Towards Pattern-aware Data Augmentation for Temporal Knowledge Graph Completion

Jiasheng Zhang
Xidian University
Xi'an, China
zjss12358@gmail.com

Deqiang Ouyang*
Chongqing University
Chongqing, China
deqiangouyang@cqu.edu.cn

Shuang Liang
Jie Shao
University of Electronic Science and Technology of China
Chengdu, China
{shuangliang,shaojie}@uestc.edu.cn

ABSTRACT

Predicting missing facts for temporal knowledge graphs (TKGs) is a fundamental task, called temporal knowledge graph completion (TKGC). One key challenge in this task is the imbalance in data distribution, where facts are unevenly spread across entities and timestamps. This imbalance can lead to poor completion performance for long-tail entities and timestamps, and unstable training due to the introduction of false negative samples. Unfortunately, few previous studies have investigated how to mitigate these effects. Moreover, for the first time, we found that existing methods suffer from model preferences, revealing that entities with specific properties (e.g., recently active) are favored by different models. Such preferences will lead to error accumulation and further exacerbate the effects of imbalanced data distribution. To alleviate the impacts of imbalanced data and model preferences, we introduce *Booster*, the first data augmentation strategy for TKGs. The unique requirements here lie in generating new samples that fit the complex semantic and temporal patterns within TKGs, and identifying hard-learning samples specific to models. Therefore, we propose a hierarchical scoring algorithm based on triadic closures within TKGs. By incorporating both global semantic patterns and local time-aware structures, the algorithm enables pattern-aware validation for new samples. Meanwhile, we propose a two-stage training approach to identify samples that deviate from the model's preferred patterns. With a frequency-based filtering strategy, this approach also helps to avoid the misleading of false negatives. Experiments justify that *Booster* can seamlessly adapt to existing TKGC models and achieve on average 4.5% performance improvement.

PVLDB Reference Format:
Jiasheng Zhang, Deqiang Ouyang, Shuang Liang, and Jie Shao. Towards Pattern-aware Data Augmentation for Temporal Knowledge Graph Completion. PVLDB, 18(10): 3573 - 3586, 2025.
doi:10.14778/3748191.3748216

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zjs123/Booster.

*Corresponding author: Deqiang Ouyang.

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:10.14778/3748191.3748216

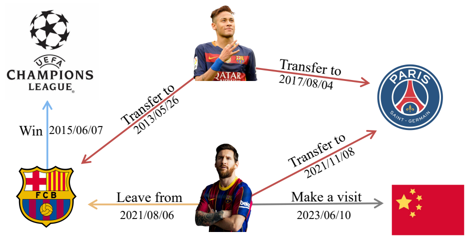


Figure 1: An illustration of temporal knowledge graph.

1 INTRODUCTION

Temporal knowledge graphs (TKGs) structure dynamic human knowledge and are widely used in applications such as event prediction [44] and recommendation systems [70]. As shown in Figure 1, a TKG is a dynamic directed graph where nodes represent real-world entities and labeled edges denote temporal relations. Each edge forms a fact (s, r, o, t), such as $(\text{Messi}, \text{Transfer to}, \text{PSG}, 2021/11/8)$.

However, due to update delays and extraction limitations [37, 43, 71], TKGs are often incomplete, missing real-world facts. Temporal knowledge graph completion (TKGC) [50] addresses this issue by predicting missing facts to enhance TKG quality and support downstream tasks [17]. Existing TKGC methods fall into two main categories: timestamp embedding models [9, 25, 58], which encode each timestamp, and dynamic embedding models [15, 52, 57], which learn evolving representations for entities and relations over time.

Despite their effectiveness, recent studies have shown that TKGs suffer from imbalanced data distribution [52], which may seriously impair the performance of TKG completion. Unfortunately, most existing methods overlook this aspect. They only report performance improvement on several metrics (e.g., mean reciprocal rank (MRR)) without thoroughly analyzing how the imbalanced data impacts their performance and how to alleviate such impacts, leading to less convincing and unsatisfactory results.

Previous limitations. The number of facts in TKGs varies greatly across entities and timestamps. While a few have rich fact descriptions, most contain only limited information, leading to a highly imbalanced data distribution. Revisiting the performance of existing completion methods, we identify two major issues caused by this imbalance: **1) Unstable training.** Most methods adopt contrastive training [53], treating observed facts as positives and all others as negatives. However, many valid facts are missing and wrongly labeled as negatives (i.e., false negatives), which misleads training and degrades performance, especially for sparsely represented entities. **2) Imbalanced performance.** The uneven distribution across timestamps leads to significant performance fluctuations, even between adjacent timestamps. Ignoring this imbalance further amplifies the inconsistency during training (Section 3.3).

While addressing the above issues, we discovered a previously overlooked problem: the **model preference issue**, which can exacerbate the effects of data imbalance. During completion, different models inherently favor entities with certain properties due to their architectural biases (Section 3.4). For instance, tensor factorization models [25] favor frequently interacted entities, while recurrent neural network-based models [52] prioritize recently active ones. These preferences hinder the models' ability to learn TKG patterns that diverge from their biases, especially when valid instances are mistakenly treated as false negatives. These limitations can severely impact downstream applications. For instance, in recommendation systems, ignoring the skewed distribution of item interactions often leads to overexposure of popular items and neglect of long-tail ones, resulting in suboptimal matches. In question answering, overlooking model preferences may cause consistent favoritism toward entities with certain attributes, compromising fairness. Such issues reveal these limitations as key bottlenecks in real-world use.

Technical challenges. Although several data augmentation methods have been proposed to address data imbalance issues in general graphs and static knowledge graphs [10, 36, 45], they face key challenges when applied to TKGs: **1) False negative filtering.** Some of them simply filter neighboring nodes as false negatives [69], failing to consider various components (i.e., entities, relations, and timestamps) and their temporal distributions within TKGs' complex graph structure. **2) New sample generation.** They generate new samples solely based on node connectivity. However, TKGs have intricate semantic and temporal patterns brought by diverse relations and time-evolving topology. New samples must therefore fit with these patterns. **3) Computational cost.** Most methods depend on resource-intensive techniques such as path searching, which become impractical with the long temporal sequences in TKGs. **4) Model preference.** They often train models directly on the augmented data without addressing model preferences, resulting in poor generalization to diverse temporal patterns.

The proposed work. We present *Booster*, the first pattern-aware data augmentation strategy specialized to TKGs to tackle the imbalanced data and model preference issues. For false negative filtering, *Booster* employs a frequency-based filtering strategy that is customized for different components of TKGs, effectively capturing both intra- and inter-component interactions as well as their temporal frequency distributions. For new sample generation, it utilizes a hierarchical scoring algorithm that jointly considers global semantic patterns shared across the TKG and temporal dynamics within local subgraphs. Moreover, to reduce the computational overhead caused by long historical contexts, we propose the use of a time-irrelevant graph and a decoupled triangle counting mechanism to accelerate the scoring. In addition, a novel time-aware perturbation strategy is developed to produce more robust augmented samples. Finally, *Booster* adopts a two-stage training paradigm: it first pre-trains the model to detect preference-deviated facts and then fine-tunes it on these identified facts, effectively enhancing the model's ability to generalize across different patterns while mitigating the effects of data imbalance. Experiments on five real-world TKGs demonstrate that *Booster* consistently improves performance by an average of 4.5% across 25 scenarios (5 backbones × 5 datasets), showcasing its generalization, while reducing the standard deviation across timestamps by 18.5%, indicating better balance over skewed temporal distribution. It also lowers performance variance across training runs by 18.8% on average, highlighting its robustness against false negatives and its effectiveness in stabilizing the training of existing models. Our main contributions are as follows:

- We make the first attempt to investigate the imbalanced data and model preference issues of TKG completions.
- We propose *Booster*, the first pattern-aware data augmentation strategy tailored to TKGs, which can generate new samples fitting TKG patterns and enhance the model's generalization ability and performance balance.
- Experimental results show that *Booster* can effectively improve the performance of existing TKGC models.

2 RELATED WORK

2.1 Temporal Knowledge Graph Completion

Temporal knowledge graph completion (TKGC) aims to predict missing facts based on observed ones. Existing approaches can be broadly categorized into two types: Timestamp embedding methods [25, 60, 64], which learn separate embeddings for entities, relations, and timestamps. For instance, HyTE [9] incorporates learnable timestamp embeddings into the TransE model's translation function [1]. TNT [25] employs 4-way tensor factorization, extended by Timeplex [22] to capture recurrent relational patterns, and TELM [56] to learn multi-vector representations via canonical decomposition. More recently, QDN [48] introduces a quadruplet distributor network to enhance factorization, while MADE [49] explores multi-curvature embeddings. Dynamic embedding methods [9, 20, 58, 65] learn time-evolving representations to capture semantic drift. DE [15] applies nonlinear operations to model diverse temporal trends. TA [13] leverages sequence models for time-specific relation embeddings. CENET [59] uses historical contrastive learning for temporal dependency modeling. Additionally, recent work incorporates graph neural networks [14] to exploit structural patterns in TKGs. TEMP [52] applies self-attention to capture spatio-temporal locality, RE-GCN [29] models temporal sequences autoregressively, and LogCL [6] learns both local and global historical structures.

Despite their progress, most methods overlook the problem of imbalanced data distribution [52]. They neither analyze its impact nor propose remedies, which may compromise result reliability. Although TILP [55] claims robustness via logic rule-based reasoning, it lacks generalizability across different TKGC models.

2.2 Graph Data Augmentation

Recent studies have proposed diverse data augmentation strategies to improve graph data quality and alleviate issues such as imbalanced distributions and degree bias [8, 19, 33, 36, 66]. For instance, AIA [41] applies adversarial masking to address distribution shifts, while GraphPatcher [23] introduces virtual nodes to mitigate degree bias. Some works extend augmentation to temporal graphs: MeTA [51] modifies temporal structures and features for robustness, TGEditor [68] performs task-guided editing, TagRec [38] perturbs temporal information adaptively, and TTDA [32] maximizes mutual information between original and augmented embeddings. However, these methods largely ignore the rich semantics introduced by heterogeneous relations, limiting their ability to generate temporally and semantically coherent samples for TKGs.

Imbalanced data has also drawn attention in static knowledge graphs [24, 35, 62]. Methods such as NSCaching [69] and DeMix [7] aim to reduce false negatives through importance sampling and self-supervised selection, respectively. Yet, they struggle with TKGs due to their neglect of temporal dynamics and reliance on computationally expensive techniques like adversarial training. Other efforts generate synthetic facts to enrich data [3, 36, 45, 63], such as KG-Mixup [39], but they fail to capture the temporal patterns of TKGs and do not address the model preferences.

In summary, while data augmentation has been explored for both dynamic and static graphs, existing approaches are either inefficient for long temporal sequences or lack the capacity to handle complex semantics in TKGs—highlighting the need for augmentation techniques specifically tailored to TKGs.

2.3 Spatial-temporal Graphs

A closely related area to TKGs is spatial-temporal graphs. They are graphs with evolutionary attributes, and many methods have been proposed to model their spatial-temporal dependencies for prediction and reasoning [11, 16, 18]. For instance, TiTConv [67] enhances sequence uniformity via temporal graph contrastive learning, RESTC [47] aligns spatial and temporal representations at the sequence level, and TF-GCL [42] improves robustness through augmented dynamic graph views. Conda [46] further adopts diffusion models to generate augmented neighborhood representations. However, they generally lack semantic annotations, thus less suitable for the rich relations in TKGs. Some studies explore spatial-temporal knowledge graphs [4, 5, 28]. SSTKG [61] uses a three-step embedding method for spatial recommendation. STKG-PLM [4] prompts the pre-trained language model to enhance the next point-of-interest recommendation. Yet, they mainly capture geographical correlations rather than conceptual entity interactions in TKGs.

3 PRELIMINARY STUDY

3.1 Temporal Knowledge Graph

A temporal knowledge graph is denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{F})$. $\mathcal{E}$ and $\mathcal{R}$ are entity set and relation set. $\mathcal{T}$ is the set of observed timestamps and $\mathcal{F}$ is the set of facts. Each tuple $(s, r, o, t) \in \mathcal{F}$ connects the subject and object entities $s, o \in \mathcal{E}$ via a relation $r \in \mathcal{R}$ in timestamp $t \in \mathcal{T}$, which means a unit knowledge (i.e., a fact).

3.2 Temporal Knowledge Graph Completion

Given an incomplete fact $(s, r, ?, t)$, the temporal knowledge graph completion task identifies the most likely object entity o_c from the candidate set $\mathcal{E}$. Each candidate fact (s, r, o_c, t) is ranked by confidence score, and the highest-ranking candidate is chosen as the new fact. The rank of the true object entity, denoted as $rank(s, r, o, t)$, is the basic metric of this task (lower is better). It indicates the position of the correct object entity o among all candidate entities $o_c \in \mathcal{E}$. Building on this, mean reciprocal rank (MRR) is calculated as the average reciprocal rank across all facts, defined as $MRR = \frac{1}{|Test|} \sum_{(s,r,o,t) \in Test} \frac{1}{rank(s,r,o,t)}$. A higher MRR value indicates better model performance.

We use this task to empirically examine the limitations of existing models, as it underpins many knowledge-enhanced applications, such as recommendation, question answering, and multi-hop reasoning, all of which can be reformulated as temporal knowledge graph completion. For example, recommendation can be cast as predicting the missing item in a quadruple like $(user, buy, ?, timestamp)$, while multi-hop reasoning involves a sequence of such predictions. Our experiments thus directly highlight how issues such as data imbalance and model preferences affect real-world performance.

3.3 Imbalanced Data Distribution

We evaluate three representative TKGC methods (i.e., DE [15], TNT [25], and TEMP [52]) on the widely used ICEWS14 dataset [2]. Our findings reveal two major limitations of existing methods: unstable training and imbalanced performance, both of which stem from the inherent data imbalance in TKGs.

3.3.1 Unstable Training. As shown in Figure 2(a), existing methods exhibit unstable training in two key aspects:

- Unstable across samples: The training effect varies significantly between samples. For example, in the TNT model, the *rank* metric of sample 1 increases over time, suggesting that training may harm its performance. Similarly, the *rank* metric of sample 2 in the DE model fluctuates around a high value, indicating limited optimization.
- Unstable across runs: The performance exhibits significant variability across different training runs. For example, in the DE model, sample 1 shows an increasing variance in *rank* over four runs, indicating that the training effect becomes more inconsistent as training progresses.

To investigate the cause of unstable training, we group facts based on the fluctuation range of the *rank* metric across four independent runs and compute the average degree of entities in each group. As shown in Figure 2(b), lower average degrees correlate with larger fluctuations and higher average *rank*, indicating that instability is more common for entities with sparse local structures. This is because existing methods typically treat observed facts as positives and all others as negatives, leading to numerous false negatives for sparsely connected entities due to missing but valid facts, which misguides the training process [69].

3.3.2 Imbalanced Performance. As shown in Figure 2(c), existing methods suffer from imbalanced performance in two key aspects:

- MRR fluctuates greatly across timestamps, with large gaps even between adjacent ones and notably poor results at certain timestamps.
- As training progresses, the standard deviation of MRR across timestamps increases, indicating that this imbalance tends to worsen over training.

To understand the cause of performance imbalance, we analyze the average degree and MRR at each timestamp (Figure 2(d)). Their fluctuation trends are synchronized—peaks and troughs align, with some displacement and scaling. This suggests that uneven data distribution over time contributes to the performance imbalance observed in existing TKGC methods.

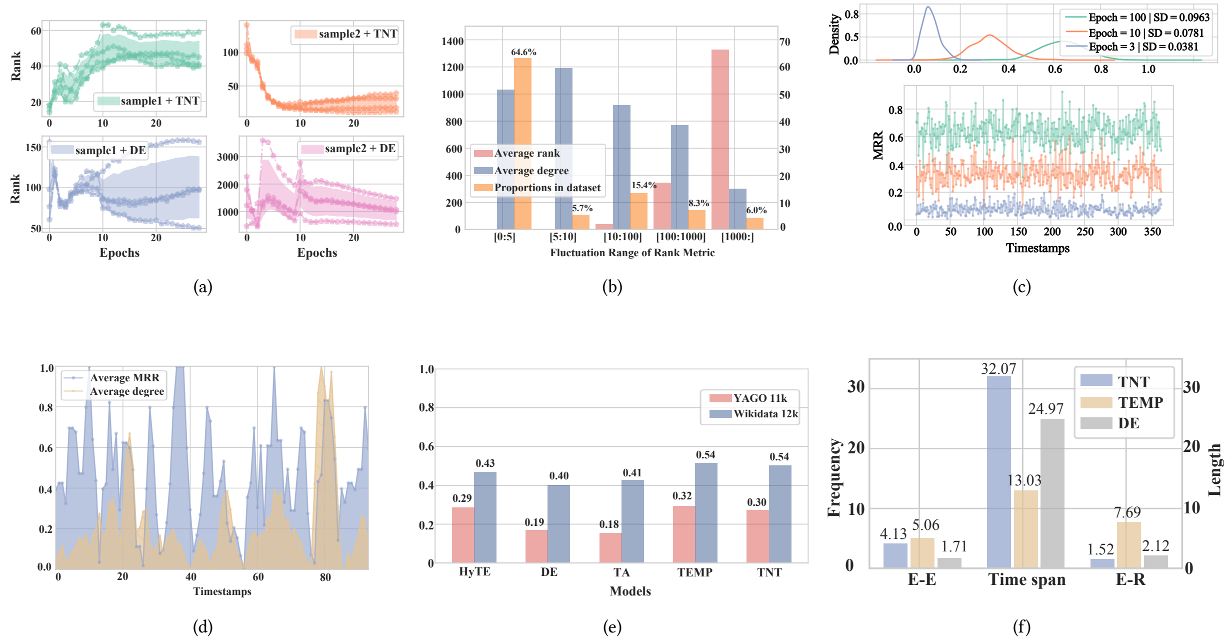


Figure 2: (a) Evolution of the *rank* metric during training, repeated independently four times. The shaded area indicates the fluctuation range across runs. (b) Average degree of samples grouped by *rank* fluctuation range. (c) TEMP's MRR over timestamps. The top plot shows the density distribution of MRR across epochs. SD denotes standard deviation. (d) MRR and average entity degree at different timestamps. (e) Proportion of positive samples among the top-10 candidates for various models. (f) Statistical characteristics of top-ranked entities across models.

3.4 Model Preference

Some studies tackle data imbalance via self-training, selecting high-scoring unlabeled samples as pseudo-positives for training [31]. However, does this approach work for TKGC models? As shown in Figure 2(e), top-ranked samples from existing TKGC models are often inaccurate, e.g., only 30% of the top-10 predictions by TNT are truly positive, suggesting self-training may introduce significant noise. While reducing the number of pseudo-positives can alleviate this to some extent, model preference remains a core challenge for self-training in TKGC, yet it has been overlooked in prior work.

We analyze the data characteristics of top-ranked entities identified by different models. Three key properties are considered:

- Entity-entity interaction frequency (E-E): The number of times a candidate entity interacts with other entities.
- Time span: The time gap between the test timestamp and the candidate entity's most recent activity.
- Entity-relation interaction frequency (E-R): The frequency of interaction between the candidate entity and the relation specified in the query.

In Figure 2(f), we observe that top-ranked entities vary significantly across models due to their different preferences. TNT favors frequently interacted entities, while TEMP prefers recently active ones or those linked to the query relation. This highlights a common issue: model preference toward entities with specific properties. Although facts in TKGs follow diverse patterns [54], such preference hinders the models from effectively learning facts that diverge from their preferred patterns. The problem worsens during self-training, where the model repeatedly selects similarly patterned samples, further reducing generalization to other patterns.

4 METHOD

4.1 Overall Architecture

The above discussions emphasize the urgent need for a data augmentation strategy tailored to TKGs, enriching imbalanced data, and alleviating misleading from false negatives and model preferences. Therefore, we propose *Booster*, a plug-and-play framework that generates pattern-aware samples to enhance TKG structure and alleviate misleading supervision via a two-stage training process.

As shown in Figure 3, *Booster* takes a TKG $\mathcal{G}$ and an untrained TKGC model $M^{(0)}$ as input. In the first stage, it uses a frequency-based filtering strategy to filter out potential false negatives, and then pre-trains $M^{(0)}$ on the remaining data. This protects the model from noisy supervision and helps identify preference-deviated samples (i.e., positive samples that the model struggles to learn).

In the second stage, a hierarchical scoring algorithm is used to further separate potential false negatives into real false negatives and hard negatives. The pre-trained model $M^{(1)}$ is then fine-tuned on these identified samples, along with preference-deviated facts, to produce the final model $M^{(2)}$. This not only enriches sparse data but also targets model-specific weaknesses, reducing the impact of distribution imbalance and inherent model preferences.

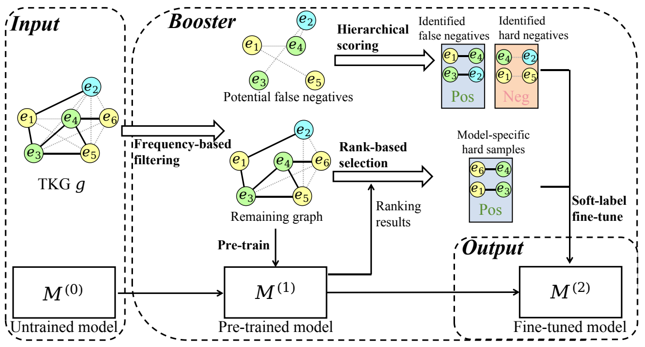


Figure 3: The conceptual illustration of *Booster*, where black solid lines represent observed facts in the TKG, and gray dashed lines denote non-existent ones. Time annotations and relations are omitted for clarity.

4.2 Frequency-based Filtering

Compared with general graph data, filtering false negatives in TKGs poses unique challenges: 1) TKGs consist of entities, relations, and timestamps, whose co-occurrence patterns span both intra- and inter-component interactions, requiring a holistic consideration in the filtering process; 2) Time annotations introduce long historical sequences and multiple temporal edges between nodes, demanding attention to their temporal distribution. To this end, we propose a frequency-based filtering method that adopts component-specific strategies to effectively identify potential false negatives.

Relation-based filtering. Relations in TKGs have significant co-occurring patterns [70]. For example, "economic sanctions" and "export restriction" are a pair of relations that often co-occur between two hostile countries. After the relation "transfer to" occurs between a player and a football club, the relation "play for" will subsequently occur between them. This inspires us that edges missing in a TKG but fitting these relation patterns are likely to be false negatives, and we can detect them by identifying these patterns. Therefore, for each relation r we construct its co-occurred relation set as $R(r) = \{r_i | (s_j, r, o_j, t_j) \in \mathcal{G}, (s_j, r_i, o_j, t'_j) \in \mathcal{G}, |t_j - t'_j| < L_r\}$. Inspired by the temporal locality of facts [21, 26] where co-occurring events within short timespans are more likely to be related, we set the hyper-parameter L_r to a small value to filter highly entangled relation pairs. We provide the analysis of L_r in Section 5. Then, for each observed fact $(s, r, o, t) \in \mathcal{G}$ we can filter its corresponding potential false negatives as $(s, r_i, o, t) \notin \mathcal{G}$ where $r_i \in R(r)$. We further refine the filtering by considering the inter-component patterns and pattern frequencies. First, entities have preferences to interact with a specific set of relations (e.g., athletes[1] are more likely to have relation "play for"), so recognizing these entity-relation interaction patterns helps exclude unrealistic combinations. Second, the higher frequency of relations within $R(r)$ indicates a more important pattern. By retaining only the top-m most frequently co-occurring relations in $R(r)$, we can filter out low-confidence patterns, reduce potential false negatives, and thus lower the time required for scoring in the next step. Therefore, we filter out the relation-based false negatives for each fact $(s, r, o, t) \in \mathcal{G}$ as:

$$RN_{(s,r,o,t)} = \{(s, r', o, t) | r' \in \widetilde{R}(r) \cap \widetilde{R}(s), (s, r', o, t) \notin \mathcal{G}\}, \quad (1)$$

where $\widetilde{R}(r)$ is the subset of $R(r)$ which only preserves top-m most frequent relations. Similarly, $\widetilde{R}(s)$ is the frequency filtered subset of $R(s) = \{r_k | (s, r_k, o_k, t_k) \in \mathcal{G}\}$, which preserves top-$m$ frequently interacted relations of entity s. For example, given an observed fact $(Putin, Consult, China, 2024/05/16)$, both relations "Visit" and "Attack" are in $\widetilde{R}(Consult)$ and tend to co-occur with relation "Consult", but since entity "Putin" is more likely to interact with entities through relation "Visit" (i.e., "Visit" in $\widetilde{R}(Putin)$), $(Putin, Visit, China, 2024/05/16)$ is more likely to be valid.

Entity-based filtering. While the above strategy filters false negatives based on relation semantics, the connectivity among entities also provides insights into the occurrence of facts. Entities often have preferences to interact with a specific set of entities (e.g., "Israel" and "Houthis in Yemen" frequently interact due to ongoing conflict), inspiring us that facts fitting entity co-occurring patterns but missing in $\mathcal{G}$ are likely to be false negatives. Therefore, for each entity e we construct its co-occurring entity set as $N(e) = \{e_i | (e, r_i, e_i, t_i) \in \mathcal{G}\}$, and filter the corresponding top-m frequency entities as $\widetilde{N}(e)$. Subsequently, we filter out the entity-based false negatives for each fact $(s, r, o, t) \in \mathcal{G}$ as:

$$EN_{(s,r,o,t)} = \{(s, r, o', t) | o' \in \widetilde{N}(s), r \in \widetilde{R}(o'), (s, r, o', t) \notin \mathcal{G}\}, \quad (2)$$

indicating entity pairs that are likely to connect through the relation r but are missing in TKG. For example, given an observed fact $(Israel, Attack, Lebanon, 2024/09/30)$, both entity "Netanyahu" and "Hizbullah" are in $\widetilde{N}(Israel)$ and likely to connect with "Israel". However, "Hizbullah" is more likely to interact with relation "attacks" (i.e., "attacks" in $\widetilde{R}(Hizbullah)$), and thus $(Israel, Attack, Hizbullah, 2024/09/30)$ is more likely to be a missing fact.

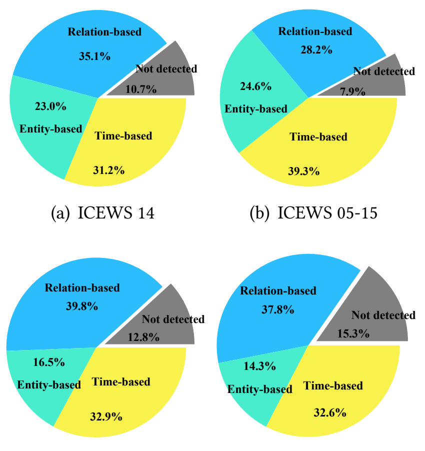
Time-based filtering. Some facts may be repeated many times over a short period, such as $(RegionA, Attack, RegionB)$. Due to the limitation of the update frequency of TKGs, repeated facts may be missing in some timestamps. We can detect false negatives by finding the omissive timestamp within the time interval where the fact repeats. Specifically, we filter out the time-based false negatives for each fact $(s, r, o, t) \in \mathcal{G}$ as:

$$TN_{(s,r,o,t)} = \{(s, r, o, t') | t' \in [x, t], (s, r, o, x) \in \mathcal{G}, (s, r, o, t') \notin \mathcal{G}\}. \quad (3)$$

Notably, we restrict that $t - x < L_t$ to ensure only focus on short-period repetitions, where $t - x$ indicates the fact repetition period and L_t is a hyper-parameter with a small value.

To evaluate the effectiveness of our filtering strategies, we randomly remove 20% of the facts from $\mathcal{G}$ and measure how many can be recovered. As shown in Figures 4(a) and (b), over 90% of the removed facts are successfully detected. Figures 4(c) and (d) further show strong performance on long-tail entities (i.e., $N(s) \leq 5$), with up to 85% detection coverage, demonstrating the robustness of our strategy under long-tail distributions. This resilience stems from the complementary nature of our three strategies: when one is less effective due to data sparsity, others can compensate. For instance, the fact $(Putin, Consult, China, 2024/05/16)$ may be identified via relation-based co-occurrence with $(Putin, Visit, China, 2024/05/16)$

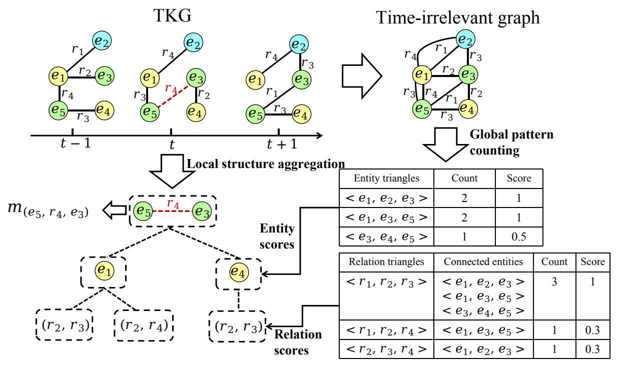
[1]Mentions of types such as "athlete" and "football club" are illustrative only, intended to clarify the intuition behind our filtering strategy, not features used by our method.



(a) ICEWS 14 (b) ICEWS 05-15

(c) YAGO 11k (d) Wikidata 12k

Figure 4: The proportion of false negative samples detected by the frequency-based filtering strategy.



Entity triangles	Count	Score
$< e_1, e_2, e_3 >$	2	1
$< e_1, e_3, e_5 >$	2	1
$< e_3, e_4, e_5 >$	1	0.5

Relation triangles	Connected entities	Count	Score
$< r_1, r_2, r_3 >$	$< e_1, e_2, e_3 >$ $< e_1, e_3, e_5 >$ $< e_3, e_4, e_5 >$	3	1
$< r_1, r_2, r_4 >$	$< e_1, e_3, e_5 >$	1	0.3
$< r_2, r_3, r_4 >$	$< e_1, e_2, e_3 >$	1	0.3

Figure 5: An example of the hierarchical scoring algorithm, where the red dashed line denotes the potential false negative fact that needs identification.

or through repeated facts such as $(Putin, Consult, China, 2024/05/15)$ and $(Putin, Consult, China, 2024/05/17)$. Since facts involving entities with rich local structures tend to yield fewer false negatives [30, 72], and our strategies show robustness to long-tail distributions, we apply filtering only to entities with sparse neighborhoods (i.e., $N(s) \le k$ or $N(o) \le k$) to reduce computational cost.

Filtering out potential false negatives prevents the model from being misled and enriches the structure by further identifying real false negatives. Moreover, since these strategies consider intrinsic patterns of TKG, the filtered samples provide fine-grained information that improves the model's performance during fine-tuning.

4.3 Hierarchical Scoring Algorithm

Identifying real false negatives in TKGs is challenging for two main reasons: (1) TKGs exhibit complex semantic and temporal patterns due to diverse relations, as well as evolving topologies, which must be properly captured for accurate recognition. (2) TKGs are often incomplete and noisy, making misidentification inevitable. Therefore, estimating the confidence of each fact is essential to avoid misleading the model with low-confidence false negatives.

In this part, we propose a novel hierarchical scoring algorithm to estimate the likelihood that potential false negatives are indeed mislabeled. To capture both global patterns and local structures in TKGs while mitigating skewed data distribution, the scoring process is divided into two stages: global pattern counting and local structure aggregation. As illustrated in Figure 5, the global stage computes scores by counting triangles in the time-irrelevant graph, while the local stage refines these scores based on each sample's temporal neighborhoods. A hierarchical design is proposed to control the complexity explosion of entity-relation combinations. To further enhance robustness, we introduce a perturbation-based technique to assess score stability and reduce noise sensitivity.

Global pattern counting. We represent semantic patterns in TKGs as triangle closures involving entities and relations. By identifying frequent triangles and estimating their intensity (i.e., the likelihood of a third edge forming given two), we capture and quantify these patterns. However, direct triangle counting in TKGs yields an overwhelming number of distinct triangles due to diverse entities, relations, and temporal orders, resulting in high computational cost and sparse, uninformative counts. To overcome this, we decompose entities and relations within triangles to reduce complexity and construct a time-irrelevant graph to mitigate temporal noise.

The time-irrelevant graph contains all the entity-relation-entity combinations observed in TKG, represented as $\mathcal{G}' = \{(s,r,o) | \{(s,r,o,t) \in \mathcal{G}\}$. By abstracting away from the non-uniform temporal distribution of facts and alleviating the burden of long historical sequences, this approach effectively captures the preferences among entities and relations while enabling efficient computation. Although temporal information is not directly considered here, the global pattern counting focuses on time-invariant conceptual patterns, while temporal dynamics are fully addressed through the local structure aggregation process. To further alleviate the complexity explosion brought by the combination of entities and relations, we first anonymize the relations in $\mathcal{G}'$ to count entity triangles. We define the number of edges between entities as $N(e_1, e_2) = |\{(e_1, r, e_2) | (e_1, r, e_2) \in \mathcal{G}'\}|$ where $|\cdot|$ means the size of the set. The number of triangles among entities can be defined as:

$$C_e(e_1, e_2, e_3) = min(N(e_1, e_2), N(e_2, e_3), N(e_1, e_3)), \tag{4}$$

which counts for the number of edges existing among three entities. Since the anonymization of relations highlights the connectivity of the graph, C_e can accurately reflect the connection preference among entities, e.g., China and Japan have more connections with South Korea than the Vatican. Finally, we obtain the entity score by normalization as:

$$S_e(e_1, e_2, e_3) = \frac{C_e(e_1, e_2, e_3) - min(C_e) + 1}{max(C_e) - min(C_e) + 1}, \tag{5}$$

where $max(C_e)$ and $min(C_e)$ are respectively the maximum number and minimum number of entity triangles in $\mathcal{G}'$.

The relation triangles indicate the interaction rules among entities (e.g., the combination of relations "launching an attack", "call for support", and "impose sanctions" describes the hostile behavior among three countries), and thus are important to identify the missing valid facts. This motivates us to anonymize the entities in $\mathcal{G}'$ to find relation triangles. Specifically, a relation triangle consists of three relations that connect the entities within an entity triangle. We define the count of each relation triangle as how many different entity triangles are connected by it, which is formally defined as:

$$C_r(r_1, r_2, r_3) = |\{(e_i, e_j, e_k) | (e_i, r_1, e_j), (e_j, r_2, e_k), (e_i, r_3, e_k) \in \mathcal{G}'\}|. \tag{6}$$

The same normalization is then employed on C_r to obtain the relation score S_r. A high entity score suggests that three entities are more likely to form a triangle, while a high relation score indicates a stronger connection among the entities through three specific relations. This effectively quantifies the patterns within TKG.

Local structure aggregation. The validity of a fact depends not only on global semantic patterns but also on its temporal dynamics. While global scores capture overall conceptual patterns in TKGs, we propose aggregating them based on the fact's local temporal context to consider its evolving nature. As shown in Figure 5, we reformulate the local structure of each potential false negative fact (s, r, o, t) into an entity layer and a relation layer, aligning with the design of global scores. The entity layer contains all entities that have interactions with both s and o within the time window L_e, denoted as $l_e(s, o, t)$. Therefore, each entity e in the entity layer can form a triangle with s and o, allowing us to estimate the validity of (s, r, o, t) based on intensities of entity triangles (i.e., the likelihood that $s - o$ will exist when $s - e$ and $o - e$ are present). To integrate relation scores, for each entity $e \in l_e(s, o, t)$, we construct its corresponding relation layer $l_r(s, o, t, e)$. Each item in relation layer is represented as (r_i, r_j), where r_i represents relations existing between s and e, and r_j represents those existing between o and e, both within L_e. Therefore, each item (r_i, r_j) in the relation layer can form a triangle with r, allowing us to estimate the validity of (s, r, o, t) based on intensities of relations triangles (i.e., the probability that s and o are connected by r given that s and o have connected with e through r_i and r_j). Afterward, we aggregate the relation layer as:

$$m_{e_i} = \sum_{(r_i, r_j) \in l_r(s,o,t,e_i)} \alpha_{i,j} \cdot S_r(r_i, r_j, r), \tag{7}$$

where $e_i \in E(s, o, t)$ represents each node in the entity layer of (s, r, o, t). We use α as a time-aware weight to emphasize facts that occurred more recently, which is defined as:

$$\alpha_{i,j} = softmax(-|t_i - t_j|), \tag{8}$$

where t_i and t_j are respectively the occurring time of facts (s, r_i, e) and (e, r_j, o). We then aggregate the entity layer as:

$$m_{(s,r,o,t)} = \sum_{e_i \in E(s,o,t)} m_{e_i} \cdot (1 + S_e(s, o, e_i)), \tag{9}$$

where $m_{(s,r,o,t)}$ is the confidence score of (s, r, o, t).

Score perturbation. Each potential false negative fact originates from an observed fact (see Section 4.2), sharing a similar local structure. Therefore, we can achieve the adaptive threshold by comparing their scores. A potential false negative f' will be identified as a real false negative if $m_{f'} > m_f$ where f is its corresponding observed fact. To address the noise introduced by the randomness of temporal dynamics, we extend our algorithm with the perturbation technique and smooth labels. For each potential false negative f', we first slightly perturb its corresponding temporal local structure (e.g., randomly repeat or remove items within layers or perturb the time-aware weights), and then calculate a set of scores $M_{f'} = \{m^1_{f'}, m^2_{f'}, ..., m^k_{f'}\}$ based on these perturbed structures. If $mean(M_{f'}) > m_f$, we will set $mean(M_{f'})$ as a smooth label for f', which emphasizes the samples with more robust patterns. The remaining facts are regarded as hard negative samples.

While data-driven methods such as HyTE [9] and DE [15] can identify false negatives and better fit data-specific properties, they have notable limitations. First, their black-box nature leads to less reliable predictions, while our method offers human-interpretable hierarchical scoring. Second, their reliance on training reduces efficiency, whereas our method is training-free. Third, they often struggle to capture diverse knowledge patterns. In contrast, *Booster* addresses this issue with built-in inductive biases through frequency-based filtering and hierarchical scoring. Although some data-driven methods also consider patterns [6, 59], they typically involve complex architectures that demand extensive training time.

4.4 Two-Stage Training

In this part, we propose a two-stage training approach to shield the model from imbalanced data and alleviate the model preferences.

Pre-training. Our filtering strategies have identified potential false negatives. To avoid misleading, we exclude them from negative samples during contrastive training. Formally, given an untrained TKGC model $M^{(0)}$, we pre-train it as

$$L_p = \sum_{(s,r,o,t) \in \mathcal{G}} -log(\frac{exp(p(s, r, o, t))}{\sum_{Neg(s,r,o,t)} exp(p(s, r, e, t))}) + \lambda, \tag{10}$$

where $Neg(s, r, o, t) = \{(s, r, e, t) | e \in \mathcal{E}, (s, r, e, t) \notin EN_{(s,r,o,t)} \cup RN_{(s,r,o,t)} \cup TN_{(s,r,o,t)}\}$ is filtered negative facts specific to $((s, r, o, t))$. λ is a regularization term and $p(\cdot)$ is the prediction score obtained by model $M^{(0)}$. The pre-trained model is denoted as $M^{(1)}$.

Since the pre-training process avoids the misleading effects of false negatives, positive samples that are ranked lower than the corresponding negative samples reflect facts that the model struggles to learn, and thus indicate its implicit preferences. Formally, for each positive sample $(s, r, o, t) \in \mathcal{G}$ with negatives $(s, r, e, t) \in Neg(s, r, o, t)$, we include (s, r, o, t) in the model-specific hard sample set $\mathcal{F}_m$ if it is not ranked above all (s, r, e, t) by $M^{(1)}$. We emphasize $\mathcal{F}_m$ during fine-tuning to correct for such preferences.

Fine-tuning. We further fine-tune the pre-trained model $M^{(1)}$ using a curated set of samples, including real false negatives, hard negatives, and model-specific hard samples, to obtain the final model $M^{(2)}$. This serves three key purposes: 1) Real false negatives are re-labeled as positives, enriching data for entities and timestamps in sparse regions. This mitigates performance imbalance from uneven data distribution. Additionally, smooth labels generated via the perturbation strategy highlight robust and stable patterns, offering finer-grained supervision for TKGC models. 2) Hard negatives closely resemble positives in structure but are invalid. Training on them improves the model's ability to distinguish subtle differences. 3) Model-specific hard samples capture patterns that the model struggles with. Fine-tuning on these forces the model to adapt, improving generalization across diverse TKG patterns. We fine-tune the pre-trained model $M^{(1)}$ as

$$L_f = \sum_{f \in \mathcal{G}^p} -l_f log\sigma(p(f)) + \sum_{f' \in \mathcal{G}^n} -log\sigma(p(f')), \qquad (11)$$

where $\mathcal{G}^p$ contains real false negatives and model-specific hard samples. $\mathcal{G}^n$ contains hard negative samples. $\sigma(\cdot)$ is the sigmoid function and $l_f = mean(M_f)$ is the smooth label of fact f.

4.5 Complexity Analysis

The additional time complexity introduced by *Booster* primarily stems from two components: the filtering strategy and the hierarchical scoring mechanism, both of which are applied to each observed fact in TKG. Suppose the TKG contains $|E|$ entities (nodes), $|R|$ relations (types of edges), $|F|$ edges and $|\mathcal{T}|$ timestamps. For the filtering phase, *Booster* applies three types of filters for each fact (s, r, o, t). The relation-based filter inspects the set of relations that co-occur with relation r, which has a worst-case size of $\Delta_R \ll |R|$, the maximum relation co-occurrence degree. The entity-based filter considers all entities that have interacted with s or o, whose neighborhoods can be bounded by the maximum entity degree $\Delta_E \ll |E|$. The time-based filter scans a fixed temporal window of length $L_t \ll |\mathcal{T}|$. Thus, the per-fact cost of filtering is $O(\Delta_R + \Delta_E + |L_t|)$.

In the hierarchical scoring phase, *Booster* first computes global scores for all entity and relation triangles. Both of them need to first traverse all the edges in the time-irrelevant graph, which has a size of $|F'| \ll |F|$. The entity counting finds the co-neighborhoods of s or o for each edge, bounded by $\Delta_E \ll |E|$. The relation counting traverses the combinations of the interacted relations of s and o, bounded by ${\Delta_E}^2$. Therefore, the overall complexity of global scores calculation is $O(|F'|{\Delta_E}^2)$. Subsequently, local structure aggregation is performed for each fact. For each fact, this process involves identifying shared neighboring entities of s and o, bounded by Δ_E, and aggregating over the relations connecting s and o to those neighbors, with up to Δ_R relations per edge. This results in a per-fact complexity of $O(\Delta_E \Delta_R^2)$. Δ_E and Δ_R are mostly very small in real-world TKGs, making the complexity of *Booster* acceptable.

5 EXPERIMENTS

We conduct experiments on 5 datasets to answer the following research questions: **RQ1**: Can *Booster* improve the performance of existing models? **RQ2**: How does each component of *Booster* contribute to performance improvement? **RQ3**: Is *Booster* efficient? **RQ4**: Can *Booster* improves the balance and stabilization of the performance?

Datasets. We evaluate *Booster* on five benchmark datasets drawn from ICEWS [2], YAGO [40], Wikidata [12], and GDELT [27]. ICEWS captures time-stamped interactions between political entities; we use two subsets: ICEWS 14 (events in 2014) and ICEWS 05-15 (events from 2005 to 2015). YAGO is a commonsense knowledge base, and YAGO 11k is a subset focusing on the top-10 most frequent time-sensitive relations. Wikidata, a collaboratively curated knowledge base derived from Wikipedia, is represented by the Wikidata 12k subset. GDELT, a large-scale political knowledge base, is also included. Each dataset is split into training, validation, and test sets with an 8:1:1 ratio. Detailed statistics are provided in Table 1.

Table 1: Statistics of datasets.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{F}\rvert$
ICEWS 14	7,128	230	365	90,730
ICEWS 05-15	10,488	251	4,017	461,329
YAGO 11k	10,623	10	2,801	20,507
Wikidata 12k	12,554	24	2,270	40,621
GDELT	500	20	366	3,419,607

Baseline models. As there are no established data augmentation baselines specifically for TKGs, we select representative methods from two closely related areas. From temporal graph augmentation, we include MeTA [51], TagRec [38], and TTDA [32], which capture the dynamic aspects of TKGs. From knowledge graph augmentation, we consider DeMix [7], NSCaching [69], and KG-Mixup [39], which focus on semantic information. These two categories provide complementary baselines for comparison. To evaluate their effectiveness, we use 5 widely adopted TKGC models as backbones: HyTE [9], TA [13], DE [15], TNT [25], and TEMP [52].

Implementation details. We adopt the official implementations of existing TKGC models as backbones. For each model, we perform grid search to tune hyper-parameters, selecting the best settings based on validation MRR. Training is conducted over 1000 epochs with 100 mini-batches per epoch. Models are pre-trained for the first 20 epochs, and then fine-tuned with early stopping. The learning rate is set to 0.001. For all models, we set the representation dimension d as 200, the size of negative sampling as 50, and the data pre-processing is unified as in TNT [25] to achieve fair comparison. The time windows L_r, L_e, and L_t are selected from $\{1, 3, 5, 10, 20\}$. We use Adagrad [34] for optimization and all experiments are conducted on a 64-bit machine with Nvidia TITAN RTX. Besides MRR, we also use Hits@k as the metric which is defined as $Hits@k = \frac{1}{|Test|} \sum_{(s,r,o,t) \in Test} ind(rank(s, r, o, t) \leq k)$, where $ind()$ is 1 if the inequality holds and 0 otherwise. Our source code is available at https://github.com/zjs123/Booster.

5.1 Overall Evaluation (RQ1)

Accuracy. Table 2 reports the performance of existing TKGC models under various data augmentation strategies. We observe that: (1) *Booster* consistently improves all backbone models, with an average gain of 4.5% and a maximum of 8.7%, both statistically significant. Notably, HyTE sees the largest boost (8.7% MRR on ICEWS 14 and 7.9% on ICEWS 05-15), likely because its timestamp-specific representation learning makes it more sensitive to data sparsity. *Booster* can enrich the sparse structure by identifying false negatives and thus achieve improvement. (2) Compared with prior graph-based augmentation methods, *Booster* achieves superior improvements across most models, outperforming strong baselines like KG-Mixup and NSCaching. Interestingly, some methods (e.g., MeTA, NSCaching) may even hurt performance due to their limited consideration of temporal semantics, introducing noise during sample generation. (3) *Booster* brings consistent gains across all datasets, confirming its adaptability to heterogeneous temporal knowledge. The gains vary with dataset sparsity: average improvements are 3.9% on YAGO 11k and 2.4% on Wikidata 12k. Figure 6(a) further shows that *Booster* remains effective on large-scale graphs.

Hyper-parameter sensitivity. In Figures 6(b) and (c), Ω^4, Δ^2, *sin*, *relu*, and temporal features are hyper-parameters of TNT and DE. We can see that *Booster* can achieve improvement with different hyper-parameters of the original model. Figure 7(a) shows how the

Table 2: Performance comparison of baseline models. The best results are boldfaced and "DA" means data augmentation.

	Dataset	ICEWS 14			ICEWS 05-15			YAGO 11k			Wikidata 12k		
TKGC models	**DA strategies**	*MRR*	*Hits@1*	*Hits@3*	*MRR*	*Hits@1*	*Hits@3*	*MRR*	*Hits@1*	*Hits@3*	*MRR*	*Hits@1*	*Hits@3*
HyTE	Without DA	0.297	0.108	0.416	0.316	0.116	0.445	0.134	0.032	0.181	0.191	0.107	0.208
	MeTA	0.293	0.105	0.410	0.319	0.117	0.449	0.132	0.031	0.178	0.186	0.105	0.201
	TagRec	0.295	0.110	0.402	0.320	0.121	0.441	0.130	0.029	0.180	0.190	0.105	0.213
	TTDA	0.306	0.115	0.416	0.325	0.139	0.437	0.128	0.026	0.182	0.194	0.109	0.212
	DeMix	0.301	0.113	0.412	0.323	0.126	0.447	0.136	0.035	0.182	0.193	0.108	0.211
	NSCaching	0.295	0.107	0.414	0.320	0.124	0.445	0.130	0.029	0.179	0.188	0.105	0.206
	KG-Mixup	0.308	0.133	**0.420**	0.325	0.145	0.446	0.136	0.037	0.175	0.192	0.107	0.210
	Booster	**0.323**	**0.176**	0.417	**0.341**	**0.223**	**0.450**	**0.142**	**0.051**	**0.182**	**0.199**	**0.112**	**0.219**
		Improve: 8.7%	P-value: 0.0114		Improve: 7.9%	P-value: 0.0101		Improve: 5.9%	P-value: 0.0225		Improve: 4.1%	P-value: 0.0217	
DE	Without DA	0.501	0.392	0.569	0.484	0.366	0.546	0.119	0.084	0.117	0.212	0.123	0.242
	MeTA	0.496	0.388	0.565	0.486	0.367	0.544	0.112	0.078	0.113	0.209	0.119	0.246
	TagRec	0.493	0.386	0.553	0.485	0.367	0.540	0.118	0.082	0.116	0.211	0.121	0.249
	TTDA	0.506	0.398	0.558	0.493	0.371	0.548	0.116	0.081	0.120	0.208	0.117	0.251
	DeMix	0.508	0.397	0.573	0.491	0.370	0.545	0.116	0.081	0.118	0.214	0.124	0.246
	NSCaching	0.503	0.394	0.570	0.482	0.365	0.541	0.115	0.080	0.116	0.217	0.125	0.248
	KG-Mixup	0.507	0.398	0.565	0.492	0.372	0.545	0.117	0.082	0.115	0.215	0.122	0.244
	Booster	**0.521**	**0.410**	**0.578**	**0.510**	**0.388**	**0.548**	**0.124**	**0.086**	**0.118**	**0.221**	**0.126**	**0.252**
		Improve: 3.9%	P-value: 0.0170		Improve: 3.5%	P-value: 0.0217		Improve: 4.2%	P-value: 0.0233		Improve: 4.2%	P-value: 0.0213	
TA	Without DA	0.409	0.295	0.466	0.492	0.376	0.544	0.110	0.072	0.108	0.188	0.109	0.210
	MeTA	0.405	0.293	0.460	0.493	0.377	0.542	0.112	0.078	0.104	0.186	0.107	0.208
	TagRec	0.412	0.294	0.465	0.505	0.383	0.551	0.110	0.073	0.106	0.192	0.111	0.214
	TTDA	0.407	0.291	0.462	0.498	0.378	0.553	0.105	0.068	0.104	0.189	0.109	0.212
	DeMix	0.411	0.292	0.473	0.498	0.380	0.545	0.106	0.071	0.103	0.184	0.106	0.210
	NSCaching	0.401	0.289	0.460	0.493	0.375	0.550	0.110	0.077	0.112	0.180	0.103	0.211
	KG-Mixup	0.412	0.290	0.475	0.502	0.381	0.544	0.113	0.075	0.110	0.191	0.110	0.210
	Booster	**0.421**	**0.298**	**0.484**	**0.513**	**0.387**	**0.566**	**0.123**	**0.081**	**0.115**	**0.205**	**0.113**	**0.214**
		Improve: 2.9%	P-value: 0.0156		Improve: 4.2%	P-value: 0.0139		Improve: 6.9%	P-value: 0.0218		Improve: 3.5%	P-value: 0.0185	
TEMP	Without DA	0.601	0.478	0.681	0.680	0.553	0.769	0.186	0.126	0.189	0.330	0.227	0.359
	MeTA	0.602	0.479	0.682	0.676	0.548	0.766	0.184	0.125	0.184	0.327	0.225	0.358
	TagRec	0.611	0.482	0.685	0.683	0.555	0.771	0.184	0.126	0.182	0.328	0.227	0.358
	TTDA	0.604	0.480	0.682	0.681	0.553	0.771	0.190	0.127	0.192	0.325	0.221	0.354
	DeMix	0.606	0.479	0.684	0.682	0.555	0.769	0.188	0.124	0.189	0.331	0.229	0.360
	NSCaching	0.598	0.476	0.677	0.678	0.550	0.764	0.180	0.121	0.187	0.325	0.220	0.356
	KG-Mixup	0.603	0.480	0.676	0.684	0.552	0.770	0.186	0.124	0.190	0.332	0.228	0.360
	Booster	**0.623**	**0.485**	**0.690**	**0.697**	**0.559**	**0.779**	**0.194**	**0.130**	**0.195**	**0.340**	**0.237**	**0.362**
		Improve: 3.6%	P-value: 0.0151		Improve: 2.5%	P-value: 0.0219		Improve: 4.3%	P-value: 0.0197		Improve: 3.3%	P-value: 0.0189	
TNT	Without DA	0.614	0.532	0.656	0.658	0.588	0.712	0.185	0.127	0.183	0.331	0.233	0.357
	MeTA	0.608	0.529	0.649	0.650	0.582	0.706	0.183	0.127	0.175	0.332	0.234	0.355
	TagRec	0.617	0.535	0.655	0.662	0.590	0.717	0.183	0.126	0.178	0.333	0.234	0.358
	TTDA	0.612	0.532	0.650	0.660	0.588	0.720	0.189	0.129	0.191	0.328	0.230	0.356
	DeMix	0.615	0.530	0.660	0.661	0.590	0.714	0.188	0.129	0.185	0.331	0.231	0.360
	NSCaching	0.605	0.526	0.649	0.652	0.584	0.709	0.180	0.123	0.176	0.327	0.228	0.355
	KG-Mixup	0.619	0.537	0.661	0.663	0.591	0.710	0.187	0.126	0.188	0.335	0.234	0.359
	Booster	**0.636**	**0.557**	**0.678**	**0.679**	**0.602**	**0.728**	**0.195**	**0.131**	**0.201**	**0.342**	**0.239**	**0.367**
		Improve: 3.5%	P-value: 0.0145		Improve: 3.2%	P-value: 0.0123		Improve: 5.4%	P-value: 0.0119		Improve: 3.3%	P-value: 0.0168	

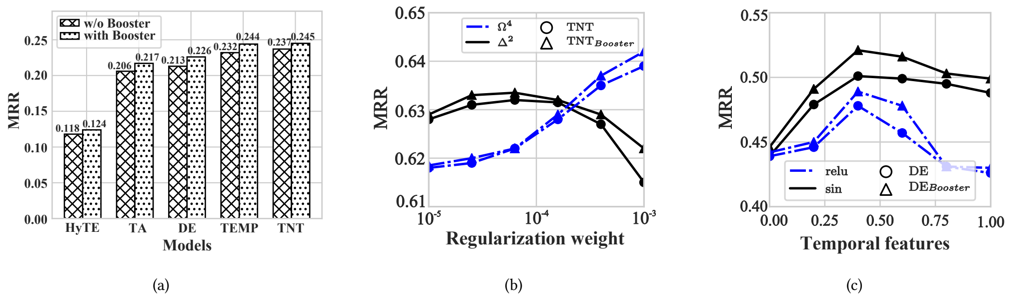


Figure 6: (a) Performance on the GDELT dataset. (b) *Booster* performance with varying hyper-parameters of the TNT model on the ICEWS 14 dataset. (c) *Booster* performance with varying hyper-parameters of the DE model on the ICEWS 14 dataset.

hyper-parameters of *Booster* affect its effectiveness. We can see that when L_e increases, *MRR* keeps increasing at first because the larger L_e helps to achieve more accurate scoring of facts. However, *MRR* drops when L_t and L_r are large. This is because the large L_t and L_r will extensively increase the number of filtered potential false negatives and thus bring more noise. Finally, Figures 7(b) and (c) show how the number of identified real false negatives changes with hyper-parameters. When L_r and Lt are large, the number of identified real false negatives increases, but when $L_r = Lt = 1$, its number gradually decreases when L_e increases, which meets our conjecture that large L_t and L_r will bring more noise.

Comparison with spatial-temporal models. As shown in Table 3, typical spatial-temporal graph and knowledge graph models fall short of *Booster*, mainly because spatial-temporal graph models

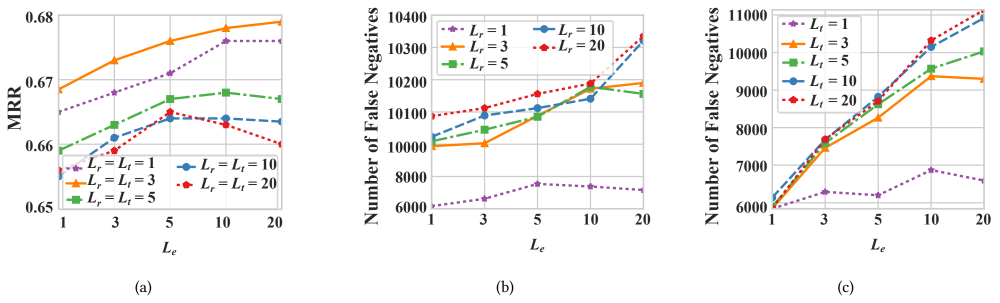


Figure 7: (a) Performance of TNT$_{Booster}$ with varying hyper-parameters on the ICEWS 05-15 dataset. (b) The number of identified false negatives with different L_e and L_r. (c) The number of identified false negatives with different L_e and L_t.

Table 3: Comparison with spatial-temporal models.

Dataset	ICEWS 14		Wikidata 12k	
Models	*Hit@1*	*Hit@10*	*Hit@1*	*Hit@10*
TiTConv	0.408	0.563	0.210	0.412
TF-GCL	0.446	0.584	0.259	0.406
Conda	0.473	0.619	0.255	0.437
SSTKG	0.519	0.693	0.282	0.486
TNT$_{Booster}$	**0.557**	**0.781**	**0.342**	**0.547**

Table 4: Results of ablation study.

Dataset	ICEWS 14		Wikidata 12k	
Variants	*Hit@1*	*Hit@10*	*Hit@1*	*Hit@10*
w/o identified false negatives	0.545	0.774	0.334	0.541
w/o identified hard negatives	0.544	0.772	0.332	0.538
w/o model-specific hard samples	0.544	0.772	0.332	0.538
w/o entity scores	0.548	0.775	0.330	0.539
w/o relation scores	0.543	0.771	0.329	0.537
w/o smooth labels	0.533	0.768	0.331	0.537
w/o smooth labels + entity scores	0.523	0.760	0.324	0.534
w/o smooth labels + relation scores	0.521	0.757	0.322	0.530
TNT$_{Booster}$	**0.557**	**0.781**	**0.342**	**0.547**

like TF-GCL and Conda fail to incorporate the rich semantics of relations, while models like SSTKG lack tailored designs for capturing complex conceptual patterns within TKGs.

5.2 Effect of Each Component (RQ2)

Ablation study. Table 4 presents the ablation results of *Booster*. Incorporating identified false negatives, hard negatives, and model-specific challenging samples during fine-tuning consistently improves performance. Removing the smooth label leads to performance drops, as low-confidence false negatives may mislead the model. Eliminating either entity or relation scores alone causes only minor degradation, highlighting the smooth label's robustness to noise. However, when the smooth label is removed, discarding the scores severely hurts performance, underscoring their importance in accurate identification. Figure 8(a) further illustrates that without the smooth label, performance becomes less stable and generally worse. As shown in Figure 8(b), fine-tuning is essential due to two reasons: (1) Pre-training only captures partial graph structure and lacks full TKG semantics; (2) Fine-tuning samples are enriched by pattern-aware heuristics, enhancing generalization to diverse TKG patterns. Additionally, identified false negatives help densify the sparse graph, benefiting long-tail entities and timestamps.

Table 5: Performance comparison of variants.

Dataset	ICEWS 05-15		Wikidata 12k	
Variants	*Hit@1*	*Hit@10*	*Hit@1*	*Hit@10*
Identifying with DE	0.595	0.814	0.234	**0.548**
Identifying with TEMP	0.596	0.816	0.235	0.545
Identifying with HyTE	0.576	0.778	0.219	0.519
Identifying with TNT	0.592	0.811	0.228	0.537
Self-training	0.582	0.791	0.230	0.539
Neighbor filtering	0.587	0.797	0.230	0.540
Recent active filtering	0.581	0.790	0.228	0.538
TNT$_{Booster}$	**0.602**	**0.823**	**0.239**	0.547

Comparison with variants. Table 5 reports the performance of *Booster* variants. To assess the effectiveness of our hierarchical scoring algorithm, we replace it with several pre-trained sophisticated models for identifying false negatives. However, these alternatives fail to surpass the original framework. As shown in Figures 8(c) and (d), retraining such models is time-consuming, further demonstrating the advantage of our design. Additionally, directly applying self-training to the TNT model leads to significant performance drops, highlighting the importance of addressing model preferences. Lastly, we evaluate our filtering strategy against commonly used ones, such as selecting neighbor entities from sparse local structures or recent active entities. None of them outperforms TNT$_{Booster}$, validating the effectiveness of our approach.

5.3 Efficiency (RQ3)

Comparison with baselines. Figures 8(c) and (d) report the training time of baseline models under the *Booster* framework. For TNT and TEMP on the Wikidata 12k dataset, training time increases only by 1/10 and 1/5, respectively, as they avoid negative sampling, with overhead mainly from filtering and scoring. Although HyTE and DE require negative sampling, the size of hard negatives grows sub-linearly with the TKG size, so the time overhead remains modest even on large datasets like ICEWS 05-15. These results show that *Booster* enhances performance with acceptable additional cost.

Throughput w.r.t. hyper-parameters. Figure 9 shows the throughput of filtering strategies and the hierarchical scoring algorithm with varying hyper-parameters. First, our proposed strategies achieve high throughput to all the hyper-parameters. The average processing time is nearly 0.5 ms per sample. Second, the throughput of our proposed strategies decreases sub-linearly when the span of the time window increases. This is because of the locality of TKGs

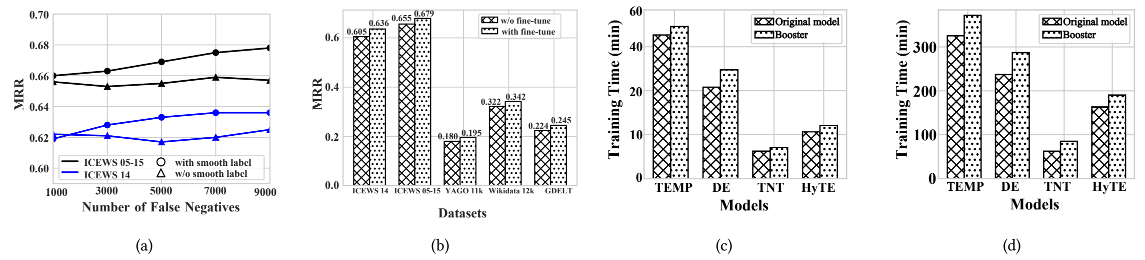


Figure 8: (a) Performance sensitivity to the identified false negatives. (b) TNT performance with and without fine-tuning. (c) Training time on the ICEWS 14 dataset. (d) Training time on the ICEWS 05-15 dataset.

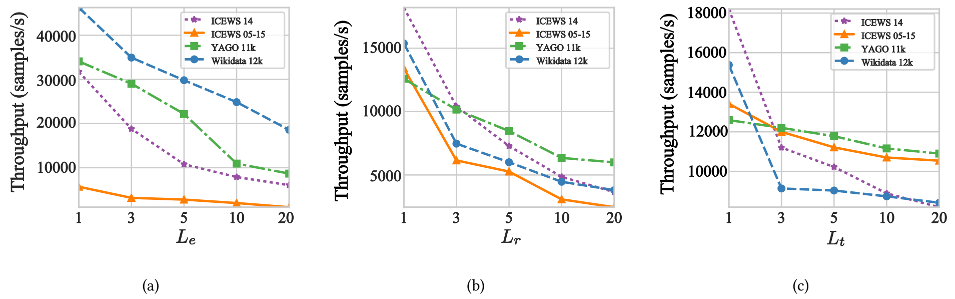


Figure 9: (a) Throughput of the *Booster* framework with varying L_e. (b) Throughput of the *Booster* framework with varying L_r.(c) Throughput of the *Booster* framework with varying L_t.

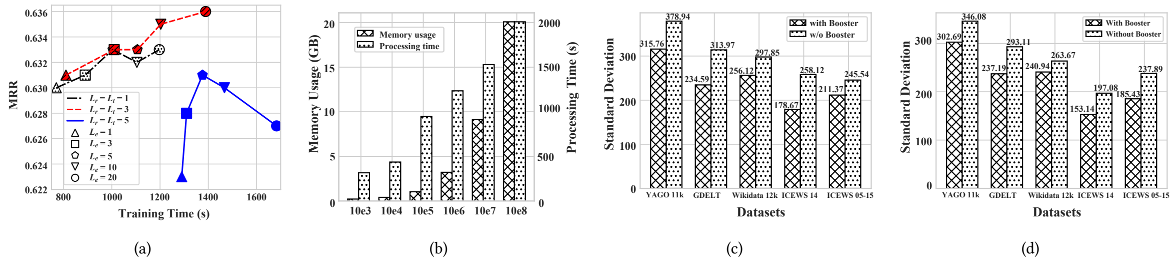


Figure 10: (a) MRR w.r.t. training time on the ICEWS 14 dataset. (b) Scalability of our proposed algorithm. (c) MRR variance on timestamps obtained by the TEMP model. (d) MRR variance on timestamps obtained by the TNT model.

that facts are mostly short-term related. Our strategies meet this property and thus can effectively filter out useless samples.

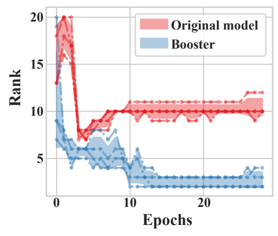
Tradeoff between MRR and training time. We conduct experiments to investigate the tradeoff between *MRR* and training time. Specifically, we set L_r and L_t as 1, 3, and 5. By varying L_e, we report the tradeoff between *MRR* and training time in Figure 10(a). We observe that when L_r and L_t are small, the MRR result increases as the training time increases. When $L_r = L_t = 5$ and L_e reaches 10, the training time keeps increasing but MRR degrades drastically.

Memory, CPU, and GPU usages. We use Psutil[2] to monitor memory and CPU usage, and GPUtil[3] to track GPU utilization. *Booster* consumes a maximum of 14.51 GB memory, with a total CPU utilization of 425% (on a 10-core CPU, where 1000% indicates full usage) and a GPU utilization rate of 34%. As shown in Figure 10(b), both memory usage and processing time grow sub-linearly.

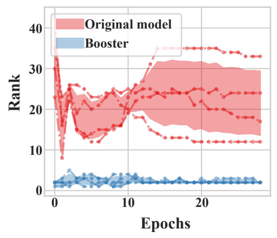
5.4 Balance and Stability (RQ4)

Balance. Table 6 presents the performance gains of *Booster* across test samples with varying degrees of sparsity. We categorize the samples by time sparsity (i.e., the number of facts per timestamp) and entity sparsity (i.e., the number of interacted entities), and then compute the average *MRR* within each group. *Booster* consistently improves performance across all sparsity levels, with more significant gains observed for sparser samples. For instance, on samples

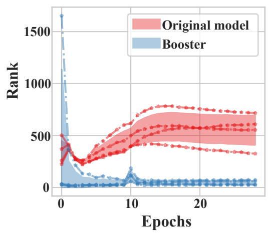
[2]https://pypi.org/project/psutil/
[3]https://pypi.org/project/GPUtil/



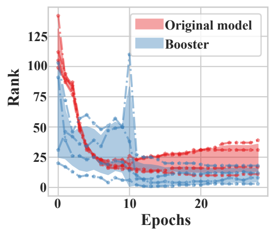
(a)



(b)



(c)

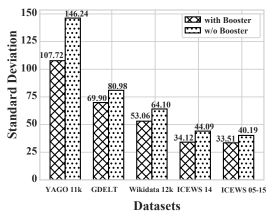


(d)

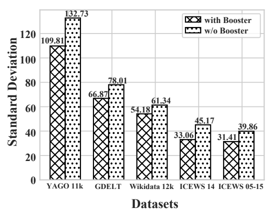
Figure 11: Change of *rank* of four test samples that are hard to be optimized by the TEMP model.

Table 6: Performance comparison on the ICEWS 14 dataset with different sparsities of timestamps and entities.

Models		HyTE	HyTE$_{Booster}$	TA	TA$_{Booster}$	TNT	TNT$_{Booster}$
	Scope	MRR	MRR	MRR	MRR	MRR	MRR
Time	[0:100]	0.255	0.291	0.421	0.447	0.598	0.621
	[100:250]	0.276	0.315	0.388	0.398	0.601	0.602
	[300:]	0.306	0.309	0.387	0.388	0.623	0.640
Entity	[0:10]	0.149	0.198	0.324	0.361	0.403	0.421
	[10:50]	0.287	0.306	0.378	0.383	0.585	0.598
	[100:]	0.343	0.357	0.411	0.428	0.634	0.639



(a)



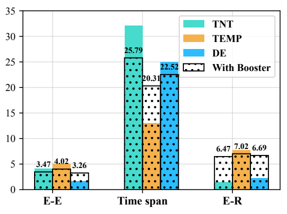
(b)

Figure 12: Variance reduction of TEMP (a) and TNT (b).

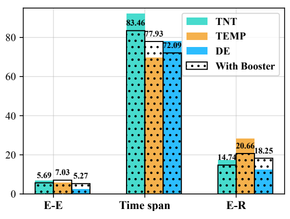
with 0–10 interacted entities, *Booster* boosts TA performance by 11.7%, compared with a 4.1% gain for those with over 100 entities. These results demonstrate that *Booster* effectively enriches sparse graph structures, leading to more balanced performance across timestamps and entities. Figures 10(c) and (d) further confirm its ability to reduce temporal performance imbalance. Given the foundational role of TKGC in real-world knowledge-enhanced applications, *Booster*'s balancing effect highlights its potential to improve the exposure of long-tail items and enhance recommendations.

Stability. To evaluate the effectiveness of *Booster* in stabilizing model performance, we randomly select four test samples whose *rank* metrics deteriorate over time during training with the TEMP model. As shown in Figure 11, these samples exhibit improved performance and reduced fluctuations when trained with the *Booster* framework. This improvement stems from *Booster*'s ability to mitigate model preference bias and reduce false negatives.

Furthermore, Figure 12 presents the standard deviation of *rank* across four independent training runs. Across all datasets, *Booster* consistently lowers variance for different baseline models, confirming its stabilizing effect. This enhanced stability not only reduces



(a)



(b)

Figure 13: Characteristics of top-ranked entities on the ICEWS 14 (a) and Wikidata (b) datasets.

model inconsistency but also improves the reliability of the system in real-world applications, where robustness and consistent output quality are critical for user adoption.

Model preference. To assess *Booster*'s effectiveness in addressing model preference, we follow the strategy in Figure 2(f) to analyze the top-ranked entities selected by each model and its *Booster*-enhanced variant. As shown in Figure 13, *Booster* helps balance entities with different characteristics, leading to fairer and more diverse recommendations in real-world scenarios.

6 CONCLUSION

In this paper, we make the first attempt to address the challenges of imbalanced data and model preference in temporal knowledge graph completion. We first reveal the limitations of existing methods through empirical analysis, and then propose *Booster*, the first pattern-aware data augmentation framework specifically designed for TKGs, to alleviate these issues. Extensive experiments on five benchmarks show that *Booster* consistently improves model performance while enhancing prediction balance. In future work, we plan to further explore the robustness of TKGC methods.

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of Chongqing (No. CSTB2023NSCQ-MSX1020), Guangxi Key Research and Development Program (No. Guike AB24010112) and National Natural Science Foundation of China (No. 62406057).

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[2] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. *ICEWS Coded Event Data*. Harvard Dataverse.

[3] Heng Chang, Jie Cai, and Jia Li. 2023. Knowledge Graph Completion with Counterfactual Augmentation. In *The Web Conference*. 2611–2620.

[4] Wei Chen, Haoyu Huang, Zhiyu Zhang, Tianyi Wang, Youfang Lin, and Liang Chang. 2025. Next-POI Recommendation via Spatial-Temporal Knowledge Graph Contrastive Learning and Trajectory Prompt. *IEEE Trans. Knowl. Data Eng.* (2025). https://doi.org/10.1109/TKDE.2025.3545958

[5] Wei Chen, Huaiyu Wan, Shengnan Guo, Haoyu Huang, Shaojie Zheng, Jiamu Li, Shuohao Lin, and Youfang Lin. 2022. Building and exploiting spatial-temporal knowledge graph for next POI recommendation. *Knowl. Based Syst.* 258 (2022), 109951.

[6] Wei Chen, Huaiyu Wan, Yuting Wu, Shuyuan Zhao, Jiayaqi Cheng, Yuxin Li, and Youfang Lin. 2024. Local-Global History-Aware Contrastive Learning for Temporal Knowledge Graph Reasoning. In *ICDE*. 733–746.

[7] Xiangnan Chen, Wen Zhang, Zhen Yao, Mingyang Chen, and Siliang Tang. 2023. Negative Sampling with Adaptive Denoising Mixup for Knowledge Graph Embedding. In *ISWC*. 253–270.

[8] Jinhao Cui, Heyan Chai, Xu Yang, Ye Ding, Binxing Fang, and Qing Liao. 2024. SGCL: Semantic-aware Graph Contrastive Learning with Lipschitz Graph Augmentation. In *ICDE*. 3028–3041.

[9] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. 2018. HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding. In *EMNLP*. 2001–2011.

[10] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data Augmentation for Deep Graph Learning: A Survey. *SIGKDD Explor.* 24, 2 (2022), 61–77.

[11] Wenying Duan, Xiaoxi He, Zimu Zhou, Lothar Thiele, and Hong Rao. 2023. Localised Adaptive Spatial-Temporal Graph Neural Network. In *KDD*. 448–458.

[12] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandecic. 2014. Introducing Wikidata to the Linked Data Web. In *ISWC*. 50–65.

[13] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *EMNLP*. 4816–4821.

[14] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.

[15] Rishab Goel, Seyed Mehran Kazemi, Marcus A. Brubaker, and Pascal Poupart. 2020. Diachronic Embedding for Temporal Knowledge Graph Completion. In *AAAI*. 3988–3995.

[16] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2022. Learning Dynamics and Heterogeneity of Spatial-Temporal Graph Data for Traffic Forecasting. *IEEE Trans. Knowl. Data Eng.* 34, 11 (2022), 5415–5428.

[17] Pankaj Gupta, Venu Satuluri, Ajeet Grewal, Siva Gurumurthy, Volodymyr Zhabiuk, Quannan Li, and Jimmy Lin. 2014. Real-Time Twitter Recommendation: Online Motif Detection in Large Dynamic Graphs. *Proc. VLDB Endow.* 7, 13 (2014), 1379–1380.

[18] Jindong Han, Weijia Zhang, Hao Liu, Tao Tao, Naiqiang Tan, and Hui Xiong. 2024. BigST: Linear Complexity Spatio-Temporal Graph Neural Network for Traffic Forecasting on Large-Scale Road Networks. *Proc. VLDB Endow.* 17, 5 (2024), 1081–1090.

[19] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. 2022. G-Mixup: Graph Data Augmentation for Graph Classification. In *ICML*, Vol. 162. 8230–8248.

[20] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021. Explainable Subgraph Reasoning for Forecasting on Temporal Knowledge Graphs. In *ICLR*.

[21] Rikui Huang, Wei Wei, Xiaoye Qu, Shengzhe Zhang, Dangyang Chen, and Yu Cheng. 2024. Confidence is not Timeless: Modeling Temporal Validity for Rule-based Temporal Knowledge Graph Forecasting. In *ACL*. 10783–10794.

[22] Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. 2020. Temporal Knowledge Base Completion: New Algorithms and Evaluation Protocols. In *EMNLP*. 3733–3747.

[23] Mingxuan Ju, Tong Zhao, Wenhao Yu, Neil Shah, and Yanfang Ye. 2023. GraphPatcher: Mitigating Degree Bias for Graph Neural Networks via Test-time Augmentation. In *NeurIPS*.

[24] Hidetaka Kamigaito and Katsuhiko Hayashi. 2022. Comprehensive Analysis of Negative Sampling in Knowledge Graph Representation Learning. In *ICML*, Vol. 162. 10661–10675.

[25] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. 2020. Tensor Decompositions for Temporal Knowledge Base Completion. In *ICLR*.

[26] Dongjin Lee, Kijung Shin, and Christos Faloutsos. 2020. Temporal locality-aware sampling for accurate triangle counting in real graph streams. *VLDB J.* 29, 6 (2020), 1501–1525.

[27] Kalev Leetaru and Philip A. Schrodt. 2013. Global Database of Events, Language and Tone. In *ISA*.

[28] Youru Li, Zhenfeng Zhu, Xiaobo Guo, Linxun Chen, Zhouyin Wang, Yinmeng Wang, Bing Han, and Yao Zhao. 2023. Learning Joint Relational Co-evolution in Spatial-Temporal Knowledge Graph for SMEs Supply Chain Prediction. In *KDD*. 4426–4436.

[29] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal Knowledge Graph Reasoning Based on Evolutional Representation Learning. In *SIGIR*. 408–417.

[30] Weibin Liao, Yifan Zhu, Yanyan Li, Qi Zhang, Zhonghong Ou, and Xuesong Li. 2025. RevGNN: Negative Sampling Enhanced Contrastive Graph Learning for Academic Reviewer Recommendation. *ACM Trans. Inf. Syst.* 43, 1 (2025), 1:1–1:26.

[31] Hongrui Liu, Binbin Hu, Xiao Wang, Chuan Shi, Zhiqiang Zhang, and Jun Zhou. 2022. Confidence May Cheat: Self-Training on Graph Neural Networks under Distribution Shift. In *WWW*. 1248–1258.

[32] Haoran Liu, Jianling Wang, Kaize Ding, and James Caverlee. 2023. Topological and Temporal Data Augmentation for Temporal Graph Networks. In *Temporal Graph Learning Workshop @ NeurIPS 2023*.

[33] Bin Lu, Ze Zhao, Xiaoying Gan, Shiyu Liang, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. 2024. Graph Out-of-Distribution Generalization With Controllable Data Augmentation. *IEEE Trans. Knowl. Data Eng.* 36, 11 (2024), 6317–6329.

[34] A. Agnes Lydia and F. Sagayaraj Francis. 2019. Adagrad - An optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci.* 6, 5 (2019), 566–568.

[35] Tiroshan Madushanka and Ryutaro Ichise. 2024. Negative Sampling in Knowledge Graph Representation Learning: A Review. *CoRR* abs/2402.19195 (2024).

[36] Saurav Manchanda. 2023. Metapath-Guided Data-Augmentation For Knowledge Graphs. In *CIKM*. 4175–4179.

[37] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. 2022. Named Entity Recognition and Relation Extraction: State-of-the-Art. *ACM Comput. Surv.* 54, 1 (2022), 20:1–20:39.

[38] Tianhao Peng, Haitao Yuan, Yongqi Zhang, Yuchen Li, Peihong Dai, Qunbo Wang, Senzhang Wang, and Wenjun Wu. 2025. TagRec: Temporal-Aware Graph Contrastive Learning With Theoretical Augmentation for Sequential Recommendation. *IEEE Trans. Knowl. Data Eng.* 37, 5 (2025), 3015–3029.

[39] Harry Shomer, Wei Jin, Wentao Wang, and Jiliang Tang. 2023. Toward Degree Bias in Embedding-Based Knowledge Graph Completion. In *WWW*. 705–715.

[40] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*. 697–706.

[41] Yongduo Sui, Qitian Wu, Jiancan Wu, Qing Cui, Longfei Li, Jun Zhou, Xiang Wang, and Xiangnan He. 2023. Unleashing the Power of Graph Data Augmentation on Covariate Distribution Shift. In *NeurIPS*.

[42] Shiyin Tan, Jingyi You, and Dongyuan Li. 2022. Temporality- and Frequency-aware Graph Contrastive Learning for Temporal Network. In *CIKM*. 1878–1888.

[43] Jizhi Tang, Yansong Feng, and Dongyan Zhao. 2019. Learning to Update Knowledge Graphs by Reading News. In *EMNLP-IJCNLP*. 2632–2641.

[44] Xing Tang, Ling Chen, Hongyu Shi, and Dandan Lyu. 2024. DHyper: A Recurrent Dual Hypergraph Neural Network for Event Prediction in Temporal Knowledge Graphs. *ACM Trans. Inf. Syst.* 42, 5 (2024), 129:1–129:23.

[45] Zhenwei Tang, Shichao Pei, Zhao Zhang, Yongchun Zhu, Fuzhen Zhuang, Robert Hoehndorf, and Xiangliang Zhang. 2022. Positive-Unlabeled Learning with Adversarial Data Augmentation for Knowledge Graph Completion. In *IJCAI*. 2248–2254.

[46] Yuxing Tian, Aiwen Jiang, Qi Huang, Jian Guo, and Yiyan Qi. 2024. Latent Diffusion-based Data Augmentation for Continuous-Time Dynamic Graph Model. In *KDD*. 2900–2911.

[47] Zhongwei Wan, Xin Liu, Benyou Wang, Jiezhong Qiu, Boyu Li, Ting Guo, Guangyong Chen, and Yang Wang. 2024. Spatio-temporal Contrastive Learning-enhanced GNNs for Session-based Recommendation. *ACM Trans. Inf. Syst.* 42, 2 (2024), 58:1–58:26.

[48] Jiapu Wang, Boyue Wang, Junbin Gao, Xiaoyan Li, Yongli Hu, and Baocai Yin. 2024. QDN: A Quadruplet Distributor Network for Temporal Knowledge Graph Completion. *IEEE Trans. Neural Networks Learn. Syst.* 35, 10 (2024), 14018–14030.

[49] Jiapu Wang, Boyue Wang, Junbin Gao, Shirui Pan, Tengfei Liu, Baocai Yin, and Wen Gao. 2024. MADE: Multicurvature Adaptive Embedding for Temporal Knowledge Graph Completion. *IEEE Trans. Cybern.* 54, 10 (2024), 5818–5831.

[50] Jiapu Wang, Boyue Wang, Meikang Qiu, Shirui Pan, Bo Xiong, Heng Liu, Linhao Luo, Tengfei Liu, Yongli Hu, Baocai Yin, and Wen Gao. 2023. A Survey on Temporal Knowledge Graph Completion: Taxonomy, Progress, and Prospects. *CoRR* abs/2308.02457 (2023).

[51] Yiwei Wang, Yujun Cai, Yuxuan Liang, Henghui Ding, Changhu Wang, Siddharth Bhatia, and Bryan Hooi. 2021. Adaptive Data Augmentation on Temporal Graphs. In *NeurIPS*. 1440–1452.

[52] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L. Hamilton. 2020. TeMP: Temporal Message Passing for Temporal Knowledge Graph Completion. In *EMNLP*. 5730–5746.

[53] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2023. Self-Supervised Learning of Graph Neural Networks: A Unified Review. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 2 (2023), 2412–2429.

[54] Hao Xin and Lei Chen. 2024. KartGPS: Knowledge Base Update with Temporal Graph Pattern-based Semantic Rules. In *ICDE*. 5075–5087.

[55] Siheng Xiong, Yuan Yang, Faramarz Fekri, and James Clayton Kerce. 2023. TILP: Differentiable Learning of Temporal Logical Rules on Knowledge Graphs. In *ICLR*.

[56] Chengjin Xu, Yung-Yu Chen, Mojtaba Nayyeri, and Jens Lehmann. 2021. Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings. In *NAACL-HLT*. 2569–2578.

[57] Chenjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020. Temporal Knowledge Graph Completion Based on Time Series Gaussian Embedding. In *ISWC*. 654–671.

[58] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Hamed Shariat Yazdi, and Jens Lehmann. 2020. TeRo: A Time-aware Knowledge Graph Embedding via Temporal Rotation. In *COLING*. 1583–1593.

[59] Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal Knowledge Graph Reasoning with Historical Contrastive Learning. In *AAAI*. 4765–4773.

[60] Jinfa Yang, Xianghua Ying, Yongjie Shi, and Bowei Xing. 2024. Tensor decompositions for temporal knowledge graph completion with time perspective. *Expert Syst. Appl.* 237, Part A (2024), 121267.

[61] Ruiyi Yang, Flora D. Salim, and Hao Xue. 2024. SSTKG: Simple Spatio-Temporal Knowledge Graph for Intepretable and Versatile Dynamic Information Embedding. In *WWW*. 551–559.

[62] Naimeng Yao, Qing Liu, Yi Yang, Weihua Li, and Quan Bai. 2023. Entity-Relation Distribution-Aware Negative Sampling for Knowledge Graph Embedding. In *ISWC*. 234–252.

[63] Yuanzhou Yao, Zhao Zhang, Yongjun Xu, and Chao Li. 2022. Data Augmentation for Few-Shot Knowledge Graph Completion from Hierarchical Perspective. In *COLING*. 2494–2503.

[64] Fu Zhang, Hongzhi Chen, Yuzhe Shi, Jingwei Cheng, and Jinghao Lin. 2024. Joint framework for tensor decomposition-based temporal knowledge graph completion. *Inf. Sci.* 654 (2024), 119853.

[65] Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. 2023. Learning Long- and Short-term Representations for Temporal Knowledge Graph Reasoning. In *WWW*. 2412–2422.

[66] Qianru Zhang, Lianghao Xia, Xuheng Cai, Siu-Ming Yiu, Chao Huang, and Christian S. Jensen. 2024. Graph Augmentation for Recommendation. In *ICDE*. 557–569.

[67] Shengzhe Zhang, Liyi Chen, Chao Wang, Shuangli Li, and Hui Xiong. 2024. Temporal Graph Contrastive Learning for Sequential Recommendation. In *AAAI*. 9359–9367.

[68] Shuaicheng Zhang, Yada Zhu, and Dawei Zhou. 2023. TGEditor: Task-Guided Graph Editing for Augmenting Temporal Financial Transaction Networks. In *ICAIF*. 219–226.

[69] Yongqi Zhang, Quanming Yao, and Lei Chen. 2021. Simple and automated negative sampling for knowledge graph embedding. *VLDB J.* 30, 2 (2021), 259–285.

[70] Yuyue Zhao, Xiang Wang, Jiawei Chen, Yashen Wang, Wei Tang, Xiangnan He, and Haiyong Xie. 2023. Time-aware Path Reasoning on Knowledge Graph for Recommendation. *ACM Trans. Inf. Syst.* 41, 2 (2023), 26:1–26:26.

[71] Xinyi Zhu, Liping Wang, Hao Xin, Xiaohan Wang, Zhifeng Jia, Jiyao Wang, Chunming Ma, and Yuxiang Zengt. 2023. T-FinKB: A Platform of Temporal Financial Knowledge Base Construction. In *ICDE*. 3671–3674.

[72] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. 2021. An Empirical Study of Graph Contrastive Learning. In *NeurIPS Datasets and Benchmarks*.