoRA, ϕ_i^* is its reference value, and $F_{t,i}$ is the corresponding diagonal Fisher importance.

Different from classical EWC, which protects historical task parameters from future updates, Session-EWC here is applied only during training of the current session to stabilize important current-session parameters.

We adopt a two-stage strategy in each session. First, a short warm-up stage optimizes only $\mathcal{L}_{\text{kgc}}$. After warm-up, the current parameters are saved as ϕ^* , and the Fisher diagonal F_i is estimated for current Session-LoRA parameters using squared gradients of $\mathcal{L}_{\text{kgc}}$ over warm-up mini-batches. Session-EWC is then activated during the remaining optimization steps.

In all experiments, warm-up lasts 200 update steps (approximately 0.3 epoch), and $\lambda_{\text{ewc}} = 50$ for all injected Session-LoRA layers.

3.5.3. Joint Optimization Objective

The overall objective for session t is

$$\mathcal{L}_{\text{total}}^{(t)} = \mathcal{L}_{\text{kgc}}^{(t)} + \mathcal{L}_{\text{EWC}}^{(t)} \quad (17)$$

The optimization variables in session t are the shared Schema-LoRA parameters and the current Session-LoRA parameters, while the backbone and all historical Session-LoRA modules remain frozen. Gradients of current Session-LoRA are computed from $\mathcal{L}_{\text{total}}^{(t)}$, projected by OGP, and then used for optimization. Meanwhile, shared Schema-LoRA is updated with a smaller learning rate. OGP controls update direction, while Session-EWC controls important-parameter drift; their combination improves adaptation stability. Algorithm A3 in Appendix A details the stable training procedure of the current Session-LoRA.

3.6. Training and Inference

To balance prediction quality and efficiency under large entity spaces, we use a two-stage pipeline: candidate retrieval and language model reranking. Given a query $(h, r, ?)$, we first use a lightweight structural retriever (RotatE) to recall top- k_{ret} candidates from the full entity set, and then rerank top- k_{rerank} candidates using the backbone together with Schema-LoRA and Session-LoRA _{t} . The retrieval size k_{ret} and reranking size k_{rerank} control the trade-off between candidate coverage and inference cost. A smaller k_{ret} may exclude the correct entity from the retrieved candidate set, thereby limiting the upper bound of the reranking stage. A larger k_{ret} improves candidate coverage but introduces more candidates into the pipeline. Similarly, increasing k_{rerank} allows the language model to compare more candidates, but it also increases inference cost because each candidate requires language model scoring. We set $k_{\text{ret}} = 100$ and $k_{\text{rerank}} = 40$ as a practical balance between candidate coverage, ranking quality, and efficiency. The detailed pseudocode of the session-aware retrieve-rerank inference procedure is provided in Algorithm A4 in Appendix A.

For tail prediction, we construct a schema-guided prompt:

$$p_{\text{tail}}(h, r, e) = \text{Template}(h, r, e, c(r)) \quad (18)$$

where $c(r)$ is a relation-specific schema description, including, when available, short-relation semantics, domain/range tendencies, and compact schema summaries derived from current-session relation statistics. Head prediction is defined analogously by replacing the missing tail with the missing head and constructing the corresponding candidate set.

In practice, we instantiate the schema-guided prompt with simple natural language templates. For tail prediction, the prompt is

Tail prediction prompt. Relation schema: [relation description]. The head entity usually belongs to [head-type tendency], and the tail entity usually belongs to [tail-type tendency]. Query: Given head entity [h] and relation [r], is [e] a plausible tail entity? Answer: true or false.

For head prediction, we use the symmetric form

Head prediction prompt. Relation schema: [relation description]. The head entity usually belongs to [head-type tendency], and the tail entity usually belongs to [tail-type tendency]. Query: Given relation [r] and tail entity [t], is [e] a plausible head entity? Answer: true or false.

Here, the relation description and type tendencies are obtained from the compact schema summary $c(r)$. The language model scores each candidate according to the likelihood of the positive answer, and candidates are then ranked by the resulting scores. The head prediction score is computed analogously.

The reranking score of candidate e is

$$s(e \mid h, r) = \text{score}(p_{\text{tail}}(h, r, e); \theta_{\text{backbone}}, \theta_{\text{sch}}, \theta_{\text{ses}, t}) \quad (19)$$

and the normalized conditional probability over candidate set $\mathcal{C}(h, r)$ is

$$P(e \mid h, r) = \frac{\exp(s(e \mid h, r))}{\sum_{e' \in \mathcal{C}(h, r)} \exp(s(e' \mid h, r))} \quad (20)$$

We compute both tail and head losses:

$$\mathcal{L}_{\text{kgc}}^{(t)} = \mathcal{L}_{\text{tail}}^{(t)} + \mathcal{L}_{\text{head}}^{(t)} \quad (21)$$

Thus, KGC is optimized as conditional ranking over a restricted candidate set instead of full-vocabulary scoring.

3.7. Complexity and Storage Overhead Analysis

The additional cost comes mainly from three components: shared Schema-LoRA, session-expanded Session-LoRA modules, and cached historical gradient subspaces for OGP. For each new session, newly trainable parameters are primarily those of the current Session-LoRA. Since the backbone stays frozen, per-session growth is much smaller than full-parameter continual tuning; specifically, per-session trainable growth is proportional to retained rank and the number of injected layers.

Shared Schema-LoRA is reused across sessions, and its schema basis is bounded by truncated decomposition and maximum-capacity control. Historical gradient subspaces are also maintained with layer-wise truncation, so OGP cache storage is bounded by the retained subspace rank per layer. Hence, both shared schema storage and historical gradient storage remain bounded over long horizons.

During training, only shared Schema-LoRA and current Session-LoRA are updated; historical Session-LoRA modules are frozen. Therefore, training cost depends mainly on the one-session low-rank scale rather than growing linearly with the number of past

sessions. During inference for session i , only the backbone, shared Schema-LoRA, and Session-LoRA $_i$ are activated.

Practically, the retrieve–rerank pipeline further reduces language model scoring cost: RotatE recalls the top-100 candidates, and only the top-40 candidates are reranked by the language model. Combined with the frozen backbone and low-rank adaptation, the full pipeline runs on a single 24 GB GPU. Overall, compared with full-parameter continual tuning, the main efficiency advantage of our method lies in bounded adapter-level parameter growth and controlled per-session storage.

4. Experiments

4.1. Experimental Setup

4.1.1. Datasets and Task Settings

To comprehensively evaluate effectiveness in both static KGC and continual KGC scenarios, we design two groups of experiments. Static experiments verify whether the proposed dual-LoRA architecture itself has strong KGC modeling capability without continual learning constraints. Continual experiments verify whether, after introducing instance-adaptation stability constraints, the model can achieve strong final averaged performance under dynamic graph evolution. All continual experiments follow the session-aware protocol defined in Section 3.1.

For static KGC, we use two widely adopted benchmarks: FB15K-237 [21] and WN18RR [22]. FB15K-237 is derived from FB15K by removing many inverse-relation patterns, making it more suitable for evaluating genuine relational reasoning ability. WN18RR is derived from WN18 by removing reciprocal relations, and it contains strong lexical semantics and hierarchical relation characteristics. These two datasets are commonly used filtered link-prediction benchmarks. Dataset statistics are reported in Table 2.

Table 2. Dataset statistics of FB15K-237 and WN18RR. Here, #Ent, #Rel, #train, #valid, and #test denote the numbers of unique entities, relations, training triples, validation triples, and test triples, respectively.

Dataset	#Ent	#Rel	#Train	#Valid	#Test
WN18RR	40,943	11	86,835	3034	3134
FB15K-237	14,541	237	272,115	17,535	20,466

For continual KGC, we use FB-CKGE and WN-CKGE, following the benchmark construction released by FastKGE. Each dataset is partitioned into five consecutive sessions to simulate realistic graph expansion over time. In each new session, the model receives only the current snapshot data and is required to adapt without revisiting the full historical training set. Snapshot statistics across sessions are shown in Table 3.

For continual evaluation, we follow the FastKGE protocol. After training on each session, the model is evaluated on the test sets of both the current and all historical sessions. We then report the average of MRR, Hits@1, Hits@3, and Hits@10 across all observed sessions at that stage. Final continual results are reported after training on the last session and aggregating performance over all historical test sets. This protocol evaluates not only new-session adaptation but also final prediction quality on historical sessions after continual training.

Table 3. Statistics of FB-CKGE and WN-CKGE under different sessions. NE, NR, and NT denote the numbers of entities, relations, and triples, respectively.

Dataset	Session	NE	NR	NT
FB-CKGE	0	7505	237	186,070
FB-CKGE	1	11,258	237	31,012
FB-CKGE	2	13,134	237	31,012
FB-CKGE	3	14,072	237	31,012
FB-CKGE	4	14,541	237	31,010
WN-CKGE	0	24,567	11	55,801
WN-CKGE	1	28,660	11	9300
WN-CKGE	2	32,754	11	9300
WN-CKGE	3	36,848	11	9300
WN-CKGE	4	40,943	11	9302

4.1.2. Baselines

In static KGC experiments, we compare with representative methods from multiple categories, including traditional embedding methods, graph neural methods, and pre-trained/large language model methods. Specifically, the baselines include: TransE, DistMult, ComplEx, RotatE, TuckER [23], HAKE [24], CompGCN, HittER [25], Pretrain-KGE [26], KG-BERT, StAR [27], MEM-KGC [28], ChatGPT zero-shot [29], ChatGPT one-shot, KICGPT, and KICGPT-tsa. However, these baselines differ substantially in backbone architecture, textual input form, candidate-generation strategy, and training budget. Therefore, the static comparison should be interpreted primarily as a broad positioning experiment rather than a strictly controlled like-for-like comparison. In particular, results from methods with substantially different training pipelines are used to contextualize the performance level of our framework, rather than to support a claim of strict superiority under identical settings.

For continual KGC, we compare against mainstream CKGC baselines used in FastKGE, including PNN [30], CWR [31], GEM [32], EMR [33], DICGRL [34], SI [35], EWC, LKGE [36], ETT-CKGE [37], and FastKGE. These methods represent several major technical routes for continual learning, such as parameter isolation, replay-based strategies, regularization-based forgetting mitigation, and parameter-efficient continual adaptation.

To ensure protocol-level comparability in the continual setting, we follow the same dataset partition and evaluation procedure as FastKGE. All baseline results are taken from the original reported numbers because these methods differ substantially in backbone model, training budget, and candidate-generation strategy. Therefore, the static comparison is mainly used to position our method among representative technical routes, whereas the most direct comparisons should be made with methods that are closer in modeling paradigm, such as KICGPT and FastKGE-style approaches.

4.1.3. Evaluation Metrics and Implementation Details

We use standard filtered link-prediction metrics, including MRR, Hits@1, Hits@3, and Hits@10, for both static and continual experiments. We follow the filtered link-prediction protocol. For each test query, other known true triples are removed from the candidate ranking, so that the model is not penalized for ranking another correct entity above the target entity. Let N be the number of test queries and rank_i be the filtered rank of the correct entity for the i -th query. Mean Reciprocal Rank (MRR) is defined as

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^N \frac{1}{\text{rank}_i}. \quad (22)$$

Hits@K measures the proportion of test queries whose correct entity is ranked within the top K:

$$\text{Hits@K} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}(\text{rank}_i \leq K). \quad (23)$$

Hits@1 measures exact top-1 prediction accuracy, while Hits@3 and Hits@10 measure whether the correct entity appears among the top-3 and top-10 predictions, respectively. MRR reflects the overall ranking quality by assigning larger scores to higher-ranked correct entities.

Our method is repeated with three random seeds under the same hyperparameter setting, and averaged results are reported. Baseline results are cited from published papers or benchmark reports, as noted below the result tables.

For implementation, we use ChatGLM3-6B [38] as the frozen backbone model. The experimental configuration and key hyperparameters are summarized in Table 4. Both Schema-LoRA and Session-LoRA use rank 8, scaling factor 16, and dropout 0.05. Schema-LoRA adopts a smaller learning rate to support slow accumulation of reusable schema priors, whereas Session-LoRA uses a relatively larger learning rate to enable fast adaptation to session-specific instance changes.

All experiments were implemented in Python 3.10.13. Model programming, training, and inference were conducted using PyTorch 2.1.2, CUDA 12.1, Hugging Face Transformers 4.36.2, and PEFT 0.7.1. Data processing and statistical analysis were performed using NumPy 1.26.4.

Table 4. Experimental configuration and key hyperparameters.

Experimental Configuration	Parameter
Backbone model	ChatGLM3-6B
GPU	NVIDIA RTX 4090 24 GB
Batch size	2
Gradient accumulation	8
Max sequence length	64
Training epochs	1 schema epoch; 1 session epoch
Schema-LoRA rank	8
Session-LoRA rank	8
Schema-LoRA learning rate	5×10^{-5}
Session-LoRA learning rate	1×10^{-4}
Session-EWC coefficient	$\lambda_{\text{ewc}} = 50$
Retrieval size	$k_{\text{ret}} = 100$
Reranking size	$k_{\text{rerank}} = 40$
Random seeds	42/43/44

During inference, we adopt the retrieve–rerank pipeline introduced in Section 3.6. Specifically, RotatE first recalls the top- k_{ret} candidate entities, and the language model reranks the top- k_{rerank} candidates. The proposed OGP and Session-EWC are applied only to the current-session Session-LoRA parameters, rather than to the frozen backbone or the shared Schema-LoRA.

4.2. Static Knowledge Graph Completion Results

Before evaluating continual adaptation, we first examine whether the proposed dual-adaptor architecture is effective for standard static KGC. The main results on FB15K-237 and WN18RR are reported in Table 5. Overall, our method achieves competitive performance on both datasets, indicating that the static experiments mainly verify that the proposed

decoupled adapter architecture remains competitive in non-continual settings and does not sacrifice standard KGC modeling ability.

Table 5. Main results on static datasets.

Dataset	FB15K-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	0.279	0.198	0.376	0.441	0.243	0.043	0.441	0.532
DistMult	0.241	0.155	0.263	0.419	0.430	0.390	0.440	0.490
ComplEx	0.247	0.158	0.275	0.428	0.440	0.410	0.460	0.510
RotatE	0.338	0.241	0.375	0.533	0.476	0.428	0.492	0.571
TuckER	0.358	0.266	0.394	0.544	0.470	0.443	0.482	0.526
HAKE	0.346	0.250	0.381	0.542	0.497	0.452	0.516	0.582
CompGCN	0.355	0.264	0.390	0.535	0.479	0.443	0.494	0.546
HittER	0.344	0.246	0.380	0.535	0.496	0.449	0.514	0.586
Pretrain-KGE	0.332	–	–	0.529	0.235	–	–	0.557
KG-BERT	–	–	–	0.420	0.216	0.041	0.302	0.524
StAR	0.263	0.171	0.287	0.452	0.364	0.222	0.436	0.647
MEM-KGC (W/O EP)	0.339	0.249	0.372	0.522	0.533	0.473	0.570	0.636
MEM-KGC (W/EP)	0.346	0.253	0.381	0.531	0.557	0.475	0.604	0.704
ChatGPT _{zero-shot}	–	0.237	–	–	–	0.190	–	–
ChatGPT _{one-shot}	–	0.267	–	–	–	0.212	–	–
KICGPT	<u>0.412</u>	<u>0.327</u>	<u>0.448</u>	0.554	0.549	0.474	0.585	0.641
KICGPT _{-tsa}	0.410	0.321	0.430	0.581	0.564	0.478	0.612	<u>0.677</u>
Ours	0.421	0.336	0.462	<u>0.564</u>	<u>0.560</u>	<u>0.476</u>	<u>0.610</u>	0.675

Note. All baseline results are cited from published papers or benchmark reports. Bold indicates the best result, and underlined values indicate the second-best result.

On FB15K-237, our method achieves MRR = 0.421, outperforming KICGPT (0.412) and KICGPT-tsa (0.410). It also obtains the best Hits@1 among the strongest LLM-based baselines, showing that the proposed framework improves fine-grained ranking quality at the top of the candidate list. On WN18RR, our method reaches MRR = 0.560, which is competitive with the strongest reported methods and close to KICGPT-tsa (0.564), while remaining stronger than most embedding-based and PLM-based baselines. These results suggest that the proposed method achieves competitive performance on both a dense relational benchmark and a semantically structured lexical benchmark.

These static results should be interpreted as a positioning experiment rather than a strict claim of overall superiority. The proposed method performs competitively with strong PLM/LLM-based KGC baselines, especially on FB15K-237, while remaining close to the strongest result on WN18RR. This indicates that the schema–session decoupled adapter design does not sacrifice standard KGC modeling ability. The main focus of this work remains continual adaptation under evolving graph sessions.

4.3. Continual Knowledge Graph Completion Results

After validating static modeling ability, we further evaluate the proposed method in the continual KGC setting over five consecutive sessions on FB-CKGE and WN-CKGE. The main results are reported in Table 6. Overall, our method achieves the best MRR on both benchmarks, showing that the proposed framework achieves strong final averaged performance over the observed sessions.

On FB-CKGE, our method achieves MRR = 0.244, outperforming FastKGE (0.223) and ETT-CKGE (0.236). On WN-CKGE, our method reaches MRR = 0.173, again exceeding FastKGE (0.159) and other compared baselines. Across the two continual datasets, our method obtains an average MRR of 0.209, compared with 0.191 for FastKGE and 0.195 for

ETT-CKGE. These results indicate that the proposed schema–session decoupled adaptation strategy improves reciprocal-rank-based ranking quality under continual graph evolution.

Table 6. Main results on continual datasets.

Method	FB-CKGE				WN-CKGE			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
PNN	0.215	0.122	0.245	0.403	0.134	0.002	0.241	0.342
CWR	0.075	0.011	0.105	0.192	0.005	0.002	0.007	0.012
GEM	0.188	0.103	0.212	0.359	0.119	0.002	0.215	0.297
EMR	0.180	0.097	0.204	0.346	0.114	0.002	0.205	0.286
DICGRL	0.149	0.091	0.160	0.261	0.057	0.001	0.155	0.166
SI	0.187	0.102	0.211	0.359	0.115	0.001	0.209	0.289
EWC	0.218	0.124	<u>0.247</u>	<u>0.410</u>	0.136	0.003	0.248	0.338
LKGE	0.208	0.113	0.238	0.403	0.144	0.007	0.259	0.347
ETT-CKGE	<u>0.236</u>	<u>0.137</u>	–	0.428	0.153	0.080	–	0.385
FastKGE	0.223	0.131	0.257	0.405	<u>0.159</u>	0.015	<u>0.287</u>	0.356
Ours	0.244	0.159	0.246	0.384	0.173	<u>0.022</u>	0.298	<u>0.371</u>

Note. All baseline results are cited from published papers or benchmark reports. Bold indicates the best result, and underlined values indicate the second-best result.

The improvements are mainly reflected in MRR and top-rank prediction. The gains are less uniform on Hits@3 and Hits@10, suggesting that the proposed method primarily improves high-confidence ranking quality rather than uniformly outperforming all baselines on every cutoff metric. This trend is consistent with our design goal: separating reusable schema knowledge from session-specific factual updates while stabilizing the instance-adaptation branch with OGP and Session-EWC.

Table 7 reports the random-seed stability of our method. The standard deviations are 0.0006 on FB-CKGE and 0.0010 on WN-CKGE, indicating that the reported MRR results are stable across different random initializations.

Table 7. Random-seed stability analysis of the proposed method on continual KGC datasets. Results are reported as MRR over three random seeds.

Dataset	Seed 1	Seed 2	Seed 3	Mean $\pm$ Std
FB-CKGE	0.244	0.244	0.243	0.2437 $\pm$ 0.0006
WN-CKGE	0.173	0.174	0.172	0.1730 $\pm$ 0.0010

Sensitivity Analysis of Retrieval and Reranking Sizes

To examine the effect of the two-stage inference parameters, we conduct a compact sensitivity analysis on FB-CKGE. Since language model reranking is computationally more expensive than structural retrieval, we vary k_{ret} and k_{rerank} using a small number of representative settings.

Table 8 reports the sensitivity analysis of k_{ret} and k_{rerank} . Compared with $k_{\text{ret}} = 50$, increasing k_{ret} to 100 improves ranking performance, indicating that sufficient retrieval coverage is important for the retrieve–rerank pipeline. Increasing k_{rerank} from 40 to 80 brings only limited additional improvement, while requiring more language model scoring. Therefore, $k_{\text{ret}} = 100$ and $k_{\text{rerank}} = 40$ are used in the main experiments as a balanced setting between candidate coverage, ranking quality, and inference efficiency.

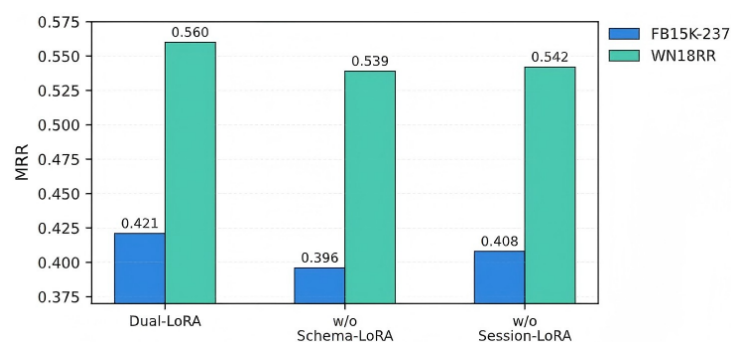
Table 8. Sensitivity analysis of k_{ret} and k_{rerank} on FB-CKGE.

k_{ret}	k_{rerank}	MRR	Hits@10
50	40	0.228	0.362
100	40	0.244	0.384
100	80	0.245	0.386

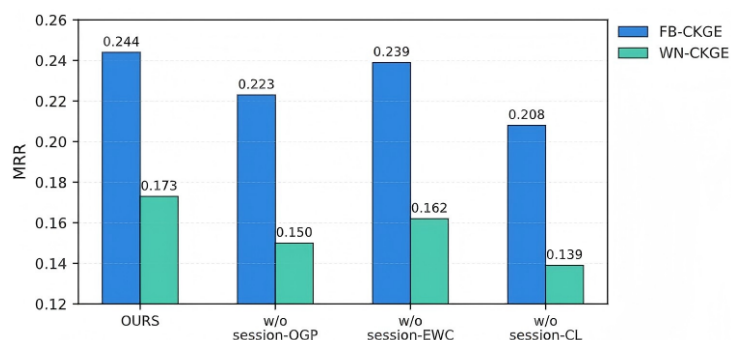
4.4. Ablation Study

To further understand the contribution of each component, we conduct two groups of ablation experiments, targeting the two central design axes of the proposed framework: knowledge-type decoupling and stable instance adaptation.

The first group of ablations evaluates the static KGC capability of the architecture by removing Schema-LoRA and Session-LoRA, respectively. As shown in Figure 4, both ablated variants perform worse than the full model on FB15K-237 and WN18RR, indicating that both branches make nontrivial contributions. Without Schema-LoRA, the model loses the ability to explicitly encode reusable schema priors such as relation patterns, type tendencies, and local structural regularities. Without Session-LoRA, the model still retains the shared schema branch but loses the flexible instance-level correction needed for precise triple discrimination. These results confirm that the two adapters are not redundant; instead, they capture complementary aspects of KGC reasoning.

**Figure 4.** Results of ablation experiments on the FB15K-237 and WN18RR datasets.

The second group of ablations evaluates the continual learning capability of the framework by removing Session-EWC and OGP, respectively. As shown in Figure 5, the full model consistently outperforms both ablated variants, demonstrating that the two stabilization mechanisms are complementary rather than interchangeable. Session-EWC constrains excessive drift of parameters that have already become important for current-session discrimination, whereas OGP suppresses harmful update directions that conflict with historical dominant gradients. Their joint action on Session-LoRA is therefore essential for stable incorporation of newly observed instance knowledge under sequential training.

**Figure 5.** Results of ablation experiments on the FB-CKGE and WN-CKGE datasets.

4.5. Result Analysis

The experimental results suggest that the proposed framework benefits mainly from the coordination of three components: schema–session knowledge decoupling, parameter-efficient adaptation, and targeted stability control. Schema-LoRA provides a shared space for reusable cross-session regularities, whereas Session-LoRA isolates session-local factual updates. OGP and Session-EWC further stabilize the instance-adaptation branch from the perspectives of gradient direction and parameter importance. This division of labor is consistent with the heterogeneous nature of evolving knowledge graphs and explains why the method improves MRR under the session-aware CKGC protocol while maintaining competitive static KGC performance.

5. Conclusions

Real-world knowledge graphs evolve continuously, requiring KGC models to adapt to new facts while retaining historical prediction ability. This paper proposes a schema–session decoupled adaptation framework for session-aware CKGC. The framework uses a frozen LLM backbone, a shared Schema-LoRA for reusable cross-session schema regularities, and session-specific Session-LoRA modules for local factual updates. To improve stability, OGP and Session-EWC are applied to the Session-LoRA branch, and a retrieve–rerank pipeline is used for efficient inference.

Experimental results show that the proposed method maintains competitive static KGC performance and improves final averaged MRR under the continual KGC setting. These results indicate that separating schema-level regularities from session-specific factual variation is beneficial for session-aware continual KGC.

At the same time, the current study has several limitations. First, the framework is developed under a session-aware CKGC setting, where the session identity of each query is available at inference time. Second, although the retrieve–rerank pipeline improves efficiency, the final prediction quality still depends on candidate coverage and prompt construction quality.

Future work can be extended in several directions. One direction is to move beyond the session-aware setting and study more flexible retrieval or routing mechanisms for session-agnostic continual KGC. Another direction is to incorporate richer schema priors, textual ontology information, or stronger structural abstraction mechanisms to better model complex and evolving relational regularities. It is also worthwhile to further improve the coordination among candidate retrieval, schema-guided prompting, and language model reranking, so as to enhance scalability and deployment efficiency in large-scale real-world knowledge graphs.

Author Contributions: Conceptualization, S.D., Y.D. and P.F.; methodology, S.D., Y.D. and H.Z.; software, S.D. and Y.D.; validation, S.D. and P.F.; formal analysis, S.D., Y.D. and H.Z.; investigation, P.F.; writing—original draft preparation, S.D. and Y.D.; writing—review and editing, S.D. and H.Z.; supervision, P.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a Key Research and Development Project of the Jilin Provincial Department of Science and Technology (Development of an Integrated Platform for Intelligent Mining of Jilin Manchurian Railway Historical Materials and Public Cultural Services Driven by Multimodal Semantics; Project No.: 20260203051SF).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors thank the anonymous reviewers for constructive comments.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Pseudocode

Algorithm A1 Overall Training Loop of the Schema–Session Decoupled CKGC Framework

Input: Continual sessions $\{\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(T)}\}$; $d_{\text{sch}}, \tau_{\text{sch}}, r_{\text{sch}}^{\max}, N_{\text{warm}}, \lambda_{\text{ewc}}$
Output: Shared adapter θ_{sch} and frozen session adapters $\{\theta_{\text{ses},1}, \dots, \theta_{\text{ses},T}\}$

- 1: Initialize $\theta_{\text{sch}}, U^{\text{sch}} \leftarrow \emptyset, \{U_l^{\text{hist}}\}_{l=1}^L \leftarrow \emptyset$
- 2: Initialize SESSIONBANK $\leftarrow \emptyset$
- 3: **for** $t = 1$ to T **do**
- 4: $U^{\text{sch}} \leftarrow \text{ALGORITHM A2}(\mathcal{S}_{\text{train}}^{(t)}, U^{\text{sch}}, d_{\text{sch}}, \tau_{\text{sch}}, r_{\text{sch}}^{\max})$
- 5: Initialize $\theta_{\text{ses},t} = \{A_l^{\text{ses},t}, B_l^{\text{ses},t}\}_{l=1}^L$
- 6: Freeze θ_{backbone} and all adapters in SESSIONBANK
- 7: $(\theta_{\text{ses},t}, \theta_{\text{sch}}, \{G_l^{(t)}\}_{l=1}^L) \leftarrow \text{ALGORITHM A3}(\mathcal{S}_{\text{train}}^{(t)}, \theta_{\text{ses},t}, \theta_{\text{sch}}, \{U_l^{\text{hist}}\}_{l=1}^L, N_{\text{warm}}, \lambda_{\text{ewc}})$
- 8: Freeze $\theta_{\text{ses},t}$ and add it to SESSIONBANK
- 9: **for** $l = 1$ to L **do**
- 10: $U_l^{\text{hist}} \leftarrow \text{Truncate}(\text{ReDecompose}([U_l^{\text{hist}}; G_l^{(t)}]))$
- 11: **end for**
- 12: **end for**
- 13: **return** $\theta_{\text{sch}}, \{\theta_{\text{ses},1}, \dots, \theta_{\text{ses},T}\}$

Algorithm A2 Shared Schema-Subspace Update (concatenate–re-decompose–truncate)

Input: $\mathcal{S}_{\text{train}}^{(t)}$, previous shared basis $U^{\text{sch}}, d_{\text{sch}}, \tau_{\text{sch}}, r_{\text{sch}}^{\max}$
Output: Updated shared basis U^{sch}

- 1: Build relation-level schema feature matrix X_t from $\mathcal{S}_{\text{train}}^{(t)}$
- 2: Normalize and project: $Z_t \leftarrow \text{Proj}_{d_{\text{sch}}}(\text{Norm}(X_t))$
- 3: Obtain current basis by SVD + energy truncation:
 $\hat{U}_t^{\text{sch}} \leftarrow \text{DominantDirections}(Z_t, \tau_{\text{sch}})$ ▷ basis definition follows Section 3.3
- 4: **if** $U^{\text{sch}} = \emptyset$ **then**
- 5: $U^{\text{sch}} \leftarrow \hat{U}_t^{\text{sch}}$
- 6: **else**
- 7: $U_{\text{cat}} \leftarrow [U^{\text{sch}}; \hat{U}_t^{\text{sch}}]$
- 8: $U^{\text{sch}} \leftarrow \text{CapTrunc}(\text{ReDecompose}(U_{\text{cat}}), \tau_{\text{sch}}, r_{\text{sch}}^{\max})$
- 9: **end if**
- 10: **return** U^{sch}

Algorithm A3 Stable Training of Current Session-LoRA with OGP and Session-EWC

Input: $\mathcal{S}_{\text{train}}^{(t)}$, current adapter $\theta_{\text{ses},t}$, shared adapter θ_{sch} , historical bases $\{U_l^{\text{hist}}\}_{l=1}^L, N_{\text{warm}}, \lambda_{\text{ewc}}$
Output: Updated $\theta_{\text{ses},t}$, updated θ_{sch} , post-training gradients $\{G_l^{(t)}\}_{l=1}^L$

- 1: **Warm-up stage** (optimize only $\mathcal{L}_{\text{kgc}}^{(t)}$; no OGP, no Session-EWC)
- 2: Initialize warm-up batch set $\mathcal{W} \leftarrow \emptyset$
- 3: **for** $n = 1$ to N_{warm} **do**
- 4: Sample mini-batch $\mathcal{B} \sim \mathcal{S}_{\text{train}}^{(t)}$; append $\mathcal{B}$ to $\mathcal{W}$
- 5: Compute $\mathcal{L}_{\text{kgc}}^{(t)}$ on $\mathcal{B}$
- 6: Update $\theta_{\text{ses},t}$ and θ_{sch} using $\nabla \mathcal{L}_{\text{kgc}}^{(t)}$ ▷ θ_{sch} uses smaller LR schedule
- 7: **end for**
- 8: Save reference parameters $\phi^* \leftarrow \theta_{\text{ses},t}$
- 9: Estimate Fisher diagonal from warm-up batches:
 $F_t \leftarrow \text{FisherDiag}(\mathcal{W}, \theta_{\text{ses},t}; \mathcal{L}_{\text{kgc}}^{(t)})$

Algorithm A3 *Cont.*

```

10: Main stage (OGP + Session-EWC)
11: for each remaining mini-batch  $\mathcal{B} \sim \mathcal{S}_{\text{train}}^{(t)}$  do
12:   Compute  $\mathcal{L}_{\text{kgc}}^{(t)}$  on  $\mathcal{B}$ 
13:    $\mathcal{L}_{\text{EWC}}^{(t)} \leftarrow \lambda_{\text{ewc}} \sum_i F_{t,i} (\phi_i - \phi_i^*)^2$ 
14:    $\mathcal{L}_{\text{total}}^{(t)} \leftarrow \mathcal{L}_{\text{kgc}}^{(t)} + \mathcal{L}_{\text{EWC}}^{(t)}$ 
15:   Compute  $\{g_l\}_{l=1}^L \leftarrow \nabla_{\theta_{\text{ses},t}} \mathcal{L}_{\text{total}}^{(t)}$ 
16:   for  $l = 1$  to  $L$  do
17:     if  $U_l^{\text{hist}} \neq \emptyset$  then
18:        $\tilde{g}_l \leftarrow g_l - U_l^{\text{hist}} (U_l^{\text{hist}})^\top g_l$ 
19:     else
20:        $\tilde{g}_l \leftarrow g_l$ 
21:     end if
22:   end for
23:   Update  $\theta_{\text{ses},t}$  with  $\{\tilde{g}_l\}_{l=1}^L$ 
24:   Update  $\theta_{\text{sch}}$  with smaller learning rate
25: end for
26: Collect post-training layer-wise gradients  $\{G_l^{(t)}\}_{l=1}^L$  for historical-subspace update
27: return  $\theta_{\text{ses},t}, \theta_{\text{sch}}, \{G_l^{(t)}\}_{l=1}^L$ 

```

Algorithm A4 Session-Aware Retrieve–Rerank Inference

Input: Query $q = (h, r, ?)$ or $(?, r, t)$ with session id t ; frozen backbone θ_{backbone} ; shared adapter θ_{sch} ; session adapter $\theta_{\text{ses},t}$; retriever RotatE; $k_{\text{ret}}, k_{\text{rerank}}$

Output: Predicted entity $\hat{e}$ and ranked candidate list

```

1: Activate inference parameters  $\{\theta_{\text{backbone}}, \theta_{\text{sch}}, \theta_{\text{ses},t}\}$ 
2: Retrieve candidate set by RotatE:
    $\mathcal{C}_{\text{ret}} \leftarrow \text{TopK}_{k_{\text{ret}}}(\text{RotatE}(q))$ 
3: if  $q = (h, r, ?)$  then ▷ tail prediction
4:    $\mathcal{C} \leftarrow \text{TopK}_{k_{\text{rerank}}}(\mathcal{C}_{\text{ret}})$ 
5:   for each  $e \in \mathcal{C}$  do
6:     Construct schema-guided prompt:
      $p_{\text{tail}}(h, r, e) = \text{Template}(h, r, e, c(r))$ 
7:     Score with backbone + Schema-LoRA + Session-LoRAt:
      $s(e | h, r) = \text{score}(p_{\text{tail}}(h, r, e); \theta_{\text{backbone}}, \theta_{\text{sch}}, \theta_{\text{ses},t})$ 
8:   end for
9:   Normalize within candidate set:
     
$$P(e | h, r) = \frac{\exp(s(e | h, r))}{\sum_{e' \in \mathcal{C}} \exp(s(e' | h, r))}$$

10: else ▷ head prediction:  $q = (?, r, t)$ 
11:    $\mathcal{C} \leftarrow \text{TopK}_{k_{\text{rerank}}}(\mathcal{C}_{\text{ret}})$ 
12:   for each  $e \in \mathcal{C}$  do
13:     Construct symmetric prompt:
      $p_{\text{head}}(e, r, t) = \text{Template}(e, r, t, c(r))$ 
14:     Score:
      $s(e | r, t) = \text{score}(p_{\text{head}}(e, r, t); \theta_{\text{backbone}}, \theta_{\text{sch}}, \theta_{\text{ses},t})$ 
15:   end for
16:   Normalize within candidate set:
     
$$P(e | r, t) = \frac{\exp(s(e | r, t))}{\sum_{e' \in \mathcal{C}} \exp(s(e' | r, t))}$$

17: end if
18: Rank candidates by normalized probability
19:  $\hat{e} \leftarrow \arg \max_{e \in \mathcal{C}} P(e | \cdot)$ 
20: return  $\hat{e}$ , ranked list over  $\mathcal{C}$ 

```

References

- Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*; Curran Associates, Inc.: Red Hook, NY, USA, 2013; pp. 2787–2795.
- Yang, B.; Yih, W.t.; He, X.; Gao, J.; Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv* **2014**, arXiv:1412.6575.
- Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; Bouchard, G. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33rd International Conference on Machine Learning*; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, NY, USA, 2016; Volume 48, pp. 2071–2080.
- Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In *Proceedings of the The Semantic Web: 15th International Conference, ESWC 2018*; Springer: Cham, Switzerland, 2018; pp. 593–607.
- Sun, Z.; Deng, Z.H.; Nie, J.Y.; Tang, J. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, New Orleans, LA, USA, 6–9 May 2019; OpenReview.net: Amherst, MA, USA, 2019.
- Vashishth, S.; Sanyal, S.; Nitin, V.; Talukdar, P. Composition-based Multi-Relational Graph Convolutional Networks. *arXiv* **2019**, arXiv:1911.03082.
- Yao, L.; Mao, C.; Luo, Y. KG-BERT: BERT for Knowledge Graph Completion. *arXiv* **2019**, arXiv:1909.03193. [[CrossRef](#)]
- Wang, L.; Zhao, W.; Wei, Z.; Liu, J. SimKGC: Simple Contrastive Knowledge Graph Completion with Pre-trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2022; pp. 4281–4294.
- Wei, Y.; Huang, Q.; Zhang, Y.; Kwok, J. KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2023; pp. 8667–8683.
- Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A. Overcoming Catastrophic Forgetting in Neural Networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [[PubMed](#)]
- Farajtabar, M.; Azizan, N.; Mott, A.; Li, A. Orthogonal Gradient Descent for Continual Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, Online, 26–28 August 2020; Chiappa, S., Calandra, R., Eds.; PMLR: Cambridge, MA, USA, 2020; Volume 108, pp. 3762–3773.
- Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv* **2021**, arXiv:2106.09685.
- Yang, G.; Liu, Y.; Zhang, L.; Zhang, L.; Xie, H.; Mao, Z. Knowledge Context Modeling with Pre-trained Language Models for Contrastive Knowledge Graph Completion. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 8619–8630.
- Liu, Y.; Cao, Y.; Lin, X.; Shang, Y.; Wang, S.; Pan, S. Enhancing Large Language Model for Knowledge Graph Completion via Structure-Aware Alignment-Tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 20981–20995.
- Liu, J.; Ke, W.; Wang, P.; Wang, J.; Gao, J.; Shang, Z.; Li, G.; Xu, Z.; Ji, K.; Li, Y. Fast and Continual Knowledge Graph Embedding via Incremental LoRA. *arXiv* **2024**, arXiv:2407.05705. [[CrossRef](#)]
- Rossi, A.; Barbosa, D.; Firmani, D.; Matinata, A.; Merialdo, P. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Trans. Knowl. Discov. Data* **2021**, *15*, 14. [[CrossRef](#)]
- Li, L.; Jin, Z.; He, Y.; Jin, D.; Zhang, Y.; Duan, H.; Zhang, X.; Tao, Z.; Nyima, T. Learning to Evolve: Bayesian-Guided Continual Knowledge Graph Embedding. In *Proceedings of the ACM Web Conference 2026*; Association for Computing Machinery: New York, NY, USA, 2026; pp. 4372–4383.
- Lee, K.H.; Choi, D.W. STARK: Structure-Aware and Adaptive Representation Learning for Continual Knowledge Graph Embedding. In *Proceedings of the ACM Web Conference 2026*; Association for Computing Machinery: New York, NY, USA, 2026. [[CrossRef](#)]
- Omeliyanenko, J.; Hotho, A.; Schlör, D. Parameter Efficient Continual Automated Knowledge Graph Completion. In *Proceedings of the Semantic Web; Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2026; Volume 16549, pp. 358–378. [[CrossRef](#)]
- Chen, Z.; Wang, X.; Li, Z.; Guo, W.; He, D.; Li, Y.; Silamu, W. KG-BiLM: Knowledge Graph Embedding via Bidirectional Language Models. In *Proceedings of the ACM Web Conference 2026*; Association for Computing Machinery: New York, NY, USA, 2026; pp. 4290–4301.
- Toutanova, K.; Chen, D. Observed versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, Beijing, China, 26–31 July 2015; pp. 57–66. [[CrossRef](#)]

22. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2018; Volume 32. [\[CrossRef\]](#)
23. Balazevic, I.; Allen, C.; Hospedales, T.M. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 5184–5193.
24. Zhang, Z.; Cai, J.; Zhang, Y.; Wang, J. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2020; Volume 34, pp. 3065–3072. [\[CrossRef\]](#)
25. Chen, S.; Liu, X.; Gao, J.; Jiao, J.; Zhang, R.; Ji, Y. HittER: Hierarchical Transformers for Knowledge Graph Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 10395–10407. [\[CrossRef\]](#)
26. Zhang, Z.; Liu, X.; Zhang, Y.; Su, Q.; Sun, X.; He, B. Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 259–266. [\[CrossRef\]](#)
27. Wang, B.; Shen, T.; Long, G.; Zhou, T.; Chang, Y. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In *Proceedings of the Web Conference 2021*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1737–1748.
28. Choi, B.; Jang, D.; Ko, Y. MEM-KGC: Masked Entity Modeling for Knowledge Graph Completion with Pre-trained Language Model. *IEEE Access* **2021**, *9*, 132025–132032. [\[CrossRef\]](#)
29. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
30. Rusu, A.A.; Rabinowitz, N.C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; Hadsell, R. Progressive Neural Networks. *arXiv* **2016**, arXiv:1606.04671. [\[CrossRef\]](#)
31. Lomonaco, V.; Maltoni, D. Continual Learning in Single-Incremental-Task Scenarios. *Neural Netw.* **2019**, *116*, 56–73.
32. Lopez-Paz, D.; Ranzato, M. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems 30*; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
33. Wang, H.; Han, X.; Liu, Z.; Sun, M. Sentence Embedding Alignment for Lifelong Relation Extraction. *arXiv* **2019**, arXiv:1909.08393. [\[CrossRef\]](#)
34. Kou, Z.; Liu, H.; Chen, H.; Li, Y.; Xu, Z.; Wang, Y. Disentangle-based Continual Graph Representation Learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 671–680.
35. Zenke, F.; Poole, B.; Ganguli, S. Continual Learning Through Synaptic Intelligence. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, Australia, 6–11 August 2017*; Precup, D., Teh, Y.W., Eds.; PMLR: Sydney, Australia, 2017; Volume 70, pp. 3987–3995.
36. Cui, Y.; Wang, Y.; Sun, Z.; Liu, W.; Jiang, Y.; Han, K.; Hu, W. Lifelong Embedding Learning and Transfer for Growing Knowledge Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2023; Volume 37, pp. 4217–4224.
37. Zhu, L.; Lan, Q.; Tian, Q.; Sun, W.; Yang, L.; Xia, L.; Xie, Y.; Xiao, X.; Duan, T.; Tao, C. ETT-CKGE: Efficient Task-Driven Tokens for Continual Knowledge Graph Embedding. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Cham, Switzerland, 2025; pp. 481–496.
38. GLM, T.; Zeng, A.; Xu, B.; Wang, B.; Zhang, C.; Yin, D.; Rojas, D.; Feng, G.; Zhao, H.; Lai, H.; et al. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. *arXiv* **2024**, arXiv:2406.12793.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.