摘 要

知識圖譜的更新仰賴人力收集及統整資料，使得更新緩慢，難以即時加入最新資訊，同時也可能因人為疏失或資料遺失造成知識圖譜不完整。因此有許多連結預測研究致力於基於已知的圖譜預測缺失或未知的連結。這些研究使得圖補全可自動進行，加速資訊更新。然而，有些資訊的影響力是會隨時間逐漸消弱的，即使這些資訊仍然是正確的，對某些下游任務（如推薦系統或問答系統）而言，過度參考這些資訊可能會造成不正確的判斷。若模型能得知資訊的影響力效期並適當的刪除或保留資訊，可以增加可用資訊並避免過度參考造成的錯誤結果。因此，本研究提出一個結合連結預測及影響力效期預測的框架 (EvoGUT, Evolutionary Graph-Updating Training) 以更新知識圖譜。除了圖譜結構及語意特徵，模型會基於實體與連結的時間資訊建構時間特徵，預測連結的存在與影響力效期。在 EvoGUT 中，每次更新圖譜時會保留具有較強影響力的連結，而影響力較弱的資訊會被移除，以維持資料的準確及新穎。

Abstract

Updating knowledge graphs typically relies on human efforts to collect and organize data, which makes the process time-consuming and limits the ability to incorporate new information in real-time. Moreover, omissions caused by human error or data loss may occur. Therefore, many link prediction models have been developed to predict missing or unknown links based on existing graphs. These methods enable graph completion to be performed automatically and accelerate the process of information updating.

However, some information is time-sensitive, which means its influence gradually diminishes over time. Although such information may remain factually correct, excessive reliance on it in some downstream tasks (such as recommendation and question-answering systems) probably leads to incorrect decisions. Therefore, enabling the model to estimate the effective lifetime of information and to selectively retain or discard it can improve information utilization and reduce errors arising from outdated references.

Therefore, this study proposes a framework that jointly models influence lifetime estimation and link prediction for knowledge graph updating. In addition to graph structure and semantic features, this framework captures temporal features from the temporal information associated with entities and links, enabling the model to determine both the existence and the influence lifetime of each link. In this way, influential links

are retained during each update, while weakly influential information is removed, thereby ensuring the accuracy and freshness of the knowledge graph.

Chapter 1 Introduction

Knowledge Graphs (KGs) consist of collections of fact triplets, where each fact is represented as (entity, relation, entity). This structured representation provides rich information for various downstream tasks, such as recommendation and question answering, thereby assisting models in making more informed decisions.

However, updating knowledge graphs depends on human efforts to collect and organize data, making the process slow and difficult to add new information in real-time. Omissions due to human negligence or data loss probably also occur. Therefore, several link prediction models have been developed to learn from existing graph features and predict missing or unknown links, allowing graph completion to be performed automatically and accelerating the updating of knowledge graphs.

Since the emergence of neural networks, many studies have harnessed their powerful learning capabilities to develop enhanced link prediction models [1–10]. Furthermore, as graphs are usually continuously evolving, traditional transductive methods present challenges for effective prediction, prompting research into temporal graphs, dynamic graphs, and inductive settings [4,6–8].

In the inductive setting, emerging KGs consist of unseen entities and links between them and have no intersection with the original KG, also called disconnected emerging KGs (DEKGs). Most research aims to learn

the structure and semantics of the original KG to predict the enclosing links between unseen entities in emerging KGs. However, in practical applications, some bridging links between the original KG and the emerging KG often exist. If this is not considered, some critical information is possible to be lost [9].

Although bridging links can be learned and predicted in the same way as enclosing links, the nodes of bridging links belong to two non-intersecting graphs, making it difficult to establish a subgraph for this link to learn structural features. Therefore, DEKG-ILP [9] is committed to capturing global semantic features through contrastive learning to compensate for the lack of structural features. Based on this, GSELI [10] combines personalized PageRank subgraph extraction and neighboring relational paths to enhance the model's ability to learn structural features.

While DEKG-ILP [9] and GSELI [10] consider the existence of bridging links, allowing the model to add more information in graph completion applications, they ignore that some information is time-sensitive. Although this information remains true after its influence lifetime, simply adding it to the knowledge graph and referring to outdated information in downstream tasks probably leads to incorrect judgments.

For example, information related to class suspension or disasters caused by a typhoon is most important and accurate during the typhoon period but gradually loses its importance afterward. If all such information is indiscriminately incorporated into the knowledge graph, downstream applications possibly subsequently rely on outdated or misleading information.

However, this does not imply that all low-impact information should be deleted. Certain aspects, such as the structure of the typhoon, its trajectory, and related characteristics, can remain informative and valuable for future reference.

Therefore, if the model is aware of the influence lifetime of information and retains it appropriately, it can increase the amount of usable information while avoiding erroneous results caused by referencing outdated knowledge.

Therefore, we propose a framework that builds upon GSELI [10] as the foundation for graph updating and incorporates influence lifetime estimation. Besides the structural and semantic features, the temporal features of the graph are learned based on the temporal information of entities and links in the graph to determine whether a link exists and how long it maintains a strong influence. During each update step, influential links are retained while low-influence information is removed, thereby ensuring the accuracy and novelty of the knowledge graph.

The primary contributions of this work are summerized as follows:

- We propose a framework designed for continuous prediction. Through evolutionary graph-updating training, the model learns to handle dynamic or unknown graph structures. It enables the model to serve as an automatic graph completion tool, reducing the need for manual annotation.
- Typical temporal graph reasoning only considers fixed entity and re-

lation sets. We extend the semi-inductive setting to temporal knowledge graphs, enabling the model to handle emerging entities and relations, making it more suitable for real-world applications.

- We design two graph updating strategies, which can be adopted for different scenarios. By retaining complete information or removing redundant information, it can maintain accuracy in various situations.

The rest of this paper is organized as follows: Chapter 2 discusses previous related works, Chapter 3 defines the problem we want to solve, Chapter 4 introduces our proposed method, Chapter 5 describes our experimental setup and results, and Chapter 6 summarizes the conclusion.

Chapter 2 Related Works

This section summarizes some outstanding works from the past and introduces them according to their application scenarios.

2.1 Transductive Link Prediction

Transductive link prediction aims to predict missing links in graphs. In this situation, all entities are known, so the key challenge is in effectively predicting the existence of links based on structural, semantic, and other information in the graph. Traditional heuristic methods can accurately predict, but rely on human design, which makes it hard to automate. Therefore, SEAL [1] uses a neural network to automatically determine the appropriate heuristic function based on the graph. Combining the accuracy of heuristic methods with the power of neural networks, the model's generality is increased. RGCN [3] considers the diversity of relations in a graph, it aggregates information according to relation, and assigns different weights. In this way, the model can fully utilize relation information to learn node representations.

2.2 Inductive Link Prediction

While many studies have demonstrated excellent performance in transductive link prediction, in the real world, the entity set in a graph is usually not fixed. New entities are added over time, and these entities often contain relatively little information. Therefore, models that focus on node information are not very effective. Although the problem can be solved via re-training, it increases the model's training cost. Furthermore, if a model can be applied to different graphs and then fine-tuned, the training cost can be reduced significantly. Consequently, many studies have focused on inductive link prediction.

GraIL [4] first proposed an inductive setting, aiming to learn logical rules independent of node semantics. This allows the model to focus on capturing structural information to establish entity-independent relation representations, achieving excellent results in predicting links between unseen entities.

2.3 Semi-Inductive Link Prediction

However, in addition to links between known entities or unknown entities, there are also links between known and unknown entities. If the graph is simply divided into two graphs, consisting of known entities and consisting of unknown entities, these links between known and unknown entities, which are called bridge links, will be ignored. Therefore, semi-inductive

link prediction, which handles both transductive and inductive link prediction, has become a focus in recent years.

MV-HRE [5] considers the diversity of nodes and relations in heterogeneous graphs, as well as the problem of data imbalance between categories. It uses multiple aspects of information to learn, including subgraph aspect, metapath aspect, and community aspect, to assist categories with few data. Meta-iKG [11] also takes similar issues into account. This work aims to strengthen new entities and relations, or rarely appearing relations. It divides relations into large-shot relations and few-shot relations according to their frequency of appearance, that is, the amount of available data, and uses meta-learning to allow large-shot relations to assist in the learning of few-shot relations.

Unseen nodes often cannot effectively predict their connections due to a lack of information, so relation embeddings are used to construct unseen node representations. DEKG-ILP [9] uses the relation composition of each node for contrastive learning to make the semantics embeddings of the relation more accurate, thereby establishing a better node representation. It also learns topological features and considers the semantic and topological information for prediction. GSELI [10] adds personalized PageRank subgraph extraction to DEKG-ILP [9], allowing the model to obtain subgraphs with closer connections when extracting for each triplet. To contruct better initial embedding for nodes, GSELI [10] combines neighboring relational path modeling proposed by SNRI [12], uses a self-attention mechanism to learn structure-based relation embeddings. It also uses GRU to learn

the contextual relationships of relations to obtain better structural features, thereby improving model performance.

2.4 Temporal Link Prediction

In the real world, the emergence of entities and relations and the evolution of graphs occur over time, which is not considered in works on static graphs. To be more fit with practice applications, some works focus on temporal link prediction, aiming to capture evolution patterns of graphs. In order to model time-varying graph structures, DySAT [7] generates dynamic node representations through joint self-attention to increase accuracy and flexibility of the model. On the other hand, CoEvoGNN [8] learns the covariance between node attributes and the overall structure of graphs to capture the mutual influence between them. TCDGE [13] proposes ToE (Timespans of Edge formation), which presents how long it takes for a triplet to form from being able to form, and combines with matrix factorization to preserve the temporal correlation between nodes.

Since the information imbalance that results from entities appearing at different times and frequencies, RE-GCN [14] integrates static and dynamic features to mitigate this limitation. It captures sequential patterns across timestamps and incorporates the static properties simultaneously to utilize two features effectively. CorDGT [15] proposes a new approach to extract high-order proximity to obtain comprehensive features, and uses the self-attention mechanism to enhance the expressive power.

However, most research assumes that all possible entities and relations are known, and can not really deal with unseen entities and relations. Thus, how to address emerging entities and relations is still an issue worth exploring.

Chapter 3 Problem Statement

In this section, we present some definitions used throughout this work and formally state the problem that our model aims to solve.

Knowledge graphs containing existing facts are widely used in several applications, such as question answering and information retrieval. However, those graphs usually record what happened without when it happened, which is not enough to learn and predict the evolution of graphs. Therefore, temporal knowledge graphs that contain temporal information are needed to capture the evolving patterns for predicting future events.

- **Temporal Knowledge Graph**

  A temporal knowledge graph G_t is a collection of facts, denoted as $G_t = \{(u, r, v, t_i) \mid u, v \in E_t, r \in R_t, 0 \leq t_i \leq t \leq T\}$, where E_t and R_t denote the entity and relation sets at t, t_i is the timestamp of each triplet formed, t is the last timestamp in the graph, and T is the last timestamp in the dataset.

Note that each link, originally represented as a triplet, is further annotated with a timestamp. As links are constantly created over time, new entities also emerge, such as a newly-debuted athlete or a newly founded company. These form the main components of graph evolution.

- **Emerging Entity Set**

  In practical applications, temporal knowledge graphs are dynamic,

with emerging entities E'_t, where $E'_t \cap E_{t-1} = \emptyset$, continuously emerging over time, leading to differences between entity sets of G_{t-1} and G_t.

However, these emerging and previously unseen entities often lack connections to existing entities. Such cases may occur when links have not yet been established, are present but unobserved, or are difficult to verify their existence, as demonstrated in the following two examples.

- **Example 1**

  In a knowledge graph that records sporting events, entities such as players, teams, sponsors, and events are connected through links representing relationships like player signing with a team or team participation in competitions. For sports news media, the events in which players or teams will participate are of primary interest, whereas teams are also concerned with player signings and sponsorships involving other teams. When new players emerge, both media organizations and teams actively monitor their movements in order to predict, as early as possible, which team they will join.

- **Example 2**

  In a protein-protein interaction network, proteins are modeled as entities and their interactions are regarded as links. When biologists discover a new protein, identifying its interaction partners and interaction mechanisms as early as possible is highly desirable, but experimental validation is costly and time-consuming. Predicting

potential interactions can narrow down the scope of verification.

In such cases, obtaining accurate connections is often difficult and costly. Therefore, predicting link existence through models can significantly reduce costs and enable real-time graph updates. Based on the above discussion, our main objective is defined as follows.

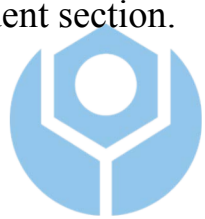
- **Problem Formulation**

  At timestamp t, given the previous temporal knowledge graph G_{t-1} and an emerging entity set E'_t, the model aims to predict the content of the graph G_t. This involves determing the appearance of new links in $S_t = E_t \times R_t \times E_t - G_{t-1}$ and the disappearance of existing links in G_{t-1}, where $E_t = E_{t-1} \cup E'_t$ and R_t are the entity and relation sets at t.

Moreover, most graph reasoning methods focus on seen entities [1,3, 14]. While some studies consider emerging entities and aim to improve model performance in semi-inductive settings [5,9,10,15], unseen relations are generally not considered. For instance, novel functions on social media, such as "virtual gifting" or "co-streaming", introduce new relations that were absent in the past. In these cases, models mentioned above either ignore triplets related to emerging relations or mispredict them as other known relations, both of which limit the scope of application scenarios. To increase model flexibility, we allow the model to deal with not only unseen entities but also unseen relations. Due to the cold-start problem of emerging relations, it is difficult to embed each emerging relation individually.

If relation embeddings are suboptimal or unavailable, it will affect the entity embeddings constructed from relation embeddings, and thus affect the model performance. Therefore, we aggregate these emerging relations and model them jointly. Specifically, all emerging relations are categorized into a single representation, denoted as r^u, and the model can predict links even when the relations were unseen during training.

In summary, we elucidate the definitions and challenges of emerging entities and relations and are dedicated to addressing these obstacles. To predict the graph G_t from the previous graph G_{t-1} under the challenges mentioned, we design a framework to model the evolution patterns of the graph, proposed in the subsequent section.



Chapter 4 Methodology

In real-world scenarios, facts continuously emerge and some of them become obsolete after a period of time. To speed up the update of graphs and reduce the effort of manual updates, we propose a framework named **EvoGUT** (**Evo**lutionary **G**raph-**U**pdating **T**raining), which is designed for temporal link prediction and knowledge graph auto-updating, as illustrated in Figure 4.1.

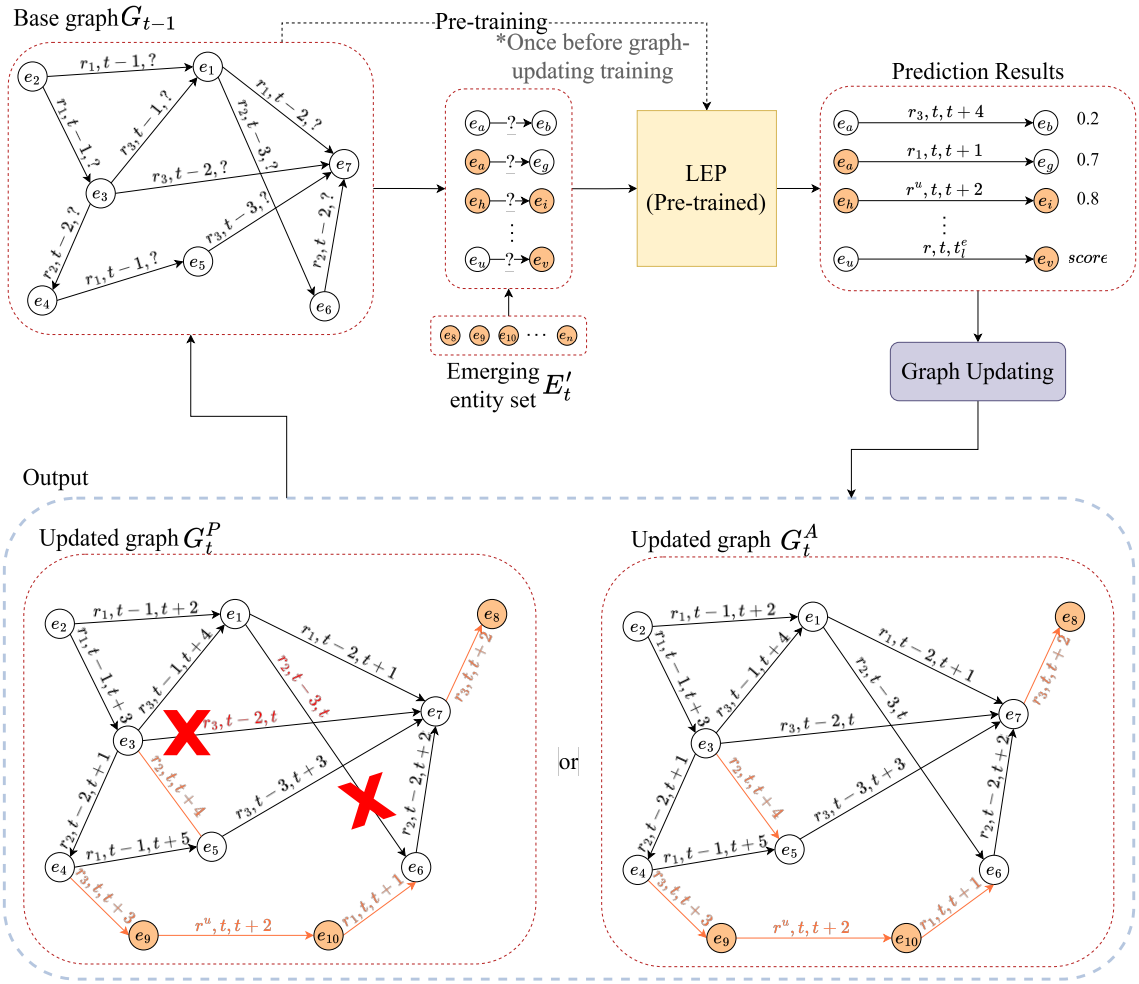


Figure 4.1: Framework of EvoGUT.

Given the base knowledge graph G_{t-1}, emerging entity set E'_t, EvoGUT aims to automatically update graphs via predicting existence and estimating the influence lifetime for each possible link, including new links in S_t and existing links in G_{t-1}. Firstly, **LEP** (**L**ink **E**xistence **P**redictor) integrates semantic, topological, and temporal features to estimate the likelihood of link existence and the influence lifetime of links from three complementary perspectives. EvoGUT updates the base knowledge graph with two updating strategies to get two candidate graphs, G_t^A and G_t^P. Finally, the next knowledge graph G_t is selected from these two graphs based on the model configuration.

4.1 Evolutionary Graph-Updating Training

Existing temporal link prediction methods, which focus on the transductive setting that assumes entities and relations are fixed, are unable to handle new content appearing over time. While semi-inductive link prediction models can handle this scenario, they seldom consider temporal information and graph evolution patterns, making it difficult to update graphs accurately. To automatically update knowledge graphs more precisely, we propose a framework that can train the model on **how to update graphs** and generate two version of updated graphs according to different strategies for processing redundant information.

Firstly, we split both the training set and validation set into two parts, pre-training and graph-updating training (GUT), as illustrated in Figure 4.2.

The first part of the training set and the validation set are used as static graphs for the initial pre-training, while the second parts are used to train the model on updating graph. In this process, the model explicitly learns graph evolution across timestamps, whereas most existing temporal graph reasoning methods typically learn from independent snapshots at each timestamp, as shown in Figure 4.3.

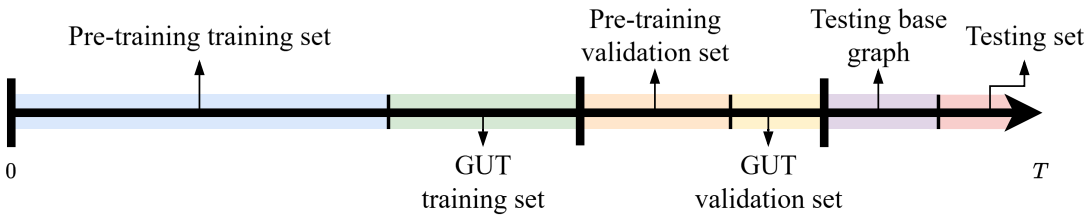


Figure 4.2: The dataset split of EvoGUT.

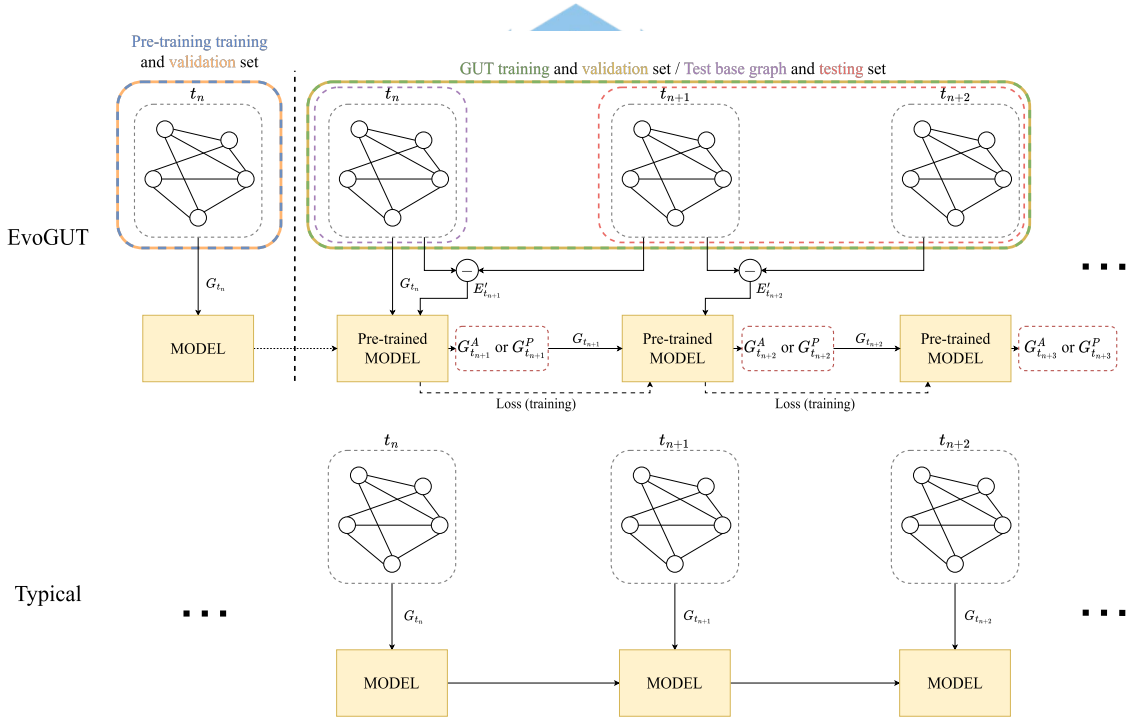


Figure 4.3: Training flow comparison between EvoGUT and typical models.

Initially, the model is trained on the static graph to learn static features

of semantics and topology. After this pre-training step, we train the model on how to update graphs. As shown in Figure 4.1, for a base graph G_{t-1}, the model generates a candidate link set $S_t^c = \{l \mid l \in S_t \wedge \phi(l) > \theta\}$, where $\phi(l)$ is the score of link l given by the model, θ is the threshold decided by the model to determine link's existence. Then the model updates G_{t-1} based on the following strategies:

$$\text{Accumulation: } G_t^A = G_{t-1} \cup S_t^c, \tag{4.1}$$

$$\text{Pruning: } G_t^P = \{l \mid l \in G_t^A \wedge t_l^e > t\}, \tag{4.2}$$

where t_l^e is the timestamp representing influence lifetime of link l.

In the accumulation strategy, all links whose scores satisfy a predefined threshold will be added to the graph, ensuring all existing information is retained regardless of its influence. In contrast, the pruning strategy jointly considers link scores and influence lifetimes, retaining only those links whose influence persists into the subsequent timestamp. The choice between Graph G_t^A and G_t^P as the input graph for the next iteration is application-dependent. For instance, in some question answering systems, it only requires information that is still valid at the current time, so the pruning strategy is preffered. As for graph completion, since it aims to record facts, it places more emphasis on information's existence rather than influence, which is more suitable for using the accumulation strategy.

By repeating these steps, the model prunes and adds triplets by minimizing the total loss, enabling it to determine the best way of updating graphs.

4.2 Link Existence Predictor

Link Existence Predictor (LEP) integrates semantic, topological, and temporal features to predict links' existence and its influence lifetime, as shown in Figure 4.4. Semantic features are learned via Contrastive Learning-based global Semantic Feature modeling (CLSF), which enhances the model's ability to express fine-grained relational semantics. Topological features are extracted through GNN-based Enhanced Local Subgraph modeling (GELS), capturing local structural relevancies within the graph. Both proposed by DEKG-ILP [9] and GSELI [10] have been performing well in unseen-entity-contained scenarios. Besides, to fully absorb the features of both sides, the relation embeddings are shared between these two aspects. So that the semantic information can be broadcast by GNN to enrich entity embeddings. Meanwhile, the topological information that implies relative location and connection information between relations can be helpful while learning semantic features. Furthermore, since most temporal knowledge graph reasoning works are limited to transductive settings that need to know all entities and relations, this leads to poor generality. In order to extend semi-inductive and inductive settings' model to temporal knowledge graphs, temporal features are modeled to not only predict existence but also estimate the influence lifetime of each link—a direction rarely explored in prior temporal knowledge graph studies. Firstly, LEP gives each link a score that reflects its likelihood of existence and an influence lifetime to reflect the time period during which the link remains informative and impactful for downstream tasks.

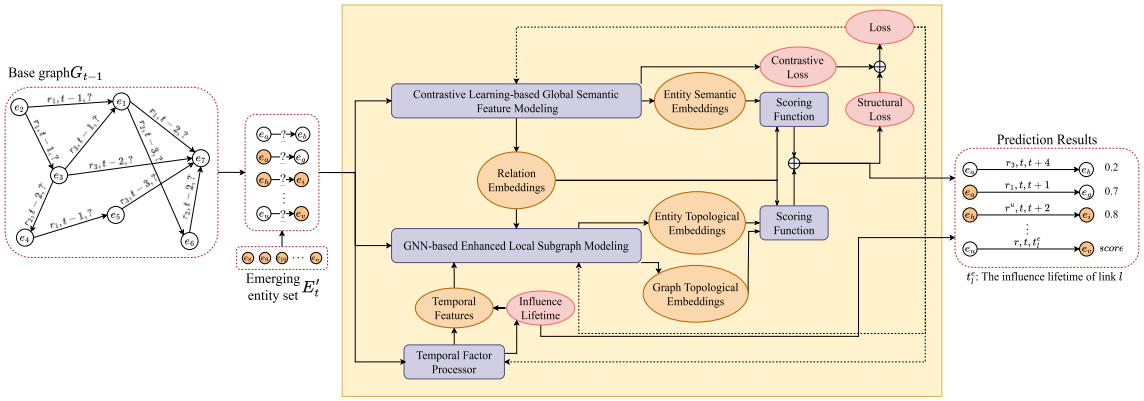


Figure 4.4: Schematic of Link Existence Predictor.

4.2.1 Semantic and Topological Embedding Learning

Semantic features constitute information in the knowledge graph, providing complementary perspectives to topological features. However, in inductive and semi-inductive scenarios, most or all entities are unseen during training, making it difficult to construct their semantic representations directly. To overcome the obstacle, most studies derive the semantic features of an entity from the relation features associated with that entity, lead to the importance of high-quality relation embeddings.

For learning the comprehensive relation features, we refer to Contrastive Learning-based Global Semantic Feature Modeling (CLSF) proposed by DEKG-ILP [9]. This module constructs entity semantic feature e_i by fusing relation embeddings based on the composition of entity i, and calculate Euclidean distance as the measure to use contrastive learning to enhance the relation embeddings.

Although Euclidean distance is a clear and widely used measure, it is

possible to be insufficient in some situations, such as when the number of links between relations of a sample is highly imbalanced. As the example illustrated in Table 4.1, the module first generates positive and negative smaples via data augmentation and then normalizes them to eliminate bias caused by numbers of links. Next, it calculates the distances between the original sample and the augmented samples, denoted as d_i^{pos} and d_i^{neg}, respectively. The goal is to minimize d_i^{pos} and maximize d_i^{neg}, making the original sample similar to the positive one and dissimilar to the negative one. In this example, after data augmentation, the positive sample contains substantially more r_3 links than the original sample. This imbalance causes d_i^{pos} to become close to d_i^{neg}. In this situation, the module needs to adjust relation embeddings to maintain a margin between d_i^{pos} and d_i^{neg}, which probably compromises the ability of the embeddings to effectively represent relational information.

Table 4.1: An example of distance calculation.

	Original			Normalized			Euclidean distance			Cosine similarity		
	r_1	r_2	r_3	r_1	r_2	r_3	d_i^{pos}	d_i^{neg}	margin	sim_i^{pos}	sim_i^{neg}	margin
e_i	1	0	1	$\frac{1}{2}$	0	$\frac{1}{2}$	-	-	-	-	-	-
e_i^{pos}	1	0	30	$\frac{1}{31}$	0	$\frac{30}{31}$	0.66149	-	0.04562	0.73027	-	0.23027
e_i^{neg}	1	1	0	$\frac{1}{2}$	$\frac{1}{2}$	0	-	0.70711		-	0.5	

We choose cosine similarity to avoid the potential problems. Since cosine similarity measures the angle between vectors, it is affected by relation composition rather than the number of links of each relation. In the same

example, replacing the distances d_i^{pos} and d_i^{neg}, we calculate the similarity between the original sample and the augmented samples, denoted as sim_i^{pos} and sim_i^{neg}. With this change, the goal becomes to maximize sim_i^{pos} and minimize sim_i^{neg}. As shown in Table 4.1, cosine similarity maintains a more obvious margin between sim_i^{pos} and sim_i^{neg}. Therefore, relation embeddings have more space to express their own semantic features.

On the other hand, learning topological features is a fundamental task of GNN-based methods, since such information captures intrinsic structural associations between entities and enables the model to predict the existence of links. Different from semantic features, topological features express patterns in graph evolution, focusing on the formation rules of the graph's physical structure, providing the model with more intuitive information to evaluate triplets more precisely.

In this part, we follow the framework, GNN-based Enhanced Local Subgraph Modeling (GELS), proposed by GSELI [10]. It proposes a PageRank-based subgraph extraction to extract subgraphs with closer association, combined with the node representation initialization method and neighboring relational path representation of SNRI [12] to obtain better subgraphs. Then, it uses RGCN [3] as the graph convolutional network to utilize structural and relational information to learn entity topological embeddings and graph topological embeddings.

4.2.2 Temporal Feature Extraction and Influence Lifetime Estimation

In contrast to conventional knowledge graphs, temporal knowledge graphs contain more explicit and implicit temporal information that can help models to predict more accurately. However, it also increases the difficulty of learning, such as modeling timestamps associated with entities and triplets, as well as capturing the evolution of graph snapshots. To effectively exploit such temporal information to improve performance on temporal knowledge graphs, we design a temporal factor processor to extract temporal features and estimate the timestamp of influence lifetime of each link.

In addition, we use ToE (Timespans of Edge formation) [13] as one of the input information to enrich the temporal features. ToE, expressing how long it takes for a triplet to form from being able to form, is calculated as:

$$t_{ToE} = t_l - \max(t_i, t_j), \tag{4.3}$$

where t_l is the timestamp that the triplet was founded, t_i and t_j are the timestamps that i and j appear in the graph. In this work, we use the timestamp of the earliest triplet containing i as t_i like:

$$t_i = \{\min(t_{l_x}) \mid \forall l_x \in S \wedge i \in l_x\}, \tag{4.4}$$

and t_j is defined in the same way. We also believe that learning relative temporal features is more helpful than learning absolute temporal features of entities and links. Therefore, except for t_ToE, all input are calculated as

the difference between temporal factors and the current time t_{curr} by:

$$t_i = |t_i - t_{curr}|. \tag{4.5}$$

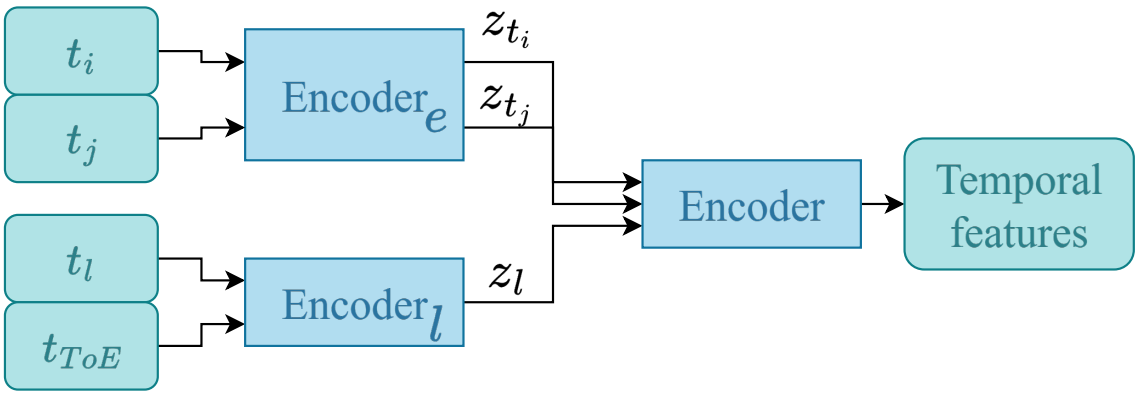


Figure 4.5: Processing pipeline of Edge Temporal Feature Learning.

For temporal feature learning, as shown in Figure 4.5, we encode each factor into embeddings for normalization, and then we adopt a linear layer as a simple encoder to encode edge temporal features F^e of link l as:

$$F_l^e = w_1 z_{t_i} + w_2 z_{t_j} + w_3 z_l + b^e, \tag{4.6}$$

where z_{t_i}, z_{t_j}, and z_l are encoded embeddings of t_i, t_j, and t_l and t_{ToE}, and w_x are trainable weights.

As discussed earlier, the influence of a link depends not only on its start time but also on its termination time, which should be considered during model learning. However, most datasets do not provide such information. To address this limitation, we estimate the influence lifetime of each link via regression over its temporal information. This design not only enriches

the information available to the model but also supplies missing temporal attributes, thereby providing a basis for updating the graph. Influence lifetime t_l^{end} is estimated as follows:

$$F_l^{t_{end}} = Regressor(t_i, t_j, t_l, t_{ToE}) = w_{t_i} t_i + w_{t_j} t_j + w_{t_l} t_l + w_{t_{ToE}} t_{ToE} + b^{t_{end}}, \tag{4.7}$$

$$t_l^{end} = F_l^{t_{end}} t_{exp}, \tag{4.8}$$

where w_{t_i}, w_{t_j}, w_{t_l}, and $w_{t_{ToE}}$ are trainable weights, $F_l^{t_{end}}$ is the embedding of the influence lifetime of link l, t_{exp} is a hyperparameter that decides the maximum of estimated influence lifetime.

Finally, we use the sum of edge temporal features F_l^e and influence lifetime embeddings $F_l^{t_{end}}$ as temporal features F_l^{temp}, and fuse it into the hidden state as follows:

$$F_l^{temp} = F_l^e + F_l^{t_{end}}, \tag{4.9}$$

$$h_i^k = \sum_{r_x \in R} \sum_{j \in \mathcal{N}_{r_x}(i)} \alpha_{ir_xj}^k W_{r_x}^k F_l^{temp} \psi(h_j^{k-1}, z_{r_x}^{k-1}), \tag{4.10}$$

$$\alpha_{ir_xj}^k = \sigma_A(W_A^k s_{ir_X} + b_A^k), \tag{4.11}$$

$$s_{ir_x} = \sigma_B(W_B[h_i^{k-1} \oplus h_j^{k-1} \oplus z_r^{k-1} \oplus z_{r_z}^{k-1}] + b_B^k), \tag{4.12}$$

where h_e^k are node hidden states of k-th layer, z_r^k are relation embeddings of k-th layer, σ_A and σ_B are activation functions, $W_{r_x}^k$ is the transformation matrix, $\psi(,)$ is the fusion function. In this way, the model can extract and diffuse temporal features to find better entity and relation embeddings.

4.3 Scoring Function, Threshold Tuning and Loss Function

Most link prediction models will calculate a score or probability for a triplet to determine whether it exists according to various features such as node features, subgraph features, relation features, and other information. In this work, we calculate scores based on semantic features and structural features, respectively, and add them as final scores.

For each triplet $l = (u, r, v)$, semantic score ϕ^{sem} is calculated by DistMult [16], denoted as:

$$\phi^{sem}(l) = \langle e_u, z_r, e_v \rangle, \tag{4.13}$$

where e_u and e_v are semantic embeddings of entity u and v, z_r is the embedding of relation r, and $\langle , \rangle$ denotes the element-wise product.

On the other hand, structural score ϕ^{str} is calculated as:

$$\phi^{str}(l) = W[h_u^K \oplus h_v^K \oplus z_r \oplus Z_{\mathcal{G}(u,r,v)}], \tag{4.14}$$

where h_u^K and h_v^K are structural embeddings of entity u and v learned by GNN, $Z_{\mathcal{G}(u,r,v)}$ is the representation of subgraph $\mathcal{G}(u, r, v)$, W is the transformation matrix.

The final score $\phi(l)$ is denoted as:

$$\phi(l) = Linear(\phi^{sem}(l), \phi^{str}(l)). \tag{4.15}$$

The existence of a link l is determined by comparing its final score $\phi(l)$

against a threshold θ. To more accurately predict the presence of links, the threshold is adaptively determined during training according to the scores of positive and negative samples, as formulated below:

$$\theta = \frac{\left(\theta_{old} + \frac{(P_a^{pos} + P_b^{neg})}{2}\right)}{2}, \tag{4.16}$$

where θ_{old} is the previous threshold, P_a^{pos} and P_b^{neg} denote the a-th and b-th percentile scores of the positive and negative samples, respectively. We expect the model to find a threshold that best fits the overall score distribution through statistics, and adjust the strictness of the threshold according to a and b. Therefore, the model calculates the boundary that distinguishes positive and negative samples in each iteration and averages it with the past threshold, thus retaining the past statistical values.

As for the loss function, in order to make the model consider both contrastive learning and structure learning, following DEKG-ILP [9] and GSELI [10], we first calculate the loss of the two separately and then add them as the total loss $\mathcal{L}$.

Contrastive learning loss $\mathcal{L}^{con}$ is defined as:

$$\mathcal{L}^{con} = \sum_{l \in S} \sum_{i \in l} \max(0, dist(e_i^{pos}, e_i) - dist(e_i^{neg}, e_i) + \gamma), \tag{4.17}$$

where S is the sample set, $dist(,)$ is the distance measured by the similarity function that is cosine similarity in this work, and γ is a hyperparameter that decides the margin.

Structure learning loss $\mathcal{L}^{str}$ is calculated as:

$$\mathcal{L}^{str} = \sum_{l_p \in S^+, l_n \in S^-} \max(0, \phi(l_n) - \phi(l_p) + \gamma), \tag{4.18}$$

where S^+ and S^- are the positive and negative sample sets.

Finally, the total loss $\mathcal{L}$ is denoted as:

$$\mathcal{L} = \mathcal{L}^{str} + \beta \mathcal{L}^{con}, \tag{4.19}$$

where β is a hyperparameter controlling the proportion of $\mathcal{L}^{con}$.

Chapter 5 Experiments

In this section, we conduct several experiments to validate the ability of the proposed EvoGUT. The primary objective is to evaluate whether the model can effectively learn to update the graph automatically and maintain performance in multi-step scenarios.

5.1 Experiment Settings

We conduct experiments under two settings. In the transductive setting, the model's performance is evaluated in the traditional static scenario. In the multi-step setting, we examine the model's ability to learn rules of graph evolution. Furthermore, we treat the link prediction problem as a binary classification and analyze the potential of the model as a graph completion tool.

In the multi-step setting, different from the typical setting, we use an independent graph as the base graph in the testing phase. This design ensures that the model's performance stems from learning evolutionary patterns sufficiently rather than relying on historical data. At the same time, the model continuously predicts triplets for multiple timestamps to examine its robustness and precision. If the model cannot accurately distinguish between positive and negative samples, the resulting errors will accumulate and affect subsequent predictions. Therefore, a model must be able

to accurately predict and self-correct to maintain excellent performance in this setting.

In the transductive setting, following the traditional setting and previous works, the base graph contains all triplets in the training and validation sets to ensure all entities and relations are known in the testing phase. But the model needs to predict triplets of several timestamps at once to prove that it has fully learned the rules of graph evolution.

For our method and GSELI [10], we set all dimensions of embeddings to 100, 64 for DEKG-ILP [9] due to the memory issue. For the hyperparameters used to tune the threshold, we set the hyperparameters a and b used to tune the threshold to 25 and 75. For GSELI [10] and DEKG-ILP [9], since margin-based ranking loss and DistMult [16] are used, the scores of positive samples usually tend to be positive, while the scores of negative samples tend to be negative or the minimum value, and there is no limit to the range of scores. Therefore, 0 is set as the threshold for both. As for the updating strategy, we choose pruning to remove the expired information from graphs. However, in the classification evaluation, link existence is determined solely by the model's final scoring function, independent of whether the corresponding triplets are retained in the updated graph afterward. For RE-GCN [14] and CorDGT [15], we follow the configuration provided by the original authors. Since CorDGT [15] does not contain relation information, we only replace the head or tail entity when negative sampling under the transductive setting.

5.2 Datasets

We test our model and baselines on two datasets, Wikidata [17,18] and ICEWS14 [19]. Wikidata [17] is created by [18] from Wikidata, and contains the expiration timestamp of each triplet. Since the data is concentrated in certain years, we only used data from 2001 to 2020. ICEWS14 [19] consists of political events that happened in 2014, recorded in Integrated Crisis Early Warning System. For both datasets, we divided them into training, validation, and testing sets according to chronological order, but with different partitions for the multi-step setting and the transductive setting. The detailed statistics are shown in Table 5.1, 5.2, 5.3 and 5.4. In the multi-step setting, to test the model's ability to learn evolutionary patterns, it uses unseen data during training as the testing base graph. As illustrated in the base graph in Table 5.1 and 5.3, its time range $\mathcal{T}$ is non-overlapping with the training and validation sets. In the transductive setting, referencing RE-GCN [14] and CorDGT [15], the testing base graph uses all data used during training. As demonstated in Table 5.2 and 5.4, the time range $\mathcal{T}$ of the base graph is the summation of the training and validation sets.

5.3 Evaluation

To evaluate the model's performance in head entity prediction, tail entity prediction, and relation prediction tasks, we follow GSELI [10] and randomly sample 50 negative samples to be ranked together with the cor-

Table 5.1: Statistics of Wikidata in the multi-step setting. $|\mathcal{T}|$ is the number of timestamps, $|\mathcal{D}|$ implies the number of links, $|\mathcal{E}|$ means the number of entities, $|\mathcal{R}|$ suggests the number of relations, and $avg.$ means the average number in each timestamp.

	Training set		**Validation set**		**Testing set**	
	Pre-training	GUT	Pre-training	GUT	Base graph	Test links
$\|\mathcal{T}\|$	7 [2001, 2007]	4 [2008, 2011]	1 [2012]	2 [2013, 2014]	3 [2015, 2017]	3 [2018, 2020]
$\|\mathcal{D}\|$	40131	22923	5093	9336	8502	2015
avg. $\|\mathcal{D}\|$	-	5730.75±322.69	-	4668±687	-	671.667±781.33
$\|\mathcal{E}\|$	27087	18452	5939	9285	8704	2365
avg. $\|\mathcal{E}\|$	-	6498.75±309.14	-	5389.5±693.5	-	798±930.89
$\|\mathcal{R}\|$	99	85	56	76	77	43
avg. $\|\mathcal{R}\|$	-	57.75±2.05	-	58.5±6.5	-	17.667

Table 5.2: Statistics of Wikidata in the transductive setting.

	Training set		**Validation set**		**Testing set**	
	Pre-training	GUT	Pre-training	GUT	Base graph	Test links
$\|\mathcal{T}\|$	10 [2001, 2010]	4 [2011, 2014]	1 [2015]	2 [2016, 2017]	17 [2001, 2017]	3 [2018, 2020]
$\|\mathcal{D}\|$	57331	20152	4133	4369	85985	2015
avg. $\|\mathcal{D}\|$	-	5038±650	-	2184.5±143.5	-	-
$\|\mathcal{E}\|$	34168	16578	4965	5246	44901	2365
avg. $\|\mathcal{E}\|$	-	5184±682.29	-	2874.5±226.5	-	-
$\|\mathcal{R}\|$	117	87	59	65	134	43
avg. $\|\mathcal{R}\|$	-	57.25±4.76	-	51.5±2.5	-	-

Table 5.3: Statistics of ICEWS14 in the multi-step setting.

	Training set		**Validation set**		**Testing set**	
	Pre-training	GUT	Pre-training	GUT	Base graph	Test links
$\|\mathcal{T}\|$	153 [0, 3648]	74 [3672, 4524]	18 [5448, 5856]	37 [5880, 6744]	73 [6768, 8496]	10 [8520, 8736]
$\|\mathcal{D}\|$	24797	11907	2637	6962	14723	1373
avg. $\|\mathcal{D}\|$	-	160.91±51.01	-	188.16±55.57	-	137.3±47.09
$\|\mathcal{E}\|$	4733	3241	1370	2131	3114	626
avg. $\|\mathcal{E}\|$	-	178.52±50.9	-	194±46.75	-	135.4±38.2
$\|\mathcal{R}\|$	84	84	78	84	84	84
avg. $\|\mathcal{R}\|$	-	35.66±6.67	-	37.51±6.52	-	35.4±5.3

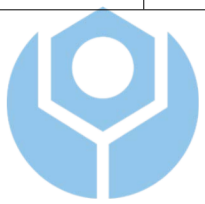


Table 5.4: Statistics of ICEWS14 in the transductive setting.

	Training set		**Validation set**		**Testing set**	
	Pre-training	GUT	Pre-training	GUT	Base graph	Test links
$\|\mathcal{T}\|$	229 [0, 5472]	72 [5496, 7200]	18 [7224, 7632]	36 [7656, 8496]	355 [0, 8496]	10 [8520, 8736]
$\|\mathcal{D}\|$	36865	13215	3657	7289	61026	1373
avg. $\|\mathcal{D}\|$	-	183.54±58.15	-	202.47±65.75	-	-
$\|\mathcal{E}\|$	5676	2900	1363	2358	6513	626
avg. $\|\mathcal{E}\|$	-	190.67±49.53	-	210.86±59.38	-	-
$\|\mathcal{R}\|$	84	84	78	83	84	84
avg. $\|\mathcal{R}\|$	-	36.81±6.66	-	38.61±6.48	-	-

responding positive sample. The model is then evaluated from two perspectives: ranking and binary classification. The ranking order of sample tests whether the model can assign higher scores to positive samples; the binary classification problem tests whether the model can correctly distinguish between positive and negative samples. The metrics used in these two aspects are as follows:

- Mean Reciprocal Rank (MRR)

  MRR is a measure of ranking accuracy that uses reciprocal rank to give lower-ranked samples lower scores; as a result, the positive samples are ranked higher, and the score is higher. It is denoted as:

$$MRR = \frac{1}{|\mathcal{D}|} \sum_{l \in \mathcal{D}} \frac{1}{rank(l)}. \tag{5.1}$$

  where $\mathcal{D}$ is the entire sample set.

- Hits ratio (Hits@k)

  Hits ratio calculates the proportion of samples whose ranking is less than or equal to k. In this study, we set k from 1, 5, 10 and calculate as:

$$Hits@k = \frac{\sum_{l \in \mathcal{D}} |rank(l) < k|}{|\mathcal{D}|}. \tag{5.2}$$

- Normalized Discounted Cumulative Gain (NDCG)

  DCG considers items' relevance and rankings to make higher-relevance items have a higher score when their rankings are higher, and it compares with the DCG of ideal ranking order (IDCG) to calculate

NDCG to assess the current ranking, defined as:

$$nDCG = \frac{DCG}{IDCG} = \frac{\sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}}{\sum_{i \in l}^{|REL_p|} \frac{2^{rel_i}-1}{\log_2(i+1)}}. \tag{5.3}$$

Since in our setting, only the positive sample is the highly relevant sample, we implement NDCG by the following equation:

$$nDCG = \frac{DCG}{IDCG} = \frac{\frac{1}{\log_2(rank(l)+1)}}{\frac{1}{\log_2(2)}}. \tag{5.4}$$

- Accurcy (ACC)

  Accuracy is the simplest metric to evaluate a classification task. It focuses on the proportion of correct prediction across all samples, denoted as:

$$Accuracy = \frac{TP + TN}{|\mathcal{D}|}, \tag{5.5}$$

  where TP is the number of positive samples predicted correctly, and TN is the number of negative samples being predicted correctly.

- Area Under Receiver Operating characterostic Curve (AUROC)

  AUROC is the area under the curve formed by FPR (False Positive Rate) and TPR (True Positive Rate). FPR represents the proportion of samples that are actually negative and are judged as positive, and TPR represents the proportion of samples that are actually positive and are judged as positive. It provides an intuitive metric for measuring a model' s ability to classify positive and negative samples.

- Area Under Precision-Recall Curve (AUPRC)

  AUPRC is the area under the curve formed by Precision and Recall.

Precision emphasizes the proportion of samples that the model classifies as positive and are actually positive, while Recall emphasizes the proportion of samples that are actually positive and were classified as positive. It considers the impact of false positives on negative samples and evaluates the model's ability to correctly classify samples with a large number of negative samples.

- F1-score

  F1-score is the harmonic mean of Precision and Recall, representing the consideration of both metrics. In this work, we calculate it as:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{5.6}$$

- Balanced accuracy (Balanced ACC)

  Balanced accuracy considers both the correct identification of positive and negative samples, requiring the model to be as accurate as possible in its judgments of both. Sacrificing accuracy for one side at the expense of the other will decrease balanced accuracy. The equation is as follows:

$$Balanced\ accuracy = \frac{sensitivity + specificity}{2} \tag{5.7}$$

  where sensitivity represents the proportion of samples that are actually positive and are judged as positive, same as Recall. Specificity represents the proportion of samples that are actually negative and are judged as negative.

5.4 Baselines

We compare MODEL with the following methods, including semi-inductive link prediction and temporal knowledge graph reasoning methods.

- DEKG-ILP [9] extends inductive link prediction to contain bridging links to apply to more situations. It uses semantic information learning by contrastive learning and topological information to predict both enclosing and bridging links.
- GSELI [10] is an improved model based on DEKG-ILP [9]. It adds a more efficient subgraph extraction module and neighboring relational paths modeling from SNRI [12], increasing prediction accuracy.
- RE-GCN [14] tries to integrate static and dynamic features into representations for temporal graph reasoning. It learns entity and relation representations by capturing structural dependencies within a single timestamp and sequential patterns across timestamps, and incorporates the static properties of the graph to contain stable features. In this way, it achieves the best performance on several datasets.
- CorDGT [15] proposes a Transformer-based model with a novel method to extract proximity more efficiently to capture comprehensive features of graphs. It employs the Poisson point process assumption to estimate temporal features and encode them with spatial features to

obtain high-order proximity, and uses the multi-head self-attention mechanism to enhance expressive power.

5.5 Results

This section presents the experimental results across three aspects to evaluate the performance of the proposed model, EvoGUT. First, we analyze ranking results under both multi-step and transductive settings to assess prediction accuracy. Second, we examine the classification performance to verify the model's ability to predict link existence. Finally, we conduct ablation studies to quantify the contributions of each component.

5.5.1 Ranking

We evaluate our model and baselines under multi-step and transductive settings by ranking order metrics.

Multi-Step Temporal Link Prediction

Tables 5.5, 5.6, and 5.7 report the experimental results of our method and baseline models under the multi-step setting. On Wikidata, although DEKG-ILP [9] and GSELI [10] are designed to handle scenarios with emerging entities, their reliance on structural features limits their performance on the sparser dataset. By enhancing GSELI [10] with temporal feature

Table 5.5: The average results of multi-step temporal link prediction.

Method	Wikidata					ICEWS14				
	MRR	Hits@1	Hits@5	Hits@10	NDCG	MRR	Hits@1	Hits@5	Hits@10	NDCG
DEKG-ILP	*20.943*	*11.824*	*30.766*	*34.307*	*34.672*	21.856	13.966	26.882	35.030	36.127
GSELI	9.677	5.103	12.363	20.810	14.087	**24.072**	*13.729*	**32.330**	*43.332*	**38.745**
EvoGUT (w/o)	39.405	26.464	51.267	**73.334**	52.313	*22.266*	12.084	*29.797*	44.478	*37.292*
EvoGUT	**51.279**	**41.279**	**59.433**	72.066	**61.346**	23.839	**14.013**	30.615	**45.747**	38.673

Table 5.6: The results of multi-step temporal link prediction on Wikidata.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG
DEKG-ILP	5.896	3.093	4.058	5.395	21.340	**48.447**	**28.339**	**81.699**	**89.271**	**60.411**	7.137	4.040	6.539	8.253	22.266
GSELI	*9.524*	*4.615*	*8.085*	*11.067*	*22.707*	*25.428*	*4.267*	*33.575*	52.146	40.284	*17.400*	*8.918*	*19.991*	*32.585*	*32.203*
EvoGUT (w/o)	42.860	24.833	69.242	**79.676**	55.424	24.129	9.778	31.986	70.848	*40.567*	51.225	44.781	52.574	69.478	60.947
EvoGUT	**61.460**	**49.244**	**73.027**	78.220	**69.499**	30.484	18.271	39.370	*60.682*	45.038	**61.892**	**56.321**	**65.903**	**77.297**	**69.500**

Table 5.7: The results of multi-step temporal link prediction on ICEWS14.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG
DEKG-ILP	28.693	19.430	37.154	44.006	41.853	7.427	1.244	7.125	18.524	24.607	**29.446**	**21.226**	36.366	42.560	41.922
GSELI	**32.370**	**19.925**	**44.970**	**57.472**	**45.957**	*10.719*	*3.836*	*11.794*	*20.561*	*27.143*	29.128	17.426	**40.226**	**51.965**	**43.135**
EvoGUT (w/o)	*26.187*	*16.522*	33.546	*45.203*	*40.732*	14.910	5.356	19.778	**36.728**	30.589	25.702	14.374	*36.067*	51.502	40.556
EvoGUT	25.904	15.306	*34.233*	52.166	40.559	**19.628**	**12.067**	**21.811**	34.047	**34.777**	*25.984*	*14.666*	35.799	*51.028*	*40.682*

learning and a new training framework, our model achieves excellent performance. On ICEWS14, DEKG-ILP [9] maintains a similar performance on average, while GSELI [10] exhibits a noticeable increase. While our model outperforms the baselines on average, its advantage is less evident compared with its performance on Wikidata, where a clearer performance margin is observed.

We speculate that this behavior arises from the intrinsic characteristics of the ICEWS14 dataset. ICEWS14 spans a relatively shorter overall time period but contains a large number of timestamps, reflecting frequent and fine-grained temporal changes. Furthermore, its content is primarily composed of news events, whose influence often decays rapidly over time. Therefore, temporal information and the duration of influence play a less critical role in this setting. Additionally, the relatively dense graph structure encourages the model to rely more heavily on structural and semantic features for prediction, rather than temporal features.

In contrast, Wikidata exhibits coarser timestamp granularity and a sparser graph structure, resulting in suboptimal structural representations. In this case, temporal features can effectively compensate for the limitations of structural information and thereby improve overall performance. However, because our model adopts a pruning-based updating strategy, less influential triplets are discarded during graph updates. On ICEWS14, where structural information is particularly important, such pruning probably leads to structural information loss, resulting in inferior performance compared to Wikidata.

For relation prediction shown in Tables 5.6 and 5.7, our model achieves suboptimal performance on Wikidata, but outperforms other methods on ICEWS14. One possible explanation is that relations are generally less sensitive to temporal information than entities. Even when the influence of certain information diminishes over time, it can still provide valuable structural cues for relation prediction. By removing such information, our model slightly decreases performance on Wikidata. In contrast, ICEWS14 is characterized by highly transient events where historical data can quickly become obsolete and accumulate as noise. In this scenario, appropriate information pruning helps reduce noise and leads to improved relation prediction performance.

Transductive Link Prediction

Table 5.8: The results of transductive temporal link prediction on Wikidata.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG
RE-GCN	9.333	2.839	9.846	18.233	25.903	**62.149**	**59.970**	*63.097*	64.576	**68.499**	*52.089*	46.203	*57.151*	59.404	*60.752*
CorDGT	**46.572**	**41.449**	48.427	49.975	**56.543**	-	-	-	-	-	30.033	23.990	33.022	35.553	42.693
DEKG-ILP	28.805	22.147	29.784	34.913	*42.023*	*55.572*	*36.426*	**84.654**	90.749	*65.872*	39.425	35.288	42.882	44.294	49.624
GSELI	*28.872*	*22.219*	*31.181*	35.202	41.911	26.616	10.980	49.035	62.161	41.540	34.522	22.036	49.890	*60.158*	47.659
EvoGUT (w/o)	20.211	13.401	21.830	*37.450*	34.911	58.090	42.473	81.398	**91.456**	67.751	55.327	*37.651*	79.308	87.766	65.385
EvoGUT	41.222	23.982	**64.826**	**73.721**	53.650	23.735	5.756	46.294	*69.884*	40.264	**71.078**	**60.741**	**84.273**	**89.796**	**77.408**

Tables 5.8 and 5.9 report the performance of our model and baselines on the two datasets. In the transductive setting, all entities and relations are known, which corresponds to a scenario favored by most temporal graph reasoning methods. Due to the reduced uncertainty, models typically achieve better performance in this setting than in the multi-step setting.

Table 5.9: The results of transductive temporal link prediction on ICEWS14.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG
RE-GCN	**84.270**	76.766	**93.839**	**96.519**	**87.941**	**72.817**	**97.925**	**85.667**	**92.615**	**77.877**	**90.006**	**84.910**	**95.703**	**97.131**	**92.303**
CorDGT	*81.387*	**76.778**	*86.992*	*89.083*	*85.173*	-	-	-	-	-	*74.060*	*70.096*	*87.678*	*91.825*	*79.867*
DEKG-ILP	82.312	*73.809*	93.183	96.053	86.456	*13.389*	2.797	*16.766*	42.520	*30.816*	84.626	77.757	93.270	96.373	88.185
GSELI	65.540	54.479	77.903	86.045	73.128	15.155	*5.287*	20.874	*37.145*	31.691	68.228	57.698	79.723	87.313	75.281
EvoGUT (w/o)	47.073	33.372	61.296	79.359	58.605	13.381	5.769	15.805	26.701	29.475	36.261	24.706	46.724	62.032	49.223
EvoGUT	47.119	29.949	68.318	78.004	58.639	5.862	0.801	4.792	11.187	22.540	34.873	21.787	48.789	57.878	47.880

On Wikidata, consistent with the observations in the multi-step setting, our model achieves excellent performance in entity prediction but performs relatively poorly in relation prediction. On ICEWS14, it lags behind other baselines. This is because our model focuses on the capability of handling dynamic, unknown structures and continuous prediction, strengthening its performance in the multi-step setting, thus sacrificing its ability in static graphs, which emphasize structural features. However, on Wikidata, it still maintains better performance in entity prediction by utilizing temporal features to compensate for the lack of structural features.

Both RE-GCN [14] and CorDGT [15], originally designed for temporal graphs, perform relatively well. However, RE-GCN [14] perform poorly in head entity prediction on Wikidata as reported in Table 5.8. This is presumably due to the inverse relations used during learning are not derived from real datasets. Since Wikidata is sparse and RE-GCN [14] does not leverage a global static graph on this dataset, it lacks aggregateble information, thus offering little help for head entity prediction. In contrast, on ICEWS14, by including a global static graph as a reference, even if the inverse relations

do not actually exist, there is sufficient information to build a good representation of them. Similarly, CorDGT [15] also relies on a global static graph, thus performing slightly worse on Wikidata, the sparser dataset.

5.5.2 Binary Classification

Table 5.10: The results of binary classification under multi-step setting on Wikidata.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	ACC	AUROC	F1-score	AUPRc	Balanced ACC	ACC	AUROC	F1-score	AUPRC	Balanced ACC	ACC	AUROC	F1-score	AUPRC	Balanced ACC
DEKG-ILP	57.578	*83.235*	5.890	*83.563*	61.986	68.800	*69.515*	3.142	*70.113*	54.213	65.648	*69.368*	3.541	*69.969*	*52.461*
GSELI	*60.592*	81.973	*6.006*	82.326	*62.235*	**72.840**	68.840	*3.751*	69.450	*55.611*	**80.822**	68.558	4.610	69.177	**59.406**
EvoGUT(w/o)	79.265	89.560	15.112	89.765	79.195	41.983	**90.273**	**5.269**	**90.464**	**60.879**	23.376	**90.167**	*3.964*	**90.359**	51.285
EvoGUT	**87.131**	**90.342**	**24.430**	**90.531**	**83.971**	*58.629*	77.704	4.914	78.142	57.051	*33.553*	88.822	**4.671**	89.041	55.158

Table 5.11: The results of binary classification under multi-step setting on ICEWS14.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	ACC	AUROC	F1-score	AUPRC	Balanced ACC	ACC	AUROC	F1-score	AUPRC	Balanced ACC	ACC	AUROC	F1-score	AUPRC	Balanced ACC
DEKG-ILP	**49.343**	79.504	4.418	79.941	54.114	**64.559**	81.384	6.508	81.749	*63.682*	*42.595*	81.647	4.251	82.336	52.737
GSELI	45.238	*86.663*	**5.001**	*86.925*	59.001	56.237	*86.697*	*6.213*	*86.806*	64.644	33.940	90.097	*4.638*	90.292	*56.605*
EvoGUT (w/o)	*32.467*	90.613	*4.507*	90.797	*56.359*	41.027	92.650	5.369	92.824	62.721	**51.952**	*88.525*	5.966	*88.750*	64.250
EvoGUT	31.469	**95.672**	4.967	**95.708**	**60.759**	*49.330*	**95.132**	**6.594**	**95.251**	**69.411**	48.158	**95.910**	**6.546**	**95.990**	**69.552**

Even though the model performs well on ranking metrics, demonstrating its ability to distinguish between positive and negative samples in a relative sense, this does not necessarily demonstrate strong performance in absolute, instance-level discrimination. Therefore, we further verify the performance of our model and baselines on the link classification task under the multi-step setting, as shown in Tables 5.10 and 5.11. The most commonly used classification metrics include accuracy, AUROC, and F1-score. However, under extreme class imbalance between positive and neg-

ative samples, accuracy and AUROC can be misleading due to the dominance of negative samples, while F1-score does not consider the accuracy of negative samples. Therefore, we additionally report AUPRC and balanced accuracy, which are more suitable for evaluating performance in highly imbalanced scenarios. AUPRC focuses on the ability to avoid misclassifying positive samples among a large number of negative samples, while balanced accuracy reflects the average ability to distinguish between positive and negative samples. These two metrics are therefore particularly suitable for evaluating graph completion performance. Our model achieves the best performance on both metrics for almost all tasks, demonstrating its strong potential as an effective graph completion tool.

In the relation prediction on Wikidata, the slight decrease in AUPRC shows the effect of considering unseen relations. Nevertheless, our model still maintains good performance than GSELI [10] and DEKG-ILP [9] since the superiority of its ability to distinguish negative samples. In tail entity prediction, while the model's ability to identify positive samples was sufficient, its ability to identify negative samples showed a significant decline compared to the other two tasks, causing a decrease in balanced accuracy. When considered alongside the ranking metrics, high-ranking performances indicate that the model gives higher scores to positive samples. However, not all negative samples are given scores below the threshold. This phenomenon is likely due to the threshold-setting strategy. When adjusting the threshold, the model uses the scores of positive and negative samples at certain percentiles as a reference for determining the threshold. To avoid overfitting, we expect the model to filter out 75% of negative sam-

ples, leaving a 25% margin. Consequently, some difficult-to-distinguish negative samples are treated as positive samples, which explains the observed decline in negative-sample discrimination.

5.5.3 Ablation Study

We conduct some ablation studies to verify each component's effect and the differences between different configurations under the multi-step setting, and show the results in Tables 5.12 and 5.13. All variants evaluated are listed as follows.

- -tf: Remove the entire temporal feature.
- -il: Remove only the influence lifetime.
- w/o GUT: Train the model without graph-updating training.
- Accumu: Use accumulation as the graph updating strategy for both training and testing.
- Gt: Use the ground truth graph as the input graph for both training and testing iterations to verify the effect of graph-updating training under ideal input conditions.
- Gt/Pruning: Use the ground truth graph as the input graph for each training iteration, but use pruning as the graph updating strategy when testing. This configuration reflects practical settings where real-time

graph updates are typically infeasible, resulting in the absence of newly updated information during graph reasoning.

- EvoGUT: Use entire temporal feature and train the model with graph-updating training. For the graph updating strategy, use pruning for both training and testing.

Table 5.12: The results of ablation studies under multi-step temporal link prediction on Wikidata.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG
-tf	21.073	3.364	48.815	76.442	38.475	11.017	3.803	15.006	18.931	26.963	**66.543**	**58.551**	*77.642*	85.202	**73.508**
-il	34.200	15.188	60.756	68.979	48.358	19.521	9.281	27.348	31.975	34.952	14.939	6.260	21.997	28.796	30.943
w/o GUT	42.860	24.833	69.242	**79.676**	55.424	*24.129*	*9.778*	*31.986*	70.848	*40.567*	51.225	44.781	52.574	69.478	60.497
Accumu	53.049	40.186	*68.312*	78.039	63.087	**48.107**	**28.055**	**75.123**	**78.487**	**59.708**	37.578	26.761	43.542	64.886	50.415
Accumu-tf	28.427	15.177	40.821	76.955	43.673	12.205	1.981	16.701	32.467	29.616	64.020	*49.376*	**80.278**	**85.448**	71.926
Accumu-il	38.611	20.078	65.484	76.803	81.976	13.010	2.794	23.438	34.446	29.505	49.374	30.352	79.013	*85.109*	60.710
Gt	21.404	3.156	53.979	78.103	38.621	8.904	3.757	6.060	10.508	25.657	22.629	3.794	51.665	83.192	39.701
Gt/Pruning	*46.729*	*30.019*	68.234	78.417	*58.388*	8.718	3.267	6.041	10.253	25.574	13.403	2.461	17.646	41.231	30.996
EvoGUT	**61.460**	**49.244**	**73.027**	*78.220*	**69.499**	30.484	18.271	39.370	*60.682*	45.038	*61.892*	56.321	65.903	77.297	*69.500*

Table 5.13: The results of ablation studies under multi-step temporal link prediction on ICEWS14.

Method	Head entity prediction					Relation prediction					Tail entity prediction				
	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG	MRR	H@1	H@5	H@10	NDCG
-tf	33.856	22.724	44.285	54.468	46.888	9.660	2.538	11.570	20.283	26.315	31.559	20.306	41.492	53.616	45.158
-il	34.571	22.209	47.394	57.175	47.656	11.504	*4.617*	12.888	20.273	27.696	38.063	25.634	49.978	61.426	50.677
w/o GUT	26.187	16.522	33.546	45.203	40.732	14.910	5.356	19.778	**36.728**	30.589	25.702	14.374	36.067	51.502	40.556
Accumu	48.155	34.003	65.311	73.606	59.126	*11.865*	4.290	14.728	24.686	28.154	52.276	*37.623*	69.336	78.216	62.593
Accumu-tf	**61.321**	**50.953**	71.484	80.220	**69.537**	10.945	4.132	11.956	22.013	27.333	**64.495**	**55.258**	73.615	*80.450*	**71.952**
Accumu-il	58.934	47.708	*71.178*	*78.797*	67.619	11.132	3.041	13.912	26.275	27.861	61.692	50.763	*73.325*	80.681	69.897
Gt	*52.445*	*35.274*	**74.597**	**80.964**	*62.974*	8.735	1.715	9.088	20.484	25.700	*54.592*	37.336	**76.718**	**85.976**	*64.881*
Gt/Pruning	19.360	6.551	32.390	49.325	35.606	11.754	2.327	*15.136*	*33.180*	*28.901*	20.703	7.436	34.241	50.761	36.747
EvoGUT	25.904	15.306	34.233	52.166	40.559	**19.628**	**12.0670**	**21.811**	34.047	**34.777**	25.984	14.666	35.799	51.028	40.682

In entity prediction, tail entity prediction relies more heavily on graph structural information than head entity prediction. Since a head entity has a high probability of establishing the same relation with several tail entities, referencing whether the head entity has similar connections with other entities provides a more stable predictive basis than temporal information. In other words, head entities are difficult to predict solely based on structural information. For example, many countries have hosted the Olympics, and the year is crucial in identifying which country hosted the Games. However, in ICEWS14, due to its frequent changes and short timestamp intervals, the discrimination of temporal features is weakened, resulting in head entity prediction performing better without relying on temporal information, as shown by the -tf and -il variants outperforming the full model in Table 5.13, together with consistent trends observed across the Accumu, the Accumu-tf, and Accumu-il variants.

Regarding the choice of graph updating strategy, the two datasets show inconsistent trends. Overall, using the accumulation strategy helps improve the overall model performance, as shown in the Accumu variant of both tables. Under this strategy, information is not removed from the graph, and even weakly influence information is continuously accumulated, thereby strengthening the quality of structural features. This effect is particularly evident on ICEWS14, which relies more heavily on structural features. However, when considering temporal features, entity prediction in Wikidata and relation prediction in ICEWS14 exhibit different trends.

For Wikidata, using the pruning strategy, which is implemented in

EvoGUT, to remove triplets whose influence has decayed can improve the accuracy of entity prediction, as shown in Table 5.12. This is because a large portion of Wikidata facts is highly time-sensitive; for example, heads of state change according to their terms, and many international events are held at different locations over time. Therefore, appropriately removing outdated information allows the model to focus more effectively on currently influential information, thereby improving prediction accuracy. However, this trend does not extend to relation prediction, as relations are generally less sensitive to temporal variation than entities, which is reflected in the performance gap between EvoGUT and Accumu variant in Table 5.12. In contrast, ICEWS14 places greater emphasis on structural features, and the accumulation strategy therefore tends to perform better, as reported in Table 5.13 where the Accumu variant excels in entity predictions over EvoGUT. However, for relation prediction, the benefits of temporal information and the influence duration are limited. Continuously, accumulating such information generally introduce noise, which can negatively affect prediction accuracy, as shown in the comparison between EvoGUT and the Accumu variant in Table 5.13.

Chapter 6 Conclusions

For the ever-evolving temporal knowledge graph, we proposed a framework for continuous prediction by training the model to update the graph step-by-step. This framework improved model robustness by simulating graph updates during training. In addition to structural and semantic features, we incorporated the temporal information and estimated influence lifetime of information as features, successfully compensating for the lack of structural features on the sparse dataset, Wikidata. Ultimately, the model learned to determine the existence of triplets through various features and can adopt different graph updating strategies for different tasks and contexts, achieving excellent performance in the multi-step setting. It demonstrated the ability to handle dynamic or unknown graph structures, as well as emerging entities and relations. Furthermore, our model showed considerable potential in graph completion. Compared to other baselines, our model can more accurately distinguish the existence of positive and negative samples in most situations, demonstrating the advantages of graph updating training. It can also serve as a graph completion tool, reducing the need for human efforts and costs.

Chapter 7 Future Works

In the future, there are several promising directions for further development. First, regarding graph-updating strategies, the current choice of strategy relies largely on empirical rules. We aim to implement reinforcement learning or other adaptive mechanisms to dynamically determine which links should be retained during the evolutionary process, rather than a fixed criterion.

Second, to achieve higher precision in handling emerging relations, additional designs or tasks are required to mitigate the relation cold-start problem. For instance, employing line graphs could transform emerging relations into emerging entities, thereby enabling the model to learn various relational features more effectively.

Lastly, research involving knowledge graphs frequently encounters high time complexity in both training and inference stages due to the immense scale of the graph. To address this, future work will explore lightweight architectures to minimize processing latency. Enhancing the efficiency of training and inference will significantly improve the model's utility as a practical tool for knowledge graph completion.

References

[1] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. NeurIPS18*, Dec. 2018.

[2] Y. Peng and J. Zhang, "LineaRE: Simple but powerful knowledge graph embedding for link prediction," in *Proc. IEEE ICDM20*, Nov. 2020.

[3] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. ESWC18*, June 2018.

[4] K. Teru, E. Denis, and W. Hamilton, "Inductive relation prediction by subgraph reasoning," in *Proc. ICML20*, July 2020.

[5] A. Mitra, P. Vijayan, S. R. Singh, D. Goswami, S. Parthasarathy, and B. Ravindran, "Revisiting link prediction on heterogeneous graphs with a multi-view perspective," in *Proc. IEEE ICDM22*, Nov./Dec. 2022.

[6] B. Ruan, C. Zhu, and W. Zhu, "A link prediction model of dynamic heterogeneous networks based on transformer," in *Proc. IEEE IJCNN22*, July 2022.

[7] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proc. ACM WSDM20*, pp. 519–527, Jan. 2020.

[8] D. Wang, Z. Zhang, Y. Ma, T. Zhao, T. Jiang, N. V. Chawla, and M. Jiang, "Modeling co-evolution of attributed and structural information in graph sequence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1817–1830, 2023.

[9] Y. Zhang, W. Wang, H. Yin, P. Zhao, W. Chen, and L. Zhao, "Disconnected emerging knowledge graph oriented inductive link prediction," in *Proc. IEEE ICDE23*, pp. 381–393, Apr. 2023.

[10] X. Liang, G. Si, J. Li, Z. An, P. Tian, F. Zhou, and X. Wang, "Integrating global semantics and enhanced local subgraph for inductive link prediction," *International Journal of Machine Learning and Cybernetics*, vol. 16, no. 3, pp. 1971–1990, 2024.

[11] S. Zheng, S. Mai, Y. Sun, H. Hu, and Y. Yang, "Subgraph-aware few-shot inductive link prediction via meta-learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 6512–6517, 2022.

[12] X. Xu, P. Zhang, Y. He, C. Chao, and C. Yan, "Subgraph neighboring relations infomax for inductive link prediction on knowledge graphs," in *Proc. IJCAI22*, pp. 2341–2347, July 2022.

[13] Y. Yang, J. Cao, M. Stojmenovic, S. Wang, Y. Cheng, C. Lum, and Z. Li, "Time-capturing dynamic graph embedding for temporal linkage evolution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 958–971, 2023.

[14] Z. Li, X. Jin, W. Li, S. Guan, J. Guo, H. Shen, Y. Wang, and X. Cheng, "Temporal knowledge graph reasoning based on evolutional representation learning," in *Proc. ACM SIGIR21*, pp. 408–417, July 2021.

[15] Z. Wang, S. Zhou, J. Chen, Z. Zhang, B. Hu, Y. Feng, C. Chen, and C. Wang, "Dynamic graph transformer with correlated spatial-temporal positional encoding," in *Proc. ACM WSDM25*, pp. 60–69, Mar. 2025.

[16] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases." arXiv:1412.6575, Dec. 2014.

[17] C. Mavromatis, P. L. Subramanyam, V. N. Ioannidis, A. Adeshina, P. R. Howard, T. Grinberg, N. Hakim, and G. Karypis, "Tempoqr: Temporal question reasoning over knowledge graphs," in *Proc. AAAI22*, pp. 5825–5833, Feb./Mar. 2022.

[18] T. Lacroix, G. Obozinski, and N. Usunier, "Tensor decompositions for temporal knowledge base completion," in *Proc. ICLR20*, Apr./May 2020.

[19] Z. Han, P. Chen, Y. Ma, and V. Tresp, "Explainable subgraph reasoning for forecasting on temporal knowledge graphs," in *Proc. ICLR21*, May 2021.