DEEP GRAPH LEARNING FOR SOCIAL-INFO DYNAMICS

BY

RUIJIE WANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Doctoral Committee:

Professor Tarek Abdelzaher, Chair
Professor Jiawei Han
Associate Professor Hanghang Tong
Professor Boleslaw K. Szymanski, Rensselaer Polytechnic Institute

ABSTRACT

Graph-structured data are prevalent in a wide range of application domains, as many data inherently demonstrate interconnected patterns. With recent advances in artificial intelligence, deep graph learning methods have revolutionized the way of modeling, processing, and learning from graphs. They are typically integrated into a three-stage pipeline: graph construction, offline model design, and online deployment. Real-world graphs, by their nature, undergo constant evolution. In an exemplifying social network scenario, the evolution exhibits in both micro scale where dynamics manifest as evolving node attributes and interaction patterns, and macro scale where graph dynamics involve the continual emergence of new nodes and edges. The typical three-stage graph learning pipeline falls short of learning latent representations of such changing, partially observed, and unreliable social information environment (referred to as social-info dynamics), and also lacks the capacity to model broader graph dynamics beyond social networks. Significant challenges need to be addressed in terms of robustness (handling data issues), efficacy (modeling temporal information), and efficiency (enabling continual model updates), corresponding to each of the three key stages.

This dissertation works on optimizing robustness, efficacy, and efficiency of current graph learning techniques, with the aim of better modeling graph dynamics. Specifically, we address the following research questions in pursuit of establishing robust, efficient, and effective learning pipelines. First, can we devise an automated graph cleaning (refinement) method to bolster model robustness against data issues? If so, can we eliminate dependency on external cues to indicate data correctness, and how should we redefine the graph cleaning objective in contrast to traditional unsupervised graph learning methods? Second, when it comes to modeling, representing, and learning a diverse range of dynamic graphs amid growing data heterogeneity and task complexity, what level of model intricacy should we explore? To be more specific, when dealing with dynamic bipartite, multi-relational, or low-resource graphs, what strategies should we employ in designing graph learning techniques to enhance the extraction of valuable information that can benefit downstream applications? Finally, is it feasible to formulate an online training policy to enhance label-efficiency for adapting the model to new nodes with limited labels and resource-efficiency for updating models on streaming data? Can we build upon a common design philosophy to guide both endeavors?

This dissertation elaborates on these core problems and their emerging temporal graph learning solutions to build robust, effective, and efficient learning pipelines that facilitate predictive analytics in changing, partially observed, and unreliable graphs.

To my lovely girlfriend, my parents, and my family, for their dedicated love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Tarek F. Abdelzaher. From the commencement of my Ph.D. journey, he has consistently provided invaluable guidance, demonstrated immense patience, and offered unwavering support. I have greatly benefited from his research vision, deep passion, and acute critical thinking. His encouragement and insightful suggestions have been instrumental in navigating the challenges of my research and throughout the various phases of my Ph.D. Without his support, the completion of this thesis would have been unattainable.

It is a genuine pleasure to express my deep appreciation to my Ph.D. thesis committee, Professor Jiawei Han, Professor Hanghang Tong, and Professor Boleslaw K. Szymanski, for providing constructive suggestions and insightful advises on my research work. It is my greatest honor to collaborate on projects, discuss research ideas, coauthor papers with them throughout the years. These collaborations and discussions have significantly broadened my perspective, fueled my passion, and enriched my intellectual depth.

I also feel extremely fortunate to have the opportunities to work with my great mentors and collaborators over the past years. Special thanks to Prof. Xinbing Wang and Prof. Weinan Zhang, for their supervision during my undergraduate study at Shanghai Jiao Tong University (SJTU). Many thanks to Bing Yin, Dr. Zheng Li, and Dr. Danqing Zhang for hosting my internships at Amazon Search, where I spent wonderful summers. Besides, I would like to extend my sincere gratitude to Prof. Shuochao Yao, Prof. Huajie Shao, Prof. Shengzhong Liu, Prof. Haiming Jin, Prof. Charith Mendis, Dr. Dongxin Liu, Yuchen Yan, Jialu Wang, Chaoqi Yang, Jingfeng Yang, Prof. Chao Zhang, Dr. Tianyu Cao, Dr. Qingyu Yin, Dr. Tong Zhao, Yufeng Wang, Chuan Wen, for their invaluable discussions and constructive advice.

I am always proud and fortunate to be a member of the Cyber-Physical Computing Group at University of Illinois at Urbana-Champaign (UIUC). I would like to thank Jinyang Li, Tianshi Wang, Dachun Sun, Yigong Hu, Jinning Li, Yizhuo Chen, Xinyi Liu, Kara Denizhan, MD Iftekharul Islam Sakib, Christina Youn for the unforgettable days and nights. Many thanks to Jianing Zhou, Hongye Jin, Fei Xue, Jiachen Li, Yifan Qiao, Ying Jing, Dr. Limin Yang, Zirui Zhao, Qingying Hao, Dr. Eli Cheung, for pleasure time and harworking days.

In addition, I would like to thank the U.S. Army Research Labs, Defense Advanced Research Projects Agency, National Science Foundation, and Amazon Research for their fnancial support to my Ph.D. studies.

Lastly and most importantly, I would like to thank my lovely girlfriend, my parents Yanjun Li and Dong Wang, as well as my entire family, for their unconditional love, comprehension, and support throughout the journey. Therefore, this dissertation is dedicated to them.

CONTENTS

Chapter 1: INTRODUCTION

Graph-structured data are prevalent in a wide range of application domains, as many data inherently demonstrate interconnected patterns. These examples encompass but are not limited to, social graphs, E-Commerce graphs, knowledge graphs, and molecular graphs. In the era of the Internet and big data, the utility of graphs has deepened further as larger and more heterogeneous graph-structured data continues to emerge. For instance, statistics reveal that approximately 5 billion users are interconnected in social networks, and over 60 billion Google-indexed webpages are linked via the Internet in 2023. Moreover, in a broader context, physical objects in the world are also interconnected as vast graphs, encompassing autonomous vehicles, a myriad of sensor networks, and large-scale computation clusters, to name a few. Therefore, computation on graphs has garnered significant interest in recent years due to their wide-ranging applicability in real-world scenarios. Over the past decade, artificial intelligence has undergone swift advancements, with deep learning techniques standing out for their significant achievements. These techniques have demonstrated tremendous progress in modeling various real-world data types, encompassing images, videos, languages, and speech. Similarly, deep graph learning methods have also revolutionized the way of modeling, processing, and learning from various types of graph-structured data. Many intelligent capabilities on graphs, such as recommendation, community detection, user classification, and disinformation detection, are therefore enabled that extend their impact not only to the research community and industry but also touch the lives of everyday people.

A typical graph learning pipeline consists of three stages: 1) *Graph construction stage*: This initial phase involves the organization of heterogeneous information derived from complex application scenarios into graph structures. An inherent advantage of utilizing graphs over other data formats lies in their innate ability to account for interdependencies among data samples. This enables the integration of diverse information from multiple sources by capturing their relationships. 2) *Offline training stage*: In the subsequent stage, a dedicated graph learning model is designed and trained to effectively model, represent, and acquire insights from the constructed graphs. A wide array of graph learning techniques has been developed for various types of constructed graphs, spanning from methods like heterogeneous information network embedding, knowledge graph embedding, and shallow graph embedding, to more recent advancements in deep graph learning methods. 3) *Online deployment stage*: Given the ongoing influx of new data during model deployment, this stage necessitates periodic, iterative model updates to refine the model's performance with the latest information. Overextended deployment periods, may even mandate a comprehensive

model retraining, potentially requiring a return to Stage 2 or a complete graph reconstruction back to Stage 1, as a non-negligible performance degradation can become evident.

The primary objective of this dissertation is to pioneer temporal graph learning techniques that enhance the aforementioned pipeline to better account for and model the dynamic nature of graphs. This is motivated by the observation that numerous real-world graphs continuously evolve, and the existing graph learning processes fall short of capturing such dynamics. Taking social networks as an example, the status is continuously evolving over time, influenced by both internal information propagation within the networks and external events in the physical world. The evolution can be observed and interpreted in two ways. On a micro scale, dynamics manifest as evolving node behaviors, encompassing changes in node attributes such as fluctuating opinions of social media users and evolving node interaction patterns like shifting consumer habits on E-Commerce platforms. Conversely, from a macro perspective, graph dynamics involve the consistent joining of new nodes and edges, infusing fresh information. Therefore, it is interesting and practical to explore the learning and extraction of dynamic latent representations of the changing, partially observed, and unreliable social information environment, which we refer to as *social info-dynamics*, that can benefit related downstream applications. More broadly, the pursuit of advancing the existing graph pipeline and learning the *graph dynamics*, becomes an intriguing yet formidable problem.

However, modeling these dynamics within real-world graphs presents non-trivial challenges, from three main perspectives. First, from a model robustness perspective, the input graph data derived from real-world scenarios is often characterized by noise and incompleteness. Data collection challenges give rise to a substantial number of *false negative issues*, where correct links are unintentionally omitted, and *false positive issues*, where non-existent links are erroneously included in the graphs. Many graph learning methods are built on implicit assumptions that the collected links are true positives and the uncollected links are true negatives, rendering them susceptible to these data issues. This challenge is further exacerbated when modeling evolutionary patterns within dynamic graphs, as the models struggle to distinguish between random fluctuations caused by data issues at each time interval and genuine evolutionary trends. Secondly, the endeavor to model temporal information and graph dynamics introduces distinctive efficacy challenges, during offline training stage. Addressing these challenges becomes progressively more demanding when dealing with different types of dynamic graphs tailored to specific application contexts and desired outcomes. These challenges intensify when dealing with dynamic bipartite graphs that mainly encompass interaction data, dynamic multi-relational graphs that consider timestamped heterogeneous connections, and dynamic low-resource graphs, where the scarcity of available links results in insufficient information for effective model training. Furthermore,

in light of the constant influx of new data during the online deployment phase, a distinctive efficiency challenge surfaces, which revolves around the effective adaptation of the model to accommodate this fresh data. In more specific terms, this challenge necessitates enhancing *label-efficiency* to model newly emerging entities with a limited number of links or labels, as well as optimizing *resource-efficiency* to enable swift model updates in order to prevent performance deterioration.

To elaborate on the above challenges, we establish the following statement for this dissertation: *By designing novel temporal graph learning techniques, we can build robust, effective, and efficient learning pipelines to facilitate predictive analytics in changing, partially observed, and unreliable graphs.* To set the scope, this dissertation is, in part, motivated by challenges arising in modeling social-info dynamics from social networks. However, our design endeavors have expanded to encompass various application domains characterized by graphs and dynamics akin to those in social networks. These domains include recommendation systems, knowledge graph reasoning, temporal event prediction, and more. To specify the applicable scenarios of this dissertation, we establish certain assumptions. Our techniques are designed to be applicable to real-world graphs that align with these assumptions. 1) *Graph dynamics assumption*: This dissertation mainly considers and models graph dynamics caused by both graph topology changes (*i.e.,* addition and deletion of nodes/edges) and graph attribute changes (*i.e.,* changes of node/edge attributes). Consequently, situations where the graphs undergo only attribute changes while retaining identical structures throughout the duration are not addressed within the scope of this dissertation. 2) *Graph noise assumption*: The main emphasis of this dissertation is on enhancing model robustness in the face of graph noise arising from issues in upstream data collection and model outputs. The noise we investigate stems from imperfect data preparation and processing rather than intentional, malicious data modification or injection aimed at degrading our model's performance. Consequently, our primary focus is not directed towards improving robustness against adversarial attacks, although our design does demonstrate resilience against some of them. 3) *Graph smoothness assumption*: While the primary goal of this dissertation is to address graph dynamics in various practical scenarios, we operate under the assumption that certain fundamental graph patterns persist over time. This ensures the continued validity of our technical designs across different time frames. These enduring graph patterns encompass power-law distribution and a fixed homophily or heterophily property, meaning that graphs do not transition between homophily and heterophily at specific time points.

With the dissertation statement and important assumptions established, we address the following research questions in pursuit of establishing robust, efficient, and effective learning pipelines. First, can we devise an automated graph cleaning (refinement) method to bolster

model robustness against graph data noise including false negative and false positive issues? If so, can we eliminate dependency on external cues to indicate data correctness, and how should we redefine the graph cleaning objective in contrast to traditional unsupervised graph learning methods? Second, when it comes to modeling, representing, and learning a diverse range of dynamic graphs amid growing data heterogeneity and task complexity, what level of model intricacy should we explore? To be more specific, when dealing with dynamic bipartite, multi-relational, or low-resource graphs, what strategies should we employ in designing graph learning techniques to enhance the extraction of valuable information that can benefit downstream applications? Finally, is it feasible to formulate an online training policy to enhance label efficiency for adapting the model to new nodes and resource efficiency for updating the model with streaming data? Can we build upon a common design philosophy to guide both endeavors?

In the following sections, we give an overview of the solutions we proposed for each of the above research problems.

1.1 ROBUST GRAPH CLEANING FOR SPECULATIVE REASONING

This paper studies graph cleaning and corresponding speculative reasoning capability on real-world graphs that contain both *false negative issue* (i.e., potential true links being excluded) and *false positive issue* (i.e., unreliable or outdated links being included). To set the scope, we mainly focus on improving robustness against real-world data issues instead of adversarial attacks to the models. We refer to automatic data issue detection joint with task-specific predictive reasoning ability as speculative reasoning, which lies the foundation of effective and efficient dynamic graph learning objective. State-of-the-art methods fall short in such ability, as they assume the correctness of a fact is solely determined by its presence in graphs, making them vulnerable to false negative/positive issues. The new reasoning task is formulated as a noisy Positive-Unlabeled learning problem. We propose a variational framework, namely nPUGraph [1], that jointly estimates the correctness of both collected and uncollected links (which we call *label posterior*) and updates model parameters during training. The label posterior estimation facilitates speculative reasoning from two perspectives. First, it improves the robustness of a label posterior-aware graph encoder against false positive links. Second, it identifies missing links to provide high-quality grounds of reasoning. They are unified in a simple yet effective self-training procedure, without requirement of any external cues to indicate link correctness. Empirically, extensive experiments on link prediction, knowledge graph reasoning, and ideology classification tasks with various degrees of false negative/positive cases demonstrate the effectiveness of nPUGraph. This work is introduced

in detail in Chapter 2.

1.2 LEARNING DYNAMIC BIPARTITE GRAPHS

This paper describes a novel graph learning model, DyDiff-VAE [2], aiming at learning dynamic bipartite graphs for information diffusion prediction task on social media. By modeling the external driven factors (which we call external stimuli) and internal driven factors (which we call social influence), DyDiff-VAE aims to extract user dynamic interests from the bipartite graphs. Inferring user interests from diffusion data lies the foundation of diffusion prediction, because users often forward the information in which they are interested or the information from those who share similar interests. Their interests also evolve over time as the result of the dynamic social influence from neighbors and the time-sensitive information gained inside/outside the social media. Existing works fail to model users' intrinsic interests from the diffusion data and assume user interests remain static along the time. DyDiff-VAE advances the state of the art in two directions: (i) We propose a dynamic encoder to infer the evolution of user interests from observed diffusion data. (ii) We propose a dual attentive decoder to estimate the propagation likelihood by integrating information from both the initial cascade content and the forwarding user sequence. Extensive experiments on four real-world datasets from Twitter and Youtube demonstrate the advantages of the proposed model. This work is introduced in detail in Chapter 3.

1.3 LEARNING DYNAMIC MULTI-RELATIONAL GRAPHS

This paper studies a generalized user behavior prediction task in a dynamic multi-relational graph. To fulfill this setting, there involves two challenges: (1) the action data for most users is scarce, while the neighbor aggregation cannot be easily extended in the scenario to enhance user representations because of data heterogeneity; (2) user preferences are dynamically evolving and shifting over time. To tackle those issues, we propose a novel *Retrieval-Enhanced Temporal Event* (RETE) [3] forecasting framework. Unlike existing methods that enhance user representations via roughly absorbing information from connected entities in the whole graph, RETE efficiently and dynamically retrieves relevant entities centrally on each user as high-quality subgraphs, preventing the noise propagation from the densely evolutionary graph structures that incorporate abundant information. And meanwhile, RETE autoregressively accumulates retrieval-enhanced user representations from each time step, to capture evolutionary patterns over time. Empirically, extensive experiments on both

the public benchmark and four real-world industrial datasets demonstrate the effectiveness of the proposed RETE method for modeling the dynamic multi-relational graphs. This work is introduced in detail in Chapter 4.

1.4 LEARNING DYNAMIC LOW-RESOURCE GRAPHS WITH KNOWLEDGE TRANSFER

This paper investigates temporal reasoning problem on scarce temporal graphs, which aims to facilitate reasoning on temporal graphs in low-resource domain by transferring knowledge from graphs in high-resource ones. The cross-network distillation ability, as a way to enable the external knowledge transfer capability, becomes increasingly crucial, in light of the unsatisfying performance of existing reasoning methods on those severely incomplete temporal graphs, especially in low-resource domain. However, it poses tremendous challenges in two aspects. First, the cross-network alignments, which serve as bridges for knowledge transfer, are usually too scarce to transfer sufficient knowledge between two graphs. Second, temporal knowledge discrepancy of the aligned entities, especially when alignments are unreliable, can mislead the knowledge distillation process. We correspondingly propose a mutually-paced knowledge distillation model MP-KD [4], where a teacher network trained on a source graph can guide the training of a student network on target graph with an alignment module. Concretely, to deal with the scarcity issue, MP-KD generates pseudo alignments between temporal graphs based on the temporal information extracted by our representation module. To maximize the efficacy of knowledge transfer and control the noise caused by the temporal knowledge discrepancy, we enhance MP-KD with a temporal cross-lingual attention mechanism to dynamically estimate the alignment strength. The two procedures are mutually paced along with model training. Extensive experiments on twelve cross-lingual knowledge transfer tasks in the EventKG benchmark demonstrate the effectiveness of the proposed MP-KD method. This work is introduced in detail in Chapter 5.

1.5 LEARNING EMERGING NODES FOR LABEL-EFFICIENT MODEL ADAPTATION

Moving to the label-efficiency objective, this paper investigate a realistic but underexplored problem, called few-shot temporal reasoning task, that aims to predict future facts for newly emerging entities based on extremely limited observations in evolving graphs. It offers practical value in applications that need to derive instant new knowledge about new entities in temporal graphs with minimal supervision. The challenges mainly come from the *few-shot*

and *time shift* properties of new entities. First, the limited observations associated with them are insufficient for training a model from scratch. Second, the potentially dynamic distributions from the initially observable facts to the future facts ask for explicitly modeling the evolving characteristics of new entities. We correspondingly propose a MetaTKGR framework [5]. Unlike prior work that relies on rigid neighborhood aggregation schemes to enhance low-data entity representation, MetaTKGR dynamically adjusts the strategies of sampling and aggregating neighbors from recent facts for new entities, through temporally supervised signals on future facts as instant feedback. Besides, such a meta temporal reasoning procedure goes beyond existing meta-learning paradigms on static graphs that fail to handle temporal adaptation with large entity variance. We further provide a theoretical analysis and propose a temporal adaptation regularizer to stabilize the meta temporal reasoning over time. Empirically, extensive experiments on three real-world temporal graphs demonstrate the superiority of the proposed method over state-of-the-art baselines by a large margin.

We further extend the framework from Euclidean space to hyperbolic space, which is advantageous for modeling emerging graph entities for two reasons: First, its geometric property of exponential expansion aligns with the rapid growth of new entities in real-world graphs; Second, it excels in capturing power-law patterns and hierarchical structures, well-suitable for new entities distributed at the peripheries of graph hierarchies and loosely connected with others through few links. We therefore propose a meta-learning framework, MetaHKG, to enable few-shot temporal reasoning within a hyperbolic space. Unlike prior hyperbolic learning works, MetaHKG addresses the challenges of effectively representing new entities and adapting model parameters by incorporating novel hyperbolic time encodings and temporal attention networks that achieve translational invariance. We also introduce a meta hyperbolic optimization algorithm to enhance model adaptation by learning both global and entity-specific parameters through bi-level optimization. Compared to its Euclidean counterpart, MetaHKG operates in a lower-dimensional space but yields a more stable and efficient adaptability towards new entities. These workw are introduced in detail in Chapter 6.

1.6 LEARNING STREAMING GRAPHS FOR RESOURCE-EFFICIENT MODEL UPDATES

This paper studies online link prediction on streaming temporal graphs, aiming to improve resource-efficiency to update deployed models on freshly acquired temporal data to ensure sustained long-term performance. State-of-the-art methods fall short in retaining and adapting informative knowledge distilled from existing data onto freshly gathered data for online updates, as they either cater exclusively to offline scenarios where all training

data is available upfront or lack sufficient modeling of temporal information and temporal graph structures during online updates. We propose a temporal meta-training framework, namely OnlineSAFE, that extracts enduringly valuable knowledge across data collection periods during the offline phase and efficiently fine-tunes the model to encode newly emerging patterns during the online phase. To this end, we design a bi-level optimization to meta-learn the model parameters that ensure sustained long-term performance and adaptability to new data, where outer/inner loops are nested to optimize the global model parameters and the fine-tuning procedure, respectively. Considering the potentially distinct distribution exhibited in the new data, we analyze and derive an empirical bound based on the PAC-Bayes theory to enhance the stability and generalizability of the online updating process. Furthermore, we investigate a simple but effective sample reduction heuristic that accelerates online updates by bypassing edge samples that lack additional information. Extensive experiments on four real-world streaming graphs demonstrate the effectiveness and efficiency of OnlineSAFE, compared with 17 state-of-the-art baselines. This work is introduced in detail in Chapter 7.

1.7 DISSERTATION ORGANIZATION

The rest of this dissertation is organized as follows: Part I (*i.e.,* Chapter 2) introduces the graph cleaning work nPUGraph to address the robustness challenge during the graph construction stage. Part II (*i.e.,* Chapter 3-5) introduces research works to address the efficacy challenge during the offline training stage, with increasing of model complexity to deal with various types of dynamic graphs. Part III (*i.e.,* Chapter 6-7) focuses on improving efficiency during online stage, in which we propose MetaTKGR and MetaHKG to improve label-efficiency for newly emerging nodes via online model adaptation, and OnlineSAFE to improve resource-efficiency for fast model update to avoid progressive performance drop. We conclude this dissertation and discuss future opportunities in Chapter 8.

Chapter 2: ROBUST GRAPH CLEANING FOR SPECULATIVE REASONING

2.1 OVERVIEW

In this chapter, we introduce a graph cleaning and corresponding speculative reasoning capability on real-world graphs that contain both *false negative issue* (i.e., potential true links being excluded) and *false positive issue* (i.e., unreliable or outdated links being included). To set the scope, we mainly focus on improving robustness against real-world data issues instead of adversarial attacks to our model generated by other attackers. We refer to automatic data issue detection joint with task-specific predictive reasoning ability as speculative reasoning, which lies the foundation of effective and efficient dynamic graph learning objective.

We aim to enable the speculative reasoning ability on real-world graphs, including but not limited to social interaction graphs [6], [7], knowledge graphs [8]–[11], etc. The fulfillment of the goal needs to address two commonly existing issues, as shown in Figure 2.1: 1) **The false negative issue** (i.e., sparse observation): Due to the graph incompleteness, facts excluded from the KG can be used as implicit grounds of reasoning. This is particularly applicable to non-obvious facts. For example, personal information such as the birthplace of politicians may be missing when constructing a political KG, as they are not explicitly stated in the political corpus [12]. However, it can be critical while reasoning personal facts like nationality. 2) **The false positive issue** (i.e., noisy observation): Facts included in the KG may be unreliable and should not be directly grounded without inspection. It can happen when relations between entities are incorrectly collected or when facts are extracted from outdated or unreliable sources. For example, Mary Elizabeth is no longer the Prime Minister of the United Kingdom, which may affect the reasoning accuracy of her current workplace. These issues generally affect both one-hop reasoning [9] and multi-hop reasoning [13]. The main focus of this paper is investigating the one-hop speculative reasoning task as it lays the basis for complicated multi-hop reasoning capability.

Speculative reasoning differs from conventional graph/KG reasoning in that the correctness of each collected/uncollected fact needs to be dynamically estimated as part of the learning process, such that the grounds of reasoning can be accordingly calibrated. Unfortunately, most existing work, if not all, lacks such inspection capability. Knowledge graph embedding methods [9], [14]–[17] and graph neural network (GNN) methods [18]–[22] can easily overfit the false negative/positive cases because of their training objective that ranks the collected facts higher than other uncollected facts in terms of plausibility. Recent attempts on uncertain graphs [23], [24] measure the uncertainty scores for facts, which can be utilized to detect false

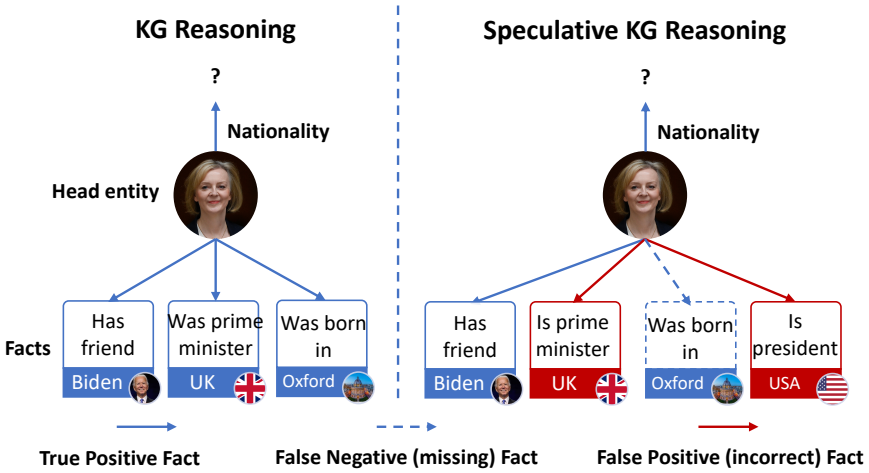


Figure 2.1: An illustrative example of speculative reasoning. Blue solid lines denote the true positive fact, blue dashed lines denote the *false negative* (missing) fact, and red solid lines denote the *false positive* (incorrect) fact. We aim to mitigate the false negative/positive issues to enable speculative reasoning.

negative/positive samples. However, they explicitly require the ground truth uncertainty scores as supervision for reasoning model training, which are usually unavailable in practice.

Motivated by these observations, we formulate the speculative reasoning task as a noisy Positive-Unlabeled learning problem. The facts contained in the graphs are seen as noisy positive samples with a certain level of label noise, and the facts excluded from the graphs are treated as unlabeled samples, which include both negative ones and possible factual ones. Instead of determining the correctness of facts before training the reasoning model without inspection, we learn the two perspectives in an end-to-end training process. To this end, we propose nPUGraph, a novel variational framework that regards the underlying correctness of collected/uncollected facts in the graphs as latent variables for the reasoning process. We jointly update model parameters and estimate the posterior likelihood of the correctness of each collected/uncollected fact (referred to as *label posterior*), through maximizing a theoretical lower bound of the log-likelihood of each fact being collected or uncollected.

The estimated label posterior further facilitates the speculative reasoning from two aspects: 1) It removes false positive facts contained in KG and improves the representation quality. We accordingly propose a label posterior-aware encoder to incorporate information only from entity neighbors induced by facts with a high posterior probability, under the assumption that the true positive facts from the collected facts provide more reliable information for reasoning. 2) It complements the grounds of reasoning by selecting missing but possibly plausible facts with high label posterior, which are iteratively added to acquire more informative samples for model training. These two procedures are ultimately unified in a simple yet

effective self-training strategy that alternates between the *data sampling based on latest label posteriors* and the *model training based on latest data samples*. Empirically, nPUGraph outperforms eleven state-of-the-art baselines on three benchmark KG data and one Twitter data we collected by large margins. Additionally, its robustness is demonstrated in speculative reasoning on data with multiple ratios of false negative/positive cases.

Our contributions are summarized as follows: (1) We open up a practical but underexplored problem space of speculative reasoning, and formulate it as a noisy Positive-Unlabeled learning task; (2) We take the first step in tackling this problem by proposing a variational framework nPUGraph to jointly optimize reasoning model parameters and estimate fact label posteriors; (3) We propose a simple yet effective self-training strategy for nPUGraph to simultaneously deal with false negative/positive issues; (4) We perform extensive evaluations to verify the effectiveness of nPUGraph on both benchmark KG and Twitter interaction data with a wide range of data perturbations.

2.2 PRELIMINARIES

2.2.1 Speculative Reasoning

A multi-relational graph (*e.g.,* KG) is denoted as $\mathcal{G} = \{(e_h, r, e_t)\} \subseteq \mathcal{S}$, where $\mathcal{S} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ denotes triple space, $\mathcal{E}$ denotes the entity set, $\mathcal{R}$ denotes the relation set. Each triple $s = (e_h, r, e_t)$ refers to that a head entity $e_h \in \mathcal{E}$ has a relation $r \in \mathcal{R}$ with a tail entity $e_t \in \mathcal{E}$. Typically, a score function $\psi(s; \Theta)$, parameterized by Θ, is designed to measure the plausibility of each potential triple $s = (e_h, r, e_t)$, and to rank the most plausible missing ones to complete the graph during inference [9], [17]. The goal of speculative reasoning is to infer the most plausible triple for each incomplete triple $(e_h, r, e_?)$ or $(e_?, r, e_t)$ given by sparse and unreliable observations in $\mathcal{G}$. In addition, it requires correctness estimation for each potential fact collected or uncollected by $\mathcal{G}$.

2.2.2 Noisy Positive-Unlabeled Learning

Positive-Unlabeled (PU) learning is a learning paradigm for training a model when only positive and unlabeled data is available [25]. We formulate the speculative reasoning task as a noisy Positive-Unlabeled learning problem, where the positive set contains potentially label noise from false facts [26].

PU Triple Distribution. For the speculative reasoning task, we aim to learn a binary classifier that maps a triple space $\mathcal{S}$ to a label space $\mathcal{Y} = \{0, 1\}$. Data are split as labeled

(collected)[1] triples $s^l \in \mathcal{S}^L$ and unlabeled (uncollected) triples $s^u \in \mathcal{S}^U$. The labeled triples are considered noisy positive samples with a certain level of label noise. The distribution of labeled triples can be represented as follows:

$$s^l \sim \beta\phi_1^l(s^l) + (1-\beta)\phi_0^l(s^l), \tag{2.1}$$

where ϕ_y^l denotes the probability of being collected over triple space $\mathcal{S}$ for the positive class ($y = 1$) and negative class ($y = 0$), and $\beta \in [0, 1)$ denotes the proportion of true positive samples in labeled data. Unlabeled triples include both negative samples and possible factual samples. The distribution of unlabeled samples can be represented as follows:

$$s^u \sim \alpha\phi_1^u(s^u) + (1-\alpha)\phi_0^u(s^u), \tag{2.2}$$

where $\phi_y^u = 1 - \phi_y^l$ denotes the probability of being uncollected, $\alpha \in [0, 1)$ denotes the positive class prior, i.e., the proportion of positive samples in unlabeled data.

PU Triple Construction. We then discuss the construction of $\mathcal{S}^L$ and $\mathcal{S}^U$ based on the collected graph $\mathcal{G}$. Triples in $\mathcal{G}$ naturally serve as labeled samples with a ratio of noise, i.e., $\mathcal{S}^L = \mathcal{G}$. For unlabeled set $\mathcal{S}^U$, However, directly using $\mathcal{S} \setminus \mathcal{G}$ as the unlabeled set $\mathcal{S}^U$ would result in too many unlabeled samples for training due to the large number of possible triples in triple space $\mathcal{S}$. Following [12], we construct $\mathcal{S}^U$ as follows: For each labeled triple $s_i^l = (e_h, r, e_t)$, we construct K unlabeled triples s_{ik}^u by replacing the head and tail respectively with other entities: $s_{ik}^u = (e_h, r, e_k^-)$ or (e_k^-, r, e_t), where e_k^- is the selected entity that ensures $s_{ik}^u \notin \mathcal{S}^L$. Initially, the construction can be randomized. During the training process, it is further improved by selecting unlabeled samples with high label posterior in a self-training scheme, so as to cover positive samples in the unlabeled set to the greatest extent.

2.3 THE DESIGN OF NPUGRAPH

2.3.1 Overview

Our approach views underlying triple labels (positive/negative) as latent variables, influencing the collection probability. Unlike the common objective of reasoning training that ranks the plausibility of the collected triples higher than uncollected ones, we instead maximize the data likelihood of each potential triple being collected or not. To this end, as shown in Figure 2.2, we propose nPUGraph framework to jointly optimize parameters and

[1] In this paper, we interchangeably use the term *labeled/unlabeled* and *collected/uncollected* with no distinction.

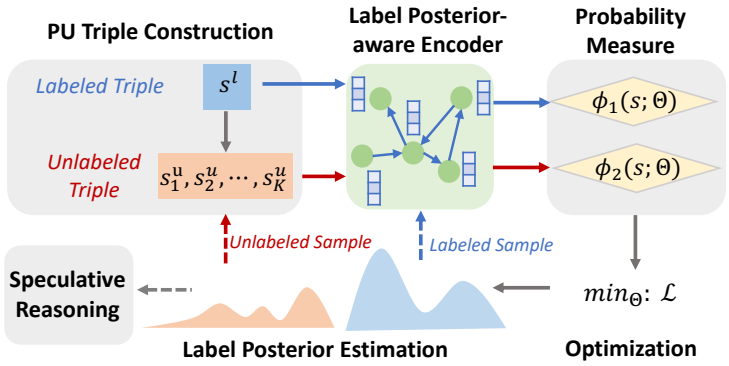


Figure 2.2: nPUGraph overview. It jointly optimizes parameters and estimates label posterior, to detect false negative/positive cases for the encoder and self-training.

infer the label posterior. During the training process, the latest label posterior estimation can be utilized by a label posterior-aware encoder, which improves the quality of representation learning by only integrating information from the entity neighbors induced by true facts. Finally, a simple yet effective self-training strategy based on label posterior is proposed, which can dynamically update neighbor sets for the encoder and sample unlabeled triplets to cover positive samples in the unlabeled set to the greatest extent for model training.

The remaining of this section is structured as follows: Section 2.3.2 first formalizes the learning objective and the variational framework for likelihood maximization. Section 2.3.3 details the label posterior-aware encoder for representation learning, followed by Section 2.3.4 that introduces the self-training strategy.

2.3.2 Noisy PU Learning on graphs

Due to the false negative/positive issues, the correctness of a fact (y) is not solely determined by its presence in a knowledge graph. nPUGraph addresses the issue by treating the underlying label as a latent variable that influences the probability of being collected or not. We, therefore, set maximizing the data collection likelihood as our objective. In such a learning paradigm, the assumptions that collected triples are correct $p(y = 1|s^l) = 1$ and uncollected triples are incorrect $p(y = 0|s^u) = 1$ are removed. We aim to train a model on labeled triples $\mathcal{S}^L$ and unlabeled ones $\mathcal{S}^U$, and infer the label posterior $p(y|s^u)$ and $p(y|s^l)$ at the same time by data likelihood maximization. The latest label posterior can help to detect false negative/positive cases during model training.

We first derive our training objective. To be more formal, the log-likelihood of each potential fact being collected or not is lower bounded by Eq. (2.3), which is given by Theorem 2.1.

Theorem 2.1. The log-likelihood of the complete data $\log p(\mathbf{S})$ is lower bounded as follows:

$$\begin{aligned}
&\log p(\mathbf{S}) \geq \mathop{\mathbb{E}}_{q(\mathbf{Y})} \left[\log p(\mathbf{S}|\mathbf{Y})\right] - \mathbb{KL}(q(\mathbf{Y})\|p(\mathbf{Y})) \\
&= \mathop{\mathbb{E}}_{s^l \in \mathcal{S}^L} \left[w^l \log[\phi_1^l(s^l)] + (1-w^l)\log[\phi_0^l(s^l)]\right] \\
&+ \mathop{\mathbb{E}}_{s^u \in \mathcal{S}^U} \left[(w^u \log[\phi_1^u(s^u)] + (1-w^u)\log[\phi_0^u(s^u)]\right] \\
&- \mathbb{KL}(\mathbf{W}^U\|\tilde{\mathbf{W}}^U) - \mathbb{KL}(\mathbf{W}^L\|\tilde{\mathbf{W}}^L) - \frac{\|\mathbf{W}^L\|_1}{|\mathcal{S}^L|} - \frac{\|\mathbf{W}^U\|_1}{|\mathcal{S}^U|},
\end{aligned} \tag{2.3}$$

where $\mathbf{S}$ denotes all labeled/unlabeled triples, $\mathbf{Y}$ is the corresponding latent variable indicating the positive/negative labels for triples, $\mathbf{W}^U = \{w_i^u\}$ denotes the point-wise probability for the uncollected triples being positive, $\mathbf{W}^L = \{w_i^l\}$ denotes the probability for the collected triples being positive. $\tilde{\mathbf{W}}^U$ and $\tilde{\mathbf{W}}^U$ are the approximation of the collection probability for uncollected/collected triples respectively, produced by nPUGraph based on the latest parameters.

Proof. Let $\log p(\mathbf{S})$ denote the log-likelihood of all potential triples being collected in the graph or not, $\mathbf{Y}$ denote the corresponding latent variable indicating the positive/negative labels. We aim to infer the label posterior $p(\mathbf{Y}|\mathbf{S})$, which can be approximated by $q(\mathbf{Y}|\mathbf{S})$. We therefore are interested at the difference between the two, measured by the Kullback–Leibler (KL) divergence as follows:

$$\mathbb{KL}(q(\mathbf{Y}|\mathbf{S})\|p(\mathbf{Y}|\mathbf{S})) = -\mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})} \log\left(\frac{p(\mathbf{S}|\mathbf{Y})p(\mathbf{Y})}{q(\mathbf{Y}|\mathbf{S})}\right) + \log p(\mathbf{S}), \tag{2.4}$$

as KL divergence is positive, we derive the lower bound of the log-likelihood as follows:

$$\begin{aligned}
&\log p(\mathbf{S}) \geq \mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})} \log\left(\frac{p(\mathbf{S}|\mathbf{Y})p(\mathbf{Y})}{q(\mathbf{Y}|\mathbf{S})}\right) \\
&\geq \mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})} \log p(\mathbf{S}|\mathbf{Y}) - \mathbb{KL}(q(\mathbf{Y}|\mathbf{S})\|p(\mathbf{Y})) - \mathop{\mathbb{E}}_{p(\mathbf{S})} q(\mathbf{Y}|\mathbf{S}),
\end{aligned} \tag{2.5}$$

which consists of three terms: triple collection probability measure, KL term and regularization of label posterior (positive). We discuss each term in detail.

Recall that the distribution of labeled triples can be represented as follows:

$$s^l \sim \beta\phi_1^l(s^l) + (1-\beta)\phi_0^l(s^l), \tag{2.6}$$

where ϕ_y^l denotes the probability of being collected over triple space $\mathcal{S}$ for the positive class ($y = 1$) and negative class ($y = 0$), and $\beta \in [0, 1)$ denotes the proportion of true positive samples in labeled data. Similarly, considering the existence of unlabeled positive triples, the distribution of unlabeled samples can be represented as follows:

$$s^u \sim \alpha\phi_1^u(s^u) + (1-\alpha)\phi_0^u(s^u), \tag{2.7}$$

where $\phi_y^u = 1 - \phi_y^l$ denotes the probability of being uncollected, $\alpha \in [0, 1)$ is the class prior or the proportion of positive samples in unlabeled data. Based on that, the first term can be detailed as follows:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{q(\mathbf{Y}|\mathbf{S})} \log p(\mathbf{S}|\mathbf{Y}) = \mathop{\mathbb{E}}_{s^l \in \mathcal{S}^L} \mathop{\mathbb{E}}_{y \in \{0,1\}} q(y|s^l) \log p(s^l|y) \\
&+ \mathop{\mathbb{E}}_{s^u \in \mathcal{S}^U} \mathop{\mathbb{E}}_{y \in \{0,1\}} q(y|s^l) \log p(s^l|y) \\
&= \mathop{\mathbb{E}}_{s^l \in \mathcal{S}^L} \left[q(y=1|s^l) \log[p(s^l|y=1))] + q(y=0|s^l) \log[p(s^l|y=0))] \right] \\
&+ \mathop{\mathbb{E}}_{s^u \in \mathcal{S}^U} \left[q(y=1|s^u) \log[p(s^u|y=1))] + q(y=0|s^u) \log[p(s^u|y=0))] \right] \\
&= \mathop{\mathbb{E}}_{s^l \in \mathcal{S}^L} \left[w^l \log[\phi_1^l(s^l)] + (1-w^l) \log[\phi_0^l(s^l)] \right] \\
&+ \mathop{\mathbb{E}}_{s^u \in \mathcal{S}^U} \left[(w^u \log[\phi_1^u(s^u)] + (1-w^u) \log[\phi_0^u(s^u)] \right],
\end{aligned} \tag{2.8}$$

where $\mathbf{W}^U = \{w^u\}$ denotes the point-wise probability for the uncollected triples being positive $q(y = 1|s^u)$, $\mathbf{W}^L = \{w^l\}$ denotes the probability for the collected triples being positive $q(y = 1|s^l)$.

We view $\mathbf{W}^U$ and $\mathbf{W}^L$ as free parameters and regularize them by $\tilde{\mathbf{W}}^U$ and $\tilde{\mathbf{W}}^L$, which are estimated posterior probability as follows:

$$\tilde{w}_i^l = \frac{\beta\phi_1^l(s_i^l)}{\beta\phi_1^l(s_i^l) + (1-\beta)\phi_0^l(s_i^l)}, \tag{2.9}$$

$$\tilde{w}_{ik}^u = \frac{\alpha\phi_1^u(s_{ik}^u)}{\alpha\phi_1^u(s_{ik}^u) + (1-\alpha)\phi_0^u(s_{ik}^u)}. \tag{2.10}$$

Therefore, the KL term in Eq. (2.5) becomes $\mathbb{KL}(\mathbf{W}^U \| \tilde{\mathbf{W}}^U) + \mathbb{KL}(\mathbf{W}^L \| \tilde{\mathbf{W}}^L)$.

Last but not least, the third term $\mathop{\mathbb{E}}_{p(\mathbf{S})} q(\mathbf{Y}|\mathbf{S}) = \|\mathbf{W}^U\|_1/|\mathcal{S}^U| + \|\mathbf{W}^L\|_1/|\mathcal{S}^L|$ regulates the total number of potential triples (including both collected ones and uncollected ones), because of the sparsity nature of graphs. Finally, we derive the lower bound of the log-likelihood, as

shown in Eq (2.3). QED.

We treat label $\mathbf{Y}$ as a latent variable and derive the lower bound for the log-likelihood, which is influenced by the prior knowledge of positive class prior α and true positive ratio β. Thus, maximizing the lower bound can jointly optimize model parameters and infer the posterior label distribution, $\mathbf{W}^U$ and $\mathbf{W}^L$. Such a learning process enables us to avoid false negative/positive issues during model training since it considers ϕ_0^l (one negative triple is collected) and ϕ_1^u (one positive triple is missing) as non-zero probability, which are determined by the latest label posterior during model training.

Probability Measure. We then specify the probability measures for positive/negative triples being collected, i.e., $\phi_1^l(\cdot)$ and $\phi_0^l(\cdot)$ ($\phi_y^u(\cdot) = 1 - \phi_y^l(\cdot)$ for $y = 1/0$). To better connect to other methods utilizing score functions for graph reasoning, we hereby utilize the sigmoid function $\sigma(\cdot)$ to directly transform the score function $\psi(s;\Theta)$, parameterized by model parameters Θ, to probability:

$$\phi_1^l(s) = \sigma(\psi_1(s;\Theta)), \quad \phi_0^l(s) = \sigma(\psi_0(s;\Theta)), \tag{2.11}$$

we hereby utilize two score functions $\psi_1(s;\Theta)$ and $\psi_0(s;\Theta)$ to measure the positive/negative triples being collected, as the influencing factors based on triple information can be different. We utilize two neural networks to approximate the probability measure, which will be detailed in Section 2.3.3.

Since we aim to detect the potential existence of positive triples in an unlabeled set, it is unnecessary to push the collection probability of all uncollected triple $\phi_y^l(s^u)$ to 0 ($\phi_y^u(s^u)$ to 1). A loose constraint is that we force the uncollection probability of a collected triple s^l lower than its corresponding uncollected triples s^u: $\phi_y^u(s^l) < \phi_y^u(s^u)$. Therefore, we adopt the pair-wise ranking measure $\phi_y^\star(s^l, s^u)$ to replace $\phi_y^u(s^u)$ as follows:

$$\phi_y^u(s^u) \to \phi_y^\star(s^l, s^u) = \sigma(\psi_y(s^u;\Theta) - \psi_y(s^l;\Theta)). \tag{2.12}$$

Maximum Probability Training. We then derive the training objective based on Eq. (2.3) The first part of Eq. (2.3) measures the probability of data being collected/uncollected. Concretely, given each collected triple $s_i^l \in \mathcal{S}^L$ and its corresponding K uncollected triples

$s^u_{ik} \in \mathcal{S}^U$, We denote the loss function measuring the probability as $\mathcal{L}_{triple}$:

$$\begin{aligned}\mathcal{L}_{triple} = -\frac{1}{K|\mathcal{S}^L|}\sum_i\sum_k (w^l_i \log[\phi^l_1(s^l_i)] \\ + (1-w^l_i)\log[\phi^l_0(s^l_i)] + w^u_{ik}\log[\phi^\star_1(s^l_i, s^u_{ik})] \\ + (1-w^u_{ik})\log[\phi^\star_0(s^l_i, s^u_{ik})]),\end{aligned} \tag{2.13}$$

where w^l_i denotes the point-wise probability for the collected triple s^l_i being positive, w^u_{ik} denotes the probability for the uncollected triple s^u_{ik} being positive. Based on the definition, the posterior probability of each collected/uncollected triple being positive can be computed as:

$$\tilde{w}^l_i = \frac{\beta\phi^l_1(s^l_i)}{\beta\phi^l_1(s^l_i) + (1-\beta)\phi^l_0(s^l_i)}, \tag{2.14}$$

$$\tilde{w}^u_{ik} = \frac{\alpha\phi^u_1(s^u_{ik})}{\alpha\phi^u_1(s^u_{ik}) + (1-\alpha)\phi^u_0(s^u_{ik})}. \tag{2.15}$$

To increase model expression ability, instead of forcing $\mathbf{W}^L = \tilde{\mathbf{W}}^L$ and $\mathbf{W}^U = \tilde{\mathbf{W}}^U$, we set $\mathbf{W}^L$ and $\mathbf{W}^U$ as free parameters and utilize the term $\mathcal{L}_{KL} = \mathbb{KL}(\mathbf{W}^L \| \tilde{\mathbf{W}}^L) + \mathbb{KL}(\mathbf{W}^U \| \tilde{\mathbf{W}}^U)$ to regularize the difference. Finally, based on Eq. (2.3), the training objective is formalized as follows:

$$\min_\Theta \mathcal{L} = \min_\Theta \mathcal{L}_{triple} + \mathcal{L}_{KL} + \mathcal{L}_{reg}, \tag{2.16}$$

where $\mathcal{L}_{reg} = \|\mathbf{W}^L\|_1 + \|\mathbf{W}^U\|_1$ can be viewed as a normalization term. Considering the sparsity property of real-world graphs, $\mathcal{L}_{reg}$ penalizes the posterior estimation that there are too many true positive facts on the graph.

2.3.3 Label Posterior-aware Encoder

We then introduce the encoder and the score functions $\psi_1(s;\Theta)$ and $\psi_0(s;\Theta)$ to measure the probability of positive/negative triples being collected, as shown in Figure 2.3. Recent work [18]–[21] has shown that integrating information from neighbors to represent entities engenders better reasoning performance. However, the message-passing mechanism is vulnerable to the false positive issue, as noise can be integrated via a link induced by a false positive fact. In light of this, we propose a label posterior-aware encoder to improve the quality of representations.

We represent each entity $e \in \mathcal{E}$ and each relation $r \in \mathcal{R}$ into a d-dimensional latent space:

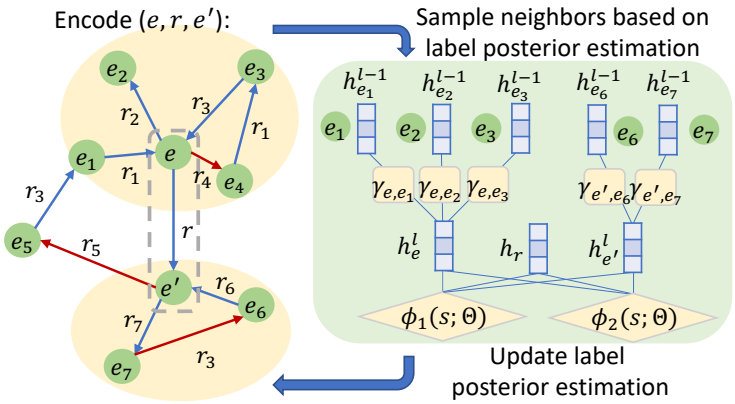


Figure 2.3: The label posterior-aware encoder. Red line denotes the detected false positive facts based on posterior, which are excluded during neighbor sampling.

$\mathbf{h}_e, \mathbf{h}_r \in \mathbb{R}^d$. To encode more information in $\mathbf{h}_e$, we first construct a neighbor set $\mathcal{N}_e$ induced by the positive facts related to entity e. The latest label posterior for collected facts $\tilde{\mathbf{W}}^L$ naturally serves this purpose, as it indicates the underlying correctness for each collected fact.

Therefore, for each entity e, we first sort the related facts by label posterior $\tilde{\mathbf{W}}^L$ and construct the neighbor set $\mathcal{N}_e(\tilde{\mathbf{W}}^L) = \{(e_i, r_i)\}$ from the top facts. Then the encoder attentively aggregates information from the collected neighbors, where the attention weights take neighbor features, relation features into account. Specifically:

$$\mathbf{h}_e^l = \mathbf{h}_e^{l-1} + \sigma\left(\sum_{(e_i,r_i)\in\mathcal{N}_e(\tilde{\mathbf{W}}^L)} \gamma_{e,e_i}^l \left(\mathbf{h}_{e_i}^{l-1}\mathbf{M}\right)\right), \tag{2.17}$$

where l denotes the layer number, $\sigma(\cdot)$ denotes the activation function, γ_{e,e_i}^l denotes the attention weight of entity e_i to the represented entity e, and $\mathbf{M}$ is the trainable transformation matrix. The attention weight γ_{e,e_i}^l is supposed to be aware of entity feature and topology feature induced by relations. We design the attention weight γ_{e,e_i}^l as follows:

$$\gamma_{e,e_i}^l = \frac{\exp(q_{e,e_i}^l)}{\sum_{\mathcal{N}_e(\tilde{\mathbf{W}}^L)} \exp(q_{e,e_k}^l)}, \quad q_{e,e_k}^l = \mathbf{a}\left(\mathbf{h}_e^{l-1} \| \mathbf{h}_{e_k}^{l-1} \| \mathbf{h}_{r_k}\right), \tag{2.18}$$

where q_{e,e_k}^l measures the pairwise importance from neighbor e_k by considering the entity embedding, neighbor embedding, and relation embedding, $\mathbf{a} \in \mathbb{R}^{3d}$ is a shared parameter in the attention.

To measure the collection probability for positive/negative triples, we utilize two multilayer

perception (MLP) to approximate score function $\psi_1(s;\Theta)$ and $\psi_0(s;\Theta)$. Specifically, for each triple $s = (e_h, r, e_t)$:

$$\psi_1(s;\Theta) = \text{MLP}_1(\mathbf{h}_s),\ \psi_0(s;\Theta) = \text{MLP}_0(\mathbf{h}_s), \tag{2.19}$$

where the MLP input $\mathbf{h}_s = [\mathbf{h}^l_{e_h} \| \mathbf{h}_r \| \mathbf{h}^l_{e_t}]$ concatenates entity embeddings and relation embedding.

2.3.4 Self-Training Strategy

The latest label posterior $\tilde{\mathbf{W}}^L$ and $\tilde{\mathbf{W}}^U$ is further utilized in a self-training strategy to enhance speculative reasoning. First, the latest posterior estimation $\tilde{\mathbf{W}}^L$ for collected links updates neighbor sets to gradually prevent the encoder effects by false positive links. Moreover, the latest estimation $\tilde{\mathbf{W}}^U$ for uncollected facts enables us to continuously sample unlabeled triplets with high label posterior to cover positive samples in the unlabeled set to the greatest extent. For each labeled triple $s^l_i = (e_h, r, e_t)$, we construct K unlabeled triples s^u_{ik} by replacing the head and tail respectively with other entities: $s^u_{ik} = (e_h, r, e^-_k)$ or (e^-_k, r, e_t), where e^-_k is the selected entity that ensures $s^u_{ik} \notin \mathcal{S}^L$. Such selection is performed by ranking the corresponding label posterior $\tilde{w}^u_{ik}$. The updates of neighbor sets and unlabeled triples based on label posterior are nested with parameter optimization during model training alternatively. The training of nPUGraph is summarized in Algorithm 2.1.

2.4 THE EVALUATION OF NPUGRAPH

2.4.1 Dataset

Dataset Information. We evaluate our proposed model based on three widely used knowledge graphs and one Twitter ineraction graph:

- **FB15K** [9] is a subset of Freebase [27], a large database containing general knowledge facts with a variety of relation types;
- **FB15K-237** [28] is a reduced version of FB15K, where inverse relations are removed;
- **WN18** [9] is a subset of WorldNet [29], a massive lexical English database that captures semantic relations between words;
- **Twitter** is an interaction graph relevant with *Russo-Ukrainian War*. Data is collected in the Twitter platform from May 1, 2022, to December 25, 2022, which records user-tweet

Algorithm 2.1: Summary of nPUGraph.

Input: The collected triple set $\mathcal{S}^L$.
Output: The model parameter Θ, predicted triples.

Construct the uncollected triple set $\mathcal{S}^U$ randomly;
Initialize the model parameter Θ and the label posterior $\tilde{\mathbf{W}}^L$ and $\tilde{\mathbf{W}}^U$ randomly;
for *each training epoch* **do**
    Construct neighbor set $\mathcal{N}_e(\tilde{\mathbf{W}}^L)$ by $\tilde{\mathbf{W}}^L$;
    Construct uncollected triple set $\mathcal{S}^U$ by $\tilde{\mathbf{W}}^U$;
    for *each collected triple* $s_i^l \in \mathcal{S}^L$ **do**
        Collect unlabeled triples $\{s_{ik}^u\}_{k=1}^K$;
        Calculate $\phi_y^l(s_i^l)$ by Eq. (2.11);
        Calculate each $\phi_y^\star(s_i^l, s_{ik}^u)$ by Eq. (2.12);
    end
    Calculate the total loss $\mathcal{L}$ by Eq. (3.17);
    Optimize model parameter: $\Theta = \Theta - \frac{\partial \mathbf{L}}{\partial \Theta}$;
    Update label posterior $\tilde{\mathbf{W}}^L$ and $\tilde{\mathbf{W}}^U$;
end

interactions and user-hashtag interactions. Thus, the graph is formed by two relations (user-tweet and user-hashtag) and multiple entities, which can be categorized into three types (user, tweet, and hashtag). When constructing the graph, thresholds will be selected to remove inactive users, tweets, and hashtags according to their occurrence frequency. We set the thresholds for the user, tweet, and hashtag as 30, 30, and 10, respectively, i.e., if a tweet has fewer than 30 interactions with users, it will be regarded as inactive and removed from the graph. Besides, the extremely frequent users and tweets are deleted as they may be generated by bots.

For all datasets, we first merge the training set and validation set as a whole. Then we simulate noisy and incomplete graphs for the training process by adding various proportions of false negative/positive cases in the merged set. After that, we partition the simulated graphs into new train/valid sets with a ratio of 7 : 3 and the test set remains the same. Table 2.1 provides an overview of the statistics of the simulated datasets corresponding to various perturbation rates and based graphs.

Dataset Perturbation. Data perturbation aims to simulate noisy and incomplete graphs from clean benchmark knowledge graphs. It consists of two aspects: First, to simulate the *false negative issue*, it randomly removes some existing links in a graph, considering the removed links as missing but potentially plausible facts. Second, to simulate the *false positive issue*, it randomly adds spurious links to the graph as unreliable or outdated facts.

We define perturbation rate, i.e., *ptb_rate* , as a proportion of modified edges in a graph

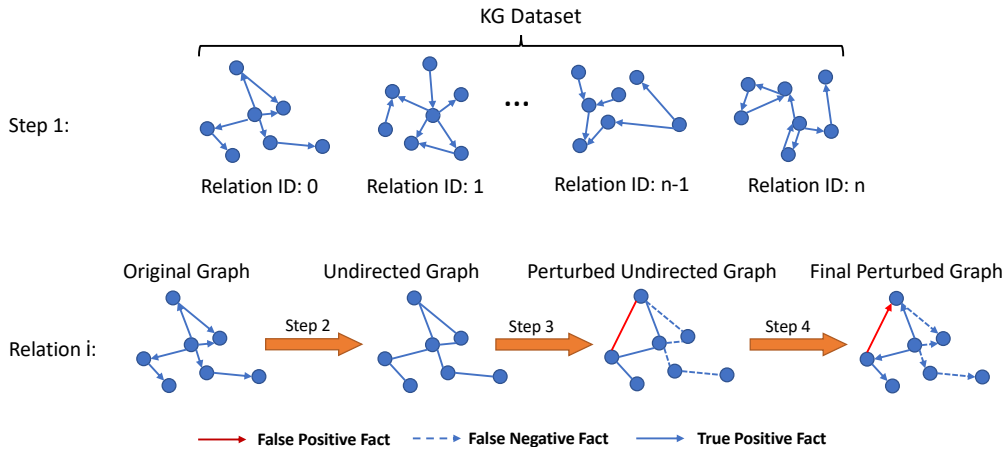


Figure 2.4: Summary of graph perturbation to construct noisy and incomplete KGs.

Table 2.1: The statistics of the datasets.

ptb_rate	Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#Train	#Valid	#Test
0.1	**FB15K**	14,951	1,345	340,968	146,129	59,071
	FB15K-237	14,541	237	184,803	79,201	20,466
	WN18	40,943	18	92,428	39,612	5,000
	Twitter	17,839	2	282,233	120,956	110,456
0.3	**FB15K**	14,951	1,345	276,940	118,688	59,071
	FB15K-237	14,541	237	149,229	63,954	20,466
	WN18	40,943	18	72,462	31,055	5,000
	Twitter	17,839	2	232,748	99,749	110,456
0.5	**FB15K**	14,951	1,345	213,380	91,448	59,071
	FB15K-237	14,541	237	113,772	48,759	20,466
	WN18	40,943	18	52,707	22,588	5,000
	Twitter	17,839	2	183,263	78,540	110,456
0.7	**FB15K**	14,951	1,345	150,485	64,493	59,071
	FB15K-237	14,541	237	78,531	33,656	20,466
	WN18	40,943	18	34,984	14,993	5,000
	Twitter	17,839	2	133,778	57,333	110,456

to control the amount of removing positive links and adding negative links. For example, if a graph has 100 links and the perturbation rate is 0.5, then we will randomly convert the positivity or negativity of 50 links. Among these 50 modified links, 10% of them will be added and the rest of them will be removed, i.e., we will randomly add 5 negative links and remove 45 positive links to generate a perturbed graph. The perturbed graph can be regarded as noisy and incomplete, leading to significant performance degradation. In our experiments, we set *ptb_rate* in a range of $\{0.1, 0.3, 0.5, 0.7\}$. The detailed perturbation process is summarized in Figure 2.4.

Table 2.2: Overall performance on noisy and incomplete graphs, with *ptb_rate = 0.3*. Average results on 5 independent runs are reported. $*$ indicates the statistically significant results over baselines, with p-value < 0.01. The best results are in boldface, and the strongest baseline performance is underlined.

Dataset	FB15K				FB15K-237				WN18			
Metrics	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1
Knowledge graph embedding methods												
TransE	0.336	0.603	0.425	0.189	0.196	0.394	0.236	0.094	0.229	0.481	0.416	0.030
TransR	0.314	0.579	0.397	0.170	0.184	0.359	0.211	0.098	0.229	0.480	0.408	0.035
DistMult	0.408	0.627	0.463	0.296	0.240	0.407	0.262	0.158	0.397	0.518	0.453	0.320
ComplEx	0.396	0.616	0.451	0.284	0.238	0.411	0.262	0.154	0.448	0.526	0.475	0.403
RotatE	0.431	0.636	0.489	0.323	0.255	0.433	0.280	0.169	0.446	0.524	0.474	0.400
Graph neural network methods on KG												
RGCN	0.154	0.307	0.164	0.078	0.141	0.276	0.145	0.075	0.362	0.464	0.412	0.300
CompGCN	0.409	0.631	0.465	0.294	0.253	0.422	0.275	0.171	0.445	0.522	0.471	0.400
Uncertain knowledge graph embedding method												
UKGE	0.311	0.556	0.337	0.189	0.172	0.233	0.128	0.081	0.241	0.447	0.309	0.119
Negative sampling methods												
NSCaching	0.371	0.576	0.424	0.265	0.190	0.329	0.208	0.121	0.306	0.401	0.334	0.255
SANS	0.372	0.599	0.434	0.252	0.243	0.416	0.267	0.158	0.453	0.528	0.479	0.409
Positive-Unlabeled learning methods on KG												
PUDA	0.403	0.623	0.458	0.291	0.234	0.394	0.255	0.156	0.382	0.499	0.444	0.306
nPUGraph	**0.486***	**0.718***	**0.534***	**0.342***	**0.287***	**0.481***	**0.315***	**0.191***	**0.493***	**0.582***	**0.519***	**0.442***
Gains (%)	*12.7*	*12.8*	*9.2*	*5.9*	*12.6*	*11.2*	*12.5*	*11.4*	*8.9*	*10.3*	*8.3*	*8.0*

2.4.2 Experimental Setup

Baselines. We compare to eleven state-of-the-art baselines: 1) KG embedding methods: **TransE** [9], **TransR** [14], **DistMult** [15], **ComplEX** [16], and **RotatE** [17]; 2) GNN methods on KG: **RGCN** [18] and **CompGCN** [21]; 3) Uncertain KG reasoning: **UKGE** [23]; 4) Negative sampling methods: **NSCaching** [30] and **SANS** [31]; 5) PU learning on KG: **PUDA** [12]. We describe the baseline models utilized in the experiments in detail:

- **TransE**[2] [9] is a translation-based embedding model, where both entities and relations are represented as vectors in the latent space. The relation is utilized as a translation operation between the subject and the object entity;
- **TransR** [14] advances TransE by optimizing modeling of n-n relations, where each entity embedding can be projected to hyperplanes defined by relations;
- **DistMult**[3] [15] is a general framework with the bilinear objective for multi-relational learning that unifies most multi-relational embedding models;
- **ComplEx** [16] introduces complex embeddings, which can effectively capture asymmetric relations while retaining the efficiency benefits of the dot product;

[2]The experiments of TransE and TransR are implemented with https://github.com/thunlp/OpenKE.

[3]The experiments of DistMult, ComplEx, and RotatE are implemented with https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding.

Table 2.3: Overall performance on noisy and incomplete graphs, with *ptb_rate = 0.1*. Average results on 5 independent runs are reported. $*$ indicates the statistically significant results over baselines, with p-value < 0.01. The best results are in boldface, and the strongest baseline performance is underlined.

Dataset	FB15K				FB15K-237				WN18			
Metrics	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1
Knowledge graph embedding methods												
TransE	0.419	0.666	0.507	0.280	0.245	0.441	0.284	0.144	0.318	0.646	0.579	0.044
TransR	0.398	0.658	0.494	0.250	0.246	0.434	0.280	0.153	0.319	0.646	0.575	0.051
DistMult	0.516	0.719	0.578	0.407	0.271	0.446	0.297	0.185	0.524	0.686	0.610	0.418
ComplEx	0.503	0.711	0.570	0.390	0.276	0.459	0.305	0.186	0.612	0.702	0.643	0.560
RotatE	0.544	0.732	0.608	0.441	0.292	0.480	0.324	0.199	0.613	0.691	0.642	0.567
Graph neural network methods on KG												
RGCN	0.196	0.372	0.209	0.110	0.169	0.317	0.177	0.097	0.483	0.625	0.554	0.396
CompGCN	0.460	0.677	0.525	0.343	0.293	0.475	0.324	0.203	0.608	0.686	0.636	0.564
Uncertain knowledge graph embedding method												
UKGE	0.338	0.425	0.321	0.233	0.231	0.411	0.204	0.110	0.381	0.541	0.407	0.331
Negative sampling method												
NSCaching	0.495	0.689	0.557	0.390	0.153	0.305	0.167	0.080	0.434	0.542	0.470	0.374
SANS	0.422	0.649	0.493	0.298	0.271	0.453	0.301	0.182	0.619	0.702	0.644	0.574
Positive-Unlabeled learning method												
PUDA	0.493	0.713	0.559	0.377	0.271	0.443	0.298	0.185	0.520	0.667	0.608	0.419
nPUGraph	**0.561***	**0.791***	**0.621***	**0.449***	**0.328***	**0.535***	**0.343***	**0.221***	**0.630***	**0.754***	**0.671***	**0.599***
Gains (%)	*3.0*	*8.1*	*2.1*	*1.9*	*12.0*	*11.4*	*5.9*	*8.7*	*1.8*	*7.4*	*4.1*	*4.4*

- **RotatE** [17] extends ComplEx by representing entities as complex vectors and relations as rotation operations in a complex vector space;
- **R-GCN**[4] [18] uses relation-specific weight matrices that are defined as linear combinations of a set of basis matrices;
- **CompGCN**[5] [21] is a framework for incorporating multi-relational information in graph convolutional networks to jointly embeds both nodes and relations in a graph;
- **UKGE**[6] [23] learns embeddings according to the confidence scores of uncertain relation facts to preserve both structural and uncertainty information of facts in the embedding space;
- **NSCaching**[7] [30] is an inexpensive negative sampling approach by using cache to keep track of high-quality negative triplets, which have high scores and rare;
- **SANS**[8] [31] utilizes the rich graph structure by selecting negative samples from a node's k-hop neighborhood for negative sampling without additional parameters and difficult adversarial optimization;

[4] https://github.com/JinheonBaek/RGCN
[5] https://github.com/malllabiisc/CompGCN
[6] https://github.com/stasl0217/UKGE
[7] https://github.com/AutoML-Research/NSCaching
[8] https://github.com/kahrabian/SANS

Table 2.4: Experimental results under *ptb_rate = 0.5*.

Dataset	FB15K				FB15K-237				WN18			
Metrics	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**
Knowledge graph embedding methods												
TransE	0.279	0.540	0.363	0.136	0.151	0.342	0.190	0.053	0.158	0.337	0.285	0.018
TransR	0.256	0.500	0.323	0.127	0.141	0.294	0.160	0.064	0.150	0.327	0.267	0.020
DistMult	0.328	0.536	0.372	0.224	0.210	0.366	0.228	0.133	0.279	0.370	0.319	0.224
ComplEx	0.322	0.525	0.365	0.220	0.201	0.358	0.220	0.123	0.314	0.373	0.335	0.280
RotatE	0.350	0.547	0.398	0.249	0.227	0.387	0.246	0.149	0.307	0.377	0.333	0.266
Graph neural network methods on KG												
RGCN	0.134	0.271	0.142	0.065	0.116	0.234	0.117	0.058	0.253	0.323	0.287	0.209
CompGCN	0.378	0.600	0.429	0.266	0.223	0.377	0.240	0.149	0.345	0.315	0.336	0.279
Uncertain knowledge graph embedding method												
UKGE	0.257	0.299	0.213	0.088	0.143	0.284	0.172	0.053	0.228	0.297	0.210	0.115
Negative sampling method												
NSCaching	0.272	0.454	0.310	0.179	0.176	0.297	0.190	0.115	0.123	0.182	0.133	0.093
SANS	0.335	0.545	0.387	0.224	0.225	0.382	0.244	0.147	0.313	0.379	0.337	0.275
Positive-Unlabeled learning method												
PUDA	0.329	0.525	0.369	0.229	0.202	0.347	0.217	0.131	0.231	0.329	0.264	0.178
nPUGraph	**0.417***	**0.663***	**0.470***	**0.291***	**0.258***	**0.433***	**0.285***	**0.171***	**0.373***	**0.443***	**0.379***	**0.327***
Gains (%)	*10.4*	*10.5*	*9.6*	*9.6*	*13.9*	*12.0*	*16.1*	*14.8*	*8.2*	*16.9*	*12.6*	*16.9*

Table 2.5: Experimental results under *ptb_rate = 0.7*.

Dataset	FB15K				FB15K-237				WN18			
Metrics	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**	**MRR**	**H@10**	**H@3**	**H@1**
Knowledge graph embedding methods												
TransE	0.219	0.457	0.294	0.087	0.104	0.273	0.128	0.020	0.114	0.229	0.199	0.020
TransR	0.195	0.396	0.248	0.087	0.096	0.217	0.108	0.036	0.096	0.207	0.167	0.016
DistMult	0.250	0.425	0.282	0.163	0.173	0.308	0.186	0.105	0.181	0.240	0.211	0.143
ComplEx	0.241	0.408	0.271	0.157	0.158	0.290	0.170	0.092	0.202	0.243	0.219	0.178
RotatE	0.271	0.439	0.309	0.185	0.198	0.337	0.212	0.129	0.192	0.250	0.212	0.159
Graph neural network methods on KG												
RGCN	0.129	0.229	0.134	0.075	0.086	0.178	0.086	0.040	0.166	0.215	0.191	0.135
CompGCN	0.354	0.578	0.402	0.243	0.193	0.325	0.204	0.129	0.212	0.262	0.229	0.183
Uncertain knowledge graph embedding method												
UKGE	0.186	0.358	0.199	0.076	0.133	0.201	0.115	0.075	0.099	0.176	0.153	0.116
Negative sampling method												
NSCaching	0.174	0.309	0.194	0.105	0.157	0.276	0.169	0.099	0.057	0.085	0.062	0.041
SANS	0.292	0.478	0.332	0.196	0.201	0.342	0.216	0.131	0.202	0.254	0.222	0.171
Positive-Unlabeled learning method												
PUDA	0.254	0.421	0.283	0.170	0.165	0.291	0.176	0.104	0.109	0.171	0.127	0.077
nPUGraph	**0.365***	**0.600***	**0.427***	**0.277***	**0.243***	**0.390***	**0.266***	**0.155***	**0.247***	**0.303***	**0.257***	**0.209***
Gains (%)	*3.0*	*3.8*	*6.3*	*13.9*	*20.9*	*14.1*	*23.0*	*18.5*	*16.8*	*15.5*	*12.3*	*14.4*

- **PUDA**[9] [12] is a KGC method to circumvent the impact of the false negative issue by tailoring positive unlabeled risk estimator and address the data sparsity issue by unifying adversarial training and PU learning under the positive-unlabeled minimax game.

Evaluation and Metrics. For each $(e_h, r, e_?)$ or $(e_?, r, e_t)$, we rank all entities at the missing position in triples, and adopt filtered mean reciprocal rank (MRR) and filtered Hits at $\{1, 3, 10\}$ as evaluation metrics [9].

[9] https://github.com/lilv98/PUDA

2.4.3 Reproducibility

Baseline Setup. All baseline models and nPUGraph are trained on the perturbed training set and validated on the perturbed valid set. We utilize *MRR* on the valid set to determine the best models and evaluate them on the clean test set. For uncertain knowledge graph embedding methods UKGE [23], since the required uncertainty scores are unavailable, we set the scores for triples in training set as 1 and 0 otherwise. The predicted uncertainty scores produced by UKGE are utilized to rank the potential triples for ranking evaluation. We train all baseline models and nPUGraph on the same GPUs (GeForce RTX 3090) and CPUs (AMD Ryzen Threadripper 3970X 32-Core Processor).

nPUGraph Setup. For model training, we utilize Adam optimizer and set the maximum number of epochs as 200. Within the first 50 epochs, we disable self-training and focus on learning suboptimal model parameters on noisy and incomplete data. After that, we start the self-training strategy, where the latest label posterior estimation is utilized to sample neighbors for the encoder and select informative unlabeled samples for model training. We set batch size as 256, the dimensions of all embeddings as 128, and the dropout rate as 0.5. For the sake of efficiency, we employ 1 neighborhood aggregation layer in the encoder.

For the setting of hyperparameter, we mainly tune positive class prior α in the range of $\{1e-1, 5e-2, 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5\}$; true positive ratio β in the range of $\{0.3, 0.2, 0.1, 0.005, 0.001\}$; learning rate in the range of $\{0.02, 0.01, 0.005, 0.001, 0.0005\}$; the number of sampled unlabeled triples for each labeled one in the range of $\{50, 40, 30, 20, 10\}$. We will publicly release our code and data upon acceptance.

2.4.4 Main Results

We first discuss the model performance on noisy and incomplete graphs, with *ptb_rate = 0.3*, as shown in Table 2.2. nPUGraph achieves consistently better results than all baseline models, with 10.3% relative improvement on average. Specifically, conventional KGE and GNN-based methods produce unsatisfying performance, as they ignore the false negative/positive issues during model training. In some cases, GNN-based ways are worse, as the message-passing mechanism is more vulnerable to false positive links. As expected, the performance of uncertain knowledge graph embedding model [23] is much worse when there are no available uncertainty scores for model training. SANS and PUDA generate competitive results in some cases, as their negative sampling strategy and PU learning objective can respectively mitigate the false negative/positive issues to some extent. Table 2.2 demonstrates the superiority of nPUGraph which addresses the false negative/positive issues

Table 2.6: Overall performance on noisy and incomplete Twitter data, with *ptb_rate = 0.3*. Average results on 5 independent runs are reported. $*$ indicates the statistically significant results over baselines, with p-value < 0.01. The best results are in boldface, and the strongest baseline performance is underlined.

Dataset	**Twitter**			
Metrics	**MRR**	**HIT@100**	**HIT@50**	**HIT@30**
Random	0.001	0.006	0.003	0.002
Knowledge graph embedding methods				
TransE	0.010	0.091	0.058	0.041
TransR	0.009	0.078	0.048	0.033
DistMult	0.021	0.091	0.065	0.052
ComplEx	0.022	0.089	0.064	0.051
RotatE	0.022	0.1115	0.077	0.059
Graph neural network methods on KG				
RGCN	0.005	0.054	0.029	0.019
CompGCN	0.014	0.089	0.059	0.044
Uncertain knowledge graph embedding method				
UKGE	0.011	0.072	0.053	0.033
Negative sampling methods				
NSCaching	0.012	0.095	0.060	0.043
SANS	0.019	0.104	0.070	0.054
Positive-Unlabeled learning method				
PUDA	0.013	0.082	0.057	0.044
nPUGraph	**0.030***	**0.127***	**0.096***	**0.074***
Gains (%)	*38.2*	*13.9*	*25.3*	*26.5*

simultaneously.

Experimental Results on Twitter Data. We discuss the model performance on noisy and incomplete Twitter data with *ptb_rate = 0.3* in this section, which is shown in Table 2.6. According to the result of Random, we can infer that all relations on Twitter are $n - n$, where relations can be $1 - n$, $n - 1$, and $n - n$ for benchmark KG. Therefore, link prediction is more challenging for Twitter data and we adopt Hits at $30, 50, 100$ as evaluation metrics, instead.

For Twitter data, nPUGraph achieves impressive performance compared with the baseline models, with 25.98% relative improvement on average. The results of Twitter data support the robustness of nPUGraph, which can mitigate false negative/positive issues not only in benchmark KG but also in the real-world social graph.

Experimental Results under Different Perturbation Rates. The experimental results under perturbation rates 0.1, 0.5, and 0.7 are shown in Table 2.3, Table 2.4, and Table 2.5, respectively. nPUGraph outperforms all baseline models for various perturbation rates, demonstrating that nPUGraph can mitigate false negative/positive issues on knowledge graphs with different degrees of noise and incompleteness. Notably, comparing these three tables, the relative improvements are more significant under higher *ptb_rate* in most cases, showing stronger robustness for nPUGraph on graphs with more false negative/positive facts.

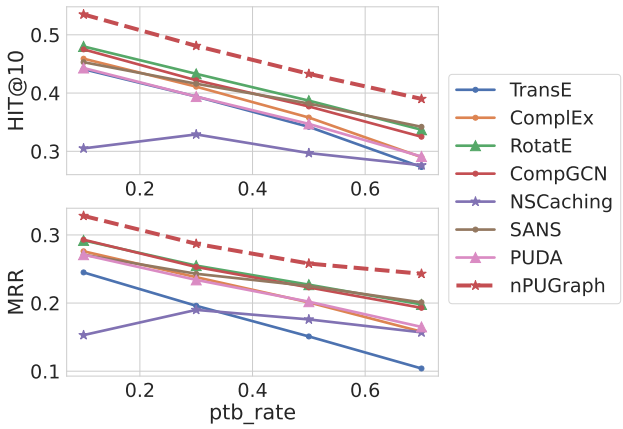


Figure 2.5: Performance with respect to various *ptb_rate* , i.e., different degrees of noise and incompleteness, on *FB15K-237*. nPUGraph exhibits impressive robustness against false negative/positive issues.

Table 2.7: Ablation Studies.

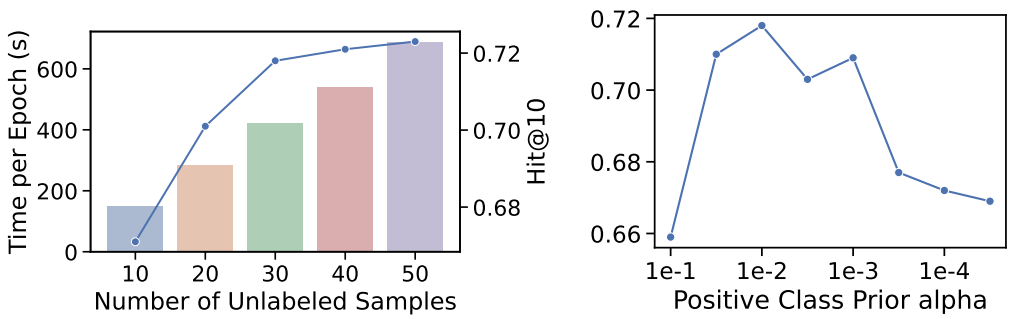
Dataset	FB15K		FB15K-237		Gains
Ablations	MRR	H@10	MRR	H@10	%
nPUGraph w/o nPU	0.401	0.619	0.230	0.407	-20.0
nPUGraph w/o LP-Encoder	0.457	0.681	0.261	0.459	-6.6
nPUGraph w/o Self-Training	0.471	0.704	0.276	0.461	-3.4
nPUGraph	0.486	0.718	0.287	0.481	-

2.4.5 Experiments under Various Degrees of Noise and Incompleteness

We investigate the performance of baseline models and nPUGraph under different degrees of noise and incompleteness. Figure 2.5 reports the performance under various *ptb_rate* , from 0.1 to 0.7, where higher *ptb_rate* means more links are perturbed as false positive/negative cases. The performance degrades as the *ptb_rate* increases for all in most cases, demonstrating that the false negative/positive issues significantly affect the reasoning performance. However, nPUGraph manages to achieve the best performance in all cases. Notably, the relative improvements are more significant under higher *ptb_rate* .

2.4.6 Model Analysis

Ablation Study. We evaluate performance improvements brought by the nPUGraph framework by following ablations: 1) **nPUGraph w/o nPU** is trained without the noisy Positive-Unlabeled framework, which instead utilizes the margin loss for model training; 2) **nPUGraph w/o LP-Encoder** eliminates the label posterior-aware encoder (LP-Encoder), which aggregates information from all neighbors instead of the sampled neighbors; 3) **nPUGraph w/o Self-Training** is trained without the proposed self-training algorithm.



(a) #unlabeled samples.

(b) Class prior α.

Figure 2.6: Model analysis w.r.t. the number of unlabeled samples and positive class prior α.

We report *MRR* and *Hit@10* over FB15K and FB15K-237 data, as shown in Table 2.7. As we can see, training the encoder without the proposed noisy Positive-Unlabeled framework will cause the performance drop, as this variant ignores the false negative/positive issues. Removing the label posterior-based neighbor sampling in the encoder will also cause performance degradation, as the information aggregation no longer distinguishes between true and false links. Such a variant can be easily influenced by the existence of false positive facts. Moreover, the last ablation result shows if the training process is further equipped with the self-training strategy, the performance will be enhanced, which verifies its effectiveness to select informative unlabeled samples for model training.

The Effect of PU Triple Construction. We then investigate the effect of PU Triple Construction on model performance, by varying different sizes of unlabeled samples from 10 to 50. Figure 2.6 shows that the performance improves as the number of unlabeled samples increases. Because more unlabeled samples can cover more false negative cases for model training. The training time grows linearly.

The Effect of Positive Class Prior α. The positive class prior α and true positive ratio β are two important hyperparameters. While β has a clear definition from real-world data, the specific value of α is unknown in advance. Figure 2.6 shows the model performance w.r.t. different values of α by grid search. Reasoning performance fluctuates a bit with different values of α since incorrect prior knowledge of α can bias the label posterior estimation and thus hurt the performance.

2.5 RELATED WORK

2.5.1 Knowledge Graph Reasoning

. Knowledge graph reasoning (KGR) aims to predict missing facts to automatically complete KG, including one-hop reasoning [9] and multi-hop reasoning [13]. It has facilitated a wide spectrum of knowledge-intensive applications [13], [32].To set the scope, we primarily focus on one-hop reasoning and are particularly interested in predicting missing entities in a partial fact. Knowledge graph embeddings achieve state-of-the-art performance [9], [14]–[17], [33]. Recently, graph neural networks (GNN) have been incorporated to enhance representation learning by aggregating neighborhood information [18]–[22], [34]. However, most approaches significantly degrade when KG are largely incomplete and contain certain errors [35], as they ignore the false negative/positive issues. Recent attempts on uncertain KG [23], [24], [36] measure the uncertainty score for facts, which can detect false negative/positive samples. However, they explicitly require the ground truth uncertainty scores for model training, which are usually unavailable in practice. Various negative sampling strategies have been explored to sample informative negative triples to facilitate model training [30], [31], [37]. However, they cannot detect false negative/positive facts. We aim to mitigate the false negative/positive issues and enable the automatic detection of false negative/positive facts during model training.

2.5.2 Positive-Unlabeled Learning

. Positive-Unlabeled (PU) learning is a learning paradigm for training a model that only has access to positive and unlabeled data, where unlabeled data includes both positive and negative samples [25], [38]. PU learning roughly includes (1) two-step solutions [26], [39]; (2) methods that consider the unlabeled samples as negative samples with label noise [40]; (3) unbiased risk estimation methods [12], [25]. Recent work further studies the setting that there exists label noise in the observed positive samples [26]. We formulate the KGR task on noisy and incomplete KG as a noisy Positive-Unlabeled learning problem and propose a variational framework for it, which relates to two-step solutions and unbiased risk estimation methods.

2.6 SUMMARY

We studied speculative reasoning based on sparse and unreliable observations, which contains both *false negative issue* and *false positive issue*. We formulated the task as a noisy Positive-Unlabeled learning problem and proposed a variational framework nPUGraph to jointly update model parameters and estimate the posterior likelihood of collected/uncollected facts being true or false, where the underlying correctness is viewed as latent variables. During the training process, a label posterior-aware encoder and a self-training strategy were proposed to further address the false positive/negative issues. We found label posterior estimation plays an important role in moving toward speculative KG reasoning in reality, and the estimation can be fulfilled by optimizing an alternative objective without additional cost. Extensive experiments verified the effectiveness of nPUGraph on both benchmark KGs and Twitter interaction data with various degrees of data perturbations.

There are certain limitations that can be concerned for further improvements. First, the posterior inference relies on the prior estimation of positive class prior α and true positive ratio β. Our experiments show that a data-driven estimation based on end-to-end model training produces worse results than a hyperparameter grid search. An automatic prior estimation is desirable for real-world applications. Moreover, in nPUGraph, we approximate the probability of negative/positive facts being collected/uncollected via neural networks, which lacks a degree of interpretability. In the future, we plan to utilize a more explainable random process depending on entity/relation features to model the collection probability distribution.

Chapter 3: LEARNING DYNAMIC BIPARTITE GRAPHS

3.1 OVERVIEW

In this chapter, we introduce Dydiff-VAE framework for information diffusion prediction. Dydiff-VAE advances the state of the arts of information diffusion prediction, a topic that receives much recent attention in several contexts, including social recommendations [41]–[43], misinformation detection [44], polarization analysis [6], [7], [45]–[47], user personalization [48], [49], among others. On social media, information is propagated rapidly through users' posting and forwarding behaviors, leading to information cascades consisting of the users who have forwarded the content (we call them *original forwarding users*) as well as the actual *information content*. Given an information cascade, the diffusion model aims to estimate the propagation likelihood for other potential users and predict the user rankings, as shown in Figure 3.1a.

On social media, users usually forward a piece of information out of their interests: On one hand, because of the widespread deployments of recommendation systems and search engines on social platforms, users are more likely to be exposed to and thereby propagate the contents matching their interests. On the other hand, users would also tendentiously pay attention to and get infected by the information from social neighbors, as a result of their shared interests and frequent interactions [50]. Therefore, how to infer user interests lies the foundation for diffusion prediction problems. However, it is non-trivial because user interests cannot be observed directly from their profiles due to various user privacy protection mechanisms, and they are highly dynamic over time. Existing neural diffusion models [51]–[54], despite demonstrating significant improvements over traditional methods built upon pre-defined propagation hypotheses [55]–[58], have two main drawbacks that oversimplify the inference of user interests. Firstly, they model user interests implicitly: they initialize user representations as random variables and optimize them by minimizing the distance between representations of cascades and forwarding users, with the objective that the learned user representations would be good approximations of user interests. However, user interests can be inferred explicitly via the diffusion data, e.g., cascade contents that users has previously forwarded. In other words, instead of building a discriminative model that learns user interests from scratch, we propose to build a generative model where user interests is modeled explicitly and serves as the latent variable that drives the diffusion process forward.

Moreover, existing works assume user representations remain static despite that user interests evolve over time in real world. Such dynamics are mainly driven by two factors:

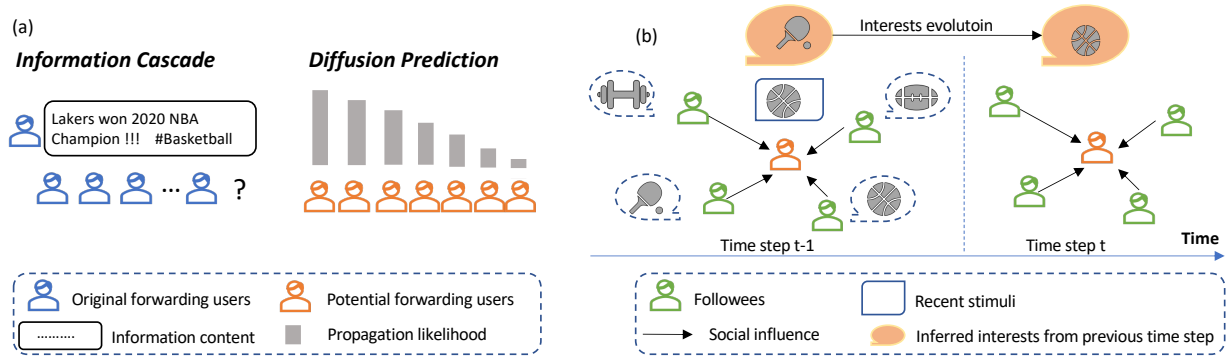


Figure 3.1: (a): An illustration of diffusion prediction task, which estimates the propagation likelihood for potential users and ranks them accordingly. (b): An illustration of dynamic user interests, which evolve over time driven by both social influence and recent stimuli.

(i) *social influence* received from their neighbors on the social networks (e.g., followees on Twitter), as individuals are likely to be influenced by their friends [59]. For instance, as shown in Figure 3.1b, a Twitter user is inferred to be interested in basketball-related topics at time step t because, in part, that one of the followees likes it at time step $t-1$. (ii) *recent stimuli* which refer to information that users gained inside/outside the social media recently. Users can gain new information by reading news, forwarding posts online and etc, which also shapes the evolution of user interests. In Figure 3.1b, another reason why the Twitter user is inferred to be interested in NBA is that (s)he reads some related tweets recently. Modeling the evolution of user interests is critical for predicting propagation behaviors, as the relevance between information cascades and potential users varies as evolution of user interests.

In this paper, we propose a dynamic generative model following the framework of variational autoencoder (VAE): DyDiff-VAE, which advances the existing diffusion models in two directions. First, we design a dynamic encoder to infer latent user interests over time by considering both social influence and recent stimuli as driven factors. We embed the graph convolutional layer [60] aggregating social influence from neighbors into the gated recurrent unit (GRU) [61], which takes recent stimuli as inputs, to learn the recurrent function of interest evolution. Second, we propose a dual attentive decoder to estimate the propagation likelihood and corresponding rankings based on the inferred latent interests. The proposed dual attention mechanism, as a replacement of the recurrent neural networks (RNNs) used in other state-of-the-arts [52], [53], [62], [63], integrates the heterogeneous information from both the user sequence and the cascade content, and overcomes back-propagation through time through efficient parallelizable attentions. Thus, it learns better representations of the information cascades and achieve higher efficiency. We conduct extensive experiments on four real-world datasets collected from the Twitter and Youtube to evaluate the performance of the

proposed DyDiff-VAE. The evaluation results demonstrate that DyDiff-VAE outperforms other compared methods including diffusion models [52]–[54], [62], recommendation models [41] and graph learning methods (dynamic/static) [64]–[66]. It achieves 43.3% relative gains on average over the best baseline on four datasets. And it demonstrates the best efficiency compared to the RNN-based diffusion models.

3.2 PRELIMINARIES

In this section, we introduce the definitions and the problem formulation in this paper. We split time span into discrete time steps, across which user interests are evolving driven by recent stimuli and social influence. According to [67], over 99% information cascade usually lasts shorter than one day, while the user interests evolve much more slowly. Thus, We assume the information cascade happens fully within one time step. We define recent stimuli, social influence, and information cascades as follows:

Definition 3.1 (**Recent Stimuli**)**.** Recent stimuli refer to the cascade contents propagated by users at the last time step. Through the propagation behaviors, users gain stimuli inside the social media to change their interests, which in turn affects their behaviors in the following time steps.

Definition 3.2 (**Social Influence**)**.** Social influence refers to factors describing how users' social neighbors influence their interests through social relations, (e.g., follower-followee relations on Twitter), which are reflected by social networks. We represent the static social network as $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, where $\mathcal{U}$ is the set of users, and $\mathcal{E}$ is the set of asymmetrical social relations among users.

Definition 3.3 (**Information Cascades**)**.** Let $I_j^{(t)}$ denotes the j-th information cascade happening at time step t. It is defined as a tuple of information content c_j and a diffusion sequence of original forwarding users in ascending order of participation time. Thus, $I_j^{(t)} = (c_j, \{u_k\}_{k=1}^K)$, where K denotes the length of the user sequence. We denote the set of information cascade starting at time step t as $\mathbb{I}^{(t)}$.

Based on the definitions above, we formulate the diffusion prediction problem as follows:

Definition 3.4 (**Diffusion Prediction**)**.** Given an information cascade $I_j^{(t)} = (c_j, \{u_k\}_{k=1}^K)$ at time step t, the diffusion prediction aims to estimate the diffusion likelihood $p(u_i|I_j^{(t)})$ for potential user $u_i \in \mathcal{U} \setminus \{u_k\}_{k=1}^K$ by utilizing the information gained from all observed cascades $\bigcup_{n=1}^{n<t} \mathbb{I}^{(n)}$ before t and the social network $\mathcal{G}$. The outcome is the propagation likelihood for the potential forwarding users as well as the corresponding rankings.

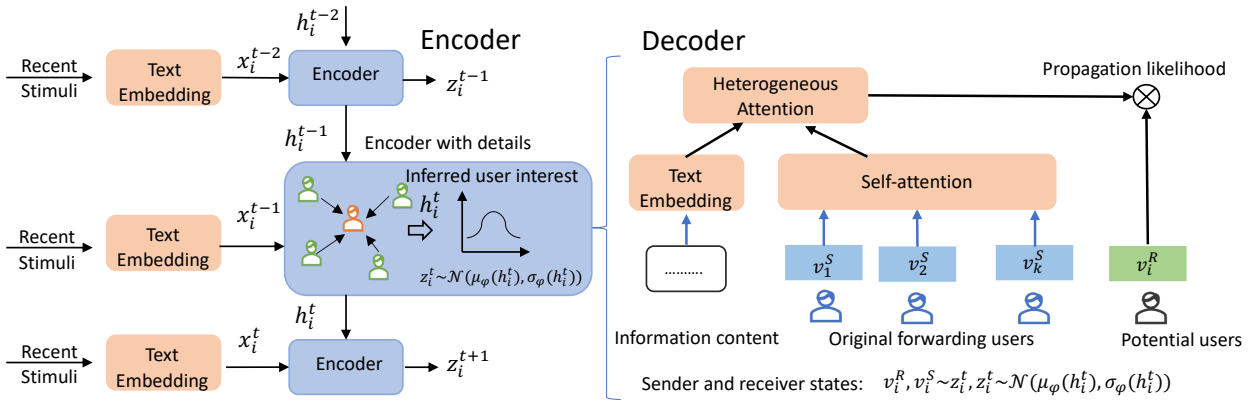


Figure 3.2: Overview of the DyDiff-VAE framework. The dynamic encoder updates the user interests a t new time step t based on the recent stimuli and social influence. The dual attention decoder represents the information cascade based on content and user sequence, and estimates the propagation likelihood.

3.3 THE DESIGN OF DYDIFF-VAE

In this section, we describe the framework of DyDiff-VAE in detail.

3.3.1 Framework Overview

At time step t, DyDiff-VAE estimates the propagation likelihood $p(u_i|I_j^{(t)})$ for the potential users given an information cascade $I_j^{(t)} = (c_j, \{u_k\}_{k=1}^K)$ that consists of the cascade content c_j and original forwarding user sequence $\{u_k\}_{k=1}^K$. The propagation likelihood is determined by the relevance between the information cascade and the potential users. We propose a variational framework for diffusion prediction, with the advance to infer evolving latent user interests $z^{(t)}$ along the time. The framework consists of two components:

1. **the dynamic encoder** (in Section 3.3.2) infers the posterior distribution of latent user interests $q_\phi(z^{(t)}|\cup_{n=1}^{n<t}\mathbb{I}^{(n)})$ at each time step by modeling recent stimuli and social influence from the historic information cascades;

2. **the dual attention decoder** (in Section 3.3.3) estimates the propagation likelihood based on the inferred latent user interest $p_\theta(u_i|I_j^{(t)}, z_i^{(t)})$ by integrating both cascade content c_j and original forwarding user sequence $\{u_k\}_{k=1}^K$.

For each information cascade $I_j^{(t)}$, the framework optimizes the Evidence Lower BOund

(ELBO) [68] $\mathcal{L}_j$ on the log likelihood of all users in the form of $\sum_i \log p(u_i|I_j^{(t)})$:

$$\begin{aligned}&\sum_i \log p(u_i|I_j^{(t)})\\&\geq \mathcal{L}_j = \sum_i \mathbb{E}_{q_\phi(z_i^{(t)})} p_\theta(u_i|I_j^{(t)}, z_i^{(t)}) - KL(q_\phi(z^{(t)})\|p(z^{(t)})),\end{aligned} \tag{3.1}$$

where $p(z_i^{(t)})$ denotes the prior distribution of $z_i^{(t)}$ approximated by unit normal distribution. For simplicity, we denote $q_\phi(z_i^{(t)})$ as the posterior distribution of user interests from our dynamic encoder in the remaining of our paper. Next, we will present the detailed designs of the encoder and decoder respectively.

3.3.2 The Dynamic Encoder

The dynamic encoder models the user interests evolution and infers the updated interests at new time steps for diffusion prediction. Two factors will influence the interests at new time steps: (1) **recent stimuli** which refer to the content users propagated recently, because users gain new information through such propagation behaviors; (2) **social influence** which describes the influence from the social neighbors (e.g., followees on Twitter) because the user interests can be influenced via the interactions with the social neighbors. The interests do not remain deterministic due to relatively nature of unpredictable and stochastic user behaviors. Considering that, at each time step, the dynamic encoder recursively models the user interests via a latent normal distribution based on the two factors from the last time step. For users at t-th time steps, let $x^{(t-1)} \in \mathbb{R}^{N\times d}$ encodes the information (s)he propagated at the last time step, $h^{(t)} \in \mathbb{R}^{N\times d}$ denotes the recurrent state, and $z^{(t)} \in \mathbb{R}^{N\times d}$ denotes the latent interest representation sampled from the normal distribution based on $h^{(t)}$. As shown in Figure 3.2, the process to infer latent interests can be summarized as:

$$h^{(t)} = f(g(h^{(t-1)}), x^{(t-1)}), \tag{3.2}$$

$$z_i^{(t)} \sim \mathcal{N}(\mu_\phi(h_i^{(t)}), \sigma_\phi(h_i^{(t)})), \tag{3.3}$$

where $f(\cdot)$ is the recurrent function to update $h^{(t)}$ at each time step, $g(\cdot)$ integrates information from the social neighbors for each users, μ_ϕ and σ_ϕ are the functions to characterize the normal distribution where $z_i^{(t)}$ is sampled. Next, we introduce the design of each part.

Recent stimuli $x^{(t-1)}$. At each time step, users would propagate some contents, and the propagated contents can influence the user interests at the next time step. To model the influence, we utilize $x^{(t-1)} \in \mathbb{R}^{N\times d}$ to encode the information from the propagated contents at

the last time step, and it can be learned from any unsupervised/supervised text embedding algorithms [69]–[71] which map raw text to the representation vectors. The choice and comparison of the text embedding module are beyond the scope of this paper. We introduce the choice of this module in experiment part, and refer interested readers to explore various text embedding methods here.

Recurrent function $f(\cdot)$. As shown in Figure 3.2, the encoder combines the social influence of $h^{(t-1)}$ and recent stimuli $x^{(t-1)}$, and updates $h^{(t)}$ for the new time step via Eq. (3.2). Intuitively, this combination should be automatically adjusted because these two factors influence the user interests differently and dynamically. Thus, we embed the graph convolutional layer into the gated recurrent unit (GRU) as the recurrent function $f(\cdot)$, where the graph convolutional layer naturally integrates information from the social neighbors via Graph Laplacian operator $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}h^{(t-1)}$, and GRU updates $h^{(t)}$ recursively. Specifically, at each time step:

$$G_u = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}h^{(t-1)}W_{hu} + W_{xu}x^{(t-1)}\right), \tag{3.4}$$

$$G_r = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}h^{(t-1)}W_{hr} + W_{xr}x^{(t-1)}\right), \tag{3.5}$$

$$\tilde{h}^t = \tanh\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}(G_r \odot h^{(t-1)})W_{hm} + W_{xm}x^{(t-1)}\right), \tag{3.6}$$

$$h^{(t)} = G_u \odot \tilde{h}^t + (1 - G_u) \odot h^{(t-1)}, \tag{3.7}$$

where σ is sigmoid function, and $\odot$ denotes the Hadamard (element-wise) product, $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is the normalized Graph Laplacian operator, G_r is the reset gate, G_u is the update gate, W are trainable parameters. The reset gate G_r determines how to form the combination $\tilde{h}^t$ from social influence and recent stimuli. And the update gate G_u controls how these two factors influence the user original interests. One can also stack multiple graph convolutional layers in the GRU in order to consider the higher-order social influence in the social networks.

Sampling from $\mathcal{N}(\mu_\phi(h_i^{(t)}), \sigma_\phi(h_i^{(t)}))$. To model the uncertainty of user's behaviors at each time step, we sample the current interest representations $z_i^{(t)}$ from the normal distributions $\mathcal{N}(\mu_\phi(h_i^{(t)}), \sigma_\phi(h_i^{(t)}))$, where $\mu_\phi(h_i^{(t)})$ and $\sigma_\phi(h_i^{(t)})$ are two linear layers learning mean and variance from $h_i^{(t)}$ respectively. With reparametrization tricks [68], we first sample ϵ from $\mathcal{N}(0, I)$ and infer $z_i^{(t)}$ by:

$$z_i^{(t)} = \mu_\phi(h_i^{(t)}) + \epsilon \odot \sigma_\phi(h_i^{(t)}) \tag{3.8}$$

In the new step, we infer the latent interests $z_i^{(t)}$ for each user. Next, we introduce how to model the relevance between the information cascades and potential users based on $z_i^{(t)}$.

3.3.3 The Dual Attention Decoder

For a specific information cascade $I_j^{(t)}$ within time step t, the dual attention decoder estimates the propagation likelihood for each potential user $p(u_i|I_j^{(t)})$. Based on inferred user interests from the encoder, the decoder first represents the information cascade, then estimates the likelihood for each potential user. The key difference of our proposed dual attention decoder from other diffusion models [51], [53], [54], [62], [63] is the consideration of both the cascade content c_j and the user sequence $\{u_k\}_{k=1}^K$. Because in reality, cascade content also determines propagation behaviors of users, especially at the early stages the user sequences are short and reflect limited information. Next, we introduce how the dual attention decoder represents the information cascade and estimate the propagation likelihood in detail.

Propagation likelihood estimation. For a single information cascade occurring at time step t: $I_j^{(t)} = (c_j, \{u_k\}_{k=1}^K)$, the propagation likelihood can be modeled as:

$$p_\theta(u_i|I_j^{(t)}, z_i^{(t)}) = \sigma(< o_j, v_i^R >), \tag{3.9}$$

where $\sigma(\cdot)$ is the sigmoid function, $< \cdot, \cdot >$ is the inner product, o_j represents the observed information cascade $I_j^{(t)}$, v_i^R represents the receiver state for potential user u_i. The same user appearing in observed information cascades and potential user set should have different states. Since in the former case the user would play as a sender who influence others, while in the latter one the user would play as a receiver who are influenced by the information cascade. This process is asymmetrical because senders are hardly influenced by receivers. For discussion convenience, we utilize the superscript S to denote the vectors in the sender state and R in the receiver state. Eq. (3.10) shows how we represent the sender state v_i^S and receive state v_i^R for u_i based on $z_i^{(t)}$.

$$v_i^S = W^S z_i^{(t)}, \ v_i^R = W^R z_i^{(t)} \tag{3.10}$$

We omit superscript t for v_i^S, v_i^R, because they both describe the states based on the same $z_i^{(t)}$ within t-th time step. Before introducing the techniques in the decoder, we want to highlight two challenges where we gain insights.

1. Some original forwarding users have noisy representations due to their sparse activities, and they may account for a large ratio due to the long tail distribution [72]. Therefore, it is critical to denoise and enhance their representations in order to characterize them more accurately.

2. The information from the information content and the original forwarding users are heterogeneous, which reflects the information from a different perspective.

Accordingly, we propose a dual attention decoder to address these challenges, where the first self-attention mechanism is designed to enhance and denoise the representations for original forwarding users, and the second heterogeneous attention mechanism is designed to aggregate information from both information content and user sequences.

Self-attention mechanism. We utilize the weighted sum of predecessors (together with the modeled user itself) as the denoised representation for each user, because users are more likely to be influenced by the predecessors within the original user sequence who they have interactions with. Thus we have:

$$v_k = \sum_{i=1}^{k} w_{ki} v_i^S, \tag{3.11}$$

where v_k denotes the denoised representation for u_k. To estimate these weights, we consider two factors. First, as social influence brings interaction, the relevance between sender u_i and receiver u_j, modeled by their inner product, may reflect influence. Second, the time elapsed between sender and receiver grows with decreasing influence. This is because the majority of social media users adopt more recent information, while often ignoring old and obsolete content [58]. Therefore, we learn the weight w_{ki} based on both users' inner product and temporal information, as follows:

$$w_{ki} = \frac{\exp\left(< (\boldsymbol{v}_i^S + PE(i)), (\boldsymbol{v}_k^R + PE(k)) >\right)}{\sum_{j=1}^{k} \exp\left(< (\boldsymbol{v}_j^S + PE(j)), (\boldsymbol{v}_k^R + PE(k)) >\right)}, \tag{3.12}$$

$$PE(k)_{2d} = \sin\left(\frac{k}{10000^{2d/D}}\right), PE(k)_{2d+1} = \cos\left(\frac{k}{10000^{2d/D}}\right), \tag{3.13}$$

where $< \cdot, \cdot >$ denotes the inner product of two vectors, $1 \leq d \leq D/2$ denotes the dimension of representations. We encode the relative position k through positional-encodings $PE(k)$ [73] as shown in Eq. (3.13).

Heterogeneous-attention mechanism. To integrate heterogeneous information from information cascade $I_j^{(t)}$, we first represent the information content c_j via the same text embedding module mentioned in Section 3.3.2, and then project it into the same space where the users are represented through one linear layer. Let v_c denote the learned vector. Then we propose an attention mechanism to integrate $(v_c, \{v_k\}_{k=1}^K)$ into the cascade representation o_j:

$$o_j = \alpha_c v_c + \sum_{k=1}^{K} \alpha_k v_k, \tag{3.14}$$

Algorithm 3.1: DyDiff-VAE training procedure.

Input: Social Network $\mathcal{G}$, Training cascades $\bigcup_{n=1}^{n<t} \mathbb{I}^t$.
Output: Model parameters ϕ and θ.

```
while model not converged do
    for Each time step t during training do
        (For the dynamic encoder:)
        Learn user recent stimuli x^(t-1) by text embedding on the content propagated
          at time step t − 1;
        Update new interests z^(t) according to Eq. (3.4) - Eq. (3.8);
        (For the dual attention decoder:)
        for Each information cascade at t time step do
            Based on z^(t), estimate propagation likelihood according to Eq (3.14) and
              Eq (3.9);
            Calculate ranking loss according to Eq. (3.16);
        end
    end
    Update the parameters φ and θ by optimizing variational ranking loss in
      Eq. (3.17);
end
```

where α_c and α_k are attention weights to combine information content and user sequence:

$$\begin{aligned} \alpha_c &= \frac{\exp(< v_c, v_c >)}{\exp(< v_c, v_c >) + \sum_{k=1}^{K} \exp(< v_c, v_k >)}, \\ \alpha_k &= \frac{\exp(< v_k, v_c >)}{\exp(< v_c, v_c >) + \sum_{k=1}^{K} \exp(< v_c, v_k >)} \end{aligned} \tag{3.15}$$

3.3.4 Optimization

DyDiff-VAE optimizes the variational lower bound on the log likelihood of all observed information cascades. For one single cascade, the objective is shown in Eq (3.1), where the first term is the reconstruction loss to estimate the users' propagation behaviors. For implementation, instead of estimating the likelihood expectation (via Monte Carlo sampling) , we use the ranking loss to reconstruct the observed cascades. For one thing, it is hard to correctly observe all forwarding users in datasets, so the estimated expectation would be biased due to noise. Also, in real world applications, (e.g., information recommendation), the ranking results are more practical than the specific likelihood. Thus, we directly optimize the weighted approximate-rank pairwise loss (WARP) [74] and expect observed forwarding users

(positive samples) to be ranked higher than the other users (negative samples). Specifically, for information cascade $I_j^{(t)}$, the ranking loss is defined as:

$$\begin{aligned} \mathcal{L}_j^{DIFF} &= \sum_{u_{i^+}} \frac{\sum_{u_{i^-}} L(rank(u_{i^+})) \cdot \mathcal{L}_j^{pair}(u_{i^+}, u_{i^-})}{rank(u_{i^+})}, \\ \mathcal{L}_j^{pair}(u_{i^+}, u_{i^-}) &= \max(0, \lambda_m - p_\theta(u_i^+ | I_j^{(t)}, z_i^{(t)}) + p_\theta(u_i^- | I_j^{(t)}, z_i^{(t)})), \end{aligned} \tag{3.16}$$

where u_{i^+} denotes the positive samples, u_{i^-} denotes the negative samples, $L(K) = \sum_{k=1}^{K} 1/k$. For each observed positive sample u_{i^+}, we expect the likelihood $p_\theta(u_i^+ | I_j^{(t)}, z_i^{(t)})$ should be larger than that of any negative samples by the margin λ_m., i.e., $\mathcal{L}_j^{pair}(u_{i^+}, u_{i^-}) = 0$. Otherwise, we penalize each pair of (u_i^+, u_i^-) because of the wrong ranking, and $L(rank(u_{i^+}))$ is the penalty weight.

Thus, our overall objective is to optimize the variational ranking loss in Eq (3.17).

$$\begin{aligned} \mathcal{L} = &\underbrace{\sum_j \mathcal{L}_j^{DIFF}}_{\text{ranking loss}} + \underbrace{\beta \cdot \sum_t \left(KL(q_\phi(z^{(t)}) \| p(z^{(t)})) \right)}_{\text{KL divergence for all time steps}} \\ &+ \underbrace{\lambda_1 \cdot \|\phi\|_2 + \lambda_2 \cdot \|\theta\|_2}_{\text{regularization}}, \end{aligned} \tag{3.17}$$

where β is the weight of KL divergence, λ_1 and λ_2 are the regularization weights for encoder and decoder parameters. Algorithm 3.1 summarizes the training procedure of DyDiff-VAE.

3.3.5 Diffusion Prediction and Ranking

We optimize the model parameters on information cascades at previous time steps, i.e., $\bigcup_{n=1}^{n<t} \mathbb{I}^n$. After convergence, DyDiff-VAE can be applied to predict the propagation likelihood and the corresponding user rankings for cascades at the new time step t. Given the observed $I_j^{(t)} = (c_j, \{u_k\}_{k=1}^K)$ at the new time step, DyDiff-VAE first infers the user new interests according to Eq. (3.8), then estimates likelihoods for all potential users according to Eq. (3.9), which induces the user rankings.

3.4 THE EVALUATION OF DYDIFF-VAE

In this section, we present and discuss our experimental results as well as ablation studies and parameter analysis on real-world datasets.

Table 3.1: Statistics of datasets used in our experiments.

Platform	Twitter			Youtube
Topic	**Election**	**Venezuela**	**While Helmet**	**Venezuela**
# Users	5,948	5,069	2,373	6,393
# Links	30,762	34,622	15,706	4,738
# Cascades	1,514	2,980	623	321
Avg. len	40.6	38.2	117.2	60.9
Start Date	2018-12	2018-12	2018-7	2018-12
End Date	2019-02	2019-03	2019-04	2019-03

3.4.1 Datasets

Since existing datasets lack the content information, we collect the following four datasets from Twitter and Youtube to evaluate our proposed model. We collect data related to *Election* [10], *Venezuela* [11], *White Helmets* [12] to form three Twitter datasets and one YouTube data set. On Twitter, each retweet sequence of a content item constitutes a diffusion cascade, while the explicit social network consists of follower-followee links. Following [53], [54], [62], we removed tweets with non-English content and preserved users with more than 10 records and cascades longer than 10. On YouTube, user interactions constitute the social networks, and a cascade corresponds to the comment sequence of a video. We collected data related to Venezuela. Based on the raw data, We extract the video descriptions as the propagated cascade contents, and then preserve users with more than 5 records and cascades longer than 10.

Dataset statistics are provided in Table 3.1. All referred IDs have been anonymized. For each dataset, we uniformly separate the time span into 6 discrete time steps and utilize cascades in the first five time steps as Training set. Val/Test are split randomly on the information cascades on 6-th time step, with ratio $1:3$. Following [52]–[54], we remove users who do not exist in Training set from Val/Test set, for we have no information of these users. We also try different time segmentation strategies, where our method consistently outperforms others. We leave a systematic study for optimal segmentation as our future work.

3.4.2 Experimental Setup

Baselines. We compare our model against state-of-the-art methods from related areas including graph representation learning (dynamic/static), social recommendation, and diffusion

[10] https://en.wikipedia.org/wiki/2018_Venezuelan_presidential_election
[11] https://en.wikipedia.org/wiki/Venezuela
[12] https://en.wikipedia.org/wiki/White_Helmets_(Syrian_Civil_War)

Table 3.2: $MAP@K$ for diffusion prediction on 4 datasets.

Platform	**Twitter**									**Youtube**		
Dataset	**Election**			**Venezuela**			**White Helmet**			**Venezuela**		
MAP	**@10**	**@50**	**@100**	**@10**	**@50**	**@100**	**@10**	**@50**	**@100**	**@10**	**@50**	**@100**
Random	0.0011	0.0010	0.0009	0.0010	0.0009	0.0010	0.0034	0.0048	0.0028	0.0019	0.0008	0.0010
Node2vec	0.0102	0.0110	0.0122	0.0105	0.0147	0.0168	0.0095	0.0136	0.0155	0.0067	0.0053	0.0041
GAE	0.0258	0.0252	0.0304	0.0229	0.0250	0.0292	0.0326	0.0267	0.0315	0.0092	0.0063	0.0058
VGRNN	0.0137	0.0108	0.0186	0.012	0.0109	0.0173	0.0158	0.0181	0.0177	0.0083	0.0073	0.0063
GraphRec	0.0224	0.0241	0.0255	0.0298	0.0280	0.0300	0.0233	0.0171	0.0177	0.0023	0.0024	0.0021
DeepDiffuse	0.0276	0.0264	0.0315	0.0171	0.0208	0.0232	0.0351	0.0252	0.0291	0.0120	0.0106	0.0119
SNIDSA	0.0299	0.0296	0.0353	0.0419	0.0376	0.0406	0.3045	0.0274	0.0277	0.0131	0.0111	0.0124
Topo-LSTM	0.0307	0.0347	0.0397	0.0325	0.0361	0.0393	0.0293	0.0262	0.0291	0.0081	0.0067	0.0071
Inf-VAE	0.0423	0.0406	0.0454	0.0393	0.0401	0.0435	0.0330	0.0257	0.0289	0.0116	0.0102	0.0094
Inf-VAE+Content	0.0428	0.0378	0.0432	0.0425	0.0416	0.0451	0.0426	0.0275	0.0298	0.0099	0.0076	0.0063
DyDiff-VAE	**0.0724***	**0.0698***	**0.0764***	**0.0602***	**0.0601***	**0.0626***	**0.0570***	**0.0577***	**0.0502***	**0.0169***	**0.0134***	**0.0141***
Imprv.	*+69.2%*	*+71.9%*	*+68.3%*	*+41.6%*	*+44.5%*	*+38.8%*	*+33.8%*	*+109.8%*	*+59.4%*	*+29.0%*	*+20.7%*	*+13.7%*

Table 3.3: $Recall@K$ for diffusion prediction on 4 datasets.

Platform	**Twitter**									**Youtube**		
Dataset	**Election**			**Venezuela**			**White Helmet**			**Venezuela**		
Recall	**@10**	**@50**	**@100**	**@10**	**@50**	**@100**	**@10**	**@50**	**@100**	**@10**	**@50**	**@100**
Random	0.0011	0.0100	0.0193	0.0010	0.0085	0.0150	0.0034	0.0185	0.0323	0.0019	0.0008	0.0010
Node2vec	0.0152	0.0723	0.1123	0.0252	0.0913	0.1420	0.0189	0.0713	0.1208	0.0066	0.0243	0.0346
GAE	0.0304	0.1200	0.2260	0.0389	0.1417	0.2418	0.0376	0.1298	0.2345	0.0061	0.0173	0.0471
VGRNN	0.0176	0.0548	0.1185	0.0199	0.1019	0.1474	0.0253	0.0874	0.1327	0.0089	0.0218	0.0386
GraphRec	0.0147	0.0837	0.1429	0.0334	0.1261	0.1915	0.0303	0.1201	0.1977	0.0027	0.0044	0.0115
DeepDiffuse	0.0382	0.1318	0.2172	0.0329	0.1337	0.1966	0.0499	0.1363	0.2076	0.0145	0.0296	0.0421
SNIDSA	0.0452	0.1232	0.2253	0.0463	0.1464	0.2199	0.0457	0.1374	0.2138	0.0132	0.0261	0.0410
Topo-LSTM	0.0506	0.1727	0.2568	0.0447	0.1419	0.2114	0.0304	0.1046	0.1965	0.0081	0.0217	0.0328
Inf-VAE	0.0462	0.1642	0.2304	0.0610	0.1586	0.2390	0.0414	0.1377	0.2041	0.0125	0.0259	0.0409
Inf-VAE+Content	0.0529	0.1599	0.2510	0.0623	0.1750	0.2541	0.0441	0.1295	0.2183	0.0106	0.0266	0.0327
DyDiff-VAE	**0.0862***	**0.2377***	**0.3376***	**0.0944***	**0.2204***	**0.2892***	**0.0809***	**0.1850***	**0.2695***	**0.0188***	**0.0371***	**0.0561***
Imprv.	*+62.9%*	*+37.6%*	*+31.5%*	*+47.4%*	*+25.9%*	*+13.8%*	*+62.1%*	*+48.7%*	*+34.4%*	*+29.7%*	*+25.3%*	*+19.1%*

prediction.

- **Node2vec** [64]: a framework that learns low-dimensional representations for nodes in a network by optimizing a neighborhood preserving objective.
- **GAE** [65]: a graph autoencoder framework consists of graph convolutional layers to represent the network and node attributes, and an inner product decoder to predict the link existence.
- **VGRNN** [66]: a variational autoencoder framework for dynamic graph learning, which represents the dynamic adjcency matrix and node attributes, and predicts the link existence.
- **GraphRec** [41]: a novel graph neural network framework for social recommendations, which coherently models interactions and opinions in the user-item graph.
- **DeepDiffuse** [53]: an attention-based RNN that operates on the subsequence of previously influenced users to predict diffusion, where the subsequence is extracted by another modified LSTM.

- **SNIDSA** [62]: a novel sequential neural network with structure attention to model both sequential nature of an information diffusion process and structural characteristics of user connection graph for diffusion prediction.
- **Topo-LSTM** [52]: a recurrent model that exploits the local propagation structure during the diffusion process through a dynamic tree-based LSTM.
- **Inf-VAE** [54]: a variational framework that combines a variation graph autoencoder and a co-attention network to jointly model social relations and sequence information.

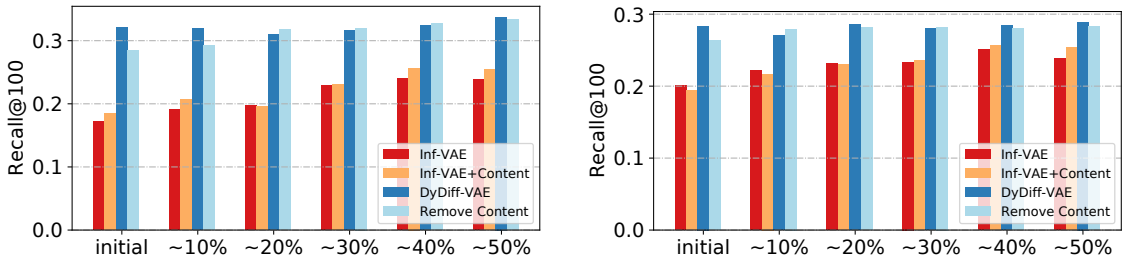
Setup. Following [53], [54], [62], [63], we evaluate our diffusion model in a retrieval setting. We rank potential users based on their propagation likelihood and utilize $MAP@K$ and $Recall@K$ as the metrics to evaluate the predicted rankings. During the evaluation, unless specifically mentioned, we keep the first half of forwarding users observed in a ground truth cascade and predict the other half of forwarding users. This ratio is defined as *seed set percentage*, and we will also report the performance comparison under various seed set percentages.

To train our model, we fix the dimension of our representation vectors as 128, regularization weight of model parameters λ_1 and λ_2 as 0.5, the KL divergence weight β as 10. Then, we tune learning rate and λ_m, by evaluating $MAP@10$ on the Val set. We choose the unsupervised Doc2Vec [69] as the text embedding module for simplicity. For Node2vec, GAE, and VGRNN, the outcomes are the user representations, we use the average aggregation function to represent the cascades, which has the best performance compared with Sum and Earliest (root representations) functions. For GraphRec, since it is designed to predict the rating scores among each user-item pair in a standard recommendation setting, we set the rating score as 1 if a user forwards the cascade, otherwise 0. We use the ranking induced by rating score for diffusion prediction. For other diffusion baselines, as stated in [54], we use the ranking induced by propagation likelihood for diffusion prediction.

3.4.3 Experimental Results

Overall Results. Table 3.2 and Table 3.3 show the performance comparisons, where the last row presents the relative improvement over the best baseline [13]. Node2vec considers the social structure information, GAE and VGRNN consider both structure information as well as user interests learnt from contents, VGRNN models the evolution of user interests, GraphRec

[13]All results are averaged over 5 independent runs with different random seeds. * indicates the statistically significant improvements over the best baseline, with p-value smaller than 10^{-6}.



(a) Twitter/Election

(b) Twitter/Venezuela

Figure 3.3: Recall@100 under various seed set percentage.

jointly models the user-cascade interactions and cascade contents in the recommendation framework, and the other diffusion baselines study the sequential information of the cascade of original forwarding users.

DyDiff-VAE significantly outperforms all the baselines on both MAP and *Recall* metrics, meaning that more forwarding users are found and they rank higher in the predicted list. The diffusion prediction is indeed a hard task, which is indicated by the extremely low scores from *Random* results, where the *Random* result could be 0.50 in some classification tasks. However, our model is able to achieve superior result. For example on Twitter/Election, we aims to retrieve $\sim$ 20 users out of $\sim$ 5k users based on initial $\sim$ 20 users. A *Recall*@100 of 0.34 is quite impressive, considering the absence of explicit propagation tree topology and the noise in the real-world diffusion processes.

Also, we find existing diffusion methods are not effective at modeling cascade content and user interests. We modified Inf-VAE to consider such information as its GCN-based encoder can be easily generalized to model user attributes. We call the modified baseline as *Inf-VAE+Content*. Results do not show a consistent improvement over the original Inf-VAE. It is worth noting that the dataset characteristics influence prediction performance. It is easier to predict on Twitter/Election dataset consisting of the polarized posts because users having different stances behave quite differently. For the Youtube dataset, more flexible social interactions and more diverse cascade content make it less predictable.

Results under various seed set percentages. A desirable property of diffusion model is that the it has robust performance when observing different ratio of original forwarding users. We vary the seed set percentage from *initial* (meaning that we only observe the initial posts) to 0.5. We use dataset *Twitter/Election* and *Twitter/Venezuela* for this study, for we require a sizable number of test examples to obtain unbiased estimates. The results are shown in Figure 3.3. Recall increases with seed set percentage as expected. We compare our results with two strong baselines. It is interesting to find that under *initial* situation, their

Table 3.4: Ablation study on framework design.

Dataset	Twitter/Election		Twitter/Venezuela	
Metrics (@100)	MAP	Recall	MAP	Recall
(1) Static encoder	0.0530	0.2660	0.0516	0.2579
(2) Remove Conv.	0.0650	0.3007	0.0485	0.2650
(3) Remove 1st attn	0.0466	0.2572	0.0327	0.2360
(4) Remove 2nd attn	0.0738	0.3337	0.0561	0.2734
(5) $W^S = W^R$	0.0693	0.3199	0.0591	0.2710
(6) DyDiff-AE	0.0746	0.3124	0.0603	0.2630
DyDiff-VAE	**0.0764**	**0.3376**	**0.0626**	**0.2892**

performance drop significantly, while DyDiff-VAE still achieves competitive performance. To dig into the reason, we show the result after removing content information (by removing the heterogeneous attention layer). We observe a performance drop in the initial situation, because in the beginning, the user sequences are too short to provide sufficient information. As the ratio increases, the performances tend to be close, because information content does not provide more information after observing the first half of the user sequence.

3.4.4 Ablation Study and Analysis

Ablation Study. We conduct an ablation study to analyze our design choices. We first examine the effectiveness of the dynamic encoder:

1. **Static encoder**. The user interests are assumed static, and the dynamic encoder becomes one-layer GCN.

2. **Remove Conv.**. We eliminate the social influence, so the user interests will be independent of other users.

Replacing the dynamic encoder as a static one will result in a significant performance drop. It proves the importance of modeling user dynamic interests on diffusion prediction. Further, while modeling such evolution, the worse performance after removing the graph convolutional layer illustrates the necessity to consider the social influence from other users. Next, we analyze the effectiveness of the dual attentive decoder by the following ablations:

3. **Remove 1st attn**. The cascade representations are directly learned on the user interests from encoder, without any denoising and enhancement.

4. **Remove 2nd attn**. Only consider the user sequence without modeling the information content. The cascade representation is the average on the denoised user representations.

5. $W^S = W^R$. Ignore the difference between the sender state and receiver state.

In our dual attentive decoder, we utilize a self-attention layer to learn the local influence and denoise the user interest representations, especially for inactive users. Removing such method (3) results in a significant performance drop. Notably, replacing the heterogenous attention layer with a mean aggregation on user sequence leads to a close performance. Because, as stated in Section 3.4.3 the information content does not provide more information after observing the first half of the user sequence. Moreover, overlooking asymmetric social influence (5) slightly deteriorates results, which indicates the existence of asymmetric influence among users. Finally, we analyze the effectiveness of the variational framework by the following ablations:

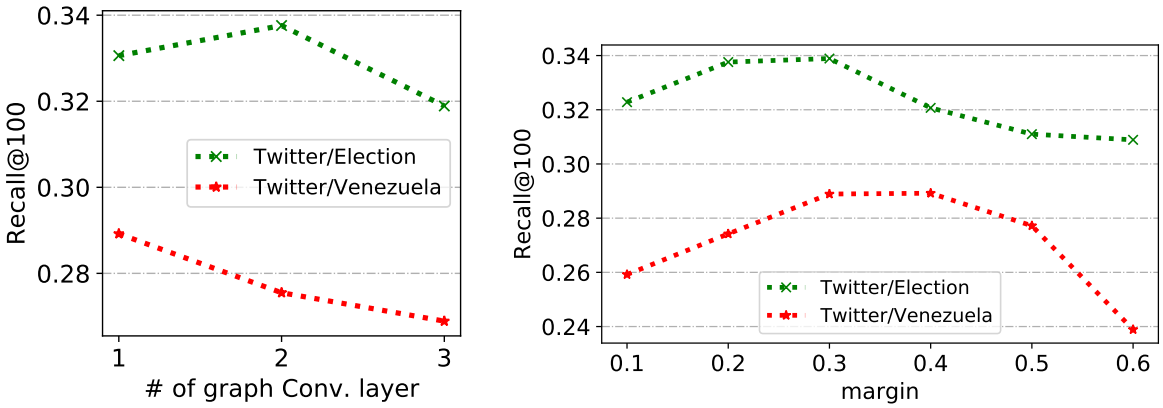
6. **DyDiff-AE**. We replace the variational framework with a autoencoder framework, and the user interests are deterministics.

As is expected, DyDiff-AE results in a worse performance, since it overlooks the uncertainty of the user interests.

Impact of graph convolution. Social influence has been proved necessary to model the user dynamic interests. We further study whether stacking more graph convolutional layers, in order to integrate higher-order social influence, can help for diffusion prediction. As shown in Figure 3.4, adding one more layer increase the *Recall*@100 on Twitter/Election dataset but have a negative impact on Twitter/Venezuela dataset. This is because, in part, that the social network of Twitter/Venezuela is far denser, and multiple convolutional layers may cause the over-smoothing issue [75], [76].

Impact of λ_m. We study the impact λ_m, as shown in Figure 3.4. λ_m is the marginal value to penalize negative users ranking higher. Large λ_m can hurt the performance, for it ignores some relevance, though is not captured in the dataset, between the cascades and negative users.

Efficiency Analysis. As mentioned earlier, efficiency is the bottleneck of sequence modeling for large-scale applications. We run diffusion baselines and DyDiff-VAE on the same workstation equipped with an Intel i9-9960X processor, 64GB memory, and one NVIDIA RTX 2080 Ti GPU, and compare execution time per epoch (average on 20) on two large datasets. Each model is trained with one GPU on the device. Figure 3.5 shows the comparison results. DyDiff-VAE is faster than the other RNN-based baselines. And it achieves the competitive efficiency as Inf-VAE with consistently superior performance.



(a) Impact of Conv. Layers

(b) Impact of λ_m

Figure 3.4: Parameter analysis evaluated by *Recall*@100.

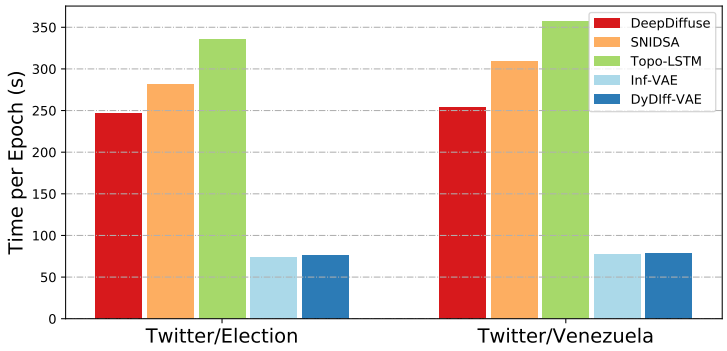


Figure 3.5: Running time comparison, unit: second (s). DyDiff-VAE is faster than recurrent models (DeepDiffuse, SNIDSA, Topo-LSTM), and close to attention model (Inf-VAE).

3.5 RELATED WORK

3.5.1 Diffusion Prediction

Early work for diffusion prediction usually holds certain assumptions about the diffusion process. For example, Independent Cascade (IC) models [55], [56] popularized the study of information diffusion with a pairwise independence assumption. By introducing stronger assumptions about time-delay information, extensions of these models [77]–[82] considered temporal evolution in the diffusion process. However, pairwise independence oversimplifies the complex nature of the diffusion process, resulting in poor performance on real-world datasets. With advances in representation learning and deep learning, recent works [52]–[54], [62], [63], [83], [84] design end-to-end frameworks to automatically learn diffusion patterns from data. They project cascades into the user-characterized space and predict the diffusion likelihood

based on the distance between cascade and user representations. Most of these models [52], [53], [62], [63], [83] employs extensions of Recurrent Neural Networks (RNNs) to extract meaningful features from user sequences, e.g., local topology among users and temporal patterns within the diffusion cascades. Recent work [54] enhances user representations, instead of initializing randomly, via a variational graph autoencoder. However, all of the above neural methods focus entirely on learning the cascade representations from user sequence, ignoring information content which has been shown meaningful in information diffusion [85]. Moreover, existing works ignore modeling dynamic user interests from diffusion data. To the best of our knowledge, DyDiff-VAE is the first framework that jointly infers the dynamic user interests and models information content for diffusion prediction.

3.5.2 Variational Autoencoder

Variational autoencoder (VAE) [68] models the generative process of data $\mathbf{x}$ from the latent variable $\mathbf{z}$. The encoder approximates the distribution of latent variable $\mathbf{z}$ by the variational posterior, and the decoder generates the data $\mathbf{x}$ given $\mathbf{z}$. VAE has superior performance in representation learning for both static data [65], [86], [87] and dynamic data [66], [88]. The key difference between DyDiff-VAE and existing VAE on modeling dynamic data is the design of the decoder. The conventional decoder aims to reconstruct the data samples or the pair-wise relevance among data samples, while our dual attentive decoder aims to reconstruct the propagation likelihood between the information cascade and potential users. It explicitly models the sequential and heterogeneous information via the dual attentive decoder from both the user sequence and the information content.

3.5.3 Sequence Modeling

Representing the information cascade requires an efficient sequence modeling of long user sequence. Recurrent Neural Networks (RNNs) are one of the most popular sequence models, which model the cross-dependency within a sequence by a special design of various gate mechanisms. Several RNN variants are proposed for handling different sequence types, (e.g., GRU and LSTM [61], [89] for long sequence modeling where the gradient vanishing and exploding problems are addressed, tree-LSTM [90] for natural language with a latent grammar structure, etc). However, RNNs scale poorly with the increasing in sequence length due to the sequential nature of back-propagation through time, which becomes the bottleneck for large-scale applications. To improve computational efficiency, alternative transformation methods that include a parallel convolutional operator [91] and self-attention [73], [92] have

been proposed. Existing diffusion models usually utilize RNNs, resulting in prohibitive costs for long cascade sequences. To achieve better efficiency, we propose an efficient and parallelizable dual attentive decoder to learn cascade representations.

3.6 SUMMARY

In this chapter, we introduced a novel dynamic variational framework that jointly considers dynamic user interests and information content for diffusion prediction. We proposed a dynamic encoder to infer the dynamic user interests from historical behaviors, and a dual attentive decoder to capture the information from both the information cascade and the sequence of original users to estimate the propagation likelihood. Our experimental results on four real-world datasets demonstrated that the proposed model outperforms other compared methods.

The work has several shortcomings that offer avenues for future extensions. First, the algorithms are described at a fixed point in time when some data have been collected and the future is to be predicted. In reality, incremental algorithms are needed where new data arrive over time. Second, we do not take into account the fact that data sets are invariably incomplete. Thus, it is likely that future cascade participants will include users never before seen in the collected data. In the future, we plan to extend DyDiff-VAE to generate realistic user profiles (in terms of social relations and interests) from the learned latent distribution for unseen users.

Chapter 4: LEARNING DYNAMIC MULTI-RELATIONAL GRAPHS

In this chapter, we extend our model design from bipartite graphs to a more complex multi-relational graphs to model the dynamics. We first introduce the preliminary analysis of multi-relational graph learning, then detail the model design of the proposed RETE framework, followed by a comprehensive evaluation of RETE framework on various real-world applications, including recommendation, temporal event forcasting.

4.1 PRELIMINARIES AND ANALYSIS

On the evolutionary multi-relational graph, user intents can be captured by integrating abundant information from connected entities [93]. Existing KG-based frameworks [94]–[96] map entities into a low-dimensional space, such that the relevance between user, query or product can be modeled via corresponding representations, i.e., $\boldsymbol{h}_u, \boldsymbol{h}_p, \boldsymbol{h}_q \in \mathbb{R}^d$. They propose various propagation mechanisms [60], [97] on whole KG to integrate abundant information for each user, which can be generally described as follows:

$$\boldsymbol{h}_u^{(l)} = \sum_{e' \in \mathcal{N}_u} \pi(u, r, e') \boldsymbol{h}_{e'}^{(l-1)}, \tag{4.1}$$

where $\mathcal{N}_u$ denotes neighbor set of user u, l denotes the number of propagation layers, triple (u, r, e') describes interaction between u and e', and $\pi(u, r, e')$ denotes aggregation weights.

Directly generalizing this family of propagation mechanisms to the event forecasting task, especially on evolutionary multi-relational graph, faces two issues: (i) As l grows to integrate higher-order information, the neighborhood size increases exponentially. A large ratio of unrelated entities (noises) are integrated, making user representations less distinguishable from each other, or even leading to over-smoothing [75], [76], [98]; (ii) evolutionary multi-relational graph shows different distribution over time as users have evolving intents and behavior patterns. Ignoring such temporal factors not only fails to capture the most recent data characteristics but also worsens the first issue due to integrating a large ratio of out-of-fashion records for learning the representations.

4.2 THE DESIGN OF RETE

In this section, we present the proposed RETE framework. We firstly start with the overview of RETE and then detail three major components. Finally, we describe the model

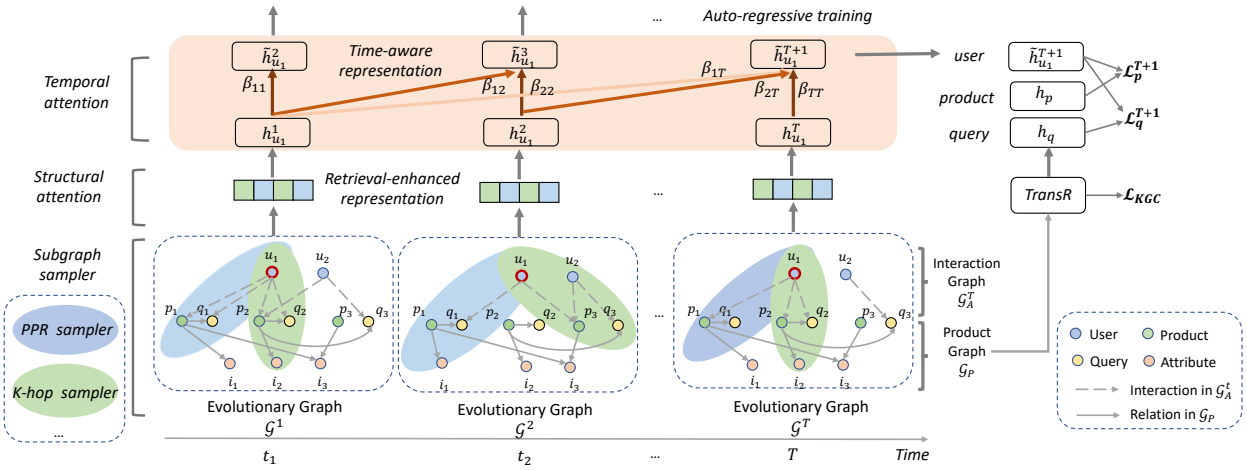


Figure 4.1: The framework of the proposed RETE model. We optimize the RETE model using the auto-regressive ranking loss and the multi-relational graph completion loss (TransR) alternatively.

optimization.

4.2.1 Overview

To address two issues mentioned in Section 4.1 simultaneously, the proposed RETE strives to satisfy the following requirements:

- *Requirement 1*: It should learn informative and discriminative user representations by considering higher-order information and filtering out large ratio of noise from the graph.
- *Requirement 2*: It should capture user intent evolution from data at different time steps, so as to produce up-to-date forecasting.

The framework of RETE is shown in Figure 4.1. Based on the collected evolutionary multi-relational graph, the key idea of RETE is to learn informative user representations at each time step from the sampled subgraphs instead of the whole graph, and combine them together via learnable temporal weights to capture user intent evolution. RETE first utilizes subgraph samplers to retrieve related entities centrally around each user u. And then it integrates rich information from the subgraphs via a structural attention module. A sequence of learned representations from all time steps are integrated via a temporal attention module. It can automatically learn the combination weights in the temporal domain so that more related (e.g., more recent) time steps are assigned larger weights, and unrelated (e.g., far away) time steps are assigned smaller weights (but not necessary to 0, as they may also

reflect long-term intents). To fulfill *Requirement 1*, the sampled subgraphs constrain the attentive information propagation within local highly-related structure instead of the whole graph, so as to consider higher-order information and filter out noise. The design of the temporal module enables us to auto-regressively learn users' up-to-date representations to meet *Requirement 2*.

4.2.2 Theoretical analysis

We then analysis how RETE satisfies the two requirements theoretically.

For *Requirement 2*, it is straightforward to show that our temporal attention module can auto-regressively consider new data and update the most up-to-date intents. To analysis *Requirement 1*, we propose the following proposition:

Proposition 4.1. For temporal event forecasting task, models able to learn informative and discriminative user representations should satisfy the following necessary conditions: let $m_{[u]}$ denote information gained by user u via n-layer of neighbor aggregation, gained information of different users should always be distinguishable: $m_{[u]} \neq m_{[v]}, (u \neq v)$, even under $n \to \infty$.

The correctness of proposition 4.1 is obvious, as models may utilize a large number of layers of neighbor aggregation to integrate higher-order and related information, especially for users with sparse records. So it should be satisfied under n up to ∞. To analyze why RETE satisfies the this condition, let $m_{[u]} = \{m^1_{[u]}, m^2_{[u]}, ..., m^t_{[u]}\}$ denote gained information by user u from different time steps. By applying theorem in [99], gained information at time step t after ∞-layer of GCN-like neighbor aggregation (used in our structural attention module) can be represented as:

$$m^t_{[u]} \to \sqrt{\frac{\delta_{[u]}(u)}{\sum_{v \in \mathcal{G}^t_{[u]}} \delta_{[u]}(v)}} \cdot \boldsymbol{e}^T_{[u]} \boldsymbol{X}_{[u]} \tag{4.2}$$

where $\delta_{[u]}(v)$ denotes degree of node v in sampled subgraph $\mathcal{G}^t_{[u]}$, $\boldsymbol{X}_{[u]}$ denotes initial entity embeddings in sampled subgraph $\mathcal{G}^t_{[u]}$, and $\boldsymbol{e}_{[u]}$ denotes eigenvector of $\mathcal{G}^t_{[u]}$ corresponding to largest eigenvalue. By using ensemble subgraph sampler, we can ensure $\mathcal{G}^t_{[u]} \neq \mathcal{G}^t_{[v]}$ even after ∞-layer of neighbor aggregation, so that $m^t_{[u]} \neq m^t_{[v]}$ in each time step, and $m_{[u]} \neq m_{[v]}$. Without such subgraph constrains, after large number of neighbor aggregations, the integrated entities easily span the whole graph, i.e., $\mathcal{G}^t_{[u]} = \mathcal{G}^t_{[v]} = \mathcal{G}^t$, making learned user representations much less distinguishable.

Next, we will introduce the subgraph samplers, the structural attention module, and the temporal attention module respectively.

4.2.3 Ensemble Subgraph Sampler

The proposed sampler aims to retrieve diverse and related entities via higher-order connections and excludes unrelated noises as order increases. To design strong indicators for such desirable subgraphs, we adopt structure-dependent Personalized PageRank (PPR) value [100] which has been recently shown effective to retrieve related nodes from homogeneous graph [99], [101], [102]. And we further ensemble it with a simple randomized k-hop sampler to retrieve entities more comprehensively. Advantages of this design, instead of utilizing trainable policy, are that it does not require reliable input features (unlike [98], it is fully feature-independent) and it achieves much higher sample efficiency (almost real-time with careful implementation. We summarize the designed sampler as follows:

- **PPR sampler**. We use the feature-independent Personalized PageRank (PPR) value. Given a target user u, our PPR sampler first computes the approximate PPR value for all other entities, then selects up to b neighborhood above threshold θ and preserves relations among selected entity set.
- **k-hop sampler**. Starting from a target user u, the k-hop sampler traverses up to k-hop connections and randomly selected up to b neighbors.
- **Ensemble sampler**. To capture a full picture of user intents, we ensemble multiple samplers under different types or with different parameters to parallelly sample several subgraphs.

4.2.4 Structural Attention Module

Without loss of generality, let s denote the number of subgraphs from the ensemble sampler. At time step t, given the sampled subgraphs $\{\mathcal{G}_{[u]}^{1t}, \cdots, \mathcal{G}_{[u]}^{st}\}$ for each user u, the structural attention module aims to integrate all useful information to learn user representations. As shown in Figure 4.2, it first extracts information from each subgraph via multi-layer graph attentions [97]. Then to combine information from different perspectives, it fuses outputs from different subgraphs together to capture the global picture of user intent.

Specifically, we first utilize L graph attention layers to propagate and integrate information within each subgraph $\mathcal{G}^{st}_{[u]}$. Each layer can be summarized in Eq. (4.3):

$$\boldsymbol{h}_u^{(l)} = \sigma\left(\sum_{v\in\mathcal{N}_u} \alpha_{uv}^{(l)} \boldsymbol{W}_V^{(l)} \boldsymbol{h}_v^{(l-1)}\right), (1 \le l \le L) \tag{4.3}$$

where $\boldsymbol{h}_u^{(l)}$ denotes user representations from layer l, $\mathcal{N}_u$ denotes neighbor set around user u on subgraph $\mathcal{G}^{st}_{[u]}$, and $\alpha_{uv}^{(l)}$ is aggregation attention weight shown in following equation:

$$\alpha_{uv}^{(l)} = \frac{\exp\left(\sigma\left(\boldsymbol{a}^T\left[\boldsymbol{W}_Q^{(l)}\boldsymbol{h}_u^{(l)} \| \boldsymbol{W}_K^{(l)}\boldsymbol{h}_v^{(l)}\right]\right)\right)}{\sum_{v'\in\mathcal{N}_u}\exp\left(\sigma\left(\boldsymbol{a}^T\left[\boldsymbol{W}_Q^{(l)}\boldsymbol{h}_u^{(l)} \| \boldsymbol{W}_K^{(l)}\boldsymbol{h}_{v'}^{(l)}\right]\right)\right)}, \tag{4.4}$$

where $\boldsymbol{W}_V^{(l)}$, $\boldsymbol{W}_K^{(l)}$, $\boldsymbol{W}_Q^{(l)}$ are shared weighted transformation applied to each entity in the subgraph, $\boldsymbol{a}$ is a weight vector parameterizing the attention function implemented as feed-forward layer, $\|$ is the concatenation operation and $\sigma(\cdot)$ is non-linear activation function.

Since information propagation is constrained within the sampled subgraphs, more attention layers can be stacked to better learn latent representations without introducing other unrelated entities and noises. We then use residual connection across L layers and graph-level pooling to better integrate information to capture user's intent at time step t:

$$\boldsymbol{h}_u^{st} = \frac{1}{|\mathcal{E}^{st}_{[u]}| \times L} \sum_{e\in\mathcal{E}^{st}_{[u]}} \sum_l \boldsymbol{h}_u^{(l)}, \tag{4.5}$$

where $\mathcal{E}^{st}_{[u]}$ denotes entity set in s-th subgraph $\mathcal{G}^{st}_{[u]}$. By doing so, we are able to integrate all useful information from the subgraphs to better represent users. To further capture a global view of user intent, ensemble samplers sample several subgraphs. From the learned representations of those subgraphs, we utilize one-layer MLP to integrate global information:

$$\boldsymbol{h}_u^t = \sigma\left(\boldsymbol{W}\left[\boldsymbol{h}_u^{1t} \| \boldsymbol{h}_u^{2t} \| \cdots \| \boldsymbol{h}_u^{st}\right]\right). \tag{4.6}$$

4.2.5 Temporal Attention Module

Given the learned user representations $\boldsymbol{H} = \{\boldsymbol{h}_u^1, \boldsymbol{h}_u^2, \cdots, \boldsymbol{h}_u^T\}$ from all time steps, we infer users' intents in a near future, i.e., $\widetilde{\boldsymbol{h}}_u^{T+1}$, for temporal event forecasting. We propose a temporal attention module to automatically capture both long-term and short-term intents by assigning different weights among h_u^t. The updating function for most recent user

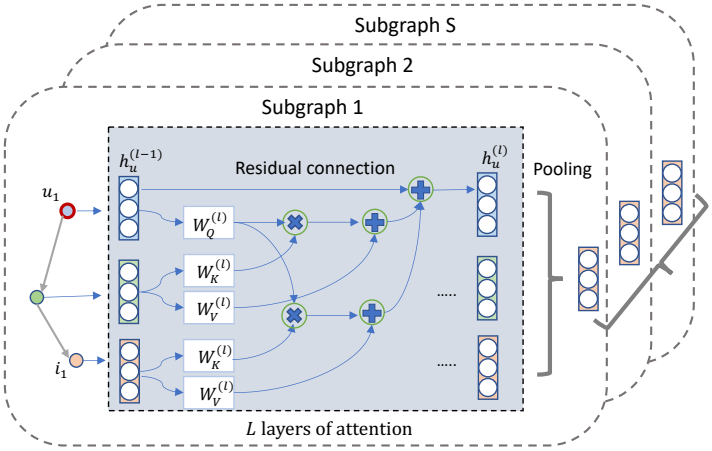


Figure 4.2: Structural attention module on subgraphs.

representation $\boldsymbol{h}_u^{T+1}$ can be expressed as follows:

$$\widetilde{\boldsymbol{h}}_u^{T+1} = \sum_{t=1}^{T} \beta_{tT} \boldsymbol{h}_u^t \boldsymbol{W}^V, \tag{4.7}$$

$$\beta_{ij} = \text{softmax}\left(\frac{\boldsymbol{H}\boldsymbol{W}^Q (\boldsymbol{H}\boldsymbol{W}^K)^T}{\sqrt{d}} + \boldsymbol{M} \right)_{ij}, \tag{4.8}$$

where $\boldsymbol{W}^Q$, $\boldsymbol{W}^L$, $\boldsymbol{W}^V$ are trainable temporal parameters, β_{ij} is learned temporal weight, d denotes dimension of user representations, and $\boldsymbol{M}$ is added to ensure auto-regressive setting, i.e., preventing future information affecting current state. We define $\boldsymbol{M}_{ij} = 0$ if $i \leq j$, otherwise $-\infty$.

By applying Eq. (4.7), we are able to not only emphasize those information related to users' short-term intents represented in $\boldsymbol{h}_u^T$, but also capture long-term intents as we integrate information from all time steps. The proposed temporal module provides better interpretability, where the temporal attention weights reflect user intent evolution and shifting. Notably, it can be auto-regressively applied to newly collected data from $T+2, T+3, \cdots$, as it automatically computes temporal combination weights in the future time steps without model modification or retraining.

4.2.6 Optimization

The proposed framework is expected to capture the evolution of user preference from the evolutionary multi-relational graph. To better represent product and query information from static product graph into the same low-dimensional space, we utilize a multi-relational graph

embedding module TransR [14] to optimize multi-relational graph completion loss $\mathcal{L}_{KGC}$. After learning user/product/query representations at time step t, i.e., $\widetilde{\boldsymbol{h}}_u^t$, $\boldsymbol{h}_p$, $\boldsymbol{h}_q$, we use inner product $\gamma_{up}^t =< \widetilde{\boldsymbol{h}}_u^t, \boldsymbol{h}_p >$ and $\gamma_{uq}^t =< \widetilde{\boldsymbol{h}}_u^t, \boldsymbol{h}_q >$ to model relevance.

We use negative sampling to accelerate and stabilize the training process. At time t, for each user-product pair (u, p^+), we randomly sample several negative samples (u, p^-), where we expect γ_{up-}^t is smaller than γ_{up+}^t by a margin. Thus, we adopt the weighted approximate-rank pairwise (WARP) loss [74] for product prediction as follows:

$$\mathcal{L}_p = \sum_{t=1}^{T} \mathbb{E}_{(u,p^+)\in\mathcal{G}^t} \sum_{p^-} \frac{L(rank(p^+)) \cdot |\lambda_m - \gamma_{up+}^t + \gamma_{up-}^t|_+}{rank(p^+)}, \tag{4.9}$$

where λ_m denotes margin value, $|\cdot|_+$ means $\max(0, \cdot)$. For each observed interaction (u, p^+), we expect the relevance score γ_{up+}^t to be larger than that of any negative samples by λ_m, i.e., $|\lambda_m - \gamma_{up+}^t + \gamma_{up-}^t|_+ = 0$. Otherwise, we penalize each pair of (p^+, p^-) because of the incorrect ranking. $L(K) = \sum_{k=1}^{K} 1/k$, $rank(p^+)$ denotes relative ranking of positive sample p^+ among negative samples p^-, and $L(rank(p^+))$ is the penalty weight. Similarly we can define WARP loss for query prediction as $\mathcal{L}_q$. Thus, the overall objective is:

$$\mathcal{L} = \mathcal{L}_p + \mathcal{L}_q + \mathcal{L}_{KGC} + \|\Theta\|_2, \tag{4.10}$$

where Θ denotes model parameters, $\mathcal{L}_{KGC}$ denotes TransR loss to represent static facts in the product graph. Rich meta-data of products forms a heterogeneous product graph $\mathcal{G}_P$, describing important attributes of each product $p \in \mathcal{P}$. Specifically, $\mathcal{G}_P = \{(e, r, e')|e \in \mathcal{P}, e' \in \mathcal{I} \bigcup \mathcal{Q}, r \in \mathcal{R}_P\}$, where $\mathcal{I}$ denotes attribute set for products, including but not limited to brand, product type and category. $\mathcal{R}_P$ denotes the relation set among them. Each triple $(e, r, e') \in \mathcal{G}_P$ represents a fact indicating that product entity e associate with tail entity e' through relation r. $\mathcal{G}_P$ also describe *mapping* relations between products and queries.

To better represent product and query information from static product graph, we utilize a knowledge graph embedding module TransR [14] to project them into the same low-dimensional space. The objective is shown below:

$$\mathcal{L}_{KGC} = \sum_{(e,r,e')\in\mathcal{G}_P} \sum_{(e,r,e^-)\in\mathcal{G}_P^-} \max\left(0, f_r(e, e') + \lambda_{KGC} - f_r\left(e, e^-\right)\right) \tag{4.11}$$

where $\mathcal{G}_P$ denotes product graph, $\mathcal{G}_P^-$ denotes negative samples, λ_{KGC} is the margin value. Following TransR, we define $f_r(h, t) = \|\mathbf{h}\mathbf{W}_r + \mathbf{r} - \mathbf{t}\mathbf{W}_r\|_2^2$, where $\mathbf{W}_r$ is trainable relation matrix.

Algorithm 4.1: The optimization process for RETE.

Input: Evolutionary multi-relational graph $\{\mathcal{G}^1, \cdots, \mathcal{G}^T\}$ and product graph $\mathcal{G}_P$.
Output: User intents $\widetilde{\boldsymbol{h}}^{T+1}$ and model parameters Θ.

Ensemble samplers sample subgraphs for users;
while *model not converged* **do**
  Optimize on product graph:
  Minimize $\mathcal{L}_{KGC}$ and update entity representation in $\mathcal{G}_P$;
  Optimize on evolutionary graph:
  for *Each time step* $t < T$ *during training* **do**
    (Auto-regressive training:)
    Learn user intent $\{\boldsymbol{h}^1, \cdots, \boldsymbol{h}^{t-1}\}$ according to Eq. (4.6);
    Update new intents $\widetilde{\boldsymbol{h}}^t$ according to Eq. (4.7);
    Calculate ranking loss $\mathcal{L}_p^t$ and $\mathcal{L}_q^t$ at time step t;
  end
  Optimizing ranking loss $\mathcal{L}_p$ and $\mathcal{L}_q$ in Eq. (4.9);
end
Update user intents $\widetilde{\boldsymbol{h}}^{T+1}$ according to Eq. (4.7);

We optimize $\mathcal{L}_{\text{KGC}}$ and $\mathcal{L}_p + \mathcal{L}_q + \|\Theta\|_2$ alternatively, where mini-batch Adam is adopted.

4.3 THE EVALUATION OF RETE

4.3.1 Datasets

We collected one public Yelp dataset and four industrial E-commerce datasets for experiments:

Yelp. The dataset is adopted in Yelp Challenge 2019 [14], which contains the interaction records between users and businesses like restaurants and bars. We utilize review history between users and businesses as user-product interactions. And we extract discriminative keyphrases from reviews as queries, so as to collect pseudo user-query interactions. We are able to show that jointly consider both pseudo queries and products can improve both performances. We propose the following procedure to collect experimental data.

1. We collect data from April 2014 to Jan 2021 and preserve those with ratings higher than 3.0, as high ratings indicate true user intents.

2. To extract keyphrases as pseudo query, we utilize both AutoPhrase [103], 2-gram, and 3-gram methods to extract keyphrases. Then we calculate TF-IDF scores for each and

[14] https://www.yelp.com/dataset/

preserve top 15000 as pseudo query pools.

3. We adopt 20-core setting to collect entities, i.e., we remove users/products/queries with fewer interactions than 20.

4. We divide the time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6).

5. To construct product graph, we first preserve critical attributes like category, location, etc to construct product-to-attribute edge, we collect frequent pairs of businesses from background data as product-to-product edge, we collect query-to-product edge also from background data per each review action.

E-commerce. We gain access to the search log data including 140 days and product attribute data. We first collect data under four specific categories: *Electronics*, *Book*, *Music* and *Beauty*. For each category, We propose the following procedure to collect experimental data.

1. We collect data from Feb 2021 to June 2021 and preserve those with specific types of actions: click, add cart, follow-on click, and purchase, with ratios of all data 11.8%, 5.1%, 4.6% and 2.0% respectively. They are preserved as they reveal strong signals of user intents. For each action, we record user, query/product, timestamp, and action type.

2. We adopt 10-core setting to collect entities, i.e., we remove users/products/queries with fewer interactions than 10.

3. We divide the time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6).

4. To construct product graph, we first preserve critical attributes like brand, model, etc to construct product-to-attribute edge, we collect frequent pairs of products within the same sessions as product-to-product edge, we collect query-to-product edge also from background data if users have interaction to one product via one search query.

We purposefully choose the two platforms because of the various time period. The e-commerce dataset emphasizes more on user short-term interest since the shopping intent is more time-sensitive. By contrast, the Yelp dataset emphasizes more on user long-term interest, since it lasts longer and a user's choice on businesses is less time-sensitive.To evaluate our framework, we divide the time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6) to train initial entities embeddings (model input), train, validate and test our model respectively. Table 4.1 summarizes the statistics of the experimental datasets.

Table 4.1: Statistics of experimental datasets.

Dataset	Public	Industrial (E-commerce)			
	Yelp	**Electronics**	**Music**	**Book**	**Beauty**
#User	22,307	5,928	7,453	37,562	47,261
#Product	16,153	10,129	12,105	61,215	51,686
#Query	9,314	8,045	6,506	25,340	25,807
Product interactions	820,219	138,607	582,651	2,976,112	1,385,366
Query interactions	800,727	58,037	213,281	718,035	200,219
#Entity	49,269	29,212	28,643	134,370	140,314
#Triplet	1,791,788	496,701	1,832,501	7,739,316	3,092,010
Time span	80 months	140 days	140 days	140 days	140 days
#Time step	28	28	28	28	28

Table 4.2: Overall performance for product prediction. Average results on 5 independent runs are reported. $*$ indicates the statistically significant improvements over the best baseline, with p-value smaller than 0.001.

Dataset	**Public**		**Industrial E-commerce**							
	Yelp		**Electronics**		**Music**		**Book**		**Beauty**	
$K = 20$	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K
FM-based Recommendation										
FM	0.0221	0.0277	0.0512	0.0713	0.0641	0.0981	0.0682	0.0964	0.1155	0.1459
NFM	0.0214	0.0281	0.0715	0.1164	0.0761	0.1005	0.0793	0.1064	0.1246	0.1591
Sequential Recommendation										
BERT4Rec	0.0422	0.0501	0.0619	0.0832	0.0537	0.0618	0.0447	0.0651	0.0827	0.1015
GRU4Rec	0.0419	0.0511	0.0742	0.0859	0.0621	0.0711	0.0412	0.0658	0.0842	0.1003
Dynamic Graph Learning										
JODIE	0.0459	0.0527	0.1399	0.1515	0.1123	0.1405	0.1401	0.1881	0.1458	0.1807
KG-based recommendation										
KGAT	0.0342	0.0403	0.1503	0.1914	0.1156	0.1301	0.1254	0.1479	0.1503	0.1893
ECFKG	0.0388	0.0495	0.1413	0.1859	0.1036	0.1246	0.1327	0.1674	0.1401	0.1799
RETE (Ours)	**0.0499***	**0.0589***	**0.1703***	**0.2120***	**0.1304***	**0.1521***	**0.1455***	**0.1976***	**0.1621***	**0.1985***
Gain	*8.71%*	*11.76%*	*13.31%*	*10.76%*	*12.80%*	*8.27%*	*3.85%*	*5.05%*	*7.85%*	*4.86%*

4.3.2 Experimental Setup

Metrics. We evaluate temporal event forecasting task in a retrieval setting, i.e., we compare the predicted top-K ranking list of products/queries with the groundtruth in the testing time steps. We adopt two widely-used evaluation protocols: *Recall@K* and *NDCG@K*. By default, we set $K = 20$.

Baselines. We compare baselines from following areas:

- FM-based recommendation, which considers the second-order feature interactions.
  - **FM** [104]. This is a basic factorization model which considers the second-order connections. We construct input features as multi-hot vectors.
  - **NFM** [105]. This is a state-of-the-art factorization model, which subsumes FM under neural networks. Specially, we employed one hidden layer to extract features from

Table 4.3: Overall performance for product prediction.

Dataset	Public		Industrial E-commerce							
	Yelp		Electronics		Music		Book		Beauty	
$K = 20$	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K
FM-based Recommendation										
FM	0.0257	0.0319	0.0481	0.0765	0.0324	0.0681	0.0862	0.1015	0.0614	0.0854
NFM	0.0244	0.0331	0.0533	0.0709	0.0583	0.1188	0.0851	0.1103	0.0673	0.0903
Sequential Recommendation										
BERT4Rec	0.0407	0.0498	0.0602	0.0877	0.0207	0.0457	0.0413	0.0882	0.0417	0.0566
GRU4Rec	0.0381	0.0477	0.0590	0.0731	0.0436	0.0599	0.0401	0.0907	0.0513	0.0602
Dynamic Graph Learning										
JODIE	0.0461	0.0617	0.0779	0.0957	0.0988	0.1364	0.1301	0.1475	0.1327	0.1495
KG-based recommendation										
KGAT	0.0431	0.0527	0.0913	0.1153	0.0823	0.1324	0.1293	0.1497	0.1299	0.1502
ECFKG	0.0397	0.0481	0.0899	0.1099	0.0897	0.1259	0.1283	0.1503	0.1214	0.1518
RETE (Ours)	**0.0507***	**0.0653***	**0.1015***	**0.1393***	**0.1033***	**0.1408***	**0.1391***	**0.1557***	**0.1487***	**0.1643***
Gain	*9.98%*	*5.83%*	*11.17%*	*20.82%*	*4.55%*	*3.22%*	*6.92%*	*4.01%*	*12.06%*	*9.31%*

inputs.

- Sequential recommendation, which considers user evolving intents overtime.
  - **GRU4Rec** [106]. This is a sequential recommendation method that utilizes gated recurrent units (GRU) to learn temporal information of user action sequences.
  - **BERT4Rec** [107]. This is a sequential recommendation method that utilizes BERT to learn temporal information of user action sequences.
- KG-based recommendation, which models heterogeneous entities and high-order connections for recommendation.
  - **ECFKG** [96]. This is an advanced collaborative filtering framework that incorporates knowledge graph for recommendation and utilizes KG embedding to learn representations.
  - **KGAT** [94]. This is an end-to-end knowledge graph attention network that explicitly models the high-order connections and heterogeneous entities and employs an attention mechanism to discriminate the importance of the neighbors.
- Dynamic graph learning: which models evolutionary interaction graph.
  - **JODIE** [108]. This is a dynamic graph method that considers co-evolution of both users and products. We encode rich side information for it via initial embedding.

Setup and Implementation. It is worth noting that all baselines are designed for recommending products. To generalize them to be able to predict queries, we train them using product and query loss separately. Those (KGAT, ECFKG) that consider knowledge graph can explicitly fuse information from both product and query. For all methods that need

initialization, we utilize a classic and lightweight knowledge embedding method, TransR, to represent multi-relational information in background data, not just IDs. For dynamic methods (BERT4Rec, GRU4Rec, JODIE) that aim to only predict the next interacted product/query, we modify the evaluation protocol to predict all possible products/queries in the following time steps.

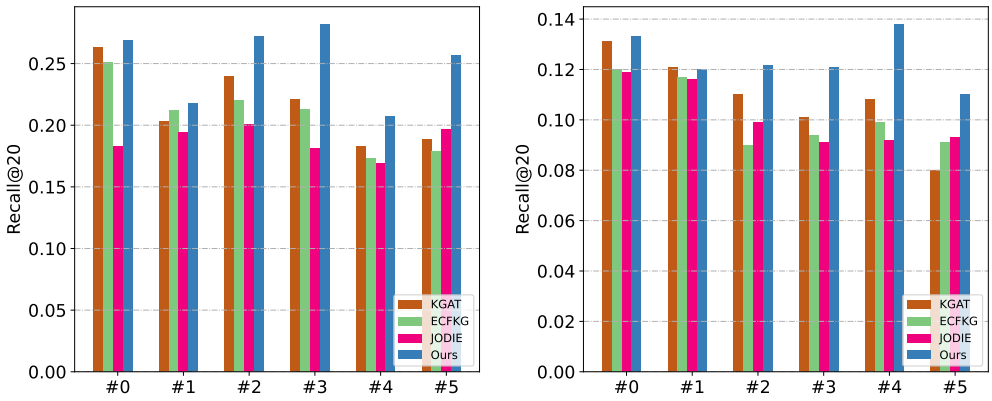
For fair comparison, we do not utilize rich semantic information from query entities and product descriptions, as all compared baselines only consider interaction records and structural side information. We fix the dimension of latent vectors of all methods as 128, and we report the average performance of the best model on the validation set. For RETE, we tune learning rate within $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$ and regularization weight $\{0.005,$
$0.05, 0.5\}$ according to *Recall*@20 of product prediction on validation set. We ensemble one PPR sampler and one randomized 3-hop sampler to retrieve subgraphs, and we stack 3 layers of graph attentions to better integrate information.

We implement RETE with Python 3.8.5. We use PyTorch 1.9.1 on CUDA 11.1 to train RETE on GPU. To implement fast subgraph sampling, we adopt methods in [99], [109] to calculate the approximate PPR value by only traversing the local region around each user. For better efficiency, we implement the sampling part with C++ and the interface between C++ and Python is via PyBind11.

4.3.3 Model Performance

Overall performance. We first compare overall performance of product prediction and query prediction with selected baselines, as shown in Table 4.2 and Table 4.3. In most cases, FM-based (FM, NFM) and sequential recommendation (BERT4Rec, GRU4Rec) methods produce poor results, as they do not explicitly consider higher-order interactions. RETE beats KG-based methods (KGAT, ECFKG) and dynamic graph learning method (JODIE) on all metrics, as we propose a better way to integrate multi-relational data in a temporal manner. Notably, on E-commerce platform, query prediction has worse performance than predicting products, while Yelp platform exhibits different pattern. We hypothesis that it is because the real queries from users on E-commerce platform are more diverse than pseudo queries extracted from Yelp review data. Also, it is harder to produce accurate prediction on Yelp platform, as Yelp data are collected from much longer period, where user intent shifting and evolution across time step are much harder to capture.

Detailed performance. Further, to investigate how does RETE perform over time, we compare the detailed performances in each testing time step (6 time step), as shown in



(a) Product. (b) Query.

Figure 4.3: Recall@20 of each test period on *Electronics*, significant improvements come from the last several time steps.

Figure 4.3. The performances in different time steps vary largely, indicating user intents are evolving and shifting. RETE can beat others in almost all time steps, and more significant improvements come from the last several time steps, which shows our proposed temporal module can capture the evolution of user preference and thus achieve better long-term performance.

Auto-regressive evaluation. As users keep interacting with E-commerce platforms, new interaction events are collected continuously. In real scenario, it is required that the deployed models can take newly collected data to update user representations without time-consuming retraining or fine-tuning. We refer it to *auto-regressive* evaluation and verify the robustness of RETE under it. As it is hard to update static models on new data without retraining, we mainly focus on comparing with dynamic models (BERT4Rec, GRU4Rec and JODIE). Given new testing data, we continuously fed them into the temporal module and evaluate the performance in the next time step. Table 4.4 reports the average performance on testing time steps. All compared models achieve improved results after considering newly collected data, as which contain more up-to-date clues to capture users' intents. RETE can achieve the best performance, showing better generalization ability and robustness for continual learning.

4.3.4 Ablation Study

To investigate how each component affects the model performance, we conduct the following ablation studies, as shown in Table 4.5: **The Effects of Various Types of Information**. Our solution constructs a temporal KG to organize multi-relational data for joint product and query prediction. As expected, removing rich attributes causes a performance

Table 4.4: Performance under the auto-regressive evaluation. Average results on six testing time steps are reported.

Task	**Product prediction**			
Dataset	**Electronics**		**Music**	
k = 20	**NDCG@K**	**Recall@K**	**NDCG@K**	**Recall@K**
BERT4Rec	0.0830	0.1232	0.0566	0.0701
GRU4Rec	0.1099	0.1201	0.0519	0.0803
JODIE	0.1801	0.1962	0.1371	0.1507
Ours	**0.1961**	**0.2414**	**0.1561**	**0.1733**
Task	**Query prediction**			
Dataset	**Electronics**		**Music**	
k = 20	**NDCG@K**	**Recall@K**	**NDCG@K**	**Recall@K**
BERT4Rec	0.0861	0.1019	0.0455	0.0634
GRU4Rec	0.0661	0.0913	0.0501	0.0633
JODIE	0.1259	0.1526	0.1203	0.1499
Ours	**0.1425**	**0.1793**	**0.1352**	**0.1631**

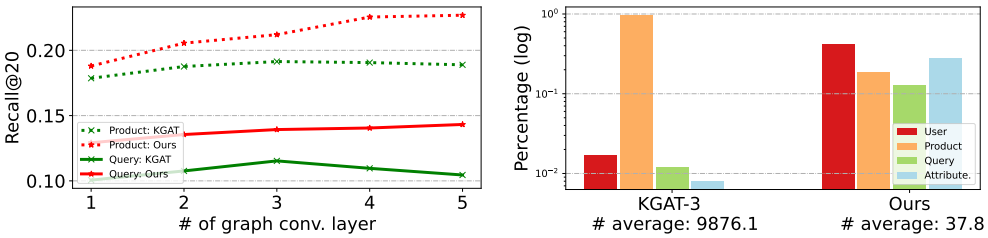
Table 4.5: Ablation study evaluated by Recall@20.

Datasets	**Electronics**		**Music**	
Ablations	**Product**	**Query**	**Product**	**Query**
Variants on how to construct input graph:				
w/o attr.	0.1686	0.1037	0.1154	0.1132
w/o query	0.1749	-	0.1335	-
w/o product	-	0.0973	-	0.0943
Variants on subgraph sampler:				
Only k-hop sampler	0.1991	0.1203	0.1363	0.1367
Only PPR sampler	0.2123	0.1381	0.1501	0.1399
Static v.s. dynamic:				
Ours (static)	0.1931	0.1183	0.1299	0.1327
Ours	**0.2120**	**0.1393**	**0.1521**	**0.1408**

hit, because they provide reliable relations among products and queries. Furthermore, we can demonstrate the mutual benefit derived from the joint query and product prediction task via the performance degradation after removing product entities and query entities, respectively. Interestingly, removing products can significantly affect query prediction performance. This is because queries are mainly connected by product nodes, and a large ratio of query connections is ignored in this ablation.

The effects of ensemble subgraph samplers. We propose an ensemble subgraph sampler to retrieve relevant entities and filter unrelated noise from the whole graph. Different samplers can capture data characteristics from different perspectives. Only using the k-hop sampler or PPR sampler can hurt the performance compared with ensembling them together. The PPR sampler behaves better than the k-hop sampler, as PPR value can better reflect the relevance among entities when raw neighbor information.

The effects of temporal module. To evaluate the impact of the temporal attention module, we compare the performance of our static variant. Our dynamic model can have



(a) Effects of # of layers.

(b) Entity type distribution.

Figure 4.4: Retrieval analysis. RETE manages to improve performance by stacking more layers, and it can retrieve much more balanced information via a reasonable number of retrieved entities.

$\sim 10\%$ relative improvements over the static variant.

Analysis of ensemble sampler.

We analyze how our ensemble subgraph sampler can improve retrieval results by collecting related higher-order entities and filtering out a large ratio of noises. As shown in Figure 4.4, unlike KGAT, RETE manages to improve performance by stacking more layers (by default, we choose 3. To investigate the quality of the retrieved entities, Figure 4.4 shows the distributions among entities types as well as average amount of the retrieved entities. RETE can integrate diverse and balanced information via retrieving a reasonable amount of entities. In contrast, KGAT with 3 layers integrates noisy information from over 9000 entities for each user, where over 96.4% integrated entities are products.

4.3.5 Temporal Analysis and Case Study

To investigate the evolution of user preference and how RETE can capture it, we select one user from the Electronics dataset with 76 event records. As shown in Figure 4.5, from the most frequent event in each time step, we can observe an obvious interest shift from mobile tablet-related items to Nintendo Switch-related items. Weights before star are used for auto-regressive forecasting. The temporal module can emphasize more on related events and less on unrelated events at new time steps.

4.4 RELATED WORK

We discuss and compare existing works from three related areas, as summarized in Table 4.6.

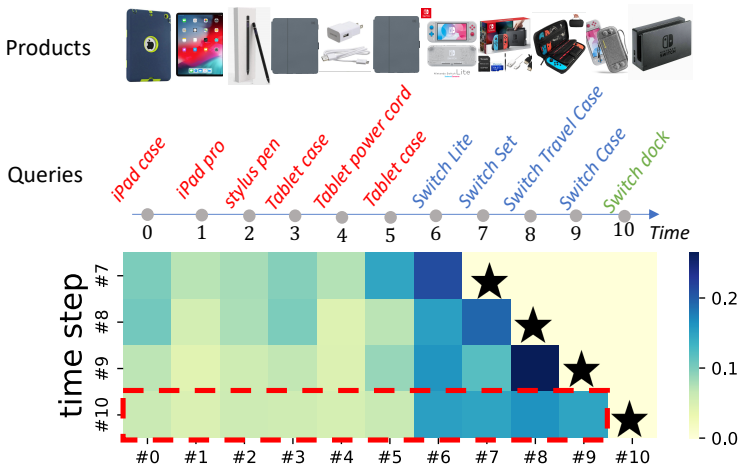


Figure 4.5: Case study of temporal attention. To forecast the new event at $t = 10$, it emphasizes more on related events, and less on unrelated events. At each time step, we report and summarize the most frequent event.

4.4.1 Product recommendation

Recommendation systems (RS) have achieved huge success on E-commerce platforms. Numerous efforts are devoted to improve RS from different perspectives: factorization machine (FM-based) models [104], [105], [116] efficiently consider abundant input features, sequential/session-based recommendation models [106], [107], [110]–[112], [117] capture user-side dynamics from long/short time periods respectively, and knowledge graph (KG-based) [94]–[96] models consider higher-order connections in multi-relational graph and produce explainable prediction. However, existing works merely focus on product-side interactions (e.g, user-to-product [104], [105], [108], [115], product-to-product [106], [107], [118], or product-to-attribute [93]–[96]) but neglect equally important search queries. We study a more generic temporal event forecasting task to jointly optimize product and query prediction on evolutionary knowledge. A few interactive recommendation works on knowledge graph optimize sequential policy for recommending products within short sessions [119]. In contrast, we focus our attention on studying how to better integrate time-aware information from evolutionary knowledge graph and exclude unrelated noises in order to improve long-term performance for both product prediction and query prediction.

4.4.2 Query suggestion

Query suggestion task aims to assistant users to better express their intents on various search engine systems. General query suggestion task includes three different objectives: query rewriting (QR) [120], [121], query auto-completion (QAC) [122], [123] and query

Table 4.6: Comparison with existing settings. Action history refers to various past users' behaviors towards products such as "Add", "Click" and "Purchase", etc. Search history denotes historical user search queries. Meta data refers to product side information like attributes.

Properties		Action history	Search history	Meta data	Temporal Info.	Multi objective
Recommendation	FM-based [104], [105]	✓	X	✓	X	Product
	Sequential / Session-based [106], [107], [110]–[112]	✓	X	X	✓	Product
	KG-based [93]–[96]	✓	X	✓	X	Product
Query suggestion [113], [114]		X	✓	X	X	Query
Dynamic Graph Learning [108], [115]		✓	X	X	✓	Product
Temporal Event Forecasting (**Ours**)		✓	✓	✓	✓	Product & Query

prediction [113], [114]. While QR and QAC focus on reformulating the queries that help the users to find better search results of current search intents, we specifically study query prediction that "recommend" queries that potentially match user intents. Existing works on recommending/predicting queries utilize search log data [114], query dependency graph [124] or interaction history with users [113]. Instead, to the best of our knowledge, we are the first to propose and study joint product and query prediction task on E-commerce. Inspired by the recent success of combining both queries and documents for jointly information retrieval in search engine systems [125], we mainly focus on exploring mutual benefits from both queries and products. One major difference is that we only consider structured graph data, instead of content, as query contents are much more private and sensitive on E-commerce.

4.4.3 Dynamic graph learning

Graph learning methods [60], [126]–[130] has been widely explored for recommendation [94], [131], user modeling [6], [44], [49], [132]–[134], etc. Recently, dynamic algorithms are proposed to better capture the temporal patterns of evolutionary graph [2], [92], [108], [127], [135]. As the constructed query product evolutionary graphs are more diverse and complicated, it brings new challenges to deal with over-smoothing issues and to capture temporal behavior patterns at the same time. Hence we aim to design a dynamic model with subgraph samplers to better learn informative and time-aware user representations. Although existing works have explored combinational power of subgraph samplers with graph neural networks on static and homogeneous/bipartite graph [98], [99], we further facilitate dynamic graph learning with subgraph samplers on the evolutionary knowledge graph.

4.5 SUMMARY

In this chapter, we explore temporal event forecasting, a new problem considering the temporal influence from both query and product to user behaviors. To enhance the sparse action information of most users and meanwhile capture the evolution of user intents, we propose a novel RETE framework to efficiently retrieve similar entities as subgraphs to enrich the user profile representation and then auto-regressively adapt it to be time-aware. We evaluate the proposed RETE method on both product-centric and query-centric event prediction tasks. Extensive experiments on both public and industrial datasets quantitatively and qualitatively demonstrate the effectiveness of the proposed method.

Chapter 5: LEARNING DYNAMIC LOW-RESOURCE GRAPHS WITH KNOWLEDGE TRANSFER

5.1 OVERVIEW

In this chapter, we investigate temporal reasoning problem on scarce temporal graphs, which aims to facilitate reasoning on temporal graphs in low-resource domain by transferring knowledge from graphs in high-resource ones. The cross-network distillation ability, as a way to enable the external knowledge transfer capability, becomes increasingly crucial, in light of the unsatisfying performance of existing reasoning methods on those severely incomplete temporal graphs, especially in low-resource domain.

Inspired by the incompleteness issue facing low-resource languages in constructing temporal graphs, we introduce a novel task named Cross-Lingual Temporal Knowledge Graph Reasoning (as shown in Figure 5.1). This task aims to alleviate the reliance on supervision for temporal graphs in low-resource languages (referred to as the target language) by transferring temporal knowledge from high-resource languages (referred to as the source language). In contrast, all the existing efforts are either limited to reasoning in monolingual temporal graphs (usually high-resource languages, e.g., English) [22], [33], [136], [137], or multilingual static KGs [138]–[140]. To the best of our knowledge, cross-lingual temporal graph reasoning that transfers temporal knowledge between temporal graphs has not been investigated.

The fulfillment of this task poses tremendous challenges in two aspects: 1) **Scarcity of cross-lingual alignment**: as the informative bridge of two separate temporal graphs, cross-lingual alignment is imperative for cross-lingual knowledge transfer [138]–[140]. However, obtaining alignments between languages is a time-consuming and resource-intensive process that heavily relies on human annotations. The transfer of knowledge through a limited number of alignments is often insufficient to fully enhance the temporal graph in the target language. 2) **Temporal knowledge discrepancy**: the information associated with two aligned entities is not necessarily identical, especially with regards to temporal patterns. Utilizing a rough approach to equate the aligned entities at all times can result in the transfer of misleading knowledge and negatively impact performance. This becomes more pronounced when the alignments are noisy and unreliable. For example, at the time step t_2, a new event about operating system "*Ventura*" from Apple company occurs in the source English temporal graph, and meanwhile there is a noisy aligned entity "*Ventura city*" in the target Japanese temporal graph. Directly pulling those two entities at this point, can inevitably introduce noise and fail to predict a set of related events in the target temporal graph. Therefore, it is crucial to dynamically regulate the alignment strength of each local graph structure over

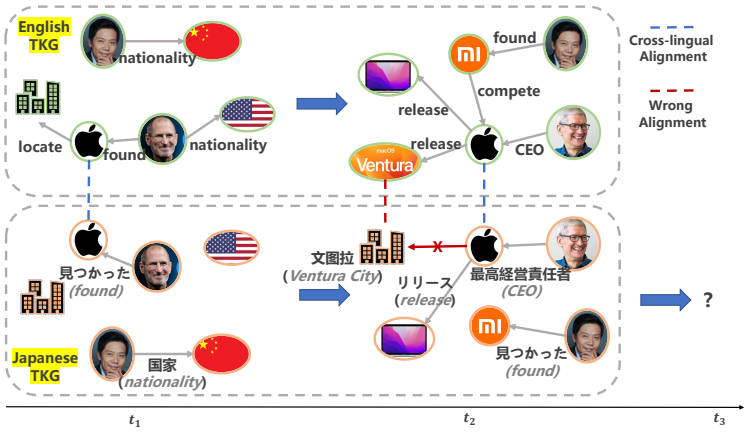


Figure 5.1: An illustrative example of cross-lingual reasoning on temporal graphs. 1) We aim to transfer knowledge from English temporal graph to Japanese temporal graph, where the English version provides more complete information; 2) Cross-lingual alignments only cover a small ratio of entities, e.g., Apple Inc; 3) Cross-lingual alignments can be noisy and misleading, e.g., A city called Ventura is linked to new macOS Ventura at t_2, introducing noise for reasoning in Japanese.

time in order to maximize the effectiveness of cross-lingual knowledge distillation.

In this chapter, we propose a novel Mutually-paced Knowledge Distillation (MP-KD) framework, where a teacher network learns more enriched temporal knowledge and reasoning skills from the source temporal graph to facilitate the learning of a student network in the low-data target one. The knowledge transfer is enabled via an alignment module, which estimates entity correspondence across languages based on temporal patterns. Firstly, to alleviate the limited language alignments (**Challenge #1**), such a knowledge distillation process is mutually paced over time. This means, on one hand, we encourage the mutually interactive learning between the teacher and student. Concretely, the alignment module between the teacher and the student learns to generate pseudo alignment between temporal graphs to maximally expand the upper bound of knowledge transfer. And subsequently, it empowers the student to encode more informative knowledge in target temporal graph, which can in turn boost the alignment module to explore more reasonable alignments as the bridge across temporal graphs. One the other hand, inspired by self-paced learning [141], [142], we make the generations as a progressively easy-to-hard process over time. We start from generating reliable pseudo data with high confidence. As time goes by, we then gradually increase the generation amount by relieving the restriction over time. Secondly, to inhibit the temporal knowledge mismatch (**Challenge #2**), the attention module can estimate the graph alignment strength distribution over time. This is achieved by a temporal cross-lingual attention in terms of the local graph structure and temporal-evolving patterns of aligned

Table 5.1: Symbols and Notations.

Symbol	Definition
(e, r, e', t)	A quadruple in graph.
$\mathcal{G}_s$, $\mathcal{G}_t$	Source graph and Target graph.
e_s, e_t	Entities in the source and target graphs.
$\Gamma_{s\leftrightarrow t}$	Alignments between the source and target graphs.
$\tilde{\mathcal{G}}_t$, $\tilde{\Gamma}_{s\leftrightarrow t}$	Incomplete target graph and alignments.
$\tilde{\mathcal{G}}_t^{ST}$, $\tilde{\Gamma}_{s\leftrightarrow t}^{ST}$	Pseudo target graph and pseudo alignment.
$f(\cdot;\Theta_s)$	Teacher network on the source graph.
$f(\cdot;\Theta_t)$	Student network on the target graph.
$g(e_s, e_t, t; \Phi)$	Alignment module measuring correspondence of (e_s, e_t) at t.
$\mathcal{L}_{\tilde{\mathcal{G}}_t}$, $\mathcal{L}_{\tilde{\mathcal{G}}_t^{ST}}$	Reasoning loss on groundtruth/pseudo target graph.
$\mathcal{L}_{\tilde{\Gamma}_{s\leftrightarrow t}}$, $\mathcal{L}_{\tilde{\Gamma}_{s\leftrightarrow t}^{ST}}$	Alignment loss on groundtruth/pseudo alignment pairs.
$\mathcal{L}_{s\to t}$	Cross-lingual reasoning loss from source graph to target graph.
$\mathcal{L}_{s\to t}^{ST}$	Cross-lingual reasoning loss on both groundtruth and pseudo data.

entities. As such, it can dynamically control the negative effect and suppress noise propagation from the source temporal graph. Moreover, we provide a theoretical convergence guarantee for the training objective on both initial ground-truth data and pseudo data. To evaluate MP-KD, we conduct extensive experiments of 12 cross-lingual temporal graph transfer tasks in multilingual EventKG graph dataset [143]. Our empirical results show that the MP-KD method outperforms state-of-the-art baselines in both with and without alignment noise settings, where only 20% of temporal events in the target KG and 10% of cross-lingual alignments are preserved.

5.2 PRELIMINARIES AND NOTATIONS

In this section, we formally define the cross-lingual temporal knowledge graph reasoning task, and summarize the notations in Table 5.1. A temporal knowledge graph can be defined as follows:

Definition 5.1 (Temporal Graphs). A temporal graph is denoted as $\mathcal{G} = \{(e, r, e', t)|t \leq T\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$, where $\mathcal{E}$ denotes the entities set, $\mathcal{R}$ denotes the relation set, $\mathcal{T}$ denotes the timestamp set, and T denotes the latest update time. Each quadruple (e, r, e', t) refers to an event that a subject entity $e \in \mathcal{E}$ has a relation $r \in \mathcal{R}$ with an object entity $e' \in \mathcal{E}$ at timestamp $t \in \mathcal{T}$.

Definition 5.2 (Multilingual graphs and Alignments). To denote multilingual graphs, we further utilize subscript to represent specific languages, i.e., $\mathcal{G}_s$ denotes graph in the source language and $\mathcal{G}_t$ denotes graph in the target language. The corresponding entities can be

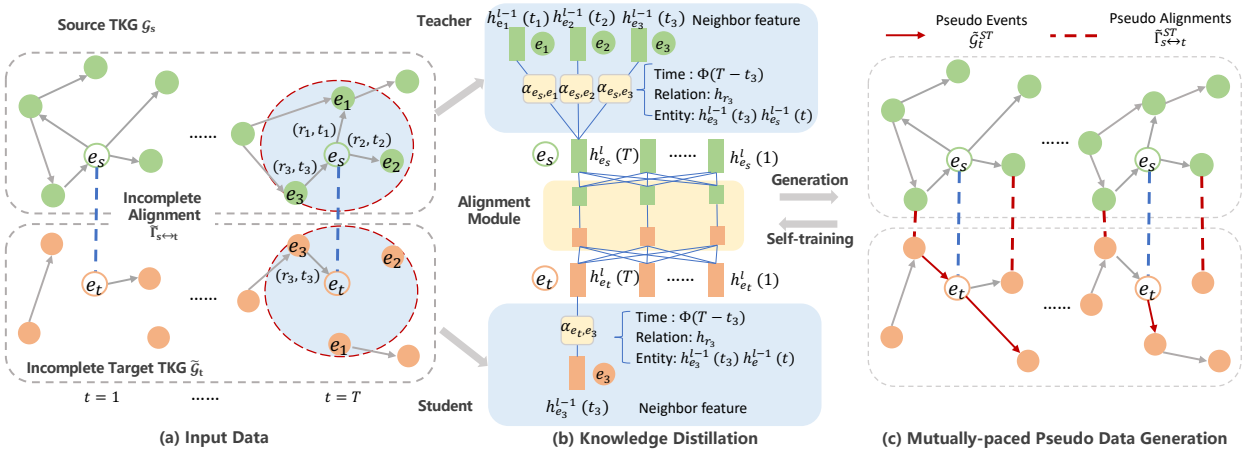


Figure 5.2: An overview of MP-KD. (a) The source graph is more complete than the target graph, and the cross-lingual alignments are also scarce; (b) A teacher/student representation module to represent source/target graph, and an alignment module for knowledge transfer; (c) Mutually-paced knowledge distillation between knowledge transfer and pseudo alignment generation.

denoted as e_s and e_t respectively. Given two different languages s,t, we have the cross-lingual alignment set $\Gamma_{s\leftrightarrow t}$. To be more practical, we further assume the graph in target language $\mathcal{G}_t$ and alignment set $\Gamma_{s\leftrightarrow t}$ are incomplete: $\tilde{\mathcal{G}}_t$, $\tilde{\Gamma}_{s\leftrightarrow t}$.

Based on the definition above, we formalize our cross-lingual reasoning task on graphs as follows:

Definition 5.3 (Cross-lingual reasoning on graphs). Given the graph $\mathcal{G}_s$ in the source language and the incomplete graph $\tilde{\mathcal{G}}_t$ in the target language before the latest update time T, and the incomplete cross-lingual alignment $\tilde{\Gamma}_{s\leftrightarrow t}$, we aim to predict future events in the target graph after time T. Concretely, we aim to predict missing entity in each future quadruple: $\{(e_t, r, ?, t)$ or $(?, r, e'_t, t)|t > T\}$ in the target graph.

5.3 THE DESIGN OF MP-KD

In this section, we present the proposed MP-KD framework for the cross-lingual temporal knowledge graph reasoning task.

5.3.1 Overview

Figure 3.2 shows an overview of MP-KD. Given the graphs in source language and target language, the teacher network and the student network first represent the source and

target graphs in a temporally evolving uni-space respectively. To facilitate training of the student, the knowledge distillation is enabled by a cross-lingual alignment module and an explicit temporal event transfer process. To deal with the scarcity issue of cross-lingual alignments, we propose a pseudo alignment generation technique to facilitate the knowledge distillation process, which is mutually-paced along with model training. To address the temporal knowledge discrepancy issue, the alignment module pulls the aligned entities close to each other based on the alignment strength which is dynamically adjusted.

Section 5.3.2 and Section 5.3.3 introduce our teacher/student network and the knowledge distillation respectively, followed by Section 5.3.4 which details how we generate pseudo alignments. Finally, Section 5.3.5 specifies our learning objective on both groundtruth data and pseudo data, and summarizes the training of MP-KD.

5.3.2 The Teacher/Student Network

We train two identical temporal representation modules on source and target graphs with different parameters. The representation module $f(\cdot;\Theta)$ parameterized by Θ is designed to measure the plausibility of each quadruple, which represents each entity e into a low-dimensional latent space at each time: $\mathbf{h}_e(t) \in \mathbb{R}^d$. On a graph $\mathcal{G}$, entities $e \in \mathcal{E}$ are evolving, as they interact with different entities over time. Such temporally interacted entities are defined as temporal neighbors. Therefore, we aim to model the temporal pattern of each entity e by encoding the changes of temporal neighbors.

Towards this goal, $f(\cdot;\Theta)$ first samples temporal neighbors $\mathcal{N}_e(t)$ from the graph for each entity $e \in \mathcal{E}$. $\mathcal{N}_e(t)$ consists of a set of the most recently interacted entities at time t. Then $f(\cdot;\Theta)$ attentively aggregates information from the temporal neighbors. Specifically, given the temporal neighbor $\mathcal{N}_e(t)$, we represent the entity e as $\mathbf{h}_e(t)$ at time t:

$$\mathbf{h}_e^l(t) = \sigma\left(\sum_{(e_i,r_i,t_i)\in\mathcal{N}_e(t)} \alpha_{e,e_i}^l \left(\mathbf{h}_{e_i}^{l-1}(t_i)\mathbf{W}\right)\right), \tag{5.1}$$

where l denotes the layer number, $\sigma(\cdot)$ denotes the activation function $ReLU$, α_{e,e_i}^l denotes the attention weight of entity e_i to the represented entity e, and $\mathbf{W}$ is the trainable transformation matrix. To aggregate from history, α_{e,e_i}^l is supposed to be aware of entity feature, time delay and topology feature induced by relations. Thus, we design the attention weight α_{e,e_i}^l as

follows:

$$\alpha^l_{e,e_i} = \frac{\exp(q^l_{e,e_i})}{\sum\limits_{(e_k,r_k,t_k)\in\mathcal{N}_e(t)} \exp(q^l_{e,e_k})}, \quad q^l_{e,e_k} = \mathbf{a}\left(\mathbf{h}^{l-1}_e \| \mathbf{h}^{l-1}_{e_k} \| \mathbf{h}_{r_k} \| \kappa(t-t_k)\right), \tag{5.2}$$

where q^l_{e,e_k} measures the pairwise importance by considering the entity embedding, relation embedding and time embedding, $\mathbf{a} \in \mathbb{R}^{4d}$ is the shared parameter in the attention mechanism. Following [144] we adopt random Fourier features as time encoding $\kappa(\Delta t)$ to reflect the time difference.

To measure plausibility of each possible quadruple, we utilize TransE [9] as the score function $f(e, r, e', t; \Theta) = -\|\mathbf{h}^l_e(t) + \mathbf{h}_r - \mathbf{h}^l_{e'}(t)\|^2$, where true quadruples should have higher scores. To optimize the parameter Θ on a graph $\mathcal{G}$, we set the objective to rank the scores of true quadruples higher than all other false quadruples produced by negative sampling:

$$\mathcal{L}_\mathcal{G} = \mathop{\mathbb{E}}_{(e,r,e',t)\in\mathcal{G}} \left[\max(0, \lambda_1 - f(e, r, e', t; \Theta) + f(e, r, e^-, t; \Theta))\right], \tag{5.3}$$

where (e, r, e^-, t) is negative samples with object e' replaced by e^-, λ_1 is the margin to distinguish positive and negative quadruples.

5.3.3 Knowledge Distillation

The incomplete target graph, $\tilde{\mathcal{G}}t$, can be used to train the corresponding parameter Θ_t through minimization of $\mathcal{L}\tilde{\mathcal{G}}_t$. However, the low-resource nature of the target language often results in an incomplete target graph, leading to suboptimal Θ_t. In light of this, we propose a knowledge distillation approach to transfer temporal knowledge from the source graph to the target graph. The proposed approach consists of two components: an alignment module that enhances Θ_t using the more informative Θ_s learned from the source graph, and an explicit temporal event transfer based on the improved parameters. This integrated approach aims to improve the completeness and quality of the target graph by leveraging the knowledge contained in the source graph.

The Alignment Module. In general, the source parameters Θ_s provide a more informative representation of each entity $e \in \mathcal{E}$ compared to the target parameters Θ_t. To take advantage of this, we utilize Θ_s to guide the optimization of Θ_t through the alignment module $g(\cdot; \Phi)$, which measures the correspondence between each pair of entities and is parameterized by Φ.

Directly pulling embeddings of aligned entities at all time steps can transfer misleading knowledge due to the temporal knowledge discrepancy. Therefore, the alignment module first

utilizes a temporal attention layer to integrate information of each entity from history in both source and target graphs, i.e., $\mathbf{H}_e^s(t), \mathbf{H}_e^t(t) \in \mathbb{R}^d$, then it pulls such integration $\mathbf{H}_e^s(t)$ close to $\mathbf{H}_e^t(t)$ instead of the initial $\mathbf{h}_e^s(t)$ and $\mathbf{h}_e^t(t)$. Moreover, the temporal integration $\mathbf{H}_e^s(t)$ and $\mathbf{H}_e^t(t)$ also encode the temporal evolution information for each entity, which can be utilized to estimate the adaptive alignment strength at different time to improve the alignment module. Concretely, the temporal integration is learned by:

$$\begin{aligned} \mathbf{H}_e^s(t) &= \text{Temporal-Attn}(\mathbf{h}_e^s(1), \mathbf{h}_e^s(2), \cdots, \mathbf{h}_e^s(t)), \\ \mathbf{H}_e^t(t) &= \text{Temporal-Attn}(\mathbf{h}_e^t(1), \mathbf{h}_e^t(2), \cdots, \mathbf{h}_e^t(t)), \end{aligned} \tag{5.4}$$

$$g(e_s, e_t, t; \Phi) = \frac{\mathbf{H}_e^s(t) \cdot \mathbf{H}_e^t(t)}{\|\mathbf{H}_e^s(t)\|_2 \cdot \|\mathbf{H}_e^t(t)\|_2}, \tag{5.5}$$

where Temporal-Attn is the temporal attention network designed to integrate information on the temporal domain. The correspondence between each pair of entities (e_s, e_t) across source and target languages at time t is measured by $g(e_s, e_t, t; \Phi)$. As the temporal knowledge for aligned entities is not identical, the alignment strength between them should vary across time t. The alignment strength is strong when the two entities share similar information, and weak when the information is dissimilar or the alignment is unreliable. This variability is achieved through the design of a trainable weight $\beta_{e,t}$ to adjust the alignment strength for different entities at different times, which is generated by a cross-lingual attention layer:

$$\beta_{e,t} = \text{Cross-Attn}(key = \mathbf{H}_e^t(1:T), query = \mathbf{H}_e^s(1:T))_{tt}. \tag{5.6}$$

We first introduce the general attention mechanism we utilized, then specify the temporal attention layer and cross-lingual attention layer for entity alignments.

Given two representation sequence from temporal domain: key sequence consisting of input representation sequence $\boldsymbol{H}_K = \{\boldsymbol{h}_K^1, \boldsymbol{h}_K^2, \cdots, \boldsymbol{h}_K^T\}$ and corresponding query sequence $\boldsymbol{H}_Q = \{\boldsymbol{h}_Q^1, \boldsymbol{h}_Q^2, \cdots, \boldsymbol{h}_Q^T\}$ from all time steps, we propose the following attention to calculate the pairwise importance:

$$\beta = \text{Attn}(key = \boldsymbol{H}_K, query = \boldsymbol{H}_Q) = \text{softmax}\left(\frac{\boldsymbol{H}_Q \boldsymbol{W}^Q (\boldsymbol{H}_K \boldsymbol{W}^K)^T}{\sqrt{d}} + \boldsymbol{M}\right), \tag{5.7}$$

where $\boldsymbol{W}^Q$, $\boldsymbol{W}^L$ are trainable temporal parameters, β is learned temporal weight indicating pairwise importance, d denotes dimension of input representations, and $\boldsymbol{M}$ is added to ensure auto-regressive setting, i.e., preventing future information affecting current state. We define $\boldsymbol{M}_{ij} = 0$ if $i \leq j$, otherwise $-\infty$.

For *temporal attention* layer, we use $\boldsymbol{h}_e = \{\boldsymbol{h}_e(1), \boldsymbol{h}_e(2), \cdots, \boldsymbol{h}_e(T)\}$ for both query and key sequence to obtain the temporal attention weights β:

$$\beta = \text{Attn}(key = \boldsymbol{h}_e, query = \boldsymbol{h}_e), \tag{5.8}$$

then the desired $\boldsymbol{H}_e(t)$ is leaned as the combination of input sequence, where $\boldsymbol{W}^V$ is a trainable matrix.:

$$\begin{aligned} \boldsymbol{H}_e(t) &= \text{Temporal-Attn}(\boldsymbol{h}_e(1), \cdots, \boldsymbol{h}_e(t)) \\ &= \sum_{i=1}^{t} \beta_{it} \boldsymbol{h}_e(i) \boldsymbol{W}^V \end{aligned} \tag{5.9}$$

For *cross-lingual attention* layer, we use $\boldsymbol{H}_e^s = \{\boldsymbol{H}_e^s(1), \cdots, \boldsymbol{H}_e^s(T)\}$ in source language as query sequence and $\boldsymbol{H}_e^t = \{\boldsymbol{H}_e^t(1), \cdots, \boldsymbol{H}_e^t(T)\}$ in target language as key sequence to obtain the attention weights β:

$$\beta_{e,t} = \text{Attn}(key = \boldsymbol{H}_e^t, query = \boldsymbol{H}_e^s)_{tt}, \tag{5.10}$$

where $\beta_{e,t}$ is trainable weight to adjust the alignment strength of different entities at different time.

To optimize the parameter Φ on the incomplete alignments $\tilde{\Gamma}_{s \leftrightarrow t}$, we set the objective in order to rank the correspondence of true alignments higher than false alignments:

$$\mathcal{L}_{\tilde{\Gamma}_{s \leftrightarrow t}} = \mathop{\mathbb{E}}_{\tilde{\Gamma}_{s \leftrightarrow t}} \left[\mathop{\mathbb{E}}_{t \in \mathcal{T}} \left[\beta_{e,t} \cdot \max(0, \lambda_2 - g(e_s, e_t, t; \Phi) + g(e_s, e_t^-, t; \Phi)) \right] \right], \tag{5.11}$$

where the entity pair $(e_s, e_t) \in \tilde{\Gamma}_{s \leftrightarrow t}$ is the aligned entities across languages, (e_s, e_t^-) is the negative samples, λ_2 is the margin value.

Temporal Event Transfer. Cross-lingual alignments offer the potential to directly transfer temporal events towards the progressive completion of the target graph. This is based on the premise that entities that are reliably aligned are likely to experience similar temporal events across languages, with the same relations.

Given an aligned pair (e_s, e_t), the temporal event $(e_t, r, e_t^?, t)$ or $(e_t^?, r, e_t, t)$ is added to the target graph if the corresponding event $(e_s, r, e_s^?, t)$ or $(e_s^?, r, e_s, t)$ exists in the source graph $\mathcal{G}_s$. To determine the missing entity $e_t^?$, we first verify if $(e_s^?, e_t^?)$ is present in the alignment set. If so, the temporal event is directly added to the target graph. Otherwise, the updated student network $f(\cdot; \Theta_t)$ is utilized to predict the missing entity and the top-1 entity is utilized to complete the temporal event. We define the set of transferred temporal events in the target graph as $\tilde{\mathcal{G}}_t^{ST}$ for ease of discussion.

5.3.4 Generating Pseudo Alignments

The limited amount of cross-lingual alignments negatively constrain the effect of the knowledge distillation process. In this section, we introduce how to generate pseudo alignments $\tilde{\Gamma}^{ST}_{s\leftrightarrow t}$ with high confidence to boost cross-lingual transfer effectiveness.

To expand the range of alignments used in the knowledge transfer process, we generate pseudo-alignments with high confidence scores and incorporate them into the training data. The confidence score for each pair of entities (e_s, e_t) is calculated as the average cosine similarity: $sim(e_s, e_t) = \mathop{\mathbb{E}}_{t}[g(e_s, e_t, t; \Phi)]$. While pair-wise similarity comparison is computationally intensive, we improve efficiency by first adding alignments for entities that are neighbors of already aligned entities $\tilde{\mathcal{E}}t = \{e_t|(e_s, e_t) \in \tilde{\Gamma}_{s\leftrightarrow t}\}$ in the target graph, as they are likely to be represented well to produce reliable alignment. Following [145], we formulate the generation process as solving the following optimization problem:

$$\begin{aligned} \max \sum_{e_t \in \mathcal{N}(\tilde{\mathcal{E}}_t)} \sum_{e_s \in \mathcal{E}_s} & sim(e_s, e_t) \cdot \phi(e_s, e_t), \\ s.t. \sum_{e_s \in \mathcal{E}_s} \phi(e_s, e_t) = 1, & \quad \sum_{e_t \in \mathcal{N}(\tilde{\mathcal{E}}_t)} \phi(e_s, e_t) = 1, \end{aligned} \tag{5.12}$$

where $\phi(e_s, e_t)$ is a binary indicator of whether to add (e_s, e_t) as pseudo alignment, $\phi(e_s, e_t) = 1$ if we choose to add this pair, otherwise $\phi(e_s, e_t) = 0$. The two constrains can guarantee each entity $e_t \in \mathcal{N}(\tilde{\mathcal{E}}_t)$ is aligned to at most one entity in source language $e_s \in \mathcal{E}_s$. Finally, all pairs that satisfying $\phi(e_s, e_t) = 1$ can be viewed as candidates to be added into alignment data. We further select the top ones in terms of $sim(e_s, e_t)$ to control the total size of pseudo alignments. Notably, in each generation, the target entities to be aligned can already have the alignment, i.e., $\mathcal{N}(\tilde{\mathcal{E}}_t) \bigcup \tilde{\mathcal{E}}_t \neq \emptyset$. In this case, we can update the existing alignments with the pseudo ones to eliminate the possible alignment noise.

5.3.5 Mutually-paced Optimization

Learning Objective. Given a source graph $\mathcal{G}_s$, the incomplete target graph $\tilde{\mathcal{G}}_t$, and the incomplete cross-lingual alignment $\tilde{\Gamma}_{s\leftrightarrow t}$, the objective of cross-lingual temporal knowledge graph reasoning $\mathcal{L}_{s\to t}$ can be summarized as follows:

$$\mathcal{L}_{s\to t} = \mathcal{L}_{\tilde{G}_t} + \mathcal{L}_{\tilde{\Gamma}_{s\leftrightarrow t}}, \tag{5.13}$$

where $\mathcal{L}_{\tilde{G}_t}$ denotes knowledge graph reasoning loss which measures the correctness of each quadruple, $\mathcal{L}_{\tilde{\Gamma}_{s\leftrightarrow t}}$ denotes the alignment loss which measures the distance of aligned entities in

Algorithm 5.1: The optimization process for MP-KD.

Input: Source TKG $\mathcal{G}_s$, incomplete target TKG $\tilde{\mathcal{G}}_t$, incomplete alignment $\tilde{\Gamma}_{s\leftrightarrow t}$.
Output: Student Model Parameter Θ_t for target TKG.

Optimize Θ_s by minimizing $\mathcal{L}_{\mathcal{G}_s}$ on source TKG;
Initialize $\Theta_t \leftarrow \Theta_s$ for target TKG;
while *model not converged* **do**
  Optimize alignment module $g(\cdot;\Phi)$:
  Minimize $\mathcal{L}^{ST}_{s\to t}$ in Eq. (5.14) w.r.t. alignment parameter Φ;
  Transfer temporal events $\tilde{\mathcal{G}}^{ST}_t$ based on updated Θ_t;
  Optimize student representation module $f_t(\cdot;\Theta_t)$:
  for *Each time step* T_i *during training* **do**
    Prepare training data $\{(e_t, r, e'_t, t)|(e_t, r, e'_t, t) \in \tilde{\mathcal{G}}_t \bigcup \tilde{\mathcal{G}}^{ST}_t \text{ and } T_i < t < T_{i+1}\}$
    Update Θ_t by minimizing $\mathcal{L}^{ST}_{s\to t}$ in Eq. (5.14);
  end
  Generate pseudo alignments $\tilde{\Gamma}^{ST}_{s\leftrightarrow t}$ based on updated Φ;
end

the uni-space. To enlarge the knowledge distillation effect, we progressively transfer temporal events $\tilde{\mathcal{G}}_t$ and generate high-quality pseudo alignment $\tilde{\Gamma}^{ST}_{s\leftrightarrow t}$. Therefore, the training objective on both ground-truth data and pseudo data $\mathcal{L}^{ST}_{s\to t}$ becomes:

$$\begin{aligned}\mathcal{L}^{ST}_{s\to t} = &\frac{|\tilde{\mathcal{G}}_t|}{|\tilde{\mathcal{G}}_t| + |\tilde{\mathcal{G}}^{ST}_t|} \cdot \mathcal{L}_{\tilde{\mathcal{G}}_t} + \frac{|\tilde{\mathcal{G}}^{ST}_t|}{|\tilde{\mathcal{G}}_t| + |\tilde{\mathcal{G}}^{ST}_t|} \cdot \mathcal{L}_{\tilde{\mathcal{G}}^{ST}_t} \\ &+ \frac{|\tilde{\Gamma}_{s\leftrightarrow t}|}{|\tilde{\Gamma}_{s\leftrightarrow t}| + |\tilde{\Gamma}^{ST}_{s\leftrightarrow t}|} \cdot \mathcal{L}_{\tilde{\Gamma}_{s\leftrightarrow t}} + \frac{|\tilde{\Gamma}^{ST}_{s\leftrightarrow t}|}{|\tilde{\Gamma}_{s\leftrightarrow t}| + |\tilde{\Gamma}^{ST}_{s\leftrightarrow t}|} \cdot \mathcal{L}_{\tilde{\Gamma}^{ST}_{s\leftrightarrow t}},\end{aligned} \tag{5.14}$$

where $|\cdot|$ denotes the set size. Eq. (5.14) formulates the learning objective on both scarce data and pseudo data for cross-lingual temporal knowledge graph reasoning in target languages. We give the convergence analysis in the following theorem:

Theorem 5.1. Let N denote the number of negative samples for optimization, ϵ denotes the portion of correct pseudo data, β denotes the proportion of pseudo data to the initial ground-truth data. As the number of negative samples $N \to \infty$, the $\mathcal{L}^{ST}_{s\to t}$ converges to its limit with an absolute deviation decaying in $O(\frac{1+\epsilon}{1+\beta} \cdot N^{-2/3})$.

Proof. In representation learning, the margin loss has been widely adopted as the similarity metric. Without loss of generality, they can be expressed in the form of Noise Contrastive Estimation (NCE) [146]. Therefore, we express $\mathcal{L}_{\mathcal{G}}$ and $\mathcal{L}_{\Gamma_{s\leftrightarrow t}}$ in the form of Noise Contrastive

Estimation (NCE) by introducing the negative sampling:

$$\mathcal{L}_{\mathcal{G}} \triangleq \mathbb{E}\left[-\log \frac{e^{f(\cdot;\Theta)/\tau}}{e^{f(\cdot;\Theta)/\tau}+\sum_{-} e^{(f^{-}(\cdot;\Theta))/\tau}}\right], \tag{5.15}$$

$$\mathcal{L}_{\Gamma_{s\leftrightarrow t}} \triangleq \mathbb{E}\left[-\log \frac{e^{g(\cdot;\Phi))/\tau}}{e^{g(\cdot;\Phi)/\tau}+\sum_{-} e^{g^{-}(\cdot;\Phi)/\tau}}\right], \tag{5.16}$$

for simplicity, $f^{-}(\cdot;\Theta)$ denotes the score for negative quadruple, and $g^{-}(\cdot;\Phi)$ denotes score for negative alignment pair.

For our training objective $\mathcal{L}^{ST}_{s\to t}$, we show the convergence analysis of four terms one by one, then prove the overall convergence results. First of all, following [147], [148], let N denote the number of negative samples per each quadruple, and we have:

$$\begin{aligned}
&\lim_{N\to\infty}\left[\mathcal{L}_{\tilde{\mathcal{G}}_t}-\log M\right] \\
&=-\frac{1}{\tau}\mathop{\mathbb{E}}_{(e_t,r,e'_t,t)\in\tilde{\mathcal{G}}_t}\left[f(\cdot;\Theta)\right] \\
&+\lim_{N\to\infty}\mathop{\mathbb{E}}_{\substack{(e_t,r,e'_t,t)\in\tilde{\mathcal{G}}_t \\ e^{-}_t\in\mathcal{E}_t}}\left[\log\left(\frac{\lambda}{N}e^{f(\cdot;\Theta)/\tau}+\frac{1}{N}\sum_{-}e^{f^{-}(\cdot;\Theta)/\tau}\right)\right] \\
&=-\frac{1}{\tau}\mathop{\mathbb{E}}_{(e_t,r,e'_t,t)\in\tilde{\mathcal{G}}_t}\left[f(\cdot;\Theta)\right]+\mathop{\mathbb{E}}_{(e_t,r,e'_t,t)\in\tilde{\mathcal{G}}_t}\left[\log\mathop{\mathbb{E}}_{e^{-}_t\in\mathcal{E}_t}\left[e^{f^{-}(\cdot;\Theta)}\right]\right],
\end{aligned} \tag{5.17}$$

where λ denotes the duplicate quadruples co-existing in both incomplete $\tilde{\mathcal{G}}_t$ and negative samples. The convergence speed is derived as follows:

For one side:

$$\mathcal{L}_{\tilde{\mathcal{G}}_t}-\log N-\lim_{N\to\infty}\left[\mathcal{L}_{\tilde{\mathcal{G}}_t}-\log N\right]\leq\frac{\lambda}{N}e^{\frac{2}{\tau}}. \tag{5.18}$$

For another side:

$$\lim_{N\to\infty}\left[\mathcal{L}_{\tilde{\mathcal{G}}_t}-\log N\right]-\left[\mathcal{L}_{\tilde{\mathcal{G}}_t}-\log N\right]\leq\frac{\lambda}{N}e^{2/\tau}+\frac{5}{4}N^{-\frac{2}{3}}e^{\frac{1}{\tau}}(e^{\frac{1}{\tau}}-e^{-\frac{1}{\tau}}). \tag{5.19}$$

We then generalize the above results to the loss term on pseudo data. Suppose ϵ of the

pseudo data are correct. Then we can have the following two inequality. For one side:

$$\begin{aligned}
&\mathcal{L}_{\tilde{\mathcal{G}}_t^{ST}} - \log N - \lim_{N\to\infty}\left[\mathcal{L}_{\tilde{\mathcal{G}}_t^{ST}} - \log N\right] \\
&\leq \epsilon \mathop{\mathbb{E}}_{e\in\mathcal{E}_t}\left[\log \frac{\mathop{\mathbb{E}}_{e^-\in\mathcal{E}_t}\left[\frac{\lambda}{N}e^{1/\tau} + e^{f^-(\cdot;\Theta_t)/\tau}\right]}{\mathop{\mathbb{E}}_{e^-\in\mathcal{E}_t} e^{f^-(\cdot;\Theta_t)/\tau}}\right] \leq \epsilon\frac{\lambda}{N}e^{\frac{2}{\tau}}
\end{aligned} \tag{5.20}$$

Therefore, for $\mathcal{L}_{s\to t}^{ST}$ in this side, we have:

$$\mathcal{L}_{s\to t}^{ST} - \log N - \lim_{N\to\infty}\left[\mathcal{L}_{s\to t}^{ST} - \log N\right] \leq \frac{1+\epsilon}{1+\beta}\frac{\lambda}{N}e^{\frac{2}{\tau}}, \tag{5.21}$$

where β is the ratio of pseudo data amount to groundtruth data amount during training.

Similarly, for another side, we have:

$$\begin{aligned}
\lim_{N\to\infty}\left[\mathcal{L}_{s\to t}^{ST} - \log N\right] - \left[\mathcal{L}_{s\to t}^{ST} - \log N\right] &\leq \frac{1+\epsilon}{1+\beta}\frac{\lambda}{N}e^{2/\tau} \\
&+ \frac{1+\epsilon}{1+\beta}\frac{5}{4}N^{-\frac{2}{3}}e^{\frac{1}{\tau}}(e^{\frac{1}{\tau}} - e^{-\frac{1}{\tau}}).
\end{aligned} \tag{5.22}$$

Therefore, we conclude that the $\mathcal{L}_{s\to t}^{ST}$ converges to its limit with an absolute deviation decaying in $O(\frac{1+\epsilon}{1+\beta}\cdot N^{-2/3})$. QED.

Mutually-paced Optimization and Generation. The MP-KD framework can be optimized by minimizing Eq. (5.14) w.r.t. Θ and Φ alternatively. The generated pseudo alignments can help the training of the representation modules by the knowledge distillation, and in turn transferring temporal events in the target graph can improves alignment module by providing high-quality representations. In light of this, we propose a mutually-paced optimization and generation procedure. Generally speaking, we iteratively generate pseudo alignments and update the representation module and alignment module respectively. To be concrete, as shown in Algorithm 3.1, we first update the alignment module $g(\cdot;\Phi)$ and transfer temporal events to transfer knowledge from source to target. Then we divide the time span into several time steps to update the student representation module from recent time step to far away ones. Finally, we generate the pseudo alignments, as the optimization Θ_t on all temporal events can improve the entity feature quality, which is beneficial for alignment prediction. Algorithm 3.1 summarizes the training procedure.

Table 5.2: Statistics of the datasets.

Languages	Entity	Relation	Quadruple	Train/Val/Test	Time
English (EN)	34,416	105	602K	602K/0K/0K	28
French (FR)	32,546	105	580K	580K/0K/0K	28
Spanish (ES)	31,808	105	316K	114K/136K/66K	40
German (DE)	27,657	105	268K	97K/114K/56K	40
Italian (IT)	23,734	94	236K	84K/100K/51K	40
Danish (DA)	15,710	94	125K	48K/50K/26K	40
Slovene (SL)	13,250	94	55K	24K/21K/10K	40
Bulgarian (BG)	3,508	105	23K	8K/9K/6K	40

5.4 THE EVALUATION OF MP-KD

We evaluate MP-KD on EventKG data [143] including 2 source languages and 6 target languages, and we aim to answer the following research questions:

- **RQ1**: How does MP-KD perform compared with state-of-the-art models on the low-resource target languages?
- **RQ2**: How do reliability of alignment information (with various noise ratio) affect model performances?
- **RQ3**: How do each component and important parameters affect MP-KD performance?

5.4.1 Datasets

Dataset Information. The commonly utilized benchmark TKGs are divided into two categories: temporal event graphs [149] and knowledge graphs where temporally associated facts have valid periods [10], [11], [150]. In this paper, we mainly evaluate MP-KD on the EventKG [143], which is a multilingual resource incorporating event-centric information extracted from several large-scale knowledge graphs such as Wikidata [11], DBpedia [150] and YAGO [10]. Each temporal event is organized as (e, r, e', t_s, t_e), where each piece of data is attached with a valid time period from start time t_s to end time t_e. Following [137], we preprocess the format such that each fact is converted to a sequence $\{(e, r, e', t_s), (e, r, e', t_s + 1), \cdots, (e, r, e', t_e)\}$ from t_s to t_e, with the minimum time unit as one step.

Splitting Scheme. We collect events during 1980 to 2022, and noisy events of early years are removed. To construct multilingual TKGs, we first preserve important entities and relations by excluding infrequent ones that have less than 20 events in each language. Then we collect the events and cross-lingual alignments.To guarantee the relation match, we only preserve

Table 5.3: Overall Performance without alignment noise. Average results on 5 independent runs are reported. $*$ indicates the statistically significant improvements over the best baseline, with p-value smaller than 0.01.

Models	Target	ES		DE		IT		DA		BG		SL		Avg.	
	Source	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
RE-GCN w/o source	**NA**	14.31	31.85	16.32	34.19	14.59	31.64	14.19	31.24	10.27	23.44	9.33	21.63	13.17	29.00
Static KG embedding methods															
TransE [9]	**EN**	11.67	26.73	15.19	31.37	9.15	21.44	12.71	23.31	10.17	23.72	9.73	21.83	11.44	24.73
	FR	12.37	27.79	14.01	28.30	11.38	23.19	10.05	22.10	11.88	23.01	10.63	22.44	11.72	24.47
	T.R.	0.84	0.86	0.89	0.87	0.70	0.71	0.80	0.73	1.07	1.00	1.09	1.02	0.88	0.85
TransR [14]	**EN**	11.88	28.66	16.01	32.01	8.14	22.07	13.34	24.73	10.33	23.51	8.89	22.12	11.43	25.52
	FR	12.01	28.32	14.58	29.51	9.93	24.66	11.90	22.64	11.98	23.44	9.27	23.88	11.61	25.41
	T.R.	0.83	0.89	0.94	0.90	0.62	0.74	0.89	0.76	1.09	1.00	0.97	1.06	0.87	0.88
DistMult [15]	**EN**	13.66	29.77	17.46	33.19	11.63	26.63	14.63	25.91	9.97	22.92	9.08	20.44	12.74	26.48
	FR	12.58	28.73	16.03	31.81	12.12	27.76	11.64	22.97	9.01	23.77	10.13	21.07	11.92	26.02
	T.R.	0.92	0.92	1.03	0.95	0.81	0.86	0.93	0.78	0.92	1.00	1.03	0.96	0.94	0.91
RotatE [17]	**EN**	12.99	28.89	19.87	35.46	15.62	30.14	13.44	25.79	11.10	22.98	11.37	23.99	14.07	27.88
	FR	13.01	29.33	17.63	34.81	14.99	31.04	11.62	23.17	10.73	23.14	11.10	24.66	13.18	27.69
	T.R.	0.91	0.91	1.15	1.03	1.05	0.97	0.88	0.78	1.06	0.98	1.20	1.12	1.03	0.96
Temporal KG embedding methods															
TA-DistMult [33]	**EN**	15.83	34.77	18.99	37.46	14.98	29.99	14.97	30.01	9.02	21.10	8.74	17.76	13.75	28.51
	FR	16.61	35.83	17.81	37.96	15.58	31.21	13.21	28.58	9.63	22.91	9.03	18.83	13.65	29.22
	T.R.	1.13	1.11	1.13	1.10	1.05	0.97	0.99	0.94	0.91	0.94	0.95	0.85	1.04	1.00
RE-Net [137]	**EN**	17.58	37.97	19.03	39.46	15.88	33.69	15.03	34.77	12.01	25.72	11.07	25.64	15.10	32.88
	FR	17.01	36.79	18.32	38.07	15.47	34.83	15.63	33.86	12.31	25.03	11.79	24.97	15.09	32.26
	T.R.	1.21	1.17	1.14	1.13	1.07	1.08	1.08	1.10	1.18	1.08	1.23	1.17	1.15	1.12
RE-GCN [22]	**EN**	16.88	36.54	19.84	40.17	16.17	34.84	15.99	35.62	12.22	26.02	10.63	23.38	15.29	32.76
	FR	17.14	37.01	19.63	41.01	16.44	35.61	15.03	33.19	11.91	25.13	11.09	22.77	15.21	32.45
	T.R.	1.19	1.15	1.21	1.19	1.12	1.11	1.09	1.10	1.17	1.09	1.16	1.07	1.16	1.12
Multilingual KG embedding methods															
KEnS [138]	**EN**	15.98	33.91	17.33	37.62	14.41	31.44	14.47	29.61	12.88	26.77	11.03	24.99	14.35	30.72
	FR	17.02	34.07	16.61	37.99	15.57	33.82	13.62	30.24	12.03	24.32	10.51	23.86	14.23	30.72
	T.R.	1.15	1.07	1.04	1.11	1.03	1.03	0.99	0.96	1.21	1.09	1.15	1.13	1.09	1.06
AlignKGC [139]	**EN**	13.59	33.19	16.44	33.14	13.71	34.07	12.13	31.07	11.33	26.63	8.32	20.77	12.59	29.81
	FR	13.90	34.71	17.14	34.81	14.97	33.65	12.07	30.44	10.92	25.31	9.64	21.28	13.11	30.03
	T.R.	0.96	1.07	1.03	0.99	0.98	1.07	0.85	0.98	1.08	1.11	0.96	0.97	0.98	1.03
SS-AGA [140]	**EN**	15.11	32.19	16.49	36.14	14.83	33.31	12.27	30.68	12.99	27.03	11.55	25.07	13.87	30.74
	FR	16.54	33.99	18.32	37.19	15.02	32.99	11.73	29.98	11.13	25.62	11.01	23.64	13.96	30.57
	T.R.	1.11	1.04	1.07	1.07	1.02	1.05	0.85	0.97	1.17	1.12	1.21	1.13	1.06	1.06
MP-KD *	**EN**	**19.51**	41.55	**22.84**	**49.30**	17.18	37.62	**18.79**	**40.01**	**14.33**	**30.13**	**13.87**	**30.30**	**17.75**	**38.15**
	FR	19.05	**42.86**	21.67	46.57	**17.92**	**39.18**	17.95	37.95	13.85	29.27	12.54	27.36	17.16	37.20
	T.R.	1.35	1.33	1.36	1.40	1.20	1.21	1.29	1.25	1.37	1.27	1.42	1.33	1.33	1.3
Gains		11%	13%	15%	20%	9%	10%	18%	12%	10%	11%	18%	18%	16%	16%

relations appearing in English TKG. We split the time span into 40 equal time steps for training, validation and testing (28/4/8), where each time step roughly lasts for one year. To focus on the prediction on existing entities during training period, and eliminate the negative effects possibly caused by the randomly appearning new entities in val/test period, we only preserve entities having events during training period, following [22]. Table 5.2 shows the dataset statistics, including 2 source languages and 6 target languages. We purposefully choose 6 different target languages with diverse characteristics in term of the TKG size, which can evaluate MP-KD from different data granularity. It is worth noting that to simulate the scarcity issue in target TKGs, the training quadruples presented in Table 5.2 are randomly selected from original TKGs, with random ratio 20%.

5.4.2 Experimental Setup

Baselines. We describe the baselines utilized in the experiments in detail:

- **TransE** [9] is a translation-based embedding model, where both entities and relations are represented as vectors in the latent space. The relation is utilized as a translation operation between the subject and the object entity;
- **TransR** [14] advances TransE by optimizing modeling of n-n relations, where each entity embedding can be projected to hyperplanes defined by relations;
- **DistMult** [15] is a general framework with bilinear objective for multi-relational learning that unifies most multi-relational embedding models;
- **RotatE** [17] represents entities as complex vectors and relations as rotation operations in a complex vector space;
- **TA-DistMult** [33] is a temporal knowledge graph reansoing method aiming at predicting missing events in history. We utilize it for predicting future events;
- **RE-NET** [137] is a generative model to predict future facts on temporal knowledge graphs, which employs a recurrent neural network to model the entity evolution, and utilizes a neighborhood aggregator to consider the connection of facts at the same time intervals;
- **RE-GCN** [22] learns the temporal representations of both entities and relations by modeling the KG sequence recurrently;
- **KEnS** [138] starts to directly improve KGR performance on static KGs given a set of seed alignment, and proposes an ensemble-based approach for the task;
- **AlignKGC** [139] jointly optimizes entity alignment loss and knowledge graph reasoning loss to improve the performance;
- **SS-AGA** [140] views alignments as new edge type and employ a relation-aware GNN with learnable attention weight to model the influence of the aligned entities.

Evaluation Protocol and Metrics. For each prediction $(e, r, ?, t)$ or $(?, r, e, t)$, we rank missing entities to evaluate the performance. Following [22], we adopt raw mean reciprocal rank (*MRR*) and raw Hits at 10 (*H@10*) as evaluation metrics. To quantitatively compare how well the transferred knowledge from the source languages can improve predictions on the

low-resource languages, we adopt Transfer Ratio ($T.R.$) to evaluate the average improvement of each method over the best baseline without knowledge transferring, i.e.:

$$T.R.(t_i) = \frac{1}{|S|} \sum_{s_i \in \mathcal{S}} \frac{\text{Model}(s_i \to t_i)}{\text{BestBaseline}(t_i)} \tag{5.23}$$

where t_i denotes each target language, $s_i \in \mathcal{S}$ denotes each source language, and $\text{BestBaseline}(t_i)$ denotes the best baseline performance on the target language t_i without any knowledge transferring, i.e., *RE-GCN w/o source*.

Implementation. For static knowledge graph reasoning methods, i.e., TransE, TransR, DistMult, and RotatE, we ignore all time information in quadruples, and view temporal knowledge graphs as static, cumulative ones. For static/temporal KG embedding methods, we merge source graph and target graph by adding one new type of relation (alignment), as they do not explicitly model cross-lingual entity alignment. For multilingual baselines, we train them on 1-to-1 knowledge transferring (instead of the original setting) for fair comparison. For static baselines, we utilize the static embeddings for predictions in all time steps. For fair comparisons, we keep the dimension of all embeddings as 128, we feed pre-trained TransE embeddings on the merge graph including both source and target TKGs to those that require initial entity/relation embeddings. We tune learning rate of baselines based on *MRR* on validation set, and we train all baseline models and MP-KD on same GPUs (Nvidia A100) and CPUs (Intel(R) Xeon(R) Platinum 8275CL).

We first utilize the source TKG to train the teacher representation module. Then we initialize the student module with the parameters of the teacher. During the training procedure, we first optimize the objective without generating pseudo data in the first 10 epoch. After that, we start to generate high-quality pseudo data. For the generation in each epoch, we gradually increase the amount of pseudo alignments from 10% to 40%, and transfer all temporal events that meet the requirement. During evaluation, we tune hyperparameters based on *MRR* on validation set, and report the performance on the test set. Next, we report the choices of hyperparameters. For model training, we utilize Adam optimizer, and set maximum number of epochs as 50. We set batch size as 256, the dimension of all embeddings as 128, and dropout rate as 0.5. For the sake of efficiency, we set number of temporal neighbors b as 8, and employ 1 neighborhood aggregation layer in temporal encoder. For TKG reasoning, we set negative sampling factor as 10. For entity alignment, we set negative sampling factor as 50. For temporal generation process, We divide time span into 4 time intervals. For model training, we mainly tune margin value λ_1, λ_2 in score functions in range $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, learning rate in range $\{0.02, 0.01, 0.005, 0.001, 0.0005\}$.

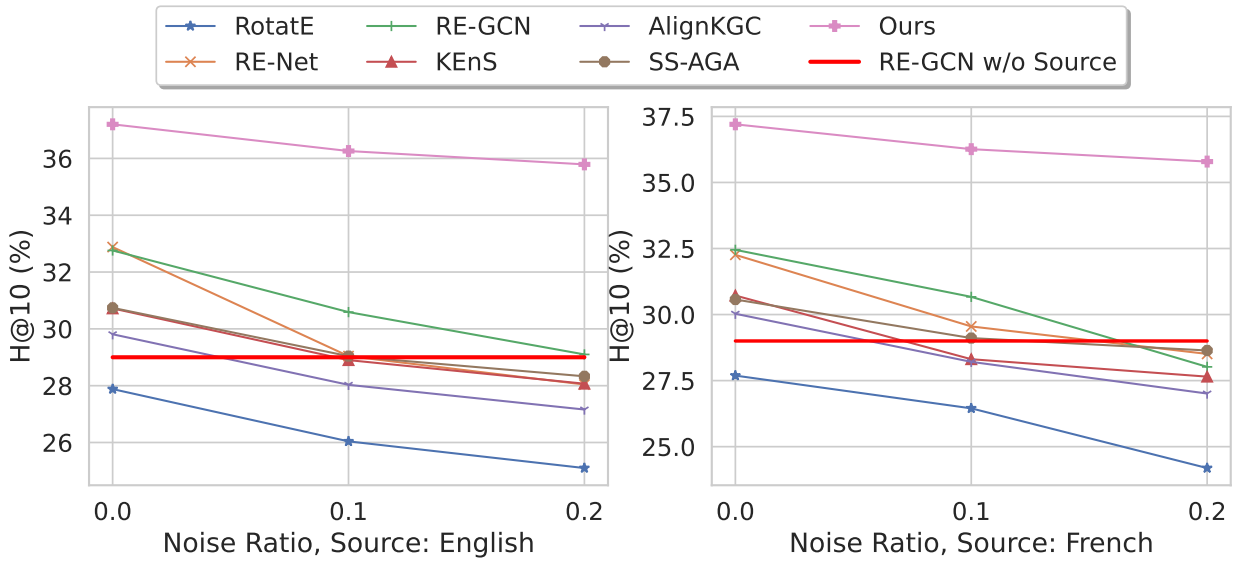


Figure 5.3: Experimental results under various alignment noise ratios. Average H@10 on 6 target languages are reported. MP-KD achieves relatively robust results, with only 3.7% relative drop, others have over 10% drop.

5.4.3 Experiments on Cross-lingual Reasoning (RQ1)

We first evaluate the model performance with incomplete cross-lingual alignments, where we randomly preserve 10% alignments of the target entities for distilling knowledge. Table 5.3 reports the overall results for the cross-lingual experiments. By utilizing only 10% cross-lingual alignments, MP-KD achieves 33% (*MRR*) and 30% (*H@10*) relative improvement over best baseline without the knowledge transferring (*RE-GCN w/o source*) on average, demonstrating the effectiveness of MP-KD in modeling alignments for knowledge transferring. Compared with ten baselines using alignments, MP-KD still achieves relative 14% relative improvements over the second best results. Specifically, we have the following observations:

- Static baseline (*TransE, TransR, DistMult, RotatE*) fail to beat *RE-GCN w/o source*, although using alignments, due the insufficient modeling of temporal information. Similarly, multilingual methods (*KEnS, AlignKGC, SS-AGA*) also produce unsatisfying results;

- All temporal baselines (*TA-DistMult, RE-Net, RE-GCN*) manage to beat *RE-GCN w/o source*, as the modeling of both temporal evolution and cross-lingual alignment can facilitate the representation learning of target entities. But the improvements are marginal compared with our model, as the effect of knowledge distillation is constrained by the limited amount of cross-lingual alignments;

- Our model consistently achieves the best performance. Through 10% alignments, MP-KD can progressively transfer temporal knowledge and generate pseudo alignments with high

Table 5.4: Ablation Studies.

Ablations	Target	ES		SL		Avg.	
	Source	MRR	H@10	MRR	H@10	MRR	H@10
MP-KD w/o Align. Strength Control	EN	17.61	38.59	13.07	28.51	16.24	37.04
	FR	17.07	39.46	13.03	28.14	16.33	36.61
	T.R.	1.21	1.23	1.27	1.21	1.21	1.22
MP-KD w Pure Training	EN	17.09	37.03	11.89	26.25	14.81	31.90
	FR	16.99	37.10	11.78	25.91	14.97	32.15
	T.R.	1.19	1.16	1.15	1.11	1.13	1.09
MP-KD w/o Pseudo Align.	EN	17.97	38.04	12.31	27.83	15.98	34.83
	FR	17.55	38.45	12.19	26.32	15.79	35.27
	T.R.	1.24	1.20	1.19	1.16	1.22	1.19
MP-KD w/o Event Transfer	EN	18.79	39.03	13.07	28.07	16.03	36.74
	FR	18.83	39.88	12.95	28.79	15.99	36.95
	T.R.	1.31	1.24	1.27	1.21	1.27	1.24
MP-KD	EN	**19.51**	41.55	**14.33**	**30.13**	**17.75**	**38.15**
	FR	19.05	**42.86**	13.85	29.27	17.16	37.20
	T.R.	1.35	1.33	1.37	1.27	1.33	1.3

confidence to boost the effect and range of the knowledge distillation;

- We also notice the uneven improvements across languages, (e.g., 40% improvements for German, 20% for Italian). We hypothesize it is because of various language dependencies with source languages.

5.4.4 Experiments under Alignment Noises (RQ2)

In reality, cross-lingual alignments can be obtained by human labeling or rule-based inference modules, which may introduce indispensable noises. We evaluate how the reliability of alignment information affects baseline models and MP-KD. In this experiment, we still utilize 10% alignments. To simulate unreliable alignments, we select a subset of alignments (measured by *Noise Ratio*) and randomly change the aligned target entity to another entity without alignment information.

We vary the noise ratio from 0.0 to 0.2 to evaluate the models performance, as shown in Figure 5.3. We report the average H@10 on 6 target languages by utilizing English TKG and French TKG respectively. As expected, with the increase of noise ratio, the performances of all compared models degrade, as the wrong alignment links mislead the knowledge transfer process. Most baselines fail to beat *RE-GCN w/o Source* even with 10% noise, and all lose with 20%*noise*, which indicates that the quality of alignments significantly influences the model effectiveness in the cross-lingual TKG reasoning task. Notably, MP-KD achieves relatively robust results, with only 3.7% performance drop, while other strong baselines have over 10% drop. This is because during the generation of pseudo alignments, MP-KD can automatically replace those unreliable ones based on the confidence score. Also, in the alignment module, MP-KD can assign small alignment strength to unreliable alignments.

5.4.5 Model Analysis (RQ3)

Ablation Study. We evaluate performance improvements brought by the MP-KD framework by following ablations:

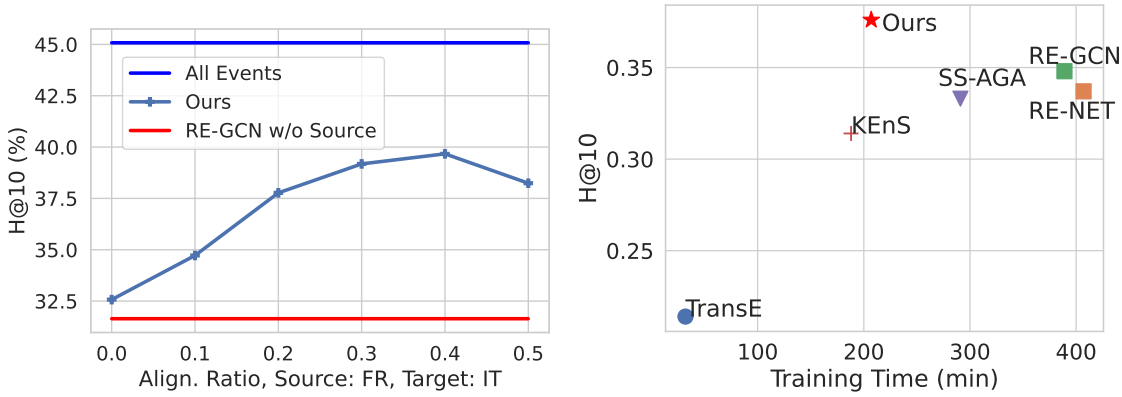
- **MP-KD w/o Align. Strength Control** uniformly set the alignment strength for all entities across all time steps;
- **MP-KD w Pure Training** optimizes the teacher-student framework without pseudo alignment generation and temporal event transfer;
- **MP-KD w/o Pseudo Align** eliminates the pseudo alignments generation process;
- **MP-KD w/o Event Transfer** eliminates the explicit transfer of temporal events.

Table 5.4 reports the results measured by $H@10$. Each component leads to performance boost. MP-KD with uniform alignment strength largely degrades performance, due to temporal knowledge discrepancy. MP-KD without pseudo data generation achieves similar performance with temporal baselines *RE-Net, RE-GCN*, because of the limited amount of cross-lingual alignments. Bother generating pseudo alignments and explicitly transferring temporal events increase the performance, and combining them together in a mutually-paced procedure (in MP-KD) can achieve the best results.

The Effect of Pseudo Alignments Ratio. To investigate the effects of the pseudo alignments on the reasoning performance, we vary the amount of pseudo alignments during training period and compare the corresponding performance measured by *H@10*, as shown in Figure 5.4. The blue line and red line show the performances of single model on complete target TKG and single model on 20% target TKG (our setting) respectively. From 0.1, MP-KD starts to generate and expand the initially available alignments. We observe a significant performance improvement, demonstrating the positive effects of the pseudo alignments. As expected, we find a performance decrease at 50%, as the added pseudo data with relatively low confidence start to introduce noise that hurt the performance.

5.4.6 Efficiency Comparison

To demonstrate the efficiency of MP-KD framework, we train MP-KD and baseline models from scratch on both target language and source language, and compare the training time. We train all baseline models and MP-KD on same GPUs (Nvidia A100) and CPUs (Intel(R) Xeon(R) Platinum 8275CL). Figure 5.4 shows that MP-KD significantly outperforms baseline models with reasonable training time. Notably, we include the pseudo data generation time.



(a) Pseudo align. analysis.

(b) Efficiency analysis.

Figure 5.4: Efficiency analysis and pseudo alignment analysis.

Compared with slow temporal models *RE-NET, RE-GCN* for knowledge graph reasoning, MP-KD is more efficient because our temporal encoder can learn temporal entity embeddings via sampled temporal neighbors at each time without using RNNs.

5.5 RELATED WORK

5.5.1 Knowledge Graph Reasoning

. Knowledge graph reasoning aims to predict missing facts to automatically complete KGs [8], [10], [11], [150]. It is mostly formulated as measuring the correctness of factual samples and negative samples by specially designed score functions [9], [17], [151]. Recently, reasoning on temporal KGs attracts a lot of interests from the community [22], [33], [136], [137]. Compared with static KG reasoning task, the main challenge lies in how to incorporate time information. Several embedding-based methods have been proposed. They encode time-dependent information of entities and relations by decoupling embeddings into static component and time-varying component [5], [152], [153], utilizing recurrent neural networks (RNNs) to adaptively learn the dynamic evolution from historical fact sequence [22], [137], or learning a sequence of evolving representations from discrete knowledge graph snapshots [22], [137], [154], [155]. However, all of the existing temporal KG reasoning models aim to extrapolate future facts based on relatively complete TKGs in high-resource languages, and how to boost reasoning performance for TKGs in low-resource languages through cross-lingual alignments is largely under-explored.

5.5.2 Multilingual KG Reasoning

. Entity alignment methods on KGs [129], [148], [156], [157] can automatically enlarge the alignments by predicting the correspondence between the two KGs. But most of them, if not all, require the relatively even completeness of two KGs to capture the structural similarities, which can not be satisfied in our case, as target TKGs are far from complete. Inspired by recent transfer learning advances [158], some recent works start to study the multilingual KG reasoning on static graphs [138]–[140], which aim to extract knowledge from several source KGs to boost the reasoning performance in the target KG, while they still require a sufficient amount of seed alignments and totally ignore the temporal information in our task. We extend this line of works on TKGs, where transferring temporal knowledge is more complex.

5.6 SUMMARY

In this chapter, we studied a realistic but underexplored cross-lingual temporal knowledge graph reasoning problem, which aims at facilitating TKG reasoning in low-resource languages by distilling knowledge from a corresponding TKG in high-resource language through a small set of entity alignments as bridges. To this end, we proposed a novel mutually-paced teacher student framework, namely MP-KD. During training, MP-KD iteratively generates pseudo alignments to expand the cross-lingual connection, as well as transfers temporal facts to facilitate student model training in low-resource languages. Our alignment module is learned to adjust the alignment strength for different entities at different time, thereby maximizing the benefits of knowledge transferring. We empirically validated the effectiveness of MP-KD on 12 language pairs of EventKG data, on which the proposed framework significantly outperforms an extensive set of state-of-the-art baselines.

Chapter 6: LEARNING EMERGING NODES FOR LABEL-EFFICIENT MODEL ADAPTATION

6.1 OVERVIEW

In this chapter, we investigate a realistic but underexplored problem, called few-shot temporal graph reasoning, that aims to predict future facts for newly emerging entities based on extremely limited observations in evolving graphs. It offers practical value in applications that need to derive instant new knowledge about new entities in temporal graphs with minimal supervision. Most prior efforts are limited to reasoning about existing entities but neglecting the commonly observed low-data regime where many new entities emerge with extremely few observations as new events or topics evolve over time [159]. This restricts their applicability.

Figure 6.1 shows the amount of new entities appearing over time. New entities continuously join the temporal graphs on three public temporal graphs, **e.g.,** 41.7**,** 61.3%**,** 9.8% **of entities on YAGO, WIKI, and ICEWS18 are new entities that firstly appear in the last** 25% **time steps**. We further investigate the amount of facts associated with those new entities. Figure 6.2 shows the corresponding distributions. **Most new entities are associated with a limited amount of facts.** Figure 6.1 and Figure 6.2 validate the practical value of our proposed setting, which aims to predict the future facts of newly emerging entities based on initial few-shot observations.

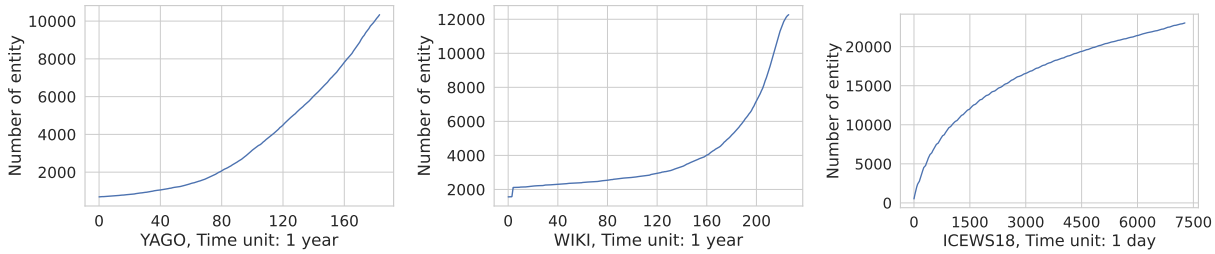


Figure 6.1: Number of entities over time. New entities continuously emerge on three public temporal graphs.

Instead, inspired by the human ability to recognize new concepts from exposure to only very few instances, we propose and solve a practical and challenging *few-shot temporal graph reasoning* problem that specifically aims to predict future facts for **newly emerging entities** with a few observed links on temporal graphs. Existing efforts [160]–[163] in similar settings simplify the task by ignoring the emergence of new entities over time and randomly simulating unseen entities on static graphs. This formulation, originating from few-shot

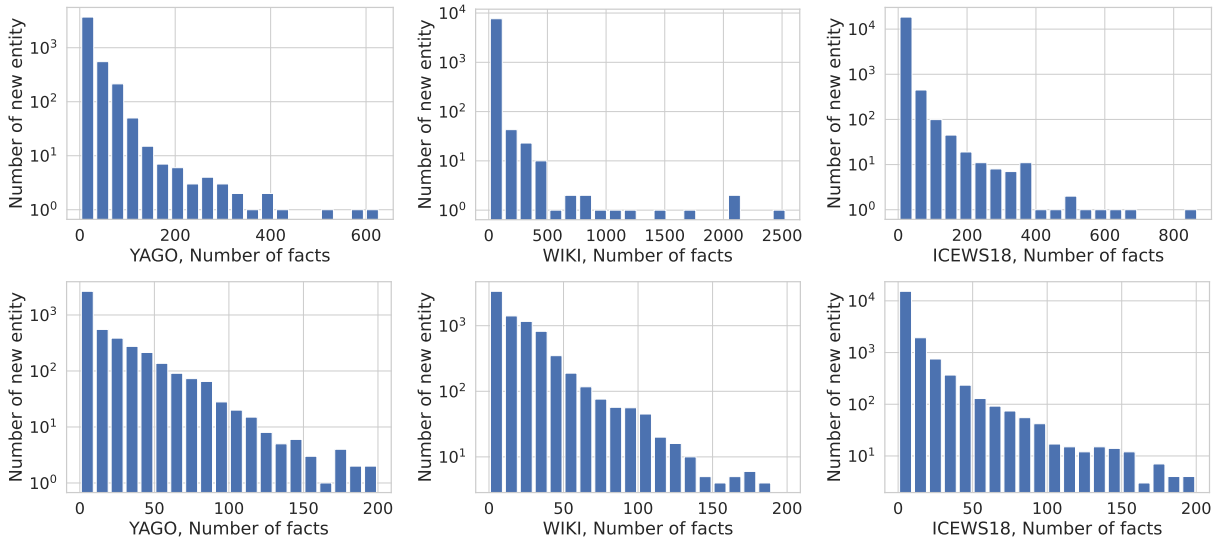


Figure 6.2: Distribution for new entity occurrences. The figures in first row show the complete distribution, and the figures in second row show the detailed distribution between [0,200]. Most of new entities are associated with limited amount of facts.

learning in the vision domain [164], [165], fails at capturing the real-world divergence between the distributions of seen and unseen entities. To the best of our knowledge, the generalized few-shot temporal reasoning task, that aims to imitate the fast learning ability of humans, has still not been formally investigated.

6.1.1 Overview of MetaTKGR

The challenges in solving this problem are two-fold: 1) **few-shot**: the extremely few facts associated with new entities cannot provide sufficient information for representation learning from scratch. 2) **time shift**: new entities usually exhibit different characteristics in the future, due to the time-evolving nature of temporal graphs. It leads models to generalize poorly in the future. Unfortunately, prior work fails to overcome these two challenges simultaneously.

On one hand, to enhance low-data entity characterization, a diverse range of neighborhood aggregation schemes that retrieve related information from other entities are explored, such as hard subgraph sampling methods (e.g., vanilla hop-based sampling [18], random sampling [166], personalized Pagerank sampling [3], [167], importance sampling [168], etc) and soft sampling via trainable attention mechanisms [136], [137], [169]. However, those rigid methods cannot well adapt to the new entities with the time-varying distributions due to the lack of instant supervision. On the other hand, recent attempts [162], [170]–[172] leverage meta-learning to improve unseen entity adaptation. While the setup for unseen entities is simulated by

randomly splitting entity sets from static graphs, they fail to handle the more realistic temporal adaptation with large entity variances.

To overcome both the aforementioned challenges, we propose a novel Meta Temporal graph Reasoning (MetaTKGR) framework, where learning the strategies for sampling and aggregating neighbors from recent facts (to enhance the new entities' predictions) can be dynamically adapted from signals of temporal supervision on their future facts as instant feedback. Intuitively, the global knowledge of sampling and aggregating can be extracted as learning to learn ability through such a meta-optimization, which takes the time-evolving nature of knowledge into consideration, avoiding progressive performance drift over time. Concretely, we formulate this learning strategy to be parameterized by a temporal encoder. On the simulated new entities during training phase, starting from the initial few-shot links, the temporal encoder can softly sample and attentively integrate relative and time-aware information from the temporal neighbors, making the strategy learning (neighbor sampling + aggregation) differentiable. The quantitative measurement of how well the current strategy performs can be reflected in the performance on predicting future facts allowing one to automatically adjust the strategy in turn. During the nested loop for meta-optimization, we firstly learn the temporal encoder on recent facts (*inner loop*), then gradually increase the difficulties by autoregressively adapting it to farther away future facts (*outer loop*). Furthermore, to better estimate the supervision signal given by future facts in different and unseen distributions, we adopt the PAC-Bayes method [173] to theoretically analyze the temporal adaptation bound on the future facts. This can serve as an adaptation regularizer to provide stability and improve the generalization ability of current strategy over time. Empirically, extensive experiments on three real-world temporal graphs validate the effectiveness of MetaTKGR, which significantly outperforms all baselines from static/temporal KG reasoning and few-shot (knowledge) graph learning areas, by up to 11.4% relative gain on average with competitive efficiency.

6.1.2 Overview of MetaHKG

New entities within real-world temporal graphs follow intriguing distributions that call for special model design. Firstly, as illustrated in Figure 6.3, the degree distribution of new entities adheres to a power-law pattern, with the majority being linked to only a few events, particularly during their initial phase. Furthermore, new entities, denoted by the blue points in Figure 6.3, display distinct hierarchical structures in contrast to existing entities denoted by the red points. They are typically distributed on the periphery of these hierarchies. Hence, effectively few-shot temporal reasoning entails not only the exponential growth of new

(a) Degree distribution. (b) Existing (red) v.s. new (blue). (c) Euclidean v.s. hyperbolic.

Figure 6.3: (a) New entity degree distribution, most of which are linked to only a few events. (b) New entities (blue) lie on the periphery of hierarchies. (c) The exponential growth in hyperbolic space and polynomial growth in Euclidean space on areas of circles.

entities but also their power-law patterns and hierarchical structures. Most existing few-shot studies [160]–[163] mainly focus on improving reasoning ability to long-tail entities randomly selected from static graphs instead of emerging entities, ignoring the aforementioned difference between existing and new entities over time. One recent study, MetaTKGR [5], investigates a similar setting to ours. However, it is built upon Euclidean spaces more suited for capturing uniform grid structures. There remains an unexplored space for few-shot temporal reasoning that simultaneously considers the rapid growth, power-law distribution, and hierarchical structures of new entities.

Hyperbolic geometry provides a promising alternative to generalize reasoning models to newly emerging entities. Characterized by constant negative curvature, hyperbolic spaces naturally accommodate the power-law patterns and hierarchical structures commonly observed in new entities. This property obviates the need for fundamental adjustments of the embedding space when new entities emerge at the periphery of hierarchies, leading to better generalizability. Moreover, hyperbolic spaces exhibit an exponential expansion property, as illustrated in Figure 6.3, wherein the area of circles grows exponentially with respect to the radius. This aligns with new entity representations that often experience exponential emergence over time. In contrast, Euclidean space demonstrates only polynomial growth concerning the radius, leading to higher embedding distortions and correspondingly more complex model updates when generalizing to new entities. Some researchers study the embedding in spherical space with positive curvature and show that a spherical embedding can better capture a cyclic structure [174], such as directional data. [175], [176] develop the Spherical Variational Autoencoders and apply them in language and document modeling. [177] proposes a spherical generative model and learns word and paragraph embeddings

jointly. However, they are not good at capturing power-law distribution and hierarchical structures of new nodes emerging over time.

Despite the success of hyperbolic representation learning in vision [178]–[181], natural language [182]–[184] and graph data [185]–[189], enabling the few-shot temporal reasoning capability in hyperbolic space still faces challenges.

- **Challenge #1: how to design operations in hyperbolic space to represent temporal graphs?** Representation learning of new entities requires hyperbolic aggregation to integrate sufficient information from few-shot neighbors and time encoding that can be generalized into the future. Existing efforts [186], [187] are unsuitable for few-shot new entities, including attention mechanism and time encoding, which call for a new model design.

- **Challenge #2: how to effectively generalize the model to new entities with adaptation on few-shot events?** New entities typically have very limited event associations, making it impractical to learn hyperbolic representations from scratch. It calls for a novel formulation of model adaptation strategies in hyperbolic space.

To further improve MetaTKGR, we propose a meta-hyperbolic learning framework MetaHKG to enable the few-shot temporal reasoning capability in a hyperbolic space. MetaHKG represents new entities by sampling and integrating information from temporal neighbors, where the inductive aggregation attentively considers entity, relation, and temporal features. The model parameters are categorized into two groups: global parameters shared across all entities and entity-specific parameters unique to each new entity. We introduce a meta hyperbolic learning strategy, which involves meta-training the parameters and subsequently fine-tuning the entity-specific parameters using a few-shot set of initial events.

To tackle the first challenge, we introduce a hyperbolic time encoding with proven translation invariance property that maps the time domain to hyperbolic space and a hyperbolic attention network that calculates attention weights in hyperbolic space. We harmonize the widely utilized score function [190] and margin loss function [9], [14] with the proposed hyperbolic operations for model training. To address the second challenge, our proposed meta-learning strategy is adept at independently learning global parameters and entity-specific parameters through a nested bi-level optimization procedure. This innovative approach enhances the model's ability to generalize to new entities within the hyperbolic space, reducing the need for extensive spatial adjustments.

Extensive empirical results on three real-world temporal graphs show that the MetaHKG method outperforms state-of-the-art baselines. Furthermore, MetaHKG excels in predicting

events taking place among new entities. Importantly, it requires lower-dimensional latent space and yields a more stable model adaptation than Euclidean counterparts in representing new entities.

6.2 PRELIMINARIES

In this section, we first formalize the definition of the few-shot temporal reasoning task on temporal graphs and then introduce the background of hyperbolic geometry. We list and explain symbols and notations in Table 6.1.

Table 6.1: Symbols and Notations.

Symbol	Definition
$\mathcal{G}^T$	A temporal graph.
$\mathcal{E}^{(0,T)}$	Entity set between $(0, T)$.
$\tilde{e} \in \mathcal{E}^{(T,T')}$	New entities between (T, T').
(e_s, r, e_o, t)	An event in TKG.
K	Amount of observable events of new entities.
d	Dimension of latent spaces.
c	Curvature of Poincaré ball space.
$\mathcal{B}^{d,c}$	Poincaré ball space.
$\mathcal{T}_{\mathbf{x}}\mathcal{B}^{d,c}$	Tangent space associated with $\mathbf{x}$.
$\mathcal{T}_{\mathbf{0}}\mathcal{B}^{d,c}$	Tangent space associated with origin $\mathbf{0}$.
$d_{\mathcal{B}}(\mathbf{x}, \mathbf{y})$	Distance in Poincaré ball space.
$\exp^c_{\mathbf{x}}(\mathbf{v})$	Exponential map from Euclidean to hyperbolic.
$\log^c_{\mathbf{x}}(\mathbf{y})$	Logarithmic map from hyperbolic to Euclidean.
$\mathbf{x} \oplus^c \mathbf{y}$	Möbius addition for two points.
$\mathbf{W} \otimes^c \mathbf{x}$	Hyperbolic linear transformation.
$\mathbf{h}^{\mathcal{B}}_e(t)$	Hyperbolic entity representation at time t.
$\phi_{\mathcal{B}}(t)$	Hyperbolic time encoding of time t.
ω_i	Trainable parameters in time encoding.
$\mathcal{N}_{\tilde{e}}(t)$	Temporal neighbors of new entity at time t.
Θ	Model parameters.
Θ_g, Θ_e	Global and entity-specific parameters.
$\mathcal{S}_{\tilde{e}}$	Support set of new entity $\tilde{e}$.
$\mathcal{Q}_{\tilde{e}}$	Query set of new entity $\tilde{e}$.
$\mathcal{L}(\Theta, \cdot)$	Predictive loss by using Θ on data $\cdot$.

.

6.2.1 Problem Definition

A temporal graph is denoted as $\mathcal{G}^T = \{(e_s, r, e_o, t) | t \leq T\} \subseteq \mathcal{E}^{(0,T)} \times \mathcal{R} \times \mathcal{E}^{(0,T)} \times \mathcal{T}$, where $\mathcal{E}^{(0,T)}$ denotes a set of entities that appear in time interval $(0, T)$, $\mathcal{R}$ denotes a set of relations, and $\mathcal{T}$ denotes the timestamp set. Each temporal link (e_s, r, e_o, t) refers to an event that a subject entity $e_s \in \mathcal{E}^{(0,T)}$ has a relation $r \in \mathcal{R}$ with an object entity $e_o \in \mathcal{E}^{(0,T)}$ at timestamp $t \in \mathcal{T}$.

As a temporal graph changes, it continually introduces new entities, causing an expansion of the entity set over time. Given a time interval (T, T'), entities that join the graph during (T, T'), i.e., $\tilde{e} \in \mathcal{E}^{(T,T')} = \mathcal{E}^{(0,T')} \setminus \mathcal{E}^{(0,T)}$ are defined as new entities on time interval (T, T'), where $\setminus$ denotes set minus. Technically, the set of relations $\mathcal{R}$ also experiences changes, and new relations may emerge as time progresses. Nevertheless, it is worth noting that most, if not all, of the relations are typically present during the training phase, implying that only a limited number of new relations emerge in the graphs [5]. Consequently, to set the scope, we focus solely on the newly emerging entities and assume that the set of relations remains static throughout this paper.

Temporal graph reasoning refers to the problem of predicting future events in the partially observed TKG. While existing few-shot models simulate unseen entities by randomly selecting from the existing entities, we focus specifically on predictions for newly emerging entities in the future. It is formalized as *few-shot temporal reasoning* task:

Definition 6.1 (**Few-shot Temporal Reasoning).** Given a partially observed temporal graph $\mathcal{G}^T$ (collected no later than T), we define entities that emerge during the following period (T, T') as new entities. For each new entity $\tilde{e} \in \mathcal{E}^{(T,T')} = \mathcal{E}^{(0,T')} \setminus \mathcal{E}^{(0,T)}$, we assume the first K associated events are available: $\{(\tilde{e}, r_i, e_i, t_i) \text{ or } (e_i, r_i, \tilde{e}, t_i)\}_{i=1}^{K}$. The task aims to predict the missing entities in the future events $(\tilde{e}, r, ?, t)$ or $(?, r, \tilde{e}, t)$ given the relation $r \in \mathcal{R}$ and specific time $T < t \leq T'$. We further assume K is a small number, *i.e.,* $K = \{1, 2, 3\}$, as a realistic setting.

6.2.2 Hyperbolic Geometry

A Riemannian manifold $(\mathcal{M}, \mathbf{g})$ is a branch of differential geometry, where $\mathcal{M}$ is a smooth manifold and $\mathbf{g}$ is a *Riemannian metric*. For each point $\mathbf{x}$ on the manifold $\mathcal{M}$, there is a tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ as the first-order approximation of $\mathcal{M}$ around $\mathbf{x}$. The Riemannian metric $\mathbf{g} = (g_{\mathbf{x}})_{\mathbf{x} \in \mathcal{M}}$ is a family of inner products defined on tangent spaces, *i.e.,* $g_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \times \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathbb{R}$.

We build our meta hyperbolic framework on the Poincaré ball model. The hyperbolic

space in d dimensions is the unique, complete, simply connected d-dimensional Riemannian manifold with constant sectional negative curvature [185]. The Poincaré ball model with negative curvature $-c$ ($c > 0$) corresponds to the Riemannian manifold $(\mathcal{B}^{d,c}, g_{\mathbf{x}}^{\mathcal{B}})$, where $\mathcal{B}^{d,c} = \{\mathbf{x} \in \mathbb{R}^d : \ c\|\mathbf{x}\| < 1\}$ is an open ball. To define the inner product, its metric tensor is $g_{\mathbf{x}}^{\mathcal{B}} = \lambda_{\mathbf{x}}^2 g^{\mathcal{E}}$, where $\lambda_{\mathbf{x}} = 2/(1 - \sqrt{c}\|\mathbf{x}\|^2)$ is the conformal factor and $g^{\mathcal{E}} = \text{diag}([1, 1, \cdots, 1])$ is the Euclidean metric tensor. When $c = 1$, the distance between two points $\mathbf{x}, \mathbf{y} \in \mathcal{B}^{d,c}$ is given as:

$$d_{\mathcal{B}}(\mathbf{x}, \mathbf{y}) = \text{arccosh}\left(1 + \frac{2\|\mathbf{x} - \mathbf{y}\|^2}{(1 - \|\mathbf{x}\|^2)(1 - \|\mathbf{y}\|^2)}\right). \tag{6.1}$$

Several other equivalent hyperbolic models exist, such as the hyperboloid and Beltrami-Klein models, and our framework can be extended to work with any hyperbolic model. However, the Poincaré ball model is picked as it provides visualizable, interpretable hyperbolic embeddings and has been extensively exploited in the past.

The map between Euclidean and hyperbolic spaces is achieved through exponential and logarithmic maps. This is a crucial ability because many input features and operations within the neural network are primarily defined within the Euclidean space. For any point $\mathbf{x} \in \mathcal{B}$, the exponential map $\exp_{\mathbf{x}}^c : \ \mathcal{T}_{\mathbf{x}}\mathcal{B}^{d,c} \to \mathcal{B}^{d,c}$ and the logarithmic map $\log_{\mathbf{x}}^c : \ \mathcal{B}^{d,c} \to \mathcal{T}_{\mathbf{x}}\mathcal{B}^{d,c}$ are defined for the tangent vector $\mathbf{v} \neq \mathbf{0}$ and the point $\mathbf{y} \neq \mathbf{0}$, respectively, as:

$$\exp_{\mathbf{x}}^c(\mathbf{v}) = \mathbf{x} \oplus^c \left(\tanh\left(\frac{\sqrt{c}\lambda_{\mathbf{x}}\|\mathbf{v}\|}{2}\right)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}\right), \tag{6.2}$$

$$\log_{\mathbf{x}}^c(\mathbf{y}) = \frac{2}{\sqrt{c}\lambda_{\mathbf{x}}}\text{arctanh}\left(\sqrt{c}\|-\mathbf{x} \oplus^c \mathbf{y}\|\right)\frac{-\mathbf{x} \oplus^c \mathbf{y}}{\|-\mathbf{x} \oplus^c \mathbf{y}\|}, \tag{6.3}$$

where c is the curvature of the Poincaré ball space, $\oplus^c$ is the Möbius addition for two points $\mathbf{x}, \mathbf{y} \in \mathcal{B}^{d,c}$:

$$\mathbf{x} \oplus^c \mathbf{y} = \frac{(1 + 2c\langle\mathbf{x}, \mathbf{y}\rangle + c|\mathbf{x}\|^2)\mathbf{x} + (1 - c\|\mathbf{x}\|^2)\mathbf{y}}{1 + 2c\langle\mathbf{x}, \mathbf{y}\rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \tag{6.4}$$

6.3 THE DESIGN OF METATKGR

In this section, we present a novel framework MetaTKGR to solve the few-shot temporal graph reasoning problem. We first introduce the basic setup of our learning objective and the overall framework, and then detail the temporal encoder module to represent new entities, followed by introduction of meta temporal reasoning for model training.

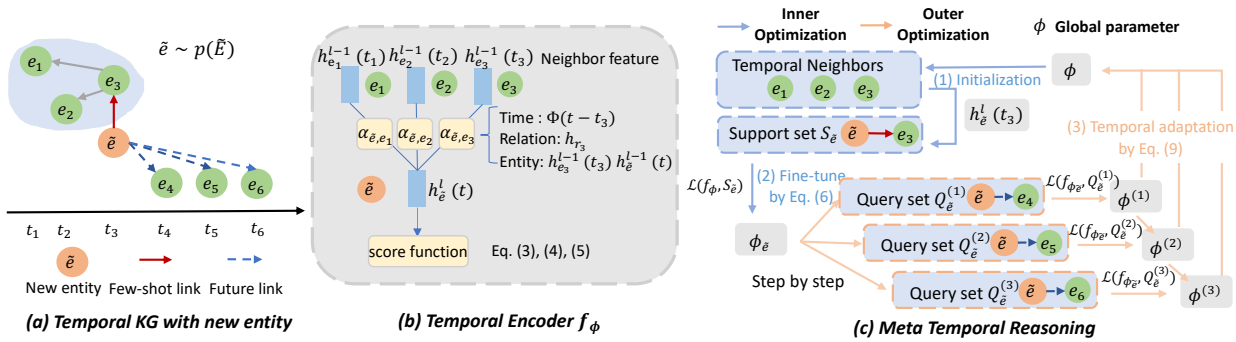


Figure 6.4: An overview of MetaTKGR framework. (a) Task illustration; (b) Temporal encoder parameterized by ϕ samples and aggregates information from temporal neighbors; (c) Meta temporal reasoning adopts a bi-level optimization to learn ϕ. In the **inner optimization**: MetaTKGR initializes entity-specific parameters from global parameters via step (1), and fine-tunes them on support set via step (2) to optimize entity-specific parameters; In the **outer optimization**, MetaTKGR optimizes global parameters on query set via step (3) to learn good and robust global parameters.

6.3.1 Learning Objective

Suppose we are given a temporal graph $\mathcal{G}^T = \{(e_s, r, e_o, t)\} \subseteq \mathcal{E}^T \times \mathcal{R} \times \mathcal{E}^T \times \mathcal{T}$ collected no later than time T. For each new entity $\tilde{e}$ appearing within a later interval (T, T'), we aim to predict future links $(\tilde{e}, r, e, t)$ or $(e, r, \tilde{e}, t)$ happening at timestamp $t \in (T, T')$. Towards this goal, we measure correctness of each possible quadruple by a score function $s(\cdot; \phi)$ parameterized by ϕ, and maximize the scores of true quadruples containing any new entities in order to rank them higher than all other false quadruples:

$$\max_{\phi} \mathbb{E}_{\tilde{e} \sim p(\tilde{\mathcal{E}})}[s(\tilde{e}, r, e, t; \phi) \text{ or } s(e, r, \tilde{e}, t; \phi)], \text{ where } \tilde{\mathcal{E}} = \mathcal{E}^{T'} \setminus \mathcal{E}^T, \ t \in (T, T'), \tag{6.5}$$

where $p(\tilde{\mathcal{E}})$ denotes distribution of all new entities, ϕ denotes parameters of model to represent entity/relation into d-dimensional space $\mathbf{h}_{\tilde{e}}, \mathbf{h}_e, \mathbf{h}_r \in \mathbb{R}^d$. To represent $\tilde{e}$ into $\mathbf{h}_{\tilde{e}}$ via ϕ , as aforementioned, the challenges lie in how to enable ϕ to encode generalized knowledge (how to sample and aggregate temporal neighbors) that can be easily adapted to new $\tilde{e}$ and achieve robust performance over time, even for a long time interval (T, T'). We thereby formulate few-shot temporal graph reasoning as a meta-learning problem to extract such knowledge.

6.3.2 MetaTKGR Framework

Figure 6.4 shows the MetaTKGR framework. To be more formal, each task corresponds to each new entity $\tilde{e}$ over distribution $p(\tilde{\mathcal{E}})$, and the links can be represented as a

chronological sequence $\{(\tilde{e}, r_i, e_i, t_i)$ or $(e_i, r_i, \tilde{e}, t_i) | t_i \in (T, T'), t_i \leq t_j$ if $i < j\}_{i=1}^{N_{\tilde{e}}}$, where $N_{\tilde{e}}$ denotes total number of links associated to $\tilde{e}$. Then we divide them into support set $\mathcal{S}_{\tilde{e}} = \{(\tilde{e}, r_i, e_i, t_i)$ or $(e_i, r_i, \tilde{e}, t_i)\}_{i=1}^{K}$, and query set $\mathcal{Q}_{\tilde{e}} = \{(\tilde{e}, r_i, e_i, t_i)$ or $(e_i, r_i, \tilde{e}, t_i)\}_{i=K+1}^{N_{\tilde{e}}}$, where K denotes the amount of initially observable facts of $\tilde{e}$.

During the meta-training phase, we simulate a set of new entities from existing entity set by assuming there are only few-shot links of these entities (support set $\mathcal{S}_{\tilde{e}}$). The model first fine-tunes parameters ϕ on $\mathcal{S}_{\tilde{e}}$ (*inner loop*), and then minimizes the predictive loss on corresponding query set $\mathcal{Q}_{\tilde{e}}$ using the updated parameters $\phi_{\tilde{e}}$ (*outer loop*). It optimizes the global sampling and aggregation strategy as generalized knowledge (i.e., learning to learn ability) by this bi-level optimization, as this procedure mimics the normal machine learning and inference process. For simplicity, to represent new entity $\tilde{e}$ along time, we denote $f_{\phi_{\tilde{e}}}$ as the temporal encoder parameterized by $\phi_{\tilde{e}}$, $\mathcal{L}(f_{\phi_{\tilde{e}}}, \mathcal{S}_{\tilde{e}})$ as model predictive loss on $\mathcal{S}_{\tilde{e}}$. Model-agnostic meta-learning (MAML) [164] can be utilized to fulfill this goal as follows:

$$\phi^* \longleftarrow \arg\min_{\phi} \mathbb{E}_{\tilde{e} \sim p(\tilde{\mathcal{E}})} \left[\mathcal{L}(f_{\phi_{\tilde{e}}}, \mathcal{Q}_{\tilde{e}}) \right], \quad \text{where } \phi_{\tilde{e}} = \phi - \eta \frac{\partial \mathcal{L}(f_{\phi}, \mathcal{S}_{\tilde{e}})}{\partial \phi}, \tag{6.6}$$

where η denotes inner-loop learning rate. However, this training strategy cannot guarantee good generalization ability over time (*challenge 2*), as they assume identical distributions between $\mathcal{S}_{\tilde{e}}$ and $\mathcal{Q}_{\tilde{e}}$, contradicting the facts that new entities are highly evolving over time on TKGs. To coherently resolve the two challenges, we advance the existing few-shot KG reasoning works by proposing:

- *Temporal encoder*, which learns the time-aware representation of each new entity $\tilde{e}$ by sampling and aggregating information from TKG neighbors on continuous domain.
- *Meta temporal reasoning*, which learns an optimal sampling and aggregating parameters via bi-level optimization (**inner optimization** and **outer optimization**). The learned parameters can be easily adapted to new entities and maintain temporal robustness.

Algorithm 6.1: Temporal Neighbor Sampler.

Input: New entity $\tilde{e}$, temporal graph $\mathcal{G}^t$, current timestamp t, neighbor budget b, time bound Δt.

Output: Temporal Neighbor $\mathcal{N}_{\tilde{e}}(t)$.

Initialize $Queue \leftarrow \tilde{e}$, $\mathcal{N}_{\tilde{e}}(t) \leftarrow \{\}$;
while $|\mathcal{N}_{\tilde{e}}(t)| < b$ *and* $Queue$ *is not empty* **do**
  $e \leftarrow Queue.dequeue()$;
  for *each* (e_s, r, e_o, t) *in* $\mathcal{G}^t$ **do**
    if $e_s == e$ *and* $t > t_{max} - \Delta t$ **then**
      Add (e_s, r, t) to $\mathcal{N}_{\tilde{e}}(t)$, Add e_s to $Queue$;
    end
    if $e_o == e$ *and* $t > t_{max} - \Delta t$ **then**
      Add (e_o, r, t) to $\mathcal{N}_{\tilde{e}}(t)$, Add e_o to $Queue$;
    end
  end
end

6.3.3 Temporal Encoder

On temporal graphs, entities are evolving over time. The temporal encoder f_ϕ is to embed each entity $\tilde{e}$ into low-dimensional latent space at each time: $\mathbf{h}_{\tilde{e}}(t) \in \mathbb{R}^d$, where it can model the temporal pattern of $\tilde{e}$ along time. By doing so, it resolves the scarcity issues caused by few-shot links to some extent. Towards this goal, f_ϕ first samples temporal neighbors from TKGs, then attentively aggregates information from the temporal neighbors of each entity, which takes neighbor feature, relation feature and time feature into account.

To better represent few-shot entities, a wider range of neighbors should be considered, as the close neighbors around the new entities are usually insufficient. Conventionally, stacking several graph neural networks (GNNs)-based layers can integrate information from multi-hop neighbors [18], [166]. In few-shot cases, such layer-by-layer procedure faces difficulties. For one thing, the limited one-hop neighbors can dominate the aggregation process, as the information of multi-hop neighbors is all propagated from them to the target entity. Also, the size of neighbors grows exponentially with the increase of GNN layers, as they collect all neighbors in each hop without strategic selection, causing severe efficiency issues. Instead, our temporal encoder first samples multi-hop neighbors via a time-bounded breadth-first-search algorithm, then aggregates information directly from the sampled neighbors in an attentive manner. As summarized in Algorithm 6.1, at each time, it selectively samples up to b multi-hop temporal

neighbors $\mathcal{N}_{\tilde{e}}(t)$ which have interactions within a recent time range Δt. Given the temporal neighbor $\mathcal{N}_{\tilde{e}}(t)$ for each new entity, temporal encoder f_ϕ represents a new entity as $\mathbf{h}_{\tilde{e}}(t)$ at time t:

$$\mathbf{h}_{\tilde{e}}^{l}(t) = \sigma\left(\sum_{(e_i,r_i,t_i)\in\mathcal{N}_{\tilde{e}}(t)} \alpha_{\tilde{e},e_i}\left(\mathbf{h}_{e_i}^{l-1}(t_i)\mathbf{W}\right)\right), \tag{6.7}$$

where l denotes the layer number, $\sigma(\cdot)$ denotes the activation function $ReLU$, $\alpha_{\tilde{e},e_i}$ denotes the attention weight of entity i to new entity $\tilde{e}$, and $\mathbf{W}$ is the trainable transformation matrix. To aggregate from history, $\alpha_{\tilde{e},e_i}$ is supposed to be aware of entity feature, time delay and topology feature induced by relations. Thus, we design $\alpha_{\tilde{e},i}$ as follows:

$$\alpha_{\tilde{e},e_i} = \frac{\exp(q_{\tilde{e},e_i})}{\sum_{(e_k,r_k,t_k)\in\mathcal{N}_{\tilde{e}}(t)}\exp(q_{\tilde{e},e_k})}, \quad q_{\tilde{e},e_i} = \mathbf{a}\left(\mathbf{h}_{\tilde{e}}^{l-1}\|\mathbf{h}_{e_i}^{l-1}\|\mathbf{h}_{r_i}\|\Phi(t-t_i)\right), \tag{6.8}$$

where $q_{\tilde{e},e_i}$ measures the pairwise importance by considering the entity embedding, relation embedding and time embedding, $\mathbf{a}\in\mathbb{R}^{4d}$ is the shared parameter in the attention mechanism. Following [144] we adopt random Fourier features as time encoding $\Phi(\Delta t)$ to reflect the time difference.

6.3.4 Meta Temporal Reasoning

Let $\mathbf{h}_{\tilde{e}}^{L}(t)$ denote the representations of new entity $\tilde{e}$ produced by f_ϕ with L layers. For each quadruple $(\tilde{e}, r, e, t)$, we employ a translation-based score function [9] to measure the correctness: $s(\tilde{e}, r, e, t; \phi) = -\|\mathbf{h}_{\tilde{e}}^{L}(t) + \mathbf{h}_r - \mathbf{h}_e^{L}(t)\|^2$. We first fine-tune f_ϕ on support set $\mathcal{S}_{\tilde{e}}$ for entity-specific parameters $f_{\phi_{\tilde{e}}}$, and then optimize training loss on query set $\mathcal{Q}_{\tilde{e}}$ to learn the global parameters shared by all entities. To calculate the predictive loss, since each set only contains positive quadruples, we perform negative sampling [9] to update MetaTKGR by training it to rank positive quadruples higher than negative ones. Specifically, taking the support set $\mathcal{S}_{\tilde{e}}$ of entity $\tilde{e}$ as an example, we construct a negative set $\mathcal{S}_{\tilde{e}}^{-} = \{(\tilde{e}, r, e^-, t) \text{ or } (e^-, r, \tilde{e}, t)\}$ by replace the ground truth entity e with a corrupted entity e^-. We then use empirical hinge loss on the given set as follows:

$$\hat{\mathcal{L}}(f_{\phi_{\tilde{e}}}, \mathcal{S}_{\tilde{e}}) = \sum_{(e_s,r,e_o,t)\in\mathcal{S}_{\tilde{e}}} \sum_{(e_s,r,e_o,t)^-\in\mathcal{S}_{\tilde{e}}^{-}} \max\left(\gamma - s(e_s,r,e_o,t) + s(e_s,r,e_o,t)^-, 0\right), \tag{6.9}$$

where $\gamma > 0$ is a margin value to distinguish positive and negative quadruples.

Algorithm 6.2: MetaTKGR: Meta-training.

Input: Temporal graph $\mathcal{G}^T \subseteq \mathcal{E}^T \times \mathcal{R} \times \mathcal{E}^T \times \mathcal{T}$, randomly initialized ϕ.

Output: Learned global parameter ϕ.

```
Simulate new entity set Ẽ from E^T;
Train model parameter φ from many-shot entities E^T \ Ẽ;
Construct S_ẽ and Q_ẽ for each new entity ẽ, where Q_ẽ = {Q_ẽ^(1), Q_ẽ^(2), ··· , Q_ẽ^(M)};
while φ not converge do
    for each time interval m do
        # Outer optimization:
        for each new entity ẽ do
            # Inner optimization:
            Calculate L̂(f_φ, S_ẽ) on S_ẽ by Eq. (6.9);
            Perform adaptation by Eq. (6.10) to update φ_ẽ;
            Calculate L̂(f_φ_ẽ, Q_ẽ^(m)) by Eq. (6.9);
        end
        Calculate temporal adaptation regularizer by Eq. (6.30);
        Update φ^(m) by Eq. (6.31);
        φ ← φ^(m)
    end
end
```

Inner optimization: For each task corresponding to new entity $\tilde{e} \sim p(\tilde{\mathcal{E}})$, we first adapt the global parameter ϕ for each new entity $\tilde{e}$ by minimizing the predictive loss on the support set $\mathcal{S}_{\tilde{e}}$:

$$\phi_{\tilde{e}} = \phi - \eta \frac{\partial \hat{\mathcal{L}}(f_\phi, \mathcal{S}_{\tilde{e}})}{\partial \phi}, \tag{6.10}$$

where η is the inner loop learning rate. By Eq. (6.10), we simulate the adaption for new entities. Updating from an optimal global parameter ϕ, the model is fine-tuned into entity-specific for new entities. Thus, it is crucial to design a meta-learning strategy to learn an optimal global parameter ϕ with a robust generalization ability over time.

Outer optimization: Since new entities evolve over time, it is not only $\mathcal{Q}_{\tilde{e}}$ but also each part of $\mathcal{Q}_{\tilde{e}}$ that follows different distributions. Thus, instead of optimizing the meta-learner on the whole query set $\mathcal{Q}_{\tilde{e}}$ at once by Eq. (6.6), we first adapt entity-specific parameters $\phi_{\tilde{e}}$ to recent facts, then gradually increase the difficulties by autoregressively adapting it to farther away future facts. Through this process, we are able to guide and stablize the training of global parameters via a temporal signal. Specifically, we split the time span of query sets

into M time intervals, where each new entity has a sequence of query set corresponding to different time intervals $\mathcal{Q}_{\tilde{e}} = \{\mathcal{Q}_{\tilde{e}}^{(1)}, \mathcal{Q}_{\tilde{e}}^{(2)}, \cdots, \mathcal{Q}_{\tilde{e}}^{(M)}\}$. We first adapt and optimize $\phi_{\tilde{e}}$ for each new entity on $\mathcal{Q}_{\tilde{e}}^{(1)}$ for $\phi^{(1)}$ encoding global knowledge for predictions in the 1-st time interval. Then we gradually adapt each $\phi_{\tilde{e}}$ fine-tuned from last time interval on farther away intervals until $\phi^{(M)}$ is learned, which maintain temporal robustness from time interval 1 to M.

We then discuss how to measure the feedback (predictive loss) from each adaptation. To simulate real scenarios, we assume query sets to follow different and unseen distributions. Taking adaptation from m-th time interval to $m+1$-th time interval as example, we view $p(\phi^{(m)})$ as prior parameter distribution and aim to learn the posterior parameter distribution $q(\phi^{(m+1)})$ conditioning on query set in $m+1$-th interval. The unseen query set distribution in $m+1$-th interval prevent us to estimate $q(\phi^{(m+1)})$ by either using unbiased empirical loss or Bayes rules. To resolve this, we instead adopt the PAC-Bayes method [173], which allows one to learn a parameter posterior that fits the observations without knowing the observations distribution. We propose the following theorem to relate real predictive loss in new time interval with its empirical verson:

Theorem 6.1 (**PAC-Bayes Generalization Bound on Temporal Adaptation).** Let $\mathcal{D}^{(m+1)} = \bigcup_{\tilde{e}\in\tilde{\mathcal{E}}} \mathcal{Q}_{\tilde{e}}^{(m+1)}$ denote all query sets in $m+1$-th time interval of all new entities from $\tilde{\mathcal{E}}$. For any $\delta \in (0,1)$ and learned parameter distribution $p(\phi^{(m)})$ in last time interval, with probability at least $1-\delta$ on $\mathcal{D}^{(m+1)}$:

$$\mathcal{L}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) \leq \hat{\mathcal{L}}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) + \sqrt{\frac{\mathbb{KL}(q(\phi^{(m+1)}) \| p(\phi^{(m)})) + \log \frac{|\mathcal{D}^{(m+1)}|}{\delta}}{2|\mathcal{D}^{(m+1)}| - 1}}, \tag{6.11}$$

where $\mathcal{L}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) = \mathbb{E}_{\tilde{e}\sim p(\tilde{\mathcal{E}})}\left[\mathcal{L}(f_{\phi_{\tilde{e}}^{(m)}}, \mathcal{Q}_{\tilde{e}}^{(m+1)})\right]$ denotes real predictive loss on new time interval with new and unknown data distribution, $\hat{\mathcal{L}}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)})$ denote the empirical version of the loss, and $\mathbb{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence.

Proof. As a realistic scenario, $\mathcal{D}^{(m+1)}$ in the new time interval follow different distribution from those in the last time interval, i.e., m-th time interval, and such new distribution is unknown at advance. To improve the generalization ability over time, we gradually adapt model parameters learned from last time interval $\phi^{(m)}$ to next time interval, resulting in updated parameters $\phi^{(m+1)}$. Formally, we view $p(\phi^{(m)})$ as prior parameter distribution and aim to learn the posterior parameter distribution $q(\phi^{(m+1)})$ conditioning on $\mathcal{D}^{(m+1)}$. The unseen distribution of $\mathcal{D}^{(m+1)}$ prevents us from estimating $q(\phi^{(m+1)})$ by either using unbiased empirical loss or Bayes rules. Therefore, for convinience of discussion, we first define the

difference of real predictive loss and its empirical extimation as follows:

$$\Delta\mathcal{L} = \mathcal{L}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) - \hat{\mathcal{L}}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}). \tag{6.12}$$

We are interested at the relation of $\Delta\mathcal{L}$ and the distribution discrenpency between $p(\phi^{(m)})$ and $p(\phi^{(m+1)})$. Towards this goal, following [173], [191], we construct the following function:

$$f(\mathcal{D}^{(m+1)}) = 2(|\mathcal{D}^{(m+1)}| - 1)\mathbb{E}_{\phi\sim q(\phi^{(m+1)})}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(q(\phi^{(m+1)})\|p(\phi^{(m)})). \tag{6.13}$$

Next, using Markov's inequality, we have:

$$p(f(\mathcal{D}^{(m+1)}) > \epsilon) = p(e^{f(\mathcal{D}^{(m+1)})} > e^{\epsilon}) \leq \frac{\mathbb{E}_{\tilde{e}}\left[e^{f(\mathcal{D}^{(m+1)})}\right]}{e^{\epsilon}}, \tag{6.14}$$

where $\mathbb{E}_{\tilde{e}}\left[e^{f(\mathcal{D}^{(m+1)})}\right]$ denotes the expectation of $e^{f(\mathcal{D}^{(m+1)})}$ w.r.t. new entity distribution. To upper bound the expectation, we have the following inequality:

$$\begin{aligned}
f(\mathcal{D}^{(m+1)}) &= 2(|\mathcal{D}^{(m+1)}| - 1)\mathbb{E}_{\phi\sim q(\phi^{(m+1)})}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(q(\phi^{(m+1)})\|p(\phi^{(m)})) && (6.15)\\
&= \mathbb{E}_{\phi\sim q(\phi^{(m+1)})}\left[\log\left(e^{2(|\mathcal{D}^{(m+1)}|-1)(\Delta\mathcal{L})^2}\frac{q(\phi^{(m+1)})}{p(\phi^{(m)})}\right)\right] && (6.16)\\
&\leq \log\left(\mathbb{E}_{\phi\sim q(\phi^{(m+1)})}\left[e^{2(|\mathcal{D}^{(m+1)}|-1)(\Delta\mathcal{L})^2}\frac{q(\phi^{(m+1)})}{p(\phi^{(m)})}\right]\right) && (6.17)\\
&= \log\left(\mathbb{E}_{\phi\sim p(\phi^{(m)})}\left[e^{2(|\mathcal{D}^{(m+1)}|-1)(\Delta\mathcal{L})^2}\right]\right), && (6.18)
\end{aligned}$$

where Jensen's inequality is utilzied to derive the inequality. Therefore, we have

$$\begin{aligned}
\mathbb{E}_{\tilde{e}}\left[e^{f(\mathcal{D}^{(m+1)})}\right] &\leq \mathbb{E}_{\tilde{e}}\mathbb{E}_{\phi\sim p(\phi^{(m)})}\left[e^{2(|\mathcal{D}^{(m+1)}|-1)(\Delta\mathcal{L})^2}\right] && (6.19)\\
&= \mathbb{E}_{\phi\sim p(\phi^{(m)})}\mathbb{E}_{\tilde{e}}\left[e^{2(|\mathcal{D}^{(m+1)}|-1)(\Delta\mathcal{L})^2}\right], && (6.20)
\end{aligned}$$

we switch the order of expectations because $p(\phi^{(m)})$ is independent to $\mathcal{D}^{(m+1)}$. Next, based on Hoeffding's inequality, we have:

$$p(\Delta\mathcal{L} > \epsilon) \leq e^{-2|\mathcal{D}^{(m+1)}|\epsilon^2}, \tag{6.21}$$

and we can further derive the following inequality:

$$\mathbb{E}_{\tilde{e}}\left[e^{f(\mathcal{D}^{(m+1)})}\right] \leq \mathbb{E}_{\phi\sim p(\phi^{(m)})}\mathbb{E}_{\tilde{e}}\left[e^{2(|\mathcal{D}^{(m+1)}|-1)(\Delta\mathcal{L})^2}\right] \leq |\mathcal{D}^{(m+1)}|. \tag{6.22}$$

Combining Eq. (6.22) and Eq. (6.14), we get:

$$p(f(\mathcal{D}^{(m+1)}) > \epsilon) \leq \frac{|\mathcal{D}^{(m+1)}|}{e^{\epsilon}} = \delta, \tag{6.23}$$

where $\delta = |\mathcal{D}^{(m+1)}|/e^{\epsilon}$. Therefore, with probability of at least $1-\delta$, we have that for all $\phi^{(m+1)}$:

$$f(\mathcal{D}^{(m+1)}) = 2(|\mathcal{D}^{(m+1)}| - 1)\mathbb{E}_{\phi \sim q(\phi^{(m+1)})}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(q(\phi^{(m+1)}) \| p(\phi^{(m)})) \tag{6.24}$$

$$\leq \frac{\log |\mathcal{D}^{(m+1)}|}{\delta}. \tag{6.25}$$

Further, by utilizing Jensen's inequality again, we have:

$$\left(\mathbb{E}_{\phi \sim q(\phi^{(m+1)})}\left[(\Delta\mathcal{L})\right]\right)^2 \leq \mathbb{E}_{\phi \sim q(\phi^{(m+1)})}\left[(\Delta\mathcal{L})^2\right] \tag{6.26}$$

$$\leq \frac{\mathbb{KL}(q(\phi^{(m+1)}) \| p(\phi^{(m)})) + \frac{\log |\mathcal{D}^{(m+1)}|}{\delta}}{2(|\mathcal{D}^{(m+1)}| - 1}. \tag{6.27}$$

Subsituting definition of $\Delta\mathcal{L}$ in Eq. 6.27, we proof the PAC-Bayes Generalization Bound on Temporal Adaptation:

$$\mathcal{L}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) \leq \hat{\mathcal{L}}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) + \sqrt{\frac{\mathbb{KL}(q(\phi^{(m+1)}) \| p(\phi^{(m)})) + \log \frac{|\mathcal{D}^{(m+1)}|}{\delta}}{2|\mathcal{D}^{(m+1)}| - 1}} \tag{6.28}$$

Based on Eq. 6.28, we define the following *temporal adaptation regularizer*:

$$\mathcal{R}(f_{\phi^{(m)}}; \mathcal{D}^{(m+1)}) \triangleq \hat{\mathcal{L}}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) + \sqrt{\frac{\mathbb{KL}(q(\phi^{(m+1)}) \| p(\phi^{(m)})) + \log \frac{|\mathcal{D}^{(m+1)}|}{\delta}}{2|\mathcal{D}^{(m+1)}| - 1}}. \tag{6.29}$$

QED.

Thus, using Theorem 6.1, we can adapt $\phi^{(m)}$ to $m+1$ time interval and estimate the feedback (predictive loss) for our meta-learning framework via the following *temporal adaptation regularizer*:

$$\mathcal{R}(f_{\phi^{(m)}}; \mathcal{D}^{(m+1)}) \triangleq \hat{\mathcal{L}}(f_{\phi^{(m)}}, \mathcal{D}^{(m+1)}) + \sqrt{\frac{\mathbb{KL}(q(\phi^{(m+1)}) \| p(\phi^{(m)})) + \log \frac{|\mathcal{D}^{(m+1)}|}{\delta}}{2|\mathcal{D}^{(m+1)}| - 1}}. \tag{6.30}$$

Remark. To interpret the temporal adaptation regularizer $\mathcal{R}(f_{\phi^{(m)}}; \mathcal{D}^{(m+1)})$, it can be viewed as a combination of empirical risk on the query set in the new time interval as well as a

regularizer of parameter distribution across time intervals in form of KL-divergence. The regularizer along time domain can guarantee that global knowledge ϕ is trained by predictive losses from all time intervals without overfitting in specific time interval. Thus, we can improve the generalization ability of our meta-learner over time by the following update step by step, from $m = 1$ to $m = M$:

$$\phi^{(m+1)} = \phi^{(m)} - \beta \frac{\partial \mathcal{R}(f_{\phi^{(m)}}; \mathcal{D}^{(m+1)})}{\partial \phi}, \tag{6.31}$$

where $\phi^{(0)}$ can be obtained by training on existing entities. The meta-training phase of MetaTKGR is summarized in Algorithm 6.2. During meta-test phase, given a new entity $\tilde{e}$, we can first fine-tune ϕ on few-shot links, and then utilize $f_{\phi_{\tilde{e}}}$ to represent $\tilde{e}$ for future predictions.

6.4 THE EVALUATION OF METATKGR

6.4.1 Datasets

We validate the proposed MetaTKGR framework on three public temporal knowledge graphs: 1) **YAGO** [10]: YAGO is a collection from the Wikipedias in multiple languages. 2)**WIKI** [11]: WIKI is a newly collected temporal knowledge graph of Wikipedias. 3)**ICEWS18** [149]: Integrated Crisis Early Warning System (ICEWS18) is the collection of coded interactions between socio-political actors which are extracted from news articles. ICEWS18 is collected from 1/1/2018 to 10/31/2018, and the minimum time unit is one day; YAGO and WIKI span much longer periods (184 years and 232 years respectively), with one year as minimum time units. They can validate our temporal framework within different time ranges. Notably, ICEWS18 represents temporal facts as quadruples (e_s, r, e_o, t), and WIKI, YAGO datasets consist of facts in format $(e_s, r, e_o, [t_s, t_e])$, where each fact is associated with a valid time range from start time t_s to end time t_e. We follow [137] to preprocess WIKI and YAGO. Specifically, we preprocess the format such that each fact is converted to a sequence $\{(e_s, r, e_o, t_s), (e_s, r, e_o, t_s + 1), \cdots, (e_s, r, e_o, t_e)\}$ from t_s to t_e, with the minimum time unit as one step. Noisy events of early years are removed (before 1786 for WIKI and 1830 for YAGO). Table 6.2 shows the detailed statistics.

Given the temporal knowledge graph, we first split the time duration into four parts with a ratio of 0.4:0.25:0.1:0.25 chronologically, then we collect the entities that firstly appear in each period as well as the associated facts as training/meta-training/meta-validation/meta-test set.

Table 6.2: Dataset statistics.

Datasets	# Entities	# Relations	# Quadruples	Time Unit
YAGO	10,623	10	201,090	1 year
WIKI	12,554	24	669,935	1 year
ICEWS18	23,033	256	281,205	1 day

For the nodes in the meta-validation and meta-test set, we assume that their first $K = 1, 2, 3$ facts are known and can be used to adapt parameters for meta-learning based methods or train/compute new entity representations for non-meta ones. The remaining facts are utilized to evaluate performance of baseline models and MetaTKGR.

6.4.2 Experimental Setup

Setup. For static knowledge graph reasoning methods, i.e., TransE, TransR, and RotatE, we ignore all time information in quadruples, and view temporal knowledge graphs as static, cumulative ones. Then we train the models with training set, meta-training set as well as first K triples of each new entity in meta-validation and meta-test sets. For temporal knowledge graph reasoning methods RE-NET and RE-GCN, we train the models with training set, meta-training set as well as first K quadruples of each new entity in meta-validation and meta-test sets. For baselines that optimize few-shot entities on static knowledge graphs, we ignore all time information in quadruples. We train them via training set and meta-training set, and adapt parameters via the first K quadruples of each new entity in meta-validation and meta-test sets. As for MetaDyGNN, since it is designed for homogeneous graphs, we adopt score function of TransE in the framework to support it on temporal knowledge graphs. Similarly, training and meta-training sets are utilized to train initial model parameters, then the few-shot facts of new entities are used to adapt entity-specific model parameters. For fair comparisons, we keep the dimension of all embeddings as 128, we feed pre-trained 1-shot TransE embeddings to those that require initial entity/relation embeddings, and we train all baseline models and MetaTKGR on same GPUs (GeForce RTX 3090) and CPUs (AMD Ryzen Threadripper 3970X 32-Core Processor).

We use training set to train MetaTKGR for parameter initialization, then we simulate few-shot tasks by utilizing meta-training set to adapt model parameters and further improve the temporal generalization ability. For new entities, we use initial $K = 1, 2, 3$ facts to fine-tune entity-specific models. During evaluation, we tune hyperparameters based on MRR on meta-validation set, and report the performance on the remaining facts on meta-test set. Next, we report the choices of hyperparameters. For model training, we utilize Adam optimizer, and set maximum number of epochs as 50. We set batch size as 20, the dimension of

all embeddings as 128, and dropout rate as 0.5. For the sake of efficiency, we set the neighbor budget b of temporal neighbor sampler as 16, and employ 1 neighborhood aggregation layer in temporal encoder. We divide query sets into 3 time intervals to simulate the real scenario. We perform a single step of gradient descent for inner loop optimization. We mainly tune margin value γ in score functions in range $\{0.3, 0.4, 0.5, 0.6, 0.7\}$, inner/outer loop learning rate η and β in range $\{0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$. For YAGO and ICEWS18, we set $\gamma = 0.5$, $\eta = \beta = 0.0001$. For WIKI, we set $\gamma = 0.4$, $\eta = 0.00005$, and $\beta = 0.0001$.

Baselines. We compare nine state-of-the-art baselines from four related areas: 1) **TransE** [9], 2) **TransR** [14], 3) **RotatE** [17]: Translation distance based embedding methods for static knowledge graphs; 4) **RE-NET** [137], 5) **RE-GCN** [22]: Temporal knowledge graph embedding methods; 6) **LAN** [161]; 7) **I-GEN** [162]; 8) **T-GEN** [162]: Few-shot methods on static knowledge graph; 9)**MetaDyGNN** [192]: A meta-learning framework for few-shot link prediction on homogeneous graphs. We describe the baseline models utilized in the experiments in detail:

- **TransE** [9] is a translation-based embedding model, where both entities and relations are represented as vectors in the latent space. The relation is utilized as a translation operation between the subject and the object entity;
- **TransR** [14] advances TransE by optimizing modeling of n-n relations, where each entity embedding can be projected to hyperplanes defined by relations;
- **RotatE** [17] represents entities as complex vectors and relations as rotation operations in a complex vector space;
- **RE-NET** [137] is a generative model to predict future facts on temporal knowledge graphs, which employs a recurrent neural network to model the entity evolution, and utilizes a neighborhood aggregator to consider the connection of facts at the same time intervals;
- **RE-GCN** [22] learns the temporal representations of both entities and relations by modeling the KG sequence recurrently;
- **LAN** [161] computes the embedding of entities by GNN-based neighboring aggregation scheme, and attention mechanisms are utilized to consider relations with neighboring information for new entities;
- **I-GEN** [162] utilizes meta-learning technique to learn the representations of new entities, which is achieved by aggregating information from neighbors attentively;

Table 6.3: The results of 3-shot temporal knowledge graph reasoning. Average results on 5 independent runs are reported. $*$ indicates the statistically significant improvements over the best baseline, with p-value smaller than 0.001.

Models	YAGO				WIKI				ICEWS18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.223	0.158	0.242	0.360	0.161	0.111	0.171	0.236	0.058	0.052	0.062	0.098
TransR	0.234	0.165	0.259	0.382	0.183	0.138	0.188	0.245	0.061	0.062	0.073	0.109
RotatE	0.241	0.182	0.278	0.409	0.232	0.171	0.223	0.284	0.078	0.074	0.082	0.128
RE-NET	0.261	0.210	0.298	0.410	0.261	0.210	0.251	0.331	0.232	0.139	0.241	0.369
RE-GCN	0.283	0.226	0.307	0.421	0.277	0.223	0.262	0.344	0.233	0.142	0.238	0.350
LAN	0.230	0.154	0.247	0.352	0.185	0.133	0.201	0.287	0.207	0.119	0.234	0.321
I-GEN	0.303	0.238	0.323	0.420	0.221	0.179	0.229	0.264	0.212	0.120	0.251	0.346
T-GEN	0.292	0.218	0.310	0.394	0.234	0.185	0.222	0.271	0.169	0.122	0.184	0.265
MetaDyGNN	0.350	0.270	0.379	0.511	0.309	0.238	0.309	0.459	0.307	0.216	0.309	0.469
MetaTKGR	**0.370***	**0.303***	**0.416***	**0.558***	**0.329***	**0.253***	**0.335***	**0.489***	**0.335***	**0.249***	**0.340***	**0.527***
Gains (%)	*5.68*	*12.21*	*9.80*	*9.19*	*6.26*	*6.38*	*8.47*	*6.35*	*9.11*	*14.85*	*9.89*	*12.4*

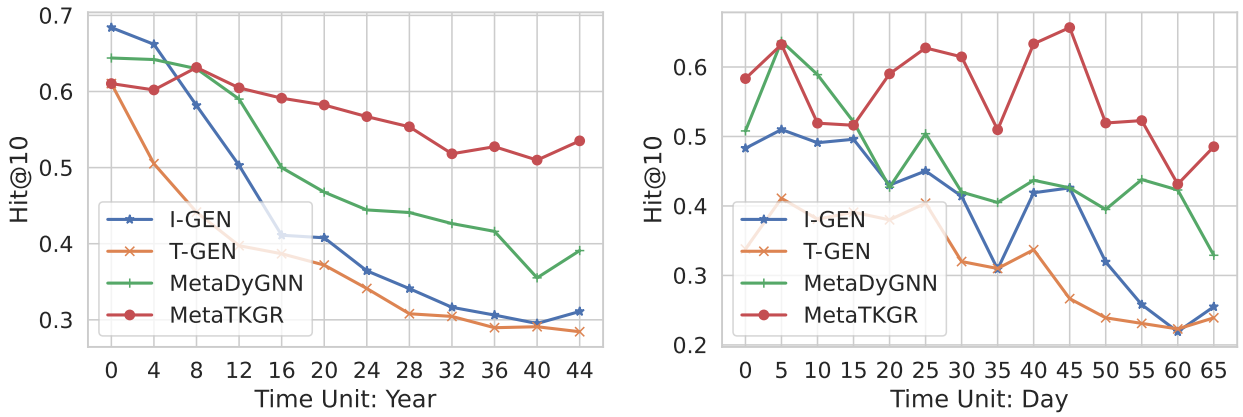
Table 6.4: The results of 1-shot and 2-shot experiment.

Models	YAGO				WIKI				ICEWS18			
	1-shot		2-shot		1-shot		2-shot		1-shot		2-shot	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
TransE	0.183	0.268	0.193	0.304	0.144	0.186	0.146	0.213	0.049	0.077	0.058	0.086
TransR	0.189	0.270	0.198	0.312	0.160	0.183	0.160	0.225	0.050	0.080	0.060	0.090
RotatE	0.215	0.280	0.210	0.359	0.175	0.190	0.201	0.268	0.068	0.098	0.070	0.091
RE-NET	0.221	0.304	0.233	0.390	0.212	0.259	0.239	0.294	0.185	0.250	0.200	0.341
RE-GCN	0.233	0.320	0.241	0.407	0.223	0.250	0.247	0.310	0.193	0.247	0.205	0.347
LAN	0.196	0.269	0.200	0.310	0.174	0.275	0.162	0.273	0.170	0.301	0.188	0.317
I-GEN	0.238	0.321	0.237	0.402	0.181	0.241	0.223	0.287	0.199	0.320	0.177	0.337
T-GEN	0.247	0.331	0.260	0.379	0.202	0.245	0.240	0.319	0.131	0.262	0.161	0.259
MetaDyGNN	0.269	0.396	0.316	0.496	0.241	0.371	0.271	0.390	0.249	0.420	0.269	0.441
MetaTKGR	**0.294***	**0.428***	**0.356***	**0.526***	**0.277***	**0.419***	**0.309***	**0.441***	**0.295***	**0.496***	**0.301***	**0.500***
Gains (%)	*9.43*	*8.04*	*12.69*	*6.14*	*14.64*	*12.93*	*14.04*	*13.20*	*18.45*	*17.87*	*11.47*	*13.39*

- **T-GEN** [162] further extends I-GEN by optimizing predictions for unseen-unseen links among unseen users. A stochastic inference is proposed to model the randomness of such links;

- **MetaDyGNN** [192] is a recent work modeling the links predictions for new nodes on homogeneous graphs. A hierarchical meta-learner is proposed to better extract global knowledge which is beneficial for link prediction task.

Evaluation Protocol and Metrics. For each prediction $(\tilde{e}, r, ?, t)$ or $(?, r, \tilde{e}, t)$, we use ranking scheme to evaluate the performance. Specifically, we rank all entities at the missing position in quadruples, and adopt mean reciprocal rank (MRR) and Hits at {1,3,10} ($H@\{1,3,10\}$) as evaluation metrics. It is worth noting that we measure the ranks in a filtered setting, where we filter other true quadruples existing in datasets, following [14], [17].



(a) Cross-shot results on YAGO.

(b) Cross-shot results on ICEWS18.

Figure 6.5: Performance of YAGO (left) and ICEWS18 (right) over time.

6.4.3 Main Results

Table 6.3 and Table 6.4 report overall result for $K = 3$ and $K = 1, 2$ few-shot experiments respectively. On average, MetaTKGR achieves 11.4% relative improvement over best baseline model, demonstrating the superiority of MetaTKGR in modeling new entity evolution in low-data regime. Baselines (*TransE, TransR, RotatE, RE-NET*) that do not optimize for new entities perform poorly on all datasets, although they are trained on both existing entities and new entities with few-shot links. *LAN, I-GEN, T-GEN* also produce unsatisfying results, despite the fact that they propose special designs to represent new entities. Note that they have worse performance in some cases than RE-NET that does not optimize for new entities, as they ignore the temporal information and perform poorly in future predictions. Compared with MetaDyGNN, MetaTKGR shows consistently better performance, because our temporal encoder can handle multi-relational graphs in low-data regimes better, and our temporal meta-learning framework can produce more robust predictions.

Performance of Prediction over Time. Next, we study the performance of MetaTKGR over time. Figure 6.5 shows the performance comparisons of 3-shot predictions over different timestamps on the *YAGO* and *ICEWS18* datasets, measured by filtered $H@10$ metrics. MetaTKGR consistently beats all strong baselines. Although the compared baselines can optimize newly emerging entities, they perform poorly in wide time intervals because they largely ignore the temporal information and the distribution discrepancy caused by evolution. We notice that the relative gains of our model get more significant with increasing time steps. It illustrates that our temporal meta-learning framework can improve the generalization ability over time. Performance on ICEWS18 fluctuates more severely. This is expected since

Table 6.5: Cross-shot results on YAGO.

Test	1-Shot (Training)			3-Shot (Training)		
	MRR	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**
1-S	0.294	0.240	0.428	0.281	0.242	0.403
3-S	0.354	0.297	0.540	0.370	0.303	0.558
5-S	0.377	0.360	0.569	0.389	0.369	0.591
R-S	0.358	0.304	0.542	0.377	0.316	0.570

the evolution of ICEWS18 is not stable due to the much shorter time unit (day).

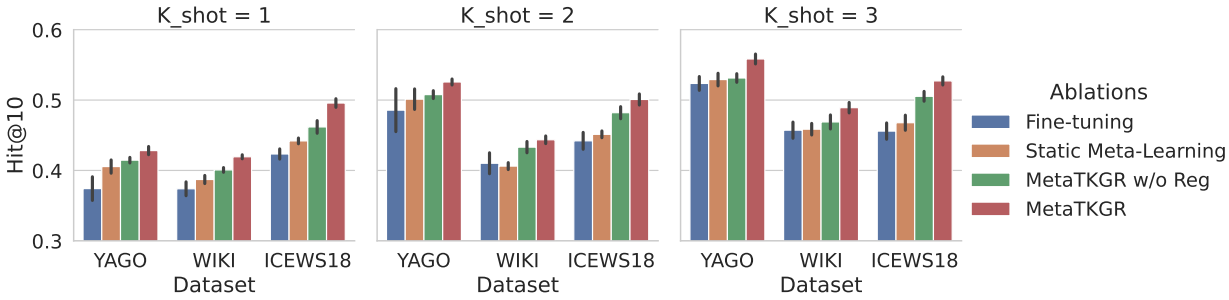


Figure 6.6: Ablation Studies, evaluated by filtered $Hit@10$.

6.4.4 Ablation Study

We evaluate performance improvements brought by the temporal meta-learning framework by following ablations: 1) *Fine-Tuning* is trained on existing entities without using any meta-learning strategy, and then fine-tuned on new entities; 2) *Static Meta-Learning* utilizes conventional MAML to train models; 3) *MetaTKGR w/o Regularizer* meta-trains model parameters on each time interval step by step, without explicitly optimizing the temporal adaptation regularizer. Figure 6.6 reports the results measured by filtered $H@10$. We can conclude that modeling the distribution discrepancy caused by the temporal evolution and optimizing the temporal adaptation regularizer can improve the performance in the future prediction. Also, simply fine-tuning the model parameters on new entities leads to poor results, as during background phase, the model does not extract generalized knowledge that can be easily adapted to new entities.

6.4.5 Cross-shot Learning

In reality, new entities are usually associated with various numbers of facts initially. Thus, the robustness of models on different numbers of shots during testing phase is critical. To

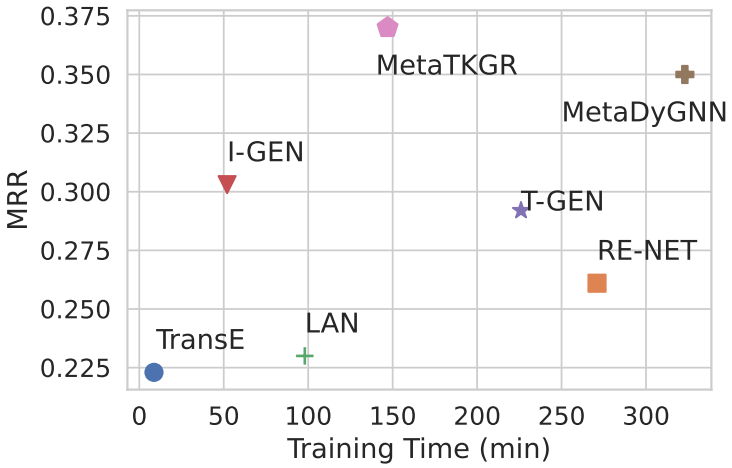


Figure 6.7: MRR over training time.

simulate such scenario, we evaluate MetaTKGR by varying the number of shots including 1-, 3-, 5-, and random-shot (R-S: between 1 and 5) during meta-training and meta-test phases. Table 6.5 reports the cross-shot learning results. As expected, with the increase of observable shots during testing phase, the performance becomes better. The differences in the number of shots used for training do not significantly affect the results, demonstrating that MetaTKGR trained with a fixed number of shots performs robustly under the various number of shots during testing.

6.4.6 Efficiency Analysis

We train MetaTKGR and baseline models from scratch on both existing entities and new entities and compare the training time, for the 3-shot experiment on YAGO. Figure 6.7 shows that MetaTKGR significantly outperforms baseline models with reasonable training time. Compared with slow temporal models *RE-NET, MetaDyGNN* for knowledge graph reasoning, MetaTKGR is more efficient because 1) our temporal encoder can learn temporal entity embeddings via sampled temporal neighbors at each continuous timestamp without using RNNs; 2) our temporal neighborhood sampler can prevent the size of multi-hop neighbors from increasing exponentially.

6.5 THE DESIGN OF METAHKG

In this section, we present the proposed MetaHKG framework for the few-shot temporal reasoning task.

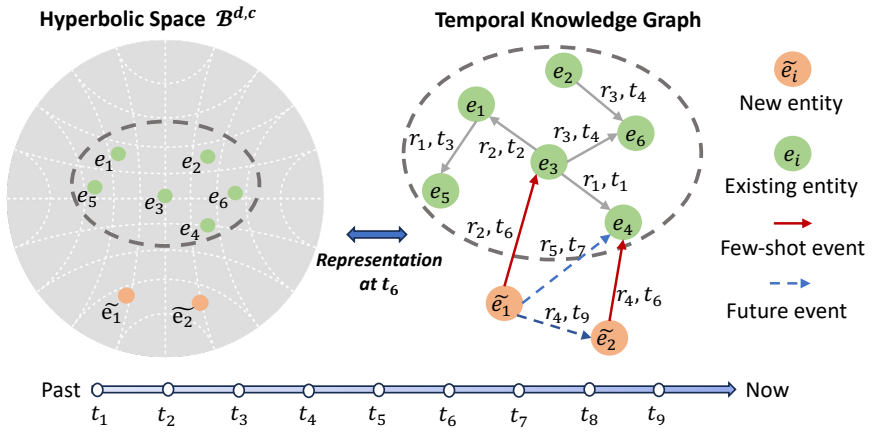


Figure 6.8: An illustrative example of a few-shot temporal reasoning task in a hyperbolic (Poincaré ball) space.

6.5.1 Overview

The challenges associated with enabling few-shot temporal reasoning capabilities in the hyperbolic space encompass two key aspects: (1) *how to learn temporal entity representations in TKGs within the hyperbolic space*, and (2) *how to effectively adapt existing model parameters to accommodate new entities using limited observations*.

To tackle the first challenge, we consider neighbor, time, and relation features in the hyperbolic space, and introduce two techniques: a hyperbolic time encoding method that maps the time domain to hyperbolic space (Section 6.5.2) and a hyperbolic attention network that integrates information from the few-shot neighbors to enhance the representation of new entities (Section 6.5.3). We incorporate the commonly used score function [190] and margin loss function [9], [14] compatible with the novel hyperbolic operations for model training (Section 6.5.4), as illustrated in Figure 6.9.

To address the second challenge, we propose a meta-learning strategy capable of separately learning global and entity-specific parameters (Section 6.5.5). This approach enhances the model's generalization ability to new entities without requiring many observational events initially, as illustrated in Figure 6.10.

6.5.2 Hyperbolic Time Encoding

To represent each new entity $\tilde{e}$ into hyperbolic latent space at each time: $\mathbf{h}_{\tilde{e}}^{\mathcal{B}}(t) \in \mathcal{B}^{d,c}$, we integrate information from temporal neighbors that are evolving over time to capture the entity evolution on hyperbolic space. To achieve the goal, we first learn a map (time encoding) $\phi_{\mathcal{B}} : \mathcal{T} \to \mathcal{B}^{d,c}$ from a time $t \in \mathcal{T}$ to a hyperbolic vector in $\mathcal{B}^{d,c}$. When modeling

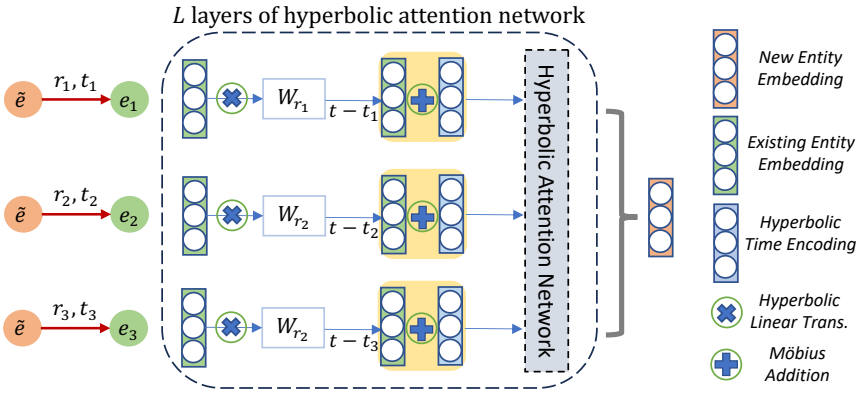


Figure 6.9: Representation learning in Poincaré ball space, with an example of a 3-shot temporal reasoning task.

the interactions between a new entity and its temporal neighbors, we primarily focus on the relative timespan rather than the absolute time value. This is because the difference between the current time and the interaction time conveys more crucial information for effectively representing the new entity at the current time. Without loss of generality, we assume t_i and t_j are two time points to be studied. Therefore, we are interested in the patterns between t_i and t_j and the corresponding inner product between time encoding $\phi_{\mathcal{B}}(t_i)$ and $\phi_{\mathcal{B}}(t_j)$.

Inspired by [5], [144], [193] that utilize the temporal kernel method and Fourier features, we define the temporal kernel $\mathcal{K}_{\mathcal{B}}(t_i, t_j) = \langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}}$, where $\langle \cdot, \cdot \rangle_{\mathcal{B}}$ denotes inner product in the hyperbolic space. To construct $\phi_{\mathcal{B}}(t)$, we first utilize its Euclidean counterpart $\phi_{\mathcal{E}}(t)$ according to [144]:

$$\phi_{\mathcal{E}}(t) = \sqrt{\frac{1}{d}} \left(\cos(\omega_1 t), \sin(\omega_1 t), \cdots, \cos(\omega_{d/2} t), \sin(\omega_{d/2} t) \right), \tag{6.32}$$

where d denotes dimension of the time encoding, $(\cdot, \cdot)$ denotes concatenation, ω_i and θ_i are trainable parameters inside the time encoding. Next we use the exponential map in Eq. (6.2) to transform $\phi_{\mathcal{E}}(t)$ into $\phi_{\mathcal{B}}(t)$:

$$\begin{aligned} \phi_{\mathcal{B}}(t) &= \exp_{\mathbf{0}}^{c}(\phi_{\mathcal{E}}(t)) \\ &= \frac{\tanh(\sqrt{cd/2})}{\sqrt{c/2d}} \left(\cos(\omega_1 t), \sin(\omega_1 t), \cdots, \cos(\omega_{d/2} t), \sin(\omega_{d/2} t) \right). \end{aligned} \tag{6.33}$$

Theorem 6.2 (Translation Invariance)**.** Let the temporal kernel be $\mathcal{K}_{\mathcal{B}}(t_i, t_j) = \langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}}$, where $\phi_{\mathcal{B}}(t)$ is defined in Eq. (6.33). Then the temporal kernel is translation-invariant, *i.e.*, $\mathcal{K}_{\mathcal{B}}(t_i + \Delta t, t_j + \Delta t) = \langle \phi_{\mathcal{B}}(t_i + \Delta t), \phi_{\mathcal{B}}(t_j + \Delta t) \rangle_{\mathcal{B}} = \mathcal{K}_{\mathcal{B}}(t_i, t_j)$, for any constant Δt.

Proof. By Eq. (6.33), the following equation holds:

$$\langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}} = \frac{\tanh^2(\sqrt{cd/2})}{cd^2/2} \langle \phi_{\mathcal{E}}(t_i), \phi_{\mathcal{E}}(t_j) \rangle \,, \tag{6.34}$$

In the Euclidean space, according to the Bochner's theorem and [144], we have $\mathcal{K}(t_i, t_j) = \langle \phi_{\mathcal{E}}(t_i), \phi_{\mathcal{E}}(t_j) \rangle = \psi(t_i - t_j)$, where there exists a function ψ that is translation invariant. Therefore, we have the following for $\mathcal{K}_{\mathcal{B}}(t_i, t_j)$:

$$\mathcal{K}_{\mathcal{B}}(t_i, t_j) = \langle \phi_{\mathcal{B}}(t_i), \phi_{\mathcal{B}}(t_j) \rangle_{\mathcal{B}} = \psi_{\mathcal{B}}(t_i - t_j), \tag{6.35}$$

where $\psi_{\mathcal{B}} = f \circ \psi$ and $f(x) = \frac{\tanh^2(\sqrt{cd/2})}{cd^2/2} x$.

QED.

Different from the existing hyperbolic time encoding for TKGs [194], [195], our time encoding can be easily generalized to new entities and to future time. Compared with the methods that utilize trainable curvatures to encode time, our method is easily generalized to new entities because the number of trainable parameters scales with the number of dimensions representing time. Our method also preserves the translation invariant property and becomes irrelevant to the absolute time. In contrast, [194], [195] utilize absolute time encoding, which cannot be easily generalized to future time beyond the training time range.

6.5.3 Hyperbolic Attention Network

The new entities are associated with a small amount of events initially. Therefore, representation learning requires neighborhood aggregation in an inductive setting to integrate sufficient information. We introduce a hyperbolic attention network with the help of hyperbolic time encoding. Let $\mathbf{h}_{\tilde{e}}^{\mathcal{B}}(t)$ denote the representation of a new entity $\tilde{e}$ at time t, $\mathcal{N}_{\tilde{e}}(t) = \{(e_i, r_i, t_i)\}$ denote the temporal neighbors at time t. For each temporal neighbor, we first compute the relation-aware message delivered by the entity e_i in an event (e_i, r_i, t_i) as follows:

$$\mathbf{h}_{e_i, r_i, t_i}^{\mathcal{B}, l}(t) = \mathbf{W}_{r_i}^{l} \otimes^{c} \mathbf{h}_{e_i}^{\mathcal{B}, l-1}(t_i) \oplus^{c} \phi_{\mathcal{B}}(t - t_i), \tag{6.36}$$

where l denotes the l-th layer, $\mathbf{W}_{r_i}^{l}$ is a relation-aware transformation matrix, $\otimes^{c}$ and $\oplus^{c}$ are interpreted as linear transformation and add operations on the hyperbolic space $\mathcal{B}^{d,c}$, and $\phi_{\mathcal{B}}(t - t_i)$ is the hyperbolic time encoding capturing the temporal pattern between (t, t_i). Thus, the effect of time delay and relation are considered in the message above. We then

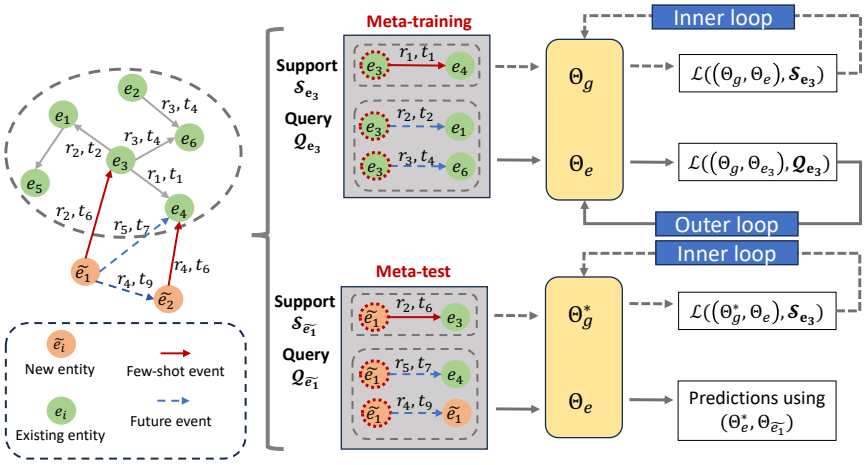


Figure 6.10: Meta hyperbolic learning strategy. Global Θ_g is fixed in the inner loop, while entity-specific Θ_e can be tuned in bi-level optimization. Finally, $(\Theta_g^*, \Theta_{\tilde{e}})$ is utilized for prediction, where $\Theta_{\tilde{e}}$ is fine-tuned on initial few-shot events.

integrate the messages from all temporal neighbors in an attentive manner, *i.e.*,:

$$\mathbf{h}_{\tilde{e}}^{\mathcal{B},l}(t) = \exp_{\mathbf{0}}^{c}\left(\sigma\left(\sum_{(e_i,r_i,t_i)\in\mathcal{N}_{\tilde{e}}(t)} \alpha_{\tilde{e},e_i} \log_{\mathbf{0}}^{c}\left(\mathbf{h}_{e_i,r_i,t_i}^{\mathcal{B},l}(t)\right)\right)\right), \tag{6.37}$$

where $\sigma(\cdot)$ denotes the activation function $ReLU$, $\alpha_{\tilde{e},e_i}$ denotes the attention weight of entity i to new entity $\tilde{e}$, $\exp_{\mathbf{0}}^{c}$ and $\log_{\mathbf{0}}^{c}$ denote the transformation mapping between the hyperbolic space $\mathcal{B}^{d,c}$ and the tangent space of the origin $\mathcal{T}_{\mathbf{0}}\mathcal{B}^{d,c}$. To aggregate from history, $\alpha_{\tilde{e},e_i}$ should be aware of entity feature, time delay, and topology feature induced by relations. Thus, we design $\alpha_{\tilde{e},i}$ as a scaled dot product attention mechanism to characterize the importance:

$$\begin{aligned}
\alpha_{\tilde{e},e_i} &= \frac{\exp(q_{\tilde{e},e_i})}{\sum_{(e_k,r_k,t_k)\in\mathcal{N}_{\tilde{e}}(t)} \exp(q_{\tilde{e},e_k})}, \\
q_{\tilde{e},e_i} &= \frac{1}{\sqrt{d}}\left(\mathbf{W}_k^l \otimes^c \mathbf{h}_{e_i,r_i,t_i}^{\mathcal{B},l}(t)\right)^T \cdot \left(\mathbf{W}_q^l \otimes^c \mathbf{h}_{\tilde{e}}^{\mathcal{B},l-1}(t)\right),
\end{aligned} \tag{6.38}$$

where $q_{\tilde{e},e_i}$ measures the pairwise importance by considering the message embedding and time embedding in the hyperbolic space. Finally, we stack L layers to aggregate information from multi-hop neighbors and obtain the hyperbolic embedding for each new entity. For the sake of simplicity, we omit the layer number and use $\mathbf{h}_{\tilde{e}}^{\mathcal{B}}(t)$ to denote the final output of our attention network. Next, we introduce how to measure the plausibility of each event and design training loss based on the embedding.

6.5.4 Score Function and Training Loss

We adapt commonly-used translation-based score function [9], [14] and margin loss to hyperbolic embedding for plausibility measurement and model training for each event (e_s, r, e_o, t). Let Θ denote model parameters, the plausibility measurement $s(e_s, r, e_o, t; \Theta)$ is formalized as follows:

$$s(e_s, r, e_o, t; \Theta) = -d_{\mathcal{B}} \left(\mathbf{h}^{\mathcal{B}}_{e_s}(t), \mathbf{h}^{\mathcal{B}}_{e_o}(t) \oplus^c \mathbf{h}_r \right)^2, \tag{6.39}$$

where $\mathbf{h}^{\mathcal{B}}_{e_s}(t)$ and $\mathbf{h}^{\mathcal{B}}_{e_o}(t)$ are hyperbolic embeddings at time t of the subject and object entities, $\mathbf{h}_r$ represents the relation embedding in hyperbolic space. Eq. (6.39) is interpreted as a translation score function in hyperbolic space.

To calculate the predictive loss, since each set only contains positive events, we perform negative sampling [9] to update MetaHKG by training it to rank positive events higher than negative ones. Specifically, taking the event (e_s, r, e_o, t) as an example, we construct a negative set (e_s, r, e^-, t) by replacing the ground truth object entity e_o with a corrupted entity e^-. The same corruption is performed on the subject entity e_s. We then use empirical hinge loss on the given set as follows:

$$\mathcal{L}(\Theta, \mathcal{G}^T) = \frac{1}{N|\mathcal{G}^T|} \sum_{\mathcal{G}^T} \sum_{i=1}^{N} \left[s(e_s, r, e_i^-, t; \Theta) - s(e_s, r, e_o, t; \Theta) + \gamma \right]_+, \tag{6.40}$$

where $\mathcal{L}(\Theta, \mathcal{G}^T)$ denotes the loss on graph $\mathcal{G}^T$ based on the model parameter Θ, $\gamma > 0$ is a margin value to distinguish positive and negative events, N is the number of negative samples per each positive event.

6.5.5 Meta Hyperbolic Learning

The proposed hyperbolic attention network naturally applies to new entities during the inference phase (in an inductive setting) so that we can represent new entities in hyperbolic space based on few-shot events. The remaining question is how to adjust model parameters on few-shot events for better reasoning performance. We adopt Model-agnostic meta-learning (MAML) [164] to improve generalization ability on new entities. More formally, each task corresponds to temporal reasoning for each new entity $\tilde{e}$ over distribution $p(\tilde{\mathcal{E}})$. The events are represented as a chronological sequence $\{(\tilde{e}, r_i, e_i, t_i)$ or $(e_i, r_i, \tilde{e}, t_i) | t_i \in (T, T'), t_i \leq t_j$ if $i < j\}_{i=1}^{N_{\tilde{e}}}$, where $N_{\tilde{e}}$ denotes total number of links associated to $\tilde{e}$. Then we divide them into support set $\mathcal{S}_{\tilde{e}} = \{(\tilde{e}, r_i, e_i, t_i)$ or $(e_i, r_i, \tilde{e}, t_i)\}_{i=1}^{K}$, and query set $\mathcal{Q}_{\tilde{e}} =$

Algorithm 6.3: MetaHKG: Training and testing process.

Input: Temporal knowledge graph $\mathcal{G}^T \subseteq \mathcal{E}^T \times \mathcal{R} \times \mathcal{E}^T \times \mathcal{T}$, randomly initialized parameters (Θ_g, Θ_e), K-shot

Output: Learned optimal parameter $(\Theta_g, \Theta_e)^*$.

Simulate new entity set $\tilde{\mathcal{E}}$ from $\mathcal{E}^{(0,T)}$ on meta-training set;
Construct $\mathcal{S}_{\tilde{e}}$ and $\mathcal{Q}_{\tilde{e}}$ for each new entity $\tilde{e}$, where $|\mathcal{S}_{\tilde{e}}| = K$;
 # Training start:
Optimize model parameter (Θ_g, Θ_e) from all events of existing entities $\mathcal{E}^{(0,T)} \setminus \tilde{\mathcal{E}}$ on training set;
while ϕ *not converge* **do**
    # Outer optimization:
    for *each new entity* $\tilde{e}$ **do**
        # Inner optimization:
        Calculate $\mathcal{L}\left((\Theta_g, \Theta_e), \mathcal{S}_{\tilde{e}}\right)$ on $\mathcal{S}_{\tilde{e}}$. by Eq. (6.39) and Eq. (6.40);
        Perform adaptation by inner loop of Eq. (6.41) to update $\phi_{\tilde{e}}$;
        Calculate $\mathcal{L}\left((\Theta_g, \Theta_{\tilde{e}}), \mathcal{Q}_{\tilde{e}}\right)$ by Eq. (6.39) and Eq. (6.40);
    end
    Update (Θ_g, Θ_e) by outer loop of Eq. (6.41);
end
 # Testing start:
for *each new entity* $\tilde{e}$ **do**
    Fix Θ_g^*, fine-tune entity-specific Θ_e on initial $\mathcal{S}_{\tilde{e}}$ into $\Theta_{\tilde{e}}$;
    Evaluate model with $(\Theta_g^*, \Theta_{\tilde{e}})$ on future events $\mathcal{Q}_{\tilde{e}}$;
end

$\{(\tilde{e}, r_i, e_i, t_i)$ or $(e_i, r_i, \tilde{e}, t_i)\}_{i=K+1}^{N_{\tilde{e}}}$, where K denotes the amount of initially observable facts of $\tilde{e}$.

As illustrated in Figure 6.10, we use a bi-level optimization to learn good values for both global and entity-specific model parameters from existing entities and further adapt entity-specific model parameters for few-shot events initially associated with new entities. Let $\Theta = (\Theta_g, \Theta_e)$ denote model parameters. Intuitively, some of the parameters are global (denoted as Θ_g) shared by all entities (both existing and new). In contrast, another set of parameters is entity-specific (denoted as Θ_e), whose optimal value varies from entity to entity. To achieve the goal, during the meta-training phase, we simulate a set of new entities from the existing entity set by assuming there are only few-shot links of these entities (support set $\mathcal{S}_{\tilde{e}}$). The model first fine-tunes entity-specific parameters Θ_e to be $\Theta_{\tilde{e}}$ on $\mathcal{S}_{\tilde{e}}$ (*inner loop*), and then minimizes the predictive loss on corresponding query set $\mathcal{Q}_{\tilde{e}}$ w.r.t. (Θ_g, Θ_e) using the updated parameters $\Theta_{\tilde{e}}$ (*outer loop*). This bi-level optimization that mimics the normal machine learning and inference process optimizes the global parameters as generalized knowledge and entity-specific parameters as good initialization for fine-tuning new entities.

The optimization is formalized as follows:

$$
\begin{aligned}
(\Theta_g^*, \Theta_e^*) \longleftarrow & \underset{(\Theta_g, \Theta_e)}{\arg\min}\, \mathbb{E}_{\tilde{e} \sim p(\tilde{e})} \left[\mathcal{L}\left((\Theta_g, \Theta_{\tilde{e}}), \mathcal{Q}_{\tilde{e}} \right) \right] \quad \text{(outer loop)}, \\
s.t. \quad & \Theta_{\tilde{e}} = \Theta_e - \eta \frac{\partial \mathcal{L}\left((\Theta_g, \Theta_e), \mathcal{S}_{\tilde{e}} \right)}{\partial \Theta_e} \quad \text{(inner loop)},
\end{aligned}
\tag{6.41}
$$

where η denotes the inner-loop learning rate. We include relation embedding, time encoding, and space curvature c for global parameters, as they are largely shared by all entities and critical to shaping the uni-space. The entity-specific parameters include entity embedding and trainable weights in the hyperbolic attention network. During the meta-test phase, given a new entity $\tilde{e}$, we fix the global parameters Θ_g^*, fine-tune Θ_e on few-shot links, and then use the model with parameters $(\Theta_g^*, \Theta_{\tilde{e}})$ to represent $\tilde{e}$ for future predictions. The meta-training/testing phase of MetaHKG is summarized in Algorithm 6.3.

6.6 THE EVALUATION OF METAHKG

We evaluate MetaHKG on three real-world TKG benchmarks, and study the following research questions:

- **RQ1**: How does MetaHKG perform compared with state-of-the-art models on the few-shot temporal reasoning task?
- **RQ2**: How does each design choice and optimization strategy affect performance and model parameters?
- **RQ3**: How does MetaHKG behave differently in hyperbolic space and Euclidean space respectively?

6.6.1 Datasets

We assess the effectiveness of the MetaHKG framework using three publicly available TKGs. Specifically, **YAGO** [10] and **WIKI** [11] contain time-varying facts, and **ICEWS18** [149] is an event-centric TKG. These datasets cover different time ranges and employ varying time units, enabling us to evaluate our framework in diverse contexts of evolution. For each graph, we first split the time duration into 0.4:0.25:0.1:0.25 chronologically, then we collect the entities that first appear in each period as well as the associated facts as training/meta-training/validation/test set. For the entities in the validation and test set, we assume that

Table 6.6: The results of 3-shot temporal reasoning on three datasets. Average results on 5 independent runs are reported. † indicates the results are taken from [5], the same is applied to Table 6.7.

Dataset	YAGO				WIKI				ICEWS18			
Model	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE †	22.3 ± 0.6	15.8 ± 0.4	24.2 ± 0.7	36.0 ± 1.1	16.1 ± 0.7	11.1 ± 0.8	17.1 ± 1.0	23.6 ± 1.1	5.8 ± 0.4	5.2 ± 0.2	6.2 ± 0.7	9.8 ± 1.1
TransR †	23.4 ± 0.9	16.5 ± 0.7	25.9 ± 1.0	38.2 ± 1.3	18.3 ± 0.4	13.8 ± 0.3	18.8 ± 0.6	24.5 ± 0.8	6.1 ± 0.7	6.2 ± 0.3	7.3 ± 0.5	10.9 ± 0.8
DistMult	25.3 ± 0.4	19.3 ± 0.9	27.1 ± 1.3	41.3 ± 0.7	22.3 ± 0.9	18.1 ± 0.9	24.4 ± 1.3	29.3 ± 0.7	9.1 ± 0.4	8.5 ± 0.3	9.9 ± 0.4	14.0 ± 0.5
HolE	23.8 ± 0.5	17.9 ± 0.6	27.1 ± 1.4	39.7 ± 1.3	24.6 ± 0.8	17.0 ± 1.1	23.1 ± 1.4	28.0 ± 1.1	8.1 ± 0.5	9.4 ± 0.8	10.7 ± 1.1	11.8 ± 1.3
ComplEX	26.7 ± 0.9	19.7 ± 0.3	29.1 ± 1.3	42.1 ± 1.1	25.1 ± 0.6	19.1 ± 1.0	23.9 ± 1.4	28.4 ± 1.0	9.3 ± 0.7	9.5 ± 0.5	10.2 ± 1.0	14.1 ± 1.0
RotatE †	24.1 ± 1.2	18.2 ± 0.7	27.8 ± 1.3	40.9 ± 1.1	23.2 ± 0.7	17.1 ± 1.0	22.3 ± 1.3	28.4 ± 1.0	7.8 ± 0.6	7.4 ± 0.7	8.2 ± 0.9	12.8 ± 1.1
AttH	26.9 ± 1.0	21.5 ± 0.8	29.5 ± 1.2	41.2 ± 0.4	24.1 ± 0.9	19.1 ± 0.3	24.6 ± 1.9	30.4 ± 1.1	25.8 ± 0.8	14.1 ± 0.6	22.9 ± 0.2	38.1 ± 0.7
TA-DistMult	24.5 ± 1.9	19.1 ± 0.8	28.7 ± 0.7	40.5 ± 1.1	25.2 ± 0.4	18.9 ± 0.7	24.5 ± 1.0	29.1 ± 0.7	17.8 ± 2.3	11.4 ± 1.1	20.1 ± 1.3	31.2 ± 1.7
RE-NET †	26.1 ± 1.0	21.0 ± 0.9	29.8 ± 0.9	41.0 ± 1.2	26.1 ± 0.6	21.0 ± 0.9	25.1 ± 1.1	33.1 ± 1.0	23.2 ± 1.2	13.9 ± 0.9	24.1 ± 1.2	36.9 ± 1.8
RE-GCN	29.3 ± 2.1	23.1 ± 1.8	31.5 ± 0.8	43.3 ± 1.7	29.4 ± 1.5	23.2 ± 0.8	25.1 ± 1.2	35.4 ± 1.1	23.3 ± 1.1	14.3 ± 1.2	25.2 ± 1.4	35.9 ± 1.1
HERCULES	26.1 ± 0.9	20.1 ± 0.6	28.1 ± 1.1	42.0 ± 1.3	26.8 ± 0.3	22.1 ± 1.1	24.9 ± 1.3	34.8 ± 1.3	24.3 ± 1.3	13.9 ± 0.8	23.8 ± 1.5	36.1 ± 1.3
LAN †	23.0 ± 1.0	15.4 ± 0.7	24.7 ± 0.9	35.2 ± 1.2	18.5 ± 1.5	13.3 ± 1.4	20.1 ± 1.1	28.7 ± 0.7	20.7 ± 0.9	11.9 ± 1.4	23.4 ± 1.5	26.5 ± 1.3
I-GEN†	30.3 ± 1.3	23.8 ± 1.1	32.3 ± 1.1	42.0 ± 1.5	22.1 ± 1.7	17.9 ± 1.3	22.9 ± 1.6	26.4 ± 2.0	21.2 ± 1.4	12.0 ± 1.2	25.1 ± 1.9	34.6 ± 2.0
T-GEN†	29.2 ± 2.7	21.8 ± 2.4	31.0 ± 2.8	39.4 ± 4.1	23.4 ± 2.8	18.5 ± 2.1	22.2 ± 3.0	27.1 ± 3.3	16.9 ± 1.6	12.2 ± 1.3	31.0 ± 2.8	26.5 ± 2.2
MetaDyGNN†	35.0 ± 1.3	27.0 ± 0.9	37.9 ± 1.1	51.1 ± 1.4	30.9 ± 0.8	23.8 ± 0.6	30.9 ± 1.1	45.9 ± 1.3	30.7 ± 0.6	21.6 ± 0.4	30.9 ± 1.1	46.9 ± 1.1
MetaTKGR†	37.0 ± 1.6	30.3 ± 1.2	41.6 ± 1.5	55.8 ± 0.8	32.9 ± 1.2	25.3 ± 1.4	33.5 ± 1.2	48.9 ± 1.4	33.5 ± 1.5	24.9 ± 1.5	34.0 ± 1.4	52.7 ± 1.0
MetaHKG	**38.9 ± 1.1**	**32.8 ± 0.9**	**43.9 ± 1.2**	**58.1 ± 0.5**	**34.1 ± 0.8**	**27.4 ± 1.0**	**36.1 ± 0.8**	**51.3 ± 1.1**	**34.6 ± 1.1**	**26.3 ± 1.2**	**35.8 ± 1.6**	**55.8 ± 0.7**

Table 6.7: The results of 1/2-shot temporal reasoning on three datasets. Average results on 5 independent runs are reported.

Dataset	YAGO				WIKI				ICEWS18			
Model	1-shot		2-shot		1-shot		2-shot		1-shot		2-shot	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
TransE †	18.3 ± 0.8	26.8 ± 0.7	19.3 ± 0.7	30.4 ± 1.3	14.4 ± 0.5	18.6 ± 1.1	14.6 ± 0.6	21.3 ± 0.9	4.9 ± 0.4	7.7 ± 0.7	5.8 ± 0.5	8.6 ± 0.5
TransR †	18.9 ± 0.7	27.0 ± 0.9	19.8 ± 0.7	31.2 ± 0.9	16.0 ± 0.8	18.3 ± 1.1	16.0 ± 0.5	22.5 ± 0.9	5.0 ± 0.5	8.0 ± 0.9	6.0 ± 0.3	9.0 ± 0.7
DistMult	21.0 ± 0.6	29.1 ± 0.7	20.9 ± 1.1	34.8 ± 0.8	16.3 ± 0.4	18.5 ± 0.3	21.4 ± 1.1	25.9 ± 0.6	7.1 ± 0.3	10.4 ± 0.5	8.1 ± 0.7	10.3 ± 0.5
HolE	22.6 ± 0.4	30.1 ± 0.6	21.8 ± 1.5	35.1 ± 0.9	18.6 ± 0.4	18.9 ± 1.2	22.8 ± 1.5	24.9 ± 1.0	6.9 ± 0.3	11.4 ± 0.8	9.1 ± 1.2	11.0 ± 0.6
ComplEX	24.9 ± 1.0	31.7 ± 0.4	23.0 ± 1.4	37.6 ± 1.2	19.1 ± 0.7	20.5 ± 0.7	24.1 ± 0.8	25.4 ± 0.6	7.9 ± 0.4	12.5 ± 0.4	10.7 ± 0.9	12.0 ± 0.9
RotatE †	21.5 ± 0.5	28.0 ± 1.1	21.0 ± 1.0	35.9 ± 0.9	17.5 ± 0.9	19.0 ± 1.1	20.1 ± 0.9	26.8 ± 0.8	6.8 ± 0.6	9.8 ± 1.0	7.0 ± 0.8	9.1 ± 0.8
AttH	21.9 ± 0.4	30.6 ± 0.7	29.5 ± 0.9	40.9 ± 0.5	21.4 ± 0.5	23.2 ± 0.1	21.5 ± 1.0	28.4 ± 0.9	10.1 ± 0.6	13.4 ± 0.5	19.7 ± 0.6	29.1 ± 0.5
TA-DistMult	21.7 ± 1.2	29.4 ± 0.3	22.4 ± 1.1	32.8 ± 1.7	18.2 ± 0.6	21.3 ± 0.4	21.9± 1.2	27.6 ± 0.8	11.7 ± 0.1	15.1 ± 1.2	17.6 ± 0.9	32.1 ± 1.1
RE-NET †	22.1 ± 0.6	30.4 ± 0.7	23.3 ± 0.8	39.0 ± 1.0	21.2 ± 1.0	25.9 ± 1.0	23.9 ± 0.5	29.4 ± 1.3	18.5 ± 1.3	25.0 ± 1.3	20.0 ± 1.0	34.1 ± 1.9
RE-GCN	23.1 ± 1.0	33.1 ± 1.9	24.5 ± 0.9	41.8 ± 1.6	22.1 ± 1.0	25.9 ± 0.9	24.9 ± 1.3	32.3 ± 0.9	19.1 ± 0.5	24.8 ± 1.0	21.9 ± 1.1	35.9 ± 1.0
HERCULES	20.7 ± 0.9	31.1 ± 0.5	21.1 ± 0.9	40.1 ± 1.1	21.3 ± 0.4	22.7 ± 1.2	22.4 ± 1.1	28.8 ± 1.0	12.1 ± 0.9	16.1 ± 0.7	18.8 ± 0.6	33.0 ± 0.3
LAN †	19.6 ± 1.5	26.9 ± 2.0	20.0 ± 0.7	31.0 ± 1.0	17.4 ± 1.6	27.5 ± 2.0	16.2 ± 1.0	27.3 ± 1.1	17.0 ± 2.1	30.1 ± 2.0	18.8 ± 0.8	31.7 ± 1.3
I-GEN†	23.8 ± 0.7	32.1 ± 1.1	23.7 ± 1.1	40.2 ± 1.7	18.1 ± 0.8	24.1 ± 1.3	22.3 ± 0.9	28.7 ± 1.3	19.9 ± 0.8	32.0 ± 1.3	17.7 ± 1.7	33.7 ± 2.0
T-GEN†	24.7 ± 2.3	33.1 ± 3.1	26.0 ± 2.9	37.9 ± 3.2	20.2 ± 1.9	24.5 ± 2.4	24.0 ± 2.3	31.9 ± 2.8	13.1 ± 3.1	26.2 ± 3.2	16.1 ± 3.2	25.9 ± 3.1
MetaDyGNN†	26.9 ± 1.0	39.6 ± 1.5	31.6 ± 0.9	49.6 ± 1.1	24.1 ± 1.3	37.1 ± 1.6	27.1 ± 1.0	39.0 ± 1.8	24.9 ± 1.2	42.0 ± 1.8	26.9 ± 1.1	44.1 ± 1.5
MetaTKGR†	29.4 ± 1.2	42.8 ± 0.9	35.6 ± 1.3	52.6 ± 1.0	27.7 ± 1.5	41.9 ± 1.0	30.9 ± 1.4	44.1 ± 1.3	29.5 ± 0.7	49.6 ± 1.2	30.0 ± 1.1	50.0 ± 1.4
MetaHKG	**31.7 ± 1.1**	**45.1 ± 1.2**	**36.9 ± 1.3**	**54.1 ± 0.9**	**29.5 ± 1.2**	**44.1 ± 0.7**	**31.8 ± 1.3**	**47.1 ± 1.0**	**31.0 ± 0.8**	**51.9 ± 1.0**	**32.1 ± 1.7**	**53.1 ± 0.8**

their first $K = 1, 2, 3$ facts are known for model adaptation. Detailed information is provided in Section 6.4.1.

6.6.2 Experimental Setup

Baselines. We compare 16 state-of-the-art baselines from three related areas:

- **Static KG reasoning**: *TransE* [9], *TransR* [14], *DistMult* [15], *HolE* [196], *ComplEX* [16], *RotatE* [17], *AttH* [197];
- **Temporal KG reasoning**: *TA-DistMult* [33], *RE-NET* [137], *RE-GCN* [22], *HERCULES* [194];
- **Few-shot KG/graph reasoning**: *LAN* [161], *I-GEN* [162], *T-GEN* [162], *MetaDyGNN* [192], *MetaTKGR* [5].

We describe the baseline models utilized in the experiments in detail:

- **TransE** [9] adopts a translation-based embedding approach, representing entities and relations as vectors in a latent space. The relations act as operators, facilitating the transition between subject and object entities;
- **TransR** [14] enhances entity-relation modeling from TransE by projecting entity embeddings onto relation-defined hyperplanes for improved representation;
- **DistMult** [15] is a neural-embedding approach designed for multi-relational representation learning. It employs a bilinear model for link prediction and leverages learned relation embeddings to extract logical rules;
- **HolE** [196] employs circular correlation to construct compositional representations within holographic embeddings for knowledge graph vector space learning;
- **ComplEX** [16] ComplEX utilizes complex-valued embeddings to enhance the capture of both symmetric and antisymmetric relations through dot products;
- **RotatE** [17] embodies entities using intricate vector representations and expresses relations through rotational operations within a complex vector space;
- **AttH** [197] is a hyperbolic knowledge graph model integrating hyperbolic reflections, rotations, and attention to capture hierarchical, logical, and complex relational patterns;
- **TA-DistMult** [33] learns time-aware representations by utilizing recurrent neural networks on time-related token sequences, combining the last hidden state with standard KG completion scoring functions;
- **RE-NET** [137] introduces a generative approach for forecasting future facts in temporal knowledge graphs using a recurrent neural network for entity evolution modeling and a neighborhood aggregator to capture interrelations at the same time intervals;
- **RE-GCN** [22] captures temporal representations of entities and relations through recurrent modeling of the knowledge graph sequence;
- **HERCULES** [194] extends the AttH model to incorporate time awareness, with the unique feature of defining the Riemannian manifold's curvature as a combination of both relation and time factors;

- **LAN** [161] generates entity embeddings through a GNN-based neighbor aggregation approach and incorporates attention mechanisms to account for relations and neighboring information for new entities;
- **I-GEN** [162] employs a meta-learning technique for acquiring representations of new entities through the attentive aggregation of information from their neighbors;
- **T-GEN** [162] is an extension of I-GEN, focusing on enhancing predictions for unseen-unseen links between unseen users through the introduction of stochastic inference to capture the randomness associated with these connections;
- **MetaDyGNN** [192] advances the modeling of link predictions for new nodes in homogeneous graphs. It introduces a hierarchical meta-learner, which enhances the extraction of global knowledge, thus benefiting the link prediction task and other aspects;
- **MetaTKGR** [5] employs dynamic neighbor sampling and aggregation strategies guided by temporally supervised signals and a temporal adaptation regularizer to ensure stable meta-temporal reasoning for new entities.

Evaluation Protocol and Metrics. In evaluating each prediction of $(\tilde{e}, r, ?, t)$ or $(?, r, \tilde{e}, t)$, we use ranking scheme. Concretely, we rank all entities at the missing position in each event and adopt mean reciprocal rank (*MRR*) and Hits at {1,3,10} (*H@{1,3,10}*) as evaluation metrics. MRR on the validation set is utilized to select optimal hyperparameters.

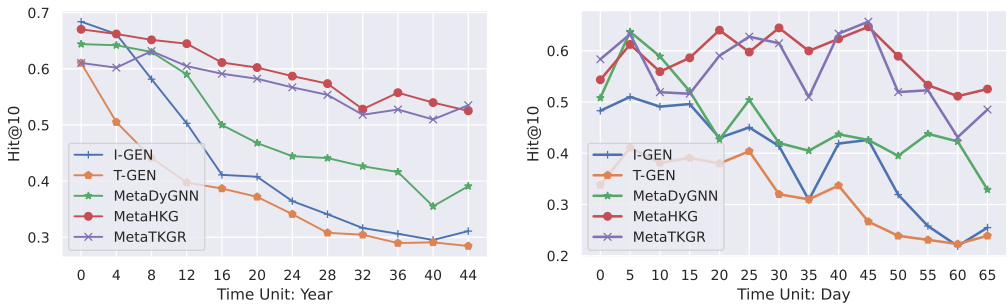
Implementation. For static knowledge graph reasoning, i.e., TransE, TransR, DistMult, HolE, ComplEX, RotatE, and AttH, we treat temporal knowledge graphs as static, cumulative datasets, and train models using the training set, meta-training set, and the first K triples of new entities in meta-validation and meta-test sets. For temporal knowledge graph reasoning methods TA-DistMult, RE-NET, RE-GCN, and HERCULES, the models are trained with the training set, meta-training set, and the first K events associated with each new entity present in the meta-validation and meta-test sets. Baseline methods optimizing few-shot entities on static knowledge graphs, i.e., LAN, I-GEN, and T-GEN, and dynamic knowledge graphs, i.e., MetaTKGR, follow a similar procedure, omitting temporal event information. In the case of MetaDyGNN, originally designed for homogeneous graphs, we adopt it to temporal knowledge graphs by incorporating TransE's score function. Initial model parameters are trained using training and meta-training sets, with entity-specific model parameters adapted based on few-shot facts of new entities. To ensure fair comparisons, we maintain a consistent embedding of 128, employ pre-trained 1-shot TransE embeddings for models requiring initial entity/relation embeddings, and conduct all training and testing on the same hardware, including GeForce RTX 3090 GPUs and AMD Ryzen Threadripper 3970X 32-Core Processors.

Figure 6.11: Performance breakdown for new-to-existing and new-to-new prediction.

We initialize our model with the training dataset and simulate few-shot tasks using the meta-training set to adapt model parameters and enhance temporal generalization. For new entities, we fine-tune entity-specific models using $K = 1, 2, 3$ initial facts. During the evaluation, we optimize hyperparameters based on MRR on the meta-validation set and report performance on the remaining facts in the meta-test set. Our model training employs that Adam optimizer with a maximum of 50 epochs, a batch size of 20, 128-dimensional embeddings, and a dropout rate of 0.5. To enhance efficiency, we set the neighbor budget b of the temporal neighbor sample to 16 and incorporate 2 neighborhood aggregation layer within the temporal encoder. In the inner loop optimization process, we conduct a single step of gradient descent. The primary hyperparameters we primarily tune include the margin value γ in the score functions, varying within the range of $0.3, 0.4, 0.5, 0.6, 0.7$. Additionally, we adjust the learning rates for both the inner and outer loops, η and β within the range of $0.1, 0.05, 0.01, 0.005, 0.001, 0.0005$. For YAGO and ICEWS18, we set $\gamma = 0.5$, $\eta = \beta = 0.0001$. For WIKI, we use $\gamma = 0.6$, $\eta = 0.0001$, and $\beta = 0.00005$.

6.6.3 Main results (RQ1)

Tables 6.6 and 6.7 present the comprehensive results for few-shot experiments with $K = 3$ and $K = 1, 2$, respectively. In each of these tasks, MetaHKG shows a remarkable 5.2% relative improvement over the best baseline model, thus underscoring the superior capacity of MetaHKG in effectively modeling the evolution of new entities under data-constrained conditions. Notably, the baselines belonging to the first two categories, which do not optimize specifically for new entities, yield suboptimal performance across all datasets. This is despite their training on both existing and new entities with a few-shot link setup. We also observe the unsatisfying performance for hyperbolic baselines *AttH* and *HERCULES*. This subpar



(a) Cross-shot results on YAGO.

(b) Cross-shot results on ICEWS18.

Figure 6.12: Performance of YAGO (left) and ICEWS18 (right) over time.

performance is primarily due to their neglect of special optimization for new entities in future predictions. In contrast, MetaHKG consistently outperforms MetaDyGNN and MetaTKGR. This success can be attributed to our method's effective utilization of hyperbolic geometry, which proves more suitable for the few-shot temporal reasoning task. Additionally, our framework enhances sample efficiency in the few-shot scenario via the meta hyperbolic framework.

Performance over Time. Subsequently, we assess the performance of MetaHKG across various timeframes. Figure 6.12 illustrates performance comparisons for 3-shot predictions at different timestamps on the *YAGO* and *ICEWS18*, with the evaluation conducted using filtered $H@10$ metrics. In these comparisons, MetaHKG consistently outperforms all strong baselines. It's worth noting that while the compared baselines can optimize their models for newly emerging entities, their performance diminishes significantly over wider time intervals. This performance drop can be attributed to their insufficient modeling of temporal information. Notably, although MetaHKG does not consider the distribution shift directly as MetaTKGR, we can still perform better in longer future predictions. And MetaHKG produces more stable prediction results over time on ICEWS18. Our hypothesis suggests that the superiority is primarily derived from the hyperbolic time encoding, which can be readily extended to accommodate future time instances.

Performance Breakdown for Event Types. The events for new entities fall into two distinct categories: *new-to-existing* events, which occur between new entities and existing ones, and *new-to-new* events, which transpire exclusively among new entities, as depicted in Figure 6.3. To provide a more detailed insight into the performance enhancement attributed to MetaHKG, Figure 6.11 offers a performance breakdown for each event type. In general, predictions for "new-to-new" events tend to yield relatively poorer results due to the greater difficulty in simultaneously optimizing representations for all few-shot entities. Nevertheless,

Table 6.8: Ablation Studies.

Dataset	YAGO		WIKI		ICEWS18	
Ablation	**MRR**	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**
MetaHKG **w/o meta-learning**	30.7	49.8	30.8	44.6	28.4	46.7
MetaHKG **w/o fine-tuning**	32.9	52.1	32.0	48.1	30.4	50.1
MetaHKG **w global meta-learning**	38.2	56.7	33.1	49.7	33.4	54.0
MetaHKG **w/o time encoding**	37.9	56.8	32.8	50.1	32.3	53.7
MetaHKG **in Euclidean space**	37.1	55.6	31.2	48.1	32.9	51.9
MetaHKG	**38.9**	**58.1**	**34.1**	**51.3**	**34.6**	**55.8**

MetaHKG surpasses all strong baselines for both event types. It is noteworthy that we observed more substantial relative improvements in the context of "new-to-new" event predictions. This is largely attributed to MetaHKG's ability to position the clusters of new entities more effectively within the hyperbolic latent space, leading to improved predictions.

6.6.4 Effect of optimization strategy (RQ2)

We evaluate performance improvements brought by the training strategy of MetaHKG framework by following variants:

- **MetaHKG w/o meta-learning** trains model parameters on all events of existing entities and few-shot events of new entities without the proposed meta-learning approach;
- **MetaHKG w/o fine-tuning** first trains model parameters on all events of existing entities, and then fine-tunes parameters on few-shot events of new entities without the proposed meta-learning approach;
- **MetaHKG w global meta-learning** meta-trains model parameters without distinguishing global parameters and entity-specific parameters;

Table 6.8 reports the results measured by $H@10$. Training model parameters on all events, including the few-shot cases, or attempting fine-tuning exclusively on new entities results in inferior outcomes. This is because they fail to extract generalized knowledge that can be effectively applied to new entities. Furthermore, the decline in performance observed when applying global meta-learning, which combines both global parameter Θ_g and entity-specific parameter Θ_e without distinction, highlights the advances of our meta-learning strategy in capturing global and entity-specific knowledge.

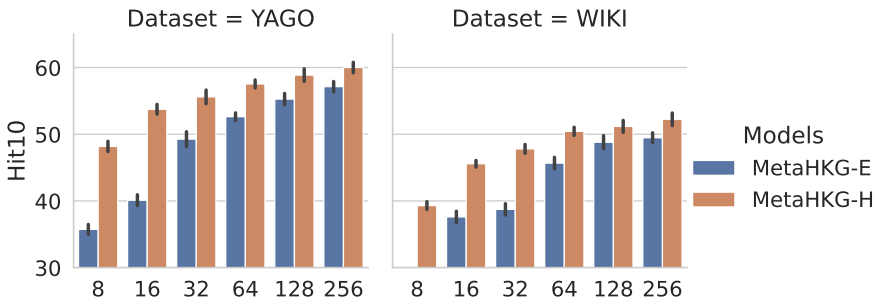


Figure 6.13: Performance comparison w.r.t. dimension.

6.6.5 Effect of Hyperbolic Space (RQ3)

Performance gains via hyperbolic embedding. We first examine the effect of hyperbolic embedding by following variants:

- **MetaHKG w/o time encoding** eliminates the utilization of time encoding where time only impacts the temporal neighbors;
- **MetaHKG in Euclidean space (MetaHKG-E)** represent entity, relation, and time into corresponding Euclidean space, where the number of dimensions is the same (128);

Table 3.4 reports the results measured by $H@10$. Removing hyperbolic time encoding hurts the performance, as we ignore temporal patterns existing in the entity interactions while representing new entities. Training MetaHKG in Euclidean space (MetaHKG-E) also leads to a performance drop. Because of power-law distribution and hierarchical structures of new entities, a higher order of distortion might be required in Euclidean space to produce competitive performance with a lower dimensional hyperbolic space.

Performance w.r.t. Dimension. We further investigate the performance comparison between MetaHKG-E (in Euclidean space) and MetaHKG-H (in hyperbolic space) by varying the dimension of latent space. Figure 6.13 reports the results. It is observed that MetaHKG-H requires a lower dimensional latent space to achieve satisfying performance, while the corresponding MetaHKG-E produces much worse results. This comparison further indicates that MetaHKG requires fewer spatial adjustments and yields a more coherent embedding space for representing new entities.

Convergence Analysis. We conducted an analysis of how the hyperbolic space contributes to the meta-learning process. Figure 6.14 depicts the loss curves concerning batch iterations during the meta-training process of both MetaHKG-E and MetaHKG-H. These loss curves represent the predictive loss on the query set during the outer loop optimization, corresponding to the global knowledge learning process. To ensure a fair comparison, we

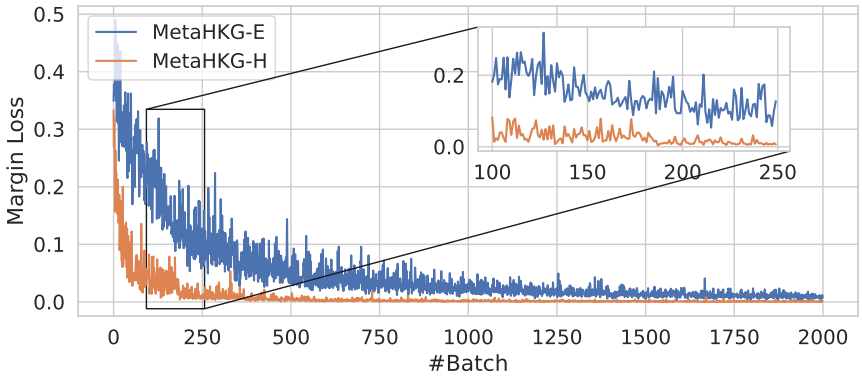


Figure 6.14: Convergence of MetaHKG-E and MetaHKG-H.

maintained the same parameter settings for both models. First and foremost, it's evident that MetaHKG-H converges more rapidly than MetaHKG-E. This suggests that learning the few-shot reasoning task in the hyperbolic space is inherently more manageable due to the alignment between its geometric properties and the distribution and structures found in Temporal Knowledge Graphs (TKGs). Additionally, we observed periodic variations in the predictive loss. When processing a batch of new entities, the predictive loss on the new query set (future events) based on fine-tuned model parameters initially increases. This is because the global parameters have never been adapted to the batch of new entities beforehand. However, the loss gradually decreases after the initial spike as the model begins to encode information about the new entities. Importantly, MetaHKG-H produces a smoother loss curve, indicating that adapting the model to new entities in the hyperbolic space is a more straightforward process.

6.7 RELATED WORK

6.7.1 Few-shot Learning on Graphs

Few-shot learning [158], [198], [199] focuses on acquiring generalized knowledge from existing tasks to derive transferable priors for new tasks, particularly when faced with limited labeled data. This includes approaches based on metric learning [200], [201] and meta-learning [164], [165], [202]. For graph reasoning, prior works study representation learning of scarce entities [160]–[162], [203] and scarce relations [170], [204]–[206]. In this paper, we are primarily interested in works to represent entities with few-shot events. The neighborhood aggregator used in inductive graph neural networks (GNNs) [18], [60], [166] can be naturally extended to represent a new entity [160]. [161], [203], [206] further extend this

line of work by utilizing attention mechanisms. However, those models are usually trained on both many-shot and few-shot entities, ending up being suboptimal for few-shot entities. Recent work [162] explores how to effectively mate-train the neighborhood aggregator to adapt global knowledge for few-shot reasoning. But they are limited to unseen entities out of simulation, not based on temporal appearance. Recent studies of temporal graph models mainly focus on improving reasoning performance for existing entities and ignore newly emerging ones [22], [137], [152]–[154], [207]. For few-shot temporal reasoning tasks, MetaTKGR [5] is a recent framework combining MAML. However, it did not consider the rapid growth of new entities and their power-law distribution and hierarchical structures. We aim to tackle the few-shot temporal reasoning task in a hyperbolic space and utilize its geometric properties.

6.7.2 Temporal graph Reasoning

Temporal graphs (TKGs) store time-varying facts in the real-world. Temporal graph reasoning aims to predict missing facts at a certain time in the future. It is mostly formulated as measuring the correctness of factual samples and negative samples by specially designed score functions [8], [9], [14], [16], [17], [33], [132]. Compared with static KG reasoning tasks [140], the main challenge lies in how to incorporate time information into the representation process. Several embedding-based methods have been proposed. They encode time-dependent information of entities and relations by decoupling embeddings into static component and time-varying component [152], [153], utilizing recurrent neural networks (RNNs) to adaptively learn the dynamic evolution from historical fact sequence [137], [207], or learning a sequence of evolving representations from discrete graph snapshots [137], [154], [208]. However, all of the existing temporal KG reasoning models aim to extrapolate future facts among existing entities, and how to predict future facts specifically for new emerging entities is largely under-explored.

6.7.3 Hyperbolic Learning

Hyperbolic geometry has found broad utility in natural language processing [182]–[184], computer vision [178]–[181], and graph learning [185]–[189], [194], [197]. In recent graph embedding developments, methods like AttH [197] and HERCULES [194] extend KG embedding into hyperbolic space, addressing both static and temporal graph reasoning by incorporating reflections, rotations, and attention mechanisms to capture hierarchical, logical, and complex relational patterns. Hyperbolic geometry's exceptional performance and adaptability in

managing hierarchical structures motivate its exploration in few-shot temporal reasoning. To the best of our knowledge, we are the first to explore the few-shot temporal reasoning capabilities within the hyperbolic space.

6.8 SUMMARY

In this chapter, We study a realistic but underexplored few-shot temporal graph reasoning problem, which aims at predicting future facts for newly emerging entities with a few facts. To this end, we propose a novel Meta Temporal graph Reasoning framework MetaTKGR. It meta-learns the global knowledge of sampling and aggregating temporal neighbors, which can be adapted quickly to new entities for future prediction. Such procedure is gradually guided by the performance on predicting future facts, from near time intervals to far away ones. We further theoretically analyze and propose a temporal adaptation regularizer to stabilize and generalize the learned knowledge on future tasks. To improve MetaTKGR, we proposed a meta hyperbolic learning framework (MetaHKG) within a hyperbolic space. We found its geometric property matches the rapid growth of new entities, effectively capturing their power-law distribution and hierarchical structures. We proposed a time encoding and an attention network to represent new entities in a hyperbolic space. They are optimized by a novel meta-learning approach to facilitate effective model adaptation. Extensive experiments verified the effectiveness of MetaTKGR and MetaHKG on three real-world temporal graphs.

Chapter 7: LEARNING STREAMING GRAPHS FOR RESOURCE-EFFICIENT MODEL UPDATES

7.1 OVERVIEW

In this chapter, we study online link prediction on streaming temporal graphs, aiming to efficiently update deployed models on freshly acquired temporal data to ensure sustained long-term performance. Temporal graphs refer to a specialized type of graph structure that captures time-varying links among nodes [144], [209], which are instrumental in modeling and understanding the evolution of diverse real-world systems that change over time, *e.g.,* dynamic social networks [210], user-product interaction systems in E-commerce [3], dynamical systems in physics [211]. Learning representations of temporal graphs for future link prediction has been attracting increasing attention, owing to the multitude of downstream applications it supports [66], [92], [108], [144], [212].

Temporal graphs exhibit a distinct property that distinguishes them from their static counterparts. This distinction arises from the continuous appearance of new information in the form of streaming links or nodes over time, driven by the underlying system evolution [209]. For instance, fresh follower-followee relationships frequently emerge on social graphs, constituting *streaming links*; while E-commerce graphs commonly see additions of new users and products, exemplifying *streaming nodes*. This dynamic nature necessitates implementing a proficient and streamlined approach for online graph model updates, accurately and efficiently reflecting the information evolvement in a streaming fashion. Unfortunately, despite the significant efforts invested into (offline) temporal graph learning [66], [92], [108], [112], [144], [212], significant performance degradation happens during streaming information updates, because the offline training paradigm lacks adequate support for elastic online model adaptation. Instead, this paper aims to develop a temporal graph learning technique that facilitates effective and efficient online updates, boosting link prediction performance in streaming scenarios, as shown in Figure 7.1.

Conventional methods, such as retraining the model on the complete dataset or fine-tuning exclusively on new data, are straightforward but exhibit unsatisfying performance. The former approach demands significant computational resources and might overlook newly introduced information [213], while the latter method is prone to overfitting on new data and could face catastrophic forgetting issues [214]. Consequently, the central hurdle in tackling online learning tasks for temporal graphs centers on devising a strategy to effectively and efficiently represent information from both pre-existing and new data. Inspired by the recent success in knowledge distillation [215] and data condensation [216], [217], our study centers around a

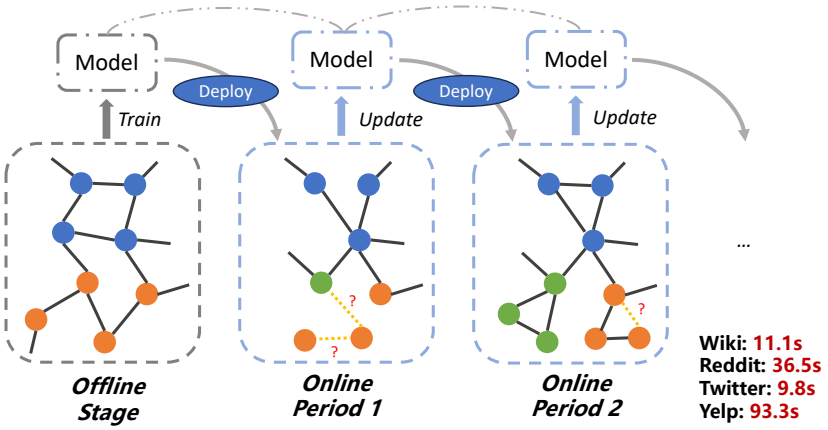


Figure 7.1: An example of streaming temporal graph, where new nodes and edges continuously emerge. We study online link prediction problem on it, which seeks to aiming to efficiently update deployed models on freshly acquired temporal data to ensure sustained long-term performance. **Shown**: Online update time of OnlineSAFE, which is faster than vanila fine-tuning with better performance.

key philosophy: *update the deployed model by retaining and adapting informative knowledge distilled from existing data onto freshly gathered data, as opposed to directly perpetuating and reutilizing the original data.* Hence, it calls for a novel training strategy that distills enduringly valuable knowledge from existing data and seamlessly adapts it to newly gathered information, ensuring efficient adaptation and robust performance over time.

The fulfillment of the novel training strategy poses challenges in two aspects: **First**, considering the model complexity of representing temporal graphs, how to distill enduringly valuable knowledge via the sophisticated models, especially for newly emerging nodes/edges, is non-trivial. Recent efforts have explored both light-weight model refreshing [213], [218], [219] and continual graph learning based on streaming graph neural networks (GNNs) [220]–[225]. However, these methods primarily focus on updating shallow and static models, and their capability of modeling temporal graphs remains limited, especially for online link prediction task. **Second**, periodical data collection necessitates fast adaptation of the distilled knowledge onto new data to update the deployed model.

To overcome both the aforementioned challenges, we propose a novel temporal meta-training framework, namely OnlineSAFE, where the strategy of aggregating information from temporal neighbors to update models on new data, parameterized by a temporal encoder, is viewed as the enduringly valuable knowledge across data collection periods. To extract such knowledge, we formulate a meta-task as adapting global knowledge on recently collected new data to update the model for future link prediction, where the quality of global knowledge can be further improved by measuring how well the updated models perform on

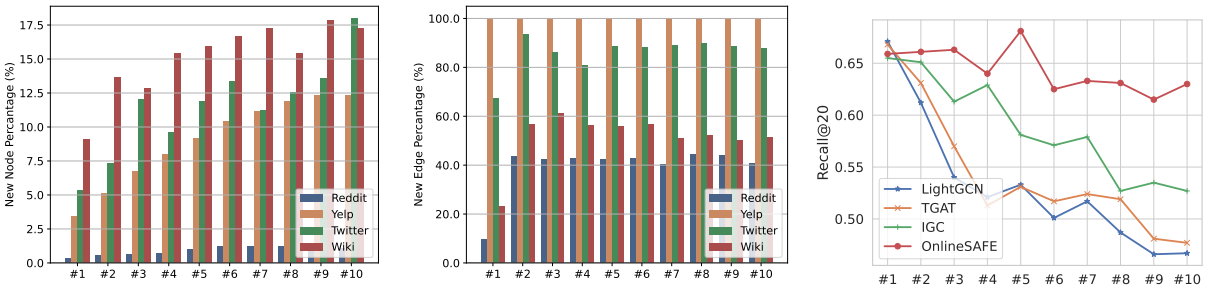
the future data as instant training signals. Such a nested meta-optimization process includes an inner-loop adaptation of global knowledge on newly collected data and an outer-loop training to meta-learn the global knowledge. After that, the extracted knowledge can be efficiently adapted to new data for model updates via several steps of lightweight fine-tuning during the online phase. Furthermore, to better estimate the supervision signal given by new data in potentially different and unseen distributions, we adopt the PAC-Bayes method [173] to analyze the temporal adaptation bound on the new data theoretically. This can serve as an adaptation regularizer to provide stability and improve the generalization ability of the current strategy over time. Finally, to improve the adaptation efficiency, we propose a simple yet effective strategy to reduce the number of training samples but sustain the adaptation performance. Empirically, extensive experiments on four real-world streaming temporal graphs validate the effectiveness of OnlineSAFE, which significantly outperforms 17 baselines, including both static/temporal graph learning methods and online graph learning methods, by up to 8.8% relative gains on average in terms of Recall and NDCG, with better efficiency than the vanilla fine-tuning strategy.

7.2 PRELIMINARIES

7.2.1 Problem Definition

Definition 7.1 (**Streaming Temporal Graphs).** A streaming temporal graph consists of a set of temporal edges $\{(u, v, t)\}$, where $u, v \in \mathcal{V}$ denotes the source and target nodes, t denotes the specific timestamp attached to the temporal edge. In reality, the temporal edges emerge continuously over time, and they are collected periodically during each collection period. Therefore, we denote $\mathcal{D}^T$ as the set of temporal edges collected at the T-th collection period, *i.e.,* $\mathcal{D}^T = \{(u, v, t)|T - 1 \leq t < T\}$. Formally, a streaming temporal graph is denoted as a sequence of edge collections $\mathcal{D} = \{\mathcal{D}^0, \mathcal{D}^1, \cdots, \mathcal{D}^T\}$, where we denote $\mathcal{G}^{T_i}$ as the corresponding graph snapshot induced by the newly collected temporal edges in $\mathcal{D}^{T_i}$.

Definition 7.2 (**Online Learning on Streaming Temporal Graphs).** At the current collection period T, let Θ^{T-1} denote the model parameters trained on the existing data and already deployed online, $\mathcal{D}^T$ and $\mathcal{G}^T$ denote the newly collected temporal edges and the corresponding graph snapshot respectively, the online learning on streaming temporal graphs aims to update the new model parameters Θ^T based on both existing and newly emerging data $\mathcal{D}^{t\leq T}$ and $\mathcal{G}^{t\leq T}$ to maximize the future predictive performance, *i.e.,* $\Theta^T \leftarrow f(\mathcal{D}^{t\leq T}, \mathcal{G}^{t\leq T}, \Theta^{T-1})$, where $f(\cdot)$ denotes the updating strategy that we study.



(a) New Node Emergence. (b) New Edge Emergence. (c) Performance on Twitter Data.

Figure 7.2: An motivating example of online model refreshing on temporal graphs. New nodes and edges continuously emerge, leading to significant performance decay. We aim to design a temporal graph learning method that enables effective and efficient online refreshing to facilitate future link prediction performance in a streaming manner.

7.2.2 Motivating example

We investigate the temporal evolution of four real-world datasets. In Figure 7.2, we illustrate the percentage of newly appearing nodes, defined as nodes that have not been present in the temporal graph before, during the online phase spanning the 1st to the 10th data collection periods, which include online validation and online testing. Notably, all the datasets exhibit a substantial influx of new nodes into their graphs over time, with particularly significant proportions for Yelp ($\sim 12\%$), Twitter ($\sim 18\%$), and Wiki ($\sim 17\%$). As expected, we also observe a substantial proportion of newly formed edges, defined as connections between two nodes that had not interacted previously, as time progresses across all four datasets. This trend is visually represented in Figure 7.2, where, notably, the Yelp dataset even reaches a point of nearly 100% new edges during the online phase.

These observations naturally lead to the question of how well the deployed model, initially trained on offline data without any fine-tuning during the online phase, performs on such dynamically evolving graphs. To address this, we selected three representative baselines from each category: *LightGCN* from the static graph learning field, *TGAT* from dynamic/temporal graph learning, and *IGC* from online graph learning. We evaluated their performance during the online periods without any adjustment. The results, shown in Figure 7.2, reveal a significant performance decline for LightGCN and TAGT as time progresses, despite their promising initial performance upon deployment. IGC also exhibits progressively deteriorating performance, even though it can update the deployed model to some extent. These outcomes suggest that existing methods struggle to effectively capture and adapt to the new information presented in evolving temporal graphs. In light of these findings, our objective is to develop a temporal graph learning method that enables efficient and effective online updates, thereby

enhancing future link prediction performance in a streaming fashion.

7.2.3 Meta-Learning Formulation

Given a set of tasks, meta-learning aims to learn general knowledge that is shared across all tasks and can be efficiently adapted to new tasks [226]. We follow model-agnostic meta-learning (MAML) [164] and formulate a task as adapting global knowledge on recently collected new data to update the model for future link prediction. Concretely, in each task at the data collection time T, the global knowledge can be adapted on a support set to update the deployed model, while a query set in the future is utilized further to improve the quality of global knowledge across time periods. The support set $\mathcal{S}^T$ consists of data from the new and several recent time periods: $\mathcal{S}^T = \bigcup_{T-T_s<t\leq T} \mathcal{D}^t$, and the query set $\mathcal{Q}^T$ consists of data from several future time periods: $\mathcal{Q}^T = \bigcup_{T<t\leq T+T_q} \mathcal{D}^t$, where T_s and T_q are hyperparameters to control the length of time periods.

7.3 THE DESIGN OF ONLINESAFE

In this section, we present a novel framework OnlineSAFE to solve the online link prediction task on streaming temporal graphs, as shown in Figure 7.3. We first introduce the basic setup of the meta-learning objective in Section 7.3.1, and then detail the attentive temporal model to represent data in Section 7.3.2, followed by the introduction of temporal meta training in Section 7.3.3 and the edge reduction for acceleration during online phase in Section 7.3.4.

7.3.1 Meta-learning Setup and Learning Objective

Our innovation mainly lies in how to distill enduringly valuable knowledge from the existing data and seamlessly adapt it to newly gathered information for efficient adaptation and robust performance over time. To this end, we formulate the online link prediction task as a meta-learning problem. Each distinct task, denoted as $(\mathcal{S}^t,\ \mathcal{Q}^t)$, is designed to facilitate model updates using the support set $\mathcal{S}^t$, with the ultimate aim of optimizing performance on the future query set $\mathcal{Q}^t$. During the meta-training phase, we simulate a set of tasks by slicing time windows within the observed time range $(0, T)$ and construct the corresponding pairs of support and query sets, with the assumption that the support set is the freshly collected data, and the query set is the future data that awaits model deployment and evaluation.

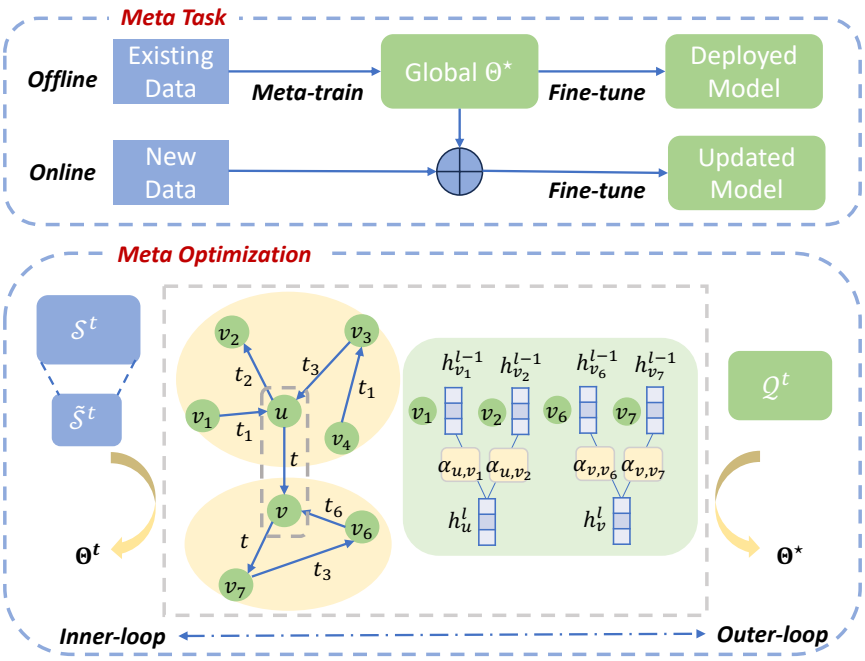


Figure 7.3: Framework overview.

Thus, we can mimic the real online updates by adapting the global knowledge, parameterized by Θ, on the support set via several fine-tuning steps (inner loop). And the global knowledge can be further improved by tuning the model on each query set (outer loop), as the quality of global knowledge can be quantitively measured by how well the model performs on future data. Such a bi-level optimization can be detailed as follows:

$$\begin{aligned} \Theta^\star &\longleftarrow \arg\min_{\Theta} \mathbb{E}_t \left[\mathcal{L}(f_{\Theta^t}, \mathcal{Q}^t)\right] \text{ (Outer Loop)}, \\ \text{where} \quad \Theta^t &= \Theta - \eta \frac{\partial \mathcal{L}(f_\Theta, \mathcal{S}^t)}{\partial \Theta} \text{ (Inner Loop)}, \end{aligned} \tag{7.1}$$

where $\mathcal{L}(f_\Theta, \cdot)$ denotes the predictive loss of model f_Θ on the respective support and query sets, Θ denotes the global knowledge across tasks, $\Theta^\star$ denotes the optimal global knowledge after the outer-loop optimization, and Θ^t denotes the updated model parameters at the t-th collection period. Towards the learning objective above, we introduce the detailed model f_Θ and the specific meta-training strategy in the next.

7.3.2 The Attentive Temporal Model

The attentive temporal model f_Θ parameterized by Θ is designed to measure the plausibility of each temporal edge, which represents each node u into a low-dimensional latent

space at each time t: $\mathbf{h}_u(t) \in \mathbb{R}^d$. On a streaming temporal graph, nodes $u \in \mathcal{E}$ evolve as they interact with different neighbors over time. Such temporally interacted nodes are defined as temporal neighbors. Therefore, we aim to model the temporal pattern of each node u by encoding the changes of temporal neighbors.

Towards this goal, f_Θ first samples temporal neighbors $\mathcal{N}_u(t) = \{(v_i, t_i) | (u, v_i, t_i) \text{ or } (v_i, u, t_i) \in \mathcal{D}^{T \leq t}\}$ from the temporal graph for each node $u \in \mathcal{E}$. $\mathcal{N}_u(t)$ consists of a set of the most recently interacted nodes at time t. Then f_Θ attentively aggregates information from the temporal neighbors. Specifically, given the temporal neighbor $\mathcal{N}_u(t)$, we represent the entity u as $\mathbf{h}_u(t)$ at time t:

$$\mathbf{h}_u^l(t) = \text{ReLU}\left(\sum_{(v_i,t_i)\in\mathcal{N}_u(t)} \alpha_{u,v_i}^l \left(\mathbf{h}_{v_i}^{l-1}(t_i)\mathbf{W}\right)\right), \tag{7.2}$$

$$\begin{aligned} \alpha_{u,v_i}^l &= \frac{\exp(q_{u,v_i}^l)}{\sum_{(v_k,t_k)\in\mathcal{N}_u(t)} \exp(q_{u,v_k}^l)}, \\ q_{u,v_k}^l &= \mathbf{a}\left(\mathbf{h}_u^{l-1} \| \mathbf{h}_{v_k}^{l-1} \| \kappa(t - t_k)\right), \end{aligned} \tag{7.3}$$

where q_{u,v_k}^l measures the pairwise importance by considering the node features of u and each v_k, and time feature, $\mathbf{a} \in \mathbb{R}^{3d}$ is the shared parameter in the attention mechanism. Following [144], we adopt random Fourier features as time encoding $\kappa(\Delta t)$ to reflect the time difference. Similarly, the edge feature, if available, can be concatenated together in the pairwise importance measurement q_{u,v_k}^l.

To measure the probability of each possible temporal edge, we utilize inner product [227] as the score function $p = \sigma((\mathbf{h}_u^t, \mathbf{h}_v^t))$, where $\sigma(\cdot)$ denotes the activation function *Sigmoid*. To optimize the parameter Θ for a task on either the support or query set, we minimize the loss for each temporal edge to train the model f_Θ. Taking the support set $\mathcal{S}^t$ as an example:

$$\mathcal{L}(f_\Theta, \mathcal{S}^t) = \mathbb{E}\left[-y_i \log(p_i) - (1 - y_i)\log(1 - p_i)\right], \tag{7.4}$$

where $y_i = 1$ if the edge $(u_i, v_i, t_i) \in \mathcal{S}^t$, and $y_i = 0$ otherwise.

7.3.3 Temporal Meta Training

The bi-level optimization procedure outlined in Eq. (7.1) emulates the iterative refinement process of the deployed model in the online setting. The inner loop optimization mirrors the strategy for swiftly adapting the model to recently acquired data by leveraging the extracted

knowledge Θ, which can be shared among tasks in different time periods. Meanwhile, the outer loop optimization is geared towards tuning the shared knowledge Θ across various tasks to the optimal one $\Theta^\star$, enhancing its capacity to encapsulate information and effectively acquire skills from the existing data. During the offline training phase, both the global knowledge Θ and the fine-tuning procedure can be learned via the nested meta-optimization procedure, which is iterated over time. During the online updating phase, the optimal knowledge $\Theta^\star$ can be further fine-tuned on the newly collected data by optimizing the inner loop.

Inner Loop Optimization. The online update of the deployed model is essentially estimating the posterior parameter distribution $p(\Theta^t|\mathcal{S}^t)$ given the newly collected data based on the prior estimation of the global knowledge $p(\Theta)$. As pointed out by [213], [214], a unique challenge arises to avoid the overfitting and catastrophic forgetting issues on the new data. Otherwise, informative knowledge presented in $p(\Theta)$ might be dominated and biased by the new observations. One might then attempt to use the Bayes rule to compute the posterior given the prior $p(\Theta)$ and the observations $\mathcal{S}^t$. However, it requires us to have already access to the likelihood of $\mathcal{S}^t$, which needs to be estimated first. To resolve this, we instead adopt the PAC-Bayes method [173], [191], which allows one to learn a parameter posterior that fits the observations without knowing the distribution of the observations. We propose the following theorem to relate real predictive loss in new time intervals with its empirical version:

Theorem 7.1 (PAC-Bayes Bound on New Data)**.** Let $p(\Theta)$ denote the prior distribution gained by the global knowledge across tasks, $q(\Theta^t)$ denote the empirical estimation of the posterior parameter distribution on new observation $p(\Theta^t|\mathcal{S}^t)$, $|\cdot|$ denote the set size. For any $\delta \in (0,1)$ and learned prior $p(\Theta)$, with probability at least $1-\delta$ over new data $\mathcal{S}^t$, the upper bound of the inner-loop predictive loss to be optimized on the new data is given by:

$$\mathcal{L}(f_\Theta, \mathcal{S}^t) \leq \hat{\mathcal{L}}(f_\Theta, \mathcal{S}^t) + \sqrt{\frac{\mathbb{KL}(q(\Theta^t)\|p(\Theta)) + \log\frac{|\mathcal{S}^t|}{\delta}}{2|\mathcal{S}^t|-1}}, \tag{7.5}$$

Proof. After the online deployment, the newly collected data $\mathcal{S}^t$ in the new time interval may follow different distribution and exhibit different patterns from those in the last time interval, i.e., $t-1$-th time interval, and such new distribution is unknown at advance. The online update of the deployed model is essentially estimating the posterior parameter distribution $p(\Theta^t|\mathcal{S}^t)$ given the newly collected data based on the prior estimation of the global knowledge $p(\Theta)$. One might then attempt to use the Bayes rule to compute the posterior given the prior $p(\Theta)$ and the observations $\mathcal{S}^t$. However, it requires us to have already access to the likelihood of $\mathcal{S}^t$, which needs to be estimated first. To resolve this, we instead adopt the

PAC-Bayes method [173], [191]. Formally, we view $p(\Theta)$ as prior parameter distribution and aim to learn the posterior parameter distribution $q(\Theta^t)$, in order to approxiamate $p(\Theta^t|\mathcal{S}^t)$ conditioning on $\mathcal{S}^t$. The unseen distribution of $\mathcal{S}^t$ prevents us from estimating $q(\Theta^t)$ by either using unbiased empirical loss or Bayes rules. Therefore, for convenience of discussion, we first define the difference between real predictive loss and its empirical estimation as follows:

$$\Delta\mathcal{L} = \mathcal{L}(f_\Theta, \mathcal{S}^t) - \hat{\mathcal{L}}(f_\Theta, \mathcal{S}^t). \tag{7.6}$$

We are interested at the relation of $\Delta\mathcal{L}$ and the distribution discrenpency between $p(\Theta)$ and $q(\Theta^t)$. Towards this goal, following [173], [191], we construct the following function:

$$f(\mathcal{S}^t) = 2(|\mathcal{S}^t| - 1)\mathbb{E}_{\Theta^t \sim q(\Theta^t)}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(q(\Theta^t)\|p(\Theta)). \tag{7.7}$$

Next, using Markov's inequality, we have:

$$p(f(\mathcal{S}^t) > \epsilon) = p(e^{f(\mathcal{S}^t)} > e^\epsilon) \leq \frac{\mathbb{E}_t\left[e^{f(\mathcal{S}^t)}\right]}{e^\epsilon}, \tag{7.8}$$

where $\mathbb{E}_t\left[e^{f(\mathcal{S}^t)}\right]$ denotes the expectation of $e^{f(\mathcal{S}^t)}$ w.r.t. new data collection period. To upper bound the expectation, we have the following inequality:

$$\begin{aligned} f(\mathcal{S}^t) &= 2(|\mathcal{S}^t| - 1)\mathbb{E}_{\Theta^t \sim q(\Theta^t)}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(q(\Theta^t)\|p(\Theta)) \\ &= \mathbb{E}_{\Theta^t \sim q(\Theta^t)}\left[\log\left(e^{2(|\mathcal{S}^t|-1)(\Delta\mathcal{L})^2}\frac{p(\Theta)}{q(\Theta^t)}\right)\right] \\ &\leq \log\left(\mathbb{E}_{\Theta^t \sim q(\Theta^t)}\left[e^{2(|\mathcal{S}^t|-1)(\Delta\mathcal{L})^2}\frac{p(\Theta)}{q(\Theta^t)}\right]\right) \\ &= \log\left(\mathbb{E}_{\Theta^t \sim p(\Theta)}\left[e^{2(|\mathcal{S}^t|-1)(\Delta\mathcal{L})^2}\right]\right), \end{aligned} \tag{7.9}$$

where Jensen's inequality is utilized to derive the inequality. Therefore, we have

$$\begin{aligned} \mathbb{E}_t\left[e^{f(\mathcal{S}^t)}\right] &\leq \mathbb{E}_t\mathbb{E}_{\Theta^t \sim p(\Theta)}\left[e^{2(|\mathcal{S}^t|-1)(\Delta\mathcal{L})^2}\right] \\ &= \mathbb{E}_{\Theta^t \sim p(\Theta)}\mathbb{E}_t\left[e^{2(|\mathcal{S}^t|-1)(\Delta\mathcal{L})^2}\right], \end{aligned} \tag{7.10}$$

the order of expectations is swapped as $p(\Theta)$ is independent to $\mathcal{S}^t$. Next, based on Hoeffding's

inequality, we have:

$$p(\Delta\mathcal{L} > \epsilon) \leq e^{-2|\mathcal{S}^t|\epsilon^2}, \tag{7.11}$$

, and we can further derive the following inequality:

$$\mathbb{E}_t\left[e^{f(\mathcal{S}^t)}\right] \leq \mathbb{E}_{\Theta^t\sim p(\Theta)}\mathbb{E}_t\left[e^{2(|\mathcal{S}^t|-1)(\Delta\mathcal{L})^2}\right] \leq |\mathcal{S}^t|. \tag{7.12}$$

Combining Eq. (7.12) and Eq. (7.8), we get:

$$p(f(\mathcal{S}^t) > \epsilon) \leq \frac{|\mathcal{S}^t|}{e^\epsilon} = \delta, \tag{7.13}$$

where $\delta = |\mathcal{S}^t|/e^\epsilon$. Therefore, with probability of at least $1-\delta$, we have that for all Θ^t:

$$\begin{aligned} f(\mathcal{S}^t) &= 2(|\mathcal{S}^t|-1)\mathbb{E}_{\Theta^t\sim q(\Theta^t)}\left[(\Delta\mathcal{L})^2\right] - \mathbb{KL}(q(\Theta^t)\|p(\Theta)) \\ &\leq \frac{\log|\mathcal{S}^t|}{\delta}. \end{aligned} \tag{7.14}$$

Further, by utilizing Jensen's inequality again, we have:

$$\begin{aligned} \left(\mathbb{E}_{\Theta^t\sim q(\Theta^t)}\left[(\Delta\mathcal{L})\right]\right)^2 &\leq \mathbb{E}_{\Theta^t\sim q(\Theta^t)}\left[(\Delta\mathcal{L})^2\right] \\ &\leq \frac{\mathbb{KL}(q(\Theta^t)\|p(\Theta)) + \frac{\log|\mathcal{S}^t|}{\delta}}{2(|\mathcal{S}^t|-1}. \end{aligned} \tag{7.15}$$

Subsituting definition of $\Delta\mathcal{L}$ in Eq. 7.15, we proof the PAC-Bayes Bound on the newly collected data with unknow distribution:

$$\mathcal{L}(f_\Theta, \mathcal{S}^t) \leq \hat{\mathcal{L}}(f_\Theta, \mathcal{S}^t) + \sqrt{\frac{\mathbb{KL}(q(\Theta^t)\|p(\Theta)) + \log\frac{|\mathcal{S}^t|}{\delta}}{2|\mathcal{S}^t|-1}}. \tag{7.16}$$

QED.

Remark. The Eq. (7.5) can be interpreted as a combination of empirical loss on the new data in the new time interval and a regularizer of parameter distribution with the global parameter distribution in the form of KL-divergence. The regularizer along the time domain can guarantee that the inner-loop fine-tuning is trained only on new data but without overfitting in specific time intervals. Thus, we can improve the generalization ability of our online fine-tuning procedure.

Outer Loop Optimization. The quality of the extracted knowledge, *i.e.,* Θ, is quantitively measured by how well the updated model (optimized on the support set) performs on the future data (the corresponding query set). Concretely, during the offline training phase,

Table 7.1: The dataset statistics.

Dataset	#Interaction	#User	#Item	Density
Wiki	157,474	8,227	1,000	0.019141
Reddit	672,447	10,000	984	0.068338
Twitter	134,291	8,780	1,333	0.011474
Yelp	1,266,728	63,228	59,375	0.000337

we optimize the predictive loss $\mathcal{L}(f_{\Theta^t}, \mathcal{Q}^t)$ w.r.t. Θ by applying the updated model (with parameter Θ^t) on the query set $\mathcal{Q}^t$, as shown in Eq. (7.1).

7.3.4 Edge Reduction for Online Update

We analyze the time complexity of the inner loop optimization, which is used for online model updating. Let b denote the number of temporal neighbors for each node, l denote the number of attention layers, N denote the negative sampling factor, and the overall complexity is $\mathcal{O}\left(lb^2N|\mathcal{S}^t|\right)$, which grows linearly with $|\mathcal{S}^t|$. To further expedite the online update process, we explore methods to diminish the size of $|\mathcal{S}^t|$ using the edge reduction technique, all while maintaining its efficacy. Given that the PAC-Bayes bound within the online objective governs the alignment of the updated model parameters with the global ones based on historical data, the updated model parameters already incorporate the information present in the historical data. This leads us to speculate that removing edges recurrently present in historical time steps might not result in the loss of pertinent information. We propose a straightforward approach that involves bypassing edges where the two nodes are already 1- or 2-hop neighbors in recent time steps. This strategy is based on the premise that such edges might not introduce any supplementary information, as the existing historical data and PAC-Bayes bound have already covered their contribution. Let $\tilde{\mathcal{S}}^t$ denote the new support set after the edge reduction, the overall complexity becomes $\mathcal{O}\left(lb^2N|\tilde{\mathcal{S}}^t|\right)$.

7.4 THE EVALUATION OF ONLINESAFE

In this section, we introduce the evaluation results of OnlineSAFE on four public datasets.

7.4.1 Datasets

We evaluate the proposed OnlineSAFE on four real-world datasets collected from Wiki, Reddit, Twitter, and Yelp, where Table 7.1 reports the important statistics of the four graphs.

- **Wiki** [228]: The dataset is a public collection of edits made by users who contributed at least 5 edits to the 1,000 most edited Wikipedia pages within a one-month period. The nodes represent human editors and wiki pages and the links represent timestamped historical edits.

- **Reddit** [228]: The public Reddit post dataset comprises one month of user posts from the 1,000 most active subreddits and the 10,000 most active users. Its nodes include users and posts, and the temporal edges are the timestamped posting requests.

- **Twitter**: The dataset collected from Twitter constitutes user-hashtag interactions, whose nodes include users and hashtags, and whose links represent who-post-what interaction records. To maintain data quality, we excluded the top 10 users and top 20 hashtags with abnormal activation levels and users/hashtags with fewer than 15 occurrences, considered invalid for analysis.

- **Yelp** [15]: The dataset is collected from the Yelp Challenge 2018 and consists of user-business interactions with ratings of 4 and 5 points after 2010, where nodes includes users and business and edges describe reviewing actions. Inactive users with fewer than 8 interactions have been excluded.

For each dataset, we sort all interactions in chronological order and split them into 40 timesteps, each containing an equal number of interactions. We further split 40 periods into 30/2/8 for *offline training/online validating/online testing* phases. To be concrete:

- **Offline training phase** is utilized to training both baselines as well as OnlineSAFE before deployment. The data in this phase are assumed to be given at once;

- **Online validating phase** is the phase where we utilize the online data to first validate the deployed models and then to update the models. The average validation performance is selected as indicator to choose important hyperparameters which are utilized for online testing phase, *e.g.,* learning rate, batch size, number of layers, etc.

- **Online testing phase** is the phase where we utilize the online data to first test the deployed models and then to update the models. The average testing performance on 8 periods are reported.

[15] https://www.yelp.com/dataset/

7.4.2 Experimental Setup

Baselines We describe the 17 state-of-the-art baselines from three related areas in detail:

- Static graph learning methods:
  - **MF** [229] has two strategies to collaboratively learning online. Full-retrain updates the model with all previous data, while fine-tuning uses only the newest period;
  - **GAE** [227] uses GCN to learn the latent representations for undirected graphs based on the auto-encoder;
  - **VGAE** [227] utilize the same framework as GAE and adds the probabilistic variants;
  - **GAT** [97] utilizes masked self-attentional layers to aggregate neighbors, improving inductive learning performance;
  - **GraphSAGE** [166] ingenuiously generates neighborhood embeddings to perform inductive tasks;
  - **GIN** [230] proposed a simple neural framework to learn simple graphs, powerfully represents their structures;
  - **LightGCN** [131] consists only of the neighbor aggregation part of GCN to improve the performance;
  - **GPS** [231] restricts positional structural encoding computing, reducing the computational complexity to linear;
- Dynamic graph learning methods:
  - **GRU4Rec** [106] applies RNN, which models the set of interactions for each period and recommends similar items;
  - **EGCN-H** [232] uses GNN to sense nodes' relationship for each snapshot and employs RNN for link prediction;
  - **EGCN-O** [232] also utilizes EGCN-H's framework but updates the model through an LSTM without input node embeddings;
  - **VGRNN** [66] incorporates a VGAE to model each snapshot in graph recurrent neural network for dynamic link prediction;
  - **Euler** [233] proposes a framework consisting of stacked model-agnostic GNN and RNN to detect the emerging links;
  - **DySAT** [92] joins a stacked temporal and spatial self-attention network, showing that the framework is robust and efficient;

– **TGAT** [144] proposed a temporal graph attention layer to induce the topological features and interactions;

- Online graph learning methods:
  – **SPMF** [218] skillfully sampled for retraining to represent long-term preference and rank the items by an optimization framework;
  – **SML** [213] introduces a meta-learning approach that captures the long and short-term embeddings, saving time and memory;
  – **IGC** [219] re-activates the previous nodes, and updates only the new-neighbor-related parameters to speed up the retrain speed;

Evaluation Protocol and Metrics. We evaluate online link prediction tasks in a retrieval setting. For each link collected during the online phase, given the observed user nodes, we compare the predicted top-K ranking list of missing items with the ground-truth item in each testing time step. We adopt two widely-used evaluation protocols: *Recall@K* and *NDCG@K*, where $K = \{5, 10, 20\}$ by default.

Implementation We utilize the package provided by SML [16] and IGC [17] to implement *MF*, *SPMF*, *SML*, and *IGC*. We use the first ten snapshots to generate the pre-trained model and the following twenty data periods to train offline. We compare retraining all the previous data and fine-tuning the new snapshot for MF method when testing online in the last ten periods, and preserve the strategy with better performance. As for SPMF, SML, and IGC, we perform online update on the last 10 snapshots. For GRU4Rec [18], we follow the similar procedure. To implement full retraining strategy, we concatenate all action sequences from all time periods together for model training. And we only consider the sequence happened within the last time period to train model, as the fine-tuning implementation. We tuned the training epochs in $\{10, 20, 50, 100, 200\}$, learning rate in $\{1, 0.5, 0.1, 0.05, 0.01, 5e-3, 1e-3, ..., 1e-8\}$. Particularly for SML, we tuned transfer learning rate in $\{1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6\}$.

For other baselines, we use the first 30 periods for offline training. When evaluating each online period, all baselines are trained using both fine-tuning strategy on the last snapshot and full-retrain strategy on all the previous ones. Specifically for static baselines, we merge all historical snapshots as one graph via *or* operation to implement the full retraining strategy. Notably, several baselines easily encounter out-of-memory (OOM) issue on large graphs,

[16] https://github.com/zyang1580/SML
[17] https://github.com/Dingseewhole/CI_LightGCN_master/
[18] https://github.com/hungpthanh/GRU4REC-pytorch

e.g., DySAT, EGCN, Euler, etc. We use the following ways to try to deal with the OOM issue: 1. we adopt edge sampling technique instead of full graph training to reduce the training samples and graph size; 2. we split the snapshots into groups and only consider the temporal dependency within each group to save GPU memory. Each group contains $K = \{3, 10\}$ consecutive snapshots. We tuned the hyper-parameters as follows: training epochs in $\{10, 20, 50, 100, 200\}$, learning rate in $\{10, 1, 1e-1, 1e-2, 1e-3, 1e-4\}$.

To construct the task set for meta-learning formulation, we set temporal edges in $K_s = 3$ consecutive historical periods as support set, and ones in $K_q = 3$ consecutive future periods as query set. Such a choice enable us to i) consider sufficient historical information to update model online; ii) utilize relatively long-term performance as instant feedback for learning the global knowledge in the outer-loop optimization; iii) maintain acceptable efficiency. During evaluation, we tune hyperparameters based on *Recall@20* on online validating phase, and report the average performance on the remaining periods on the online testing phase. Next, we report the choices of hyperparameters. For model training, we utilize Adam optimizer, and set maximum number of epochs as 200 for offline training. We set batch size as 32, the dimension of all embeddings as 128, and dropout rate as 0.5. For the sake of efficiency, we set the neighbor budget b of temporal neighbor sampler as 16, and employ 2 neighborhood aggregation layers in temporal encoder. We perform 20 steps of gradient descent for inner loop optimization. We mainly tune inner/outer loop learning rate η and β in range $\{0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$. For Wiki, Reddit and Twitter, we set $\eta = \beta = 0.0001$. For Yelp, we set $\eta = 0.0005$, and $\beta = 0.0001$. We also compare the margin loss with the cross-entropy loss utilzied in Eq (7.4), with tuning value in range $\{0.3, 0.4, 0.5, 0.6, 0.7\}$. Finally we found training with the cross-entropy loss still achieves better performance, although training with the margin loss is faster.

7.4.3 Main Results

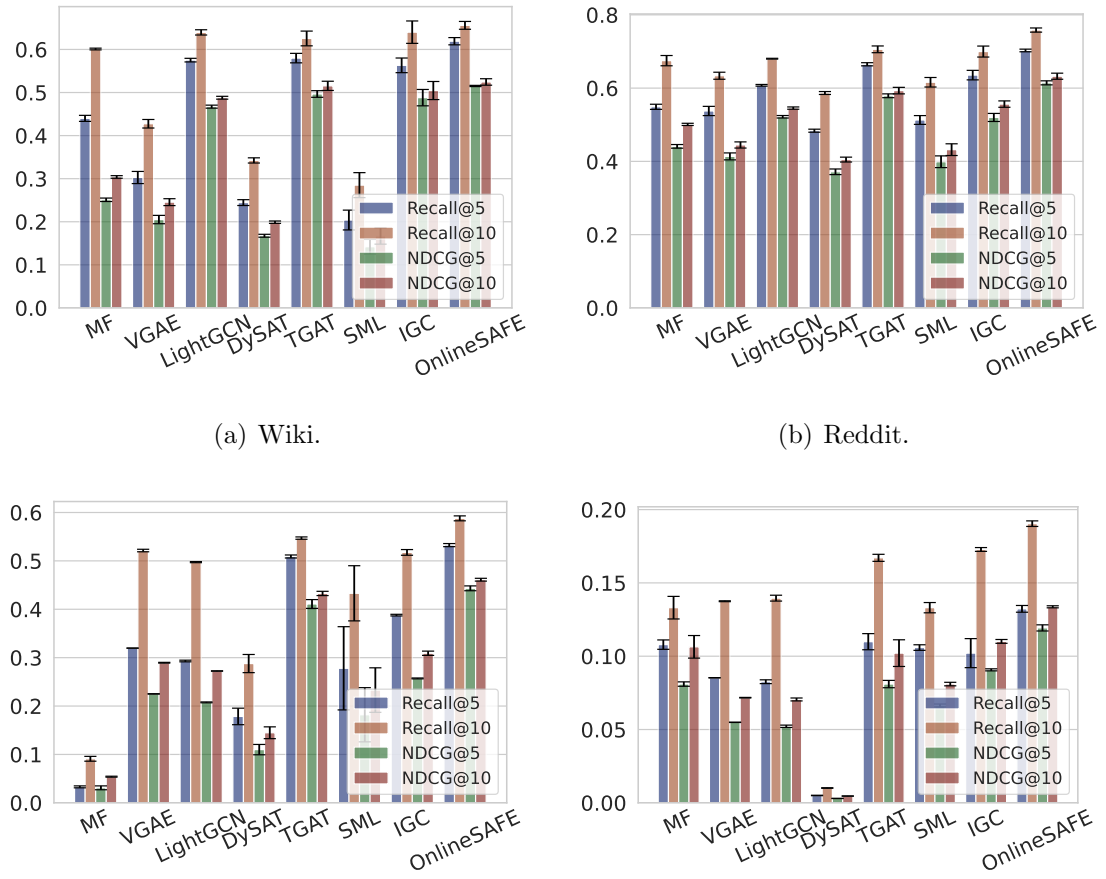
In this section, we report the complete experimental results of the online link prediction task in Table 7.2 and Figure 7.4, measured by $Recall@\{5, 10, 20\}$ and $NDCG@\{5, 10, 20\}$. Average results on 5 independent runs with different random seeds are reported. $*$ indicates the statistically significant improvements over the best baseline, with p-value smaller than 0.001. For baselines tailored for offline training scenarios, we utilize both retraining and fine-tuning techniques, and retaining the approach with better performance. Detailed options utilized for each baseline are reported. We also report the detailed standard deviation for all models. OnlineSAFE can beat all state-of-the-art baselines on all datasets, and we observe that OnlineSAFE and most baselines are stable with small standard deviations.

Table 7.2: The overall performance of baseline models and OnlineSAFE. Average results of 5 independent runs are reported. For baselines tailored for offline training scenarios, we utilize both retraining and fine-tuning techniques, and retaining the approach with better performance.

Dataset	Wiki		Reddit		Twitter		Yelp	
Performance	Recall@20	NDCG@20	NDCG@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
MF	0.662 ± 0.013	0.319 ± 0.005	0.750 ± 0.017	0.522 ± 0.004	0.172 ± 0.007	0.099 ± 0.002	0.183 ± 0.002	0.125 ± 0.002
GAE	0.481± 0.018	0.221 ± 0.012	0.708 ± 0.013	0.481 ± 0.017	0.600 ± 0.000	0.307 ± 0.001	0.213 ± 0.001	0.091 ± 0.000
VGAE	0.544 ± 0.010	0.275 ± 0.008	0.717 ± 0.008	0.466 ± 0.008	0.609 ± 0.001	0.312 ± 0.001	0.212 ± 0.001	0.090 ± 0.001
GAT	0.540 ± 0.007	0.328 ± 0.012	0.706 ± 0.007	0.490 ± 0.007	0.340 ± 0.009	0.154 ± 0.023	0.091 ± 0.002	0.038 ± 0.001
GraphSAGE	0.660 ± 0.014	0.423 ± 0.011	0.745 ± 0.002	0.513 ± 0.007	0.437 ± 0.046	0.208 ± 0.022	0.108 ± 0.001	0.047 ± 0.001
GIN	0.409 ± 0.016	0.193 ± 0.006	0.568 ± 0.055	0.345 ± 0.050	0.519 ± 0.009	0.287 ± 0.008	0.220 ± 0.001	0.093 ± 0.001
LightGCN	0.698 ± 0.003	0.503 ± 0.003	0.741 ± 0.001	0.561 ± 0.003	0.568 ± 0.001	0.291 ± 0.001	0.224 ± 0.003	0.092 ± 0.001
GPS	0.601 ± 0.027	0.358 ± 0.019	0.737 ± 0.016	0.526 ± 0.019	0.491 ± 0.056	0.260 ± 0.030	OOM	OOM
GRU4Rec	0.080 ± 0.001	0.046 ± 0.001	0.048 ± 0.001	0.037 ± 0.000	0.056 ± 0.000	0.030 ± 0.000	0.026 ± 0.000	0.010 ± 0.000
EGCN-H/O	0.089 ± 0.002	0.039 ± 0.001	0.471 ± 0.007	0.285 ± 0.004	0.250 ± 0.010	0.124 ± 0.007	OOM	OOM
VGRNN	0.048 ± 0.030	0.025 ± 0.016	0.389 ± 0.073	0.193± 0.036	0.389 ± 0.073	0.193 ± 0.036	0.165 ± 0.014	0.071 ± 0.006
Euler	0.040 ± 0.010	0.018 ± 0.005	0.484 ± 0.032	0.242 ± 0.017	0.600 ± 0.003	0.334 ± 0.002	0.070 ± 0.021	0.028 ± 0.010
DySAT	0.442 ± 0.010	0.224 ± 0.002	0.668 ± 0.002	0.426 ± 0.007	0.410 ± 0.011	0.176 ± 0.011	0.020 ± 0.000	0.007 ± 0.000
TGAT	0.664 ± 0.010	0.529 ± 0.008	0.744 ± 0.011	0.618 ± 0.017	0.604 ± 0.010	0.440 ± 0.003	0.242 ± 0.002	0.136 ± 0.002
SPMF	0.585 ± 0.007	0.358 ± 0.006	0.741 ± 0.001	0.507 ± 0.002	0.022 ± 0.003	0.007 ± 0.001	0.166 ± 0.001	0.100 ± 0.001
SML	0.374 ± 0.023	0.190 ± 0.017	0.704 ± 0.013	0.455 ± 0.016	0.500 ± 0.071	0.250 ± 0.049	0.177 ± 0.010	0.111 ± 0.003
IGC	0.685 ± 0.014	0.526 ± 0.011	0.754 ± 0.011	0.589 ± 0.010	0.577 ± 0.002	0.335 ± 0.005	0.248 ± 0.008	0.132 ± 0.006
OnlineSAFE	**0.716 ± 0.017**	**0.575 ± 0.005**	**0.807 ± 0.006**	**0.645 ± 0.010**	**0.644 ± 0.002**	**0.475 ± 0.002**	**0.272 ± 0.005**	**0.157 ± 0.002**
Gains (%)	*2.6*	*8.7*	*7.0*	*4.4*	*5.9*	*8.0*	*9.8*	*15.4*

Insterestingly, we found most of the better stategy of baselines on Wiki and Reddit are full-retraining, while those on Twitter and Yelp are fine-tuning. This might indicates that Twitter and Yelp exhibit more obvious temporal evolution patterns than Wiki and Reddit, which is consistent with our observation in Figure 7.2. Therefore, fine-tuning stategy might be better than full-retraining at capturing new patterns in the newly collected data, while full-retraining donot suffer from overfitting issues on new data, thus achieves better performance than fine-tuning on Wiki and Reddit. OnlineSAFE with temporal meta training strategy consistently outperforms all baseline models, exhibiting an average relative improvement of 8.8%. Comparing detailed performance in Figure 7.2 and average performance in Table 7.2, LightGCN and TAGE with respective fine-tuning and full-retraining produce better performance than the counterparts without any model tuning on new data, showing the necessity of the online model update.

Intriguingly, certain prominent dynamic models, like GRU4Rec, EGCN, VGRNN, and Euler, demonstrate unexpectedly suboptimal results, even underperforming the static baseline models. We hypothesize that it is caused by too long a graph sequence, where these dynamic models struggle to effectively extract valuable knowledge from the historical data that are still useful on new data patterns. While TGAT and IGC (an online algorithm based on LightGCN) generally outperform other baselines, their results still fall short of our approach. This deficiency could be attributed to the absence of a tailored online updating strategy specifically for the temporal module. The superior performance of OnlineSAFE demonstrates the efficacy of the proposed temporal meta-training strategy for online link prediction.



(a) Wiki.

(b) Reddit.

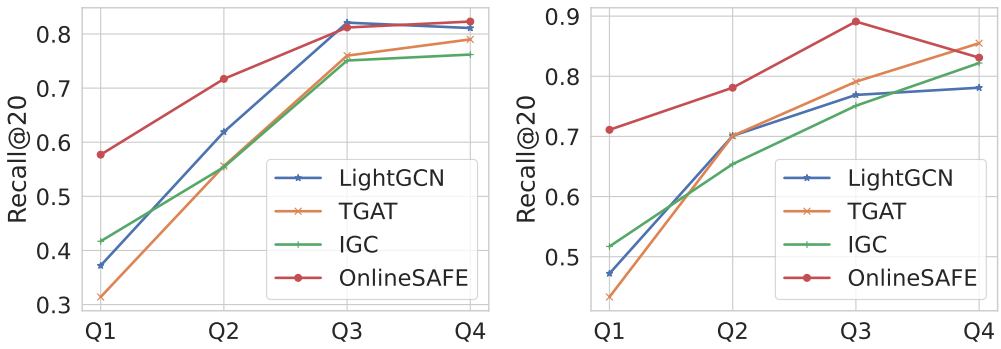
(c) Twitter.

(d) Yelp.

Figure 7.4: Detailed Recall@K and NDCG@K when $K = \{5, 10\}$. The average results of 5 independent runs are reported. OnlineSAFE achieves the best performance, with 8.8% relative gains over the second-best results on average.

7.4.4 Performance of Diverse Node Groups

The key indicator to evaluate the online algorithm is how well the model performs on newly emerging nodes with a few edges, as new nodes continuously join the graph in the real world. We divide nodes into quartiles by their degree levels, which is the number of participating edges per node. Figure 7.5 shows the performance distribution for each node group, from Q1 (lowest degree level) to Q4 (highest degree level), compared with three strong baselines. The major improvement of OnlineSAFE comes from the nodes with few links (Q1, Q2), as it can borrow useful information from the high-resource nodes to the low-resource nodes via the extraction of global knowledge.



(a) Wiki. (b) Reddit.

Figure 7.5: Performance across node quartiles on degree level. (Q1: lowest, Q4: highest). OnlineSAFE has higher gains for nodes with sparse connections (quartiles Q1-Q3).

Table 7.3: Ablation studies measured by Recall@20 and NDCG@20. Online update time is also reported.

Dataset	**Wiki**			**Reddit**		
Metrics	**Recall**	**NDCG**	**Time (s)**	**Recall**	**NDCG**	**Time (s)**
Retraining	0.669	0.540	*321.9*	0.744	0.628	*1395.8*
Fine-tuning	0.657	0.531	*12.7*	0.730	0.611	*40.5*
OnlineSAFE	0.716	0.575	*11.1*	0.807	0.645	*36.5*
w/o PAC-Bayes	0.701	0.569	*9.4*	0.793	0.623	*33.4*
w/o Edge Reduction	**0.722**	**0.589**	*22.3*	**0.820**	**0.660**	*70.3*
w/ Random Reduction	0.690	0.557	*9.9*	0.770	0.629	*32.9*

7.4.5 Ablation Study

We conduct the following ablation studies to evaluate performance improvements brought by the temporal meta-training strategy 1) **Retraining** directly retrains our attentive temporal model on all data; 2) **Fine-tuning** fine-tunes our attentive temporal model only on new data; 3) **OnlineSAFE w/o PAC-Bayes Bound** removes the PAC-Bayes bound in the inner loop optimization; 4) **OnlineSAFE w/o Edge Reduction** update the global model on all edges in the new data without skipping any sample; 5) **OnlineSAFE w/ Random Reduction** reduce the same amount of training samples during the online phase by random selection.

Table 7.3 shows the evaluation results. Both retraining and fine-tuning approaches for the attentive temporal model fail to outperform OnlineSAFE. The retraining strategy might be dominated by the existing data, leading to an excessive focus on historical patterns and potentially overlooking new patterns. Conversely, the fine-tuning strategy can easily result in overfitting on the new data. Upon excluding the PAC-Bayes bound from the inner loop objective, we observe a decline in performance, underscoring the bound's role in fortifying the stability and generalizability of the online updating procedure. We also delve into the impact

of the simple edge reduction heuristic. While OnlineSAFE without this heuristic yields a marginal performance boost, it prolongs online training time by up to 2× times. Substituting the heuristic with a random strategy significantly compromises performance. These outcomes underscore the value of our straightforward yet effective edge reduction algorithm, which accelerates the online update process without inflicting significant performance degradation.

7.4.6 Efficiency Analysis

In Table 3.4, we also provide the online updating times for each variant, quantifying the time taken per online epoch in seconds. As anticipated, the direct retraining strategy necessitates significantly more time. Notably, OnlineSAFE also accomplishes swifter online updates compared to simple fine-tuning. This is due to the edge reduction algorithm employed by OnlineSAFE, which effectively reduces the volume of training samples during the online phase in contrast to the standard fine-tuning approach. The efficacy of the edge reduction heuristic is further highlighted when contrasted with the variant that omits it. This heuristic effectively halves the online running time without introducing a notable drop in performance. This is because the omitted edges fail to contribute substantially new information for model training, and their contribution is already covered in the global knowledge regulated by the PAC-Bayes bound.

7.5 RELATED WORK

7.5.1 Learning on Streaming Temporal Graphs

Learning on temporal graphs and predicting future links have been attracting numerous research interests in the community [92], [106], [108], [144], [212], [232], because of its profound impacts on a wide range of applications. The initial attempts explore Hawkes process [234], temporal random walks [235], [236], recurrent neural networks (RNN) [108], [169], among others, to encode the temporal information in the node representations. Recently, deep learning models based on graph neural networks (GNNs) [60], [227] have achieved the state-of-the-art performance, which is divided into two categories: *discrete methods* that organize the temporal graphs as snapshot sequences and utilize sequential modeling including RNNs (*e.g.,* EGCN [232], Euler [233], VGRNN [66]) and transformers (*e.g.,* DySAT [92]) to learn the evolution of node representations, *continuous methods* that learn temporal representations for nodes based on temporal neighbor sampling and temporal encoding (*e.g.,* TGAT [144],

TGN [212]). However, despite the numerous existing efforts, how to effectively and efficiently update temporal graph models on the streaming setting still remains relatively unexplored.

Continual Graph Learning (CGL) field mainly studies how to learn new patterns incrementally on evolving graphs [214], where typical techniques can be divided into regularization-based methods [237], [238], replay-based methods [224], [239], and architecture-based methods [220], [225]. While most of the efforts focus primarily on node classification tasks, some works have already studied link prediction tasks in the streaming setting. SPMF [218] skillfully sampled for retraining to represent long-term preference and rank the items by an optimization framework. SML [213] introduces a meta-learning approach that captures the long and short-term embeddings, saving time and memory. IGC [219] re-activates the previous nodes and updates only the new-neighbor-related parameters to speed up the retrain speed. However, these methods primarily focus on updating shallow and static models, and their applicability to addressing the challenges of modeling temporal graphs remains limited. In this paper, we aim to design a temporal graph learning method that enables effective and efficient online updating to facilitate future link prediction performance in a streaming manner.

7.5.2 Meta-learning

Given a set of tasks, meta-learning aims to learn general knowledge that is shared across all tasks and can be efficiently adapted to new tasks [226]. In this paper, we formulate the online learning task as a meta-learning problem and utilize model-agnostic meta-learning (MAML) [164] to address the challenges. MAML learns the general knowledge as the model initialization during the meta-training phase via the alternative bi-level optimization with an inner loop on *support set* and an outer loop on *query set*. During the meta-testing phase, MAML fine-tunes the parameters with a small number of training instances and thus adapt to a new learning task efficiently. Recently, meta-learning was integrated with graph neural network models for few-shot predictions on graphs, *e.g.,* Meta-Graph [172], G-Meta [171], and MetaDyGNN [192]. However, most works are designed for few-shot learning on static graphs. How to aggregate neighbors considering time factors and optimize for long-term performance in the online setting are unsolved.

7.6 SUMMARY

In this chapter, we studied a realistic online link prediction problem on streaming temporal graphs, which aims to effectively and efficiently update the deployed model on the newly

collected graph data. To this end, we proposed a novel temporal meta-training framework OnlineSAFE. It meta-learns the global knowledge of sampling and aggregating temporal neighbors, which can be adapted quickly to new data for future prediction via fine-tuning steps. Such bi-level optimization is nested and alternated during the offline training to mimic the online scenario. We further theoretically analyzed and utilized a PAC-Bayes bound to stabilize and generalize the online updating procedure. We empirically validated the effectiveness of OnlineSAFE on four real-world temporal knowledge graphs, on which the proposed framework significantly outperforms an extensive set of SOTA baselines.

Chapter 8: CONCLUSIONS AND FUTURE WORK

In this dissertation, we have explored how to design novel temporal graph learning techniques to build robust, effective, and efficient learning pipelines to facilitate predictive analytics in changing, partially observed, and unreliable graphs.

- From the **robustness** perspective, our approach was guided by the insight that false negative and false positive issues could be mitigated through a self-adaptive mechanism. To address this, we introduced a variational framework that treats the underlying correctness of facts as latent variables. This framework enables the simultaneous updating of model parameters and the estimation of posterior likelihood for determining the veracity of collected and uncollected facts. Throughout the training process, we put forth a label posterior-aware encoder and implemented a self-training strategy to further mitigate false positive and false negative issues. Our investigation revealed the critical role that label posterior estimation plays in advancing knowledge graph reasoning in real-world scenarios. Notably, we established that this estimation can be achieved by optimizing an alternative objective without incurring additional computational costs.

- From the **efficacy** perspective, as we grapple with the increasing diversity of data and escalating task complexity, our research delved into various mechanisms aimed at improving the modeling, representation, and learning of a wide spectrum of graph types. This spectrum encompasses dynamic bipartite graphs, dynamic multi-relational graphs, and dynamic low-resource graphs. In our investigations, we examined the role of external stimuli and the influence of online social interactions in shaping graph dynamics. We introduced a range of temporal models to capture crucial temporal information, and we explored the impact and utilization of external knowledge sources to enrich the modeling of data-scarce graphs. Our methods have exhibited effectiveness across a broad array of applications, including information diffusion prediction, recommendation systems, and temporal knowledge graph reasoning.

- From the **efficiency** perspective, we have effectively illustrated the viability of a unified design philosophy. This philosophy involves representing global knowledge as model parameters, which can be readily adapted to new data distributions through efficient fine-tuning. This adaptability has the potential to enhance both label-efficiency, facilitating model adaptation for emerging entities, and resource-efficiency, enabling swift model updates with newly collected data to maintain performance levels. To implement this concept, we introduced MetaTKGR in Euclidean space and MetaHKG

in hyperbolic space, enhancing the model's capacity for few-shot temporal reasoning. We also proposed OnlineSAFE to facilitate rapid model updates in streaming settings.

Looking forward to the future, we will continue to study the following directions in achieving build robust, effective, and efficient learning pipelines.

During the graph construction stage, addressing the robustness challenge opens up intriguing avenues for future research opportunities. One promising direction is the development of explainable data cleaning techniques that employ a differentiable approach to learn inference rules. This can be achieved through end-to-end optimization, allowing the model to not only identify and correct false negative and false positive issues but also provide transparent and interpretable explanations for its cleaning decisions. Such an approach would enhance the model's trustworthiness and its capacity to interact with domain experts, ultimately contributing to a more robust graph learning pipeline. Furthermore, the exploration of techniques for automatically generating and refining these inference rules presents an interesting research opportunity. Leveraging external clues or knowledge to guide the rule generation process could lead to more accurate and context-aware data cleaning methods. By extending this line of research, we can pave the way for enhanced graph learning in noisy and dynamic environments, which is of paramount importance in various real-world applications, from social networks to recommendation systems and beyond.

During the offline training stage, first, addressing the issue of distributed training for large graphs, especially in scenarios where temporal edges cannot be readily transferred due to privacy or throughput constraints, is an ongoing challenge. Future research could explore novel techniques for training graph models across distributed systems while respecting privacy constraints, developing efficient edge transfer mechanisms, or employing advanced federated learning approaches to ensure the seamless and secure adaptation of models in privacy-sensitive environments. Moreover, the dynamic nature of graphs is not limited to the conventional temporal structures. New types of graphs, such as heterophily graphs, emerge in diverse application scenarios. Investigating how to effectively model and learn these unique graph types presents a promising avenue for research. Understanding the dynamics and inherent characteristics of heterophily graphs and devising tailored graph learning techniques can yield valuable insights and solutions for various domains, including social networks and recommendation systems. Additionally, the realm of graph learning is continually expanding to encompass a wide range of applications and tasks. Future research opportunities involve exploring how to leverage temporal information for graph generation, enabling the creation of realistic and dynamic graphs that mirror the underlying dynamics of various domains. This research area could lead to advancements in generating graphs for simulations, creating

dynamic knowledge graphs, or constructing predictive models for evolving network structures.

During the online deployment stage, first, in terms of label-efficiency, there is room for substantial growth in research directed at enhancing model adaptation to new relations, new classes, and new domains. Developing innovative strategies for enabling models to learn from a limited amount of labeled data while maintaining high predictive accuracy is a pivotal research direction. This includes exploring techniques for transfer learning, domain adaptation, few-shot learning, and zero-shot learning within the graph learning context. Moreover, exploring unique property and combinationary benefits of various geometry for representing newly emerging entities is promising. These advancements can pave the way for more versatile and agile graph models that adapt to the ever-evolving nature of real-world data. Resource-efficiency is another critical facet of graph learning research. As graphs grow in size and complexity, it becomes imperative to minimize the computational demands of graph learning models. This encompasses several dimensions, including reducing the required time, memory, and energy for training and inference. Future research in this area can involve the design of more efficient layers, such as attention mechanisms that can operate effectively on large neighbor sets, leading to faster and more scalable graph models. Additionally, techniques like improved deduplication methods, model distillation, and more efficient memory access mechanisms can contribute to more resource-efficient graph learning. These innovations not only make graph learning more accessible but also align with the broader goals of sustainability and eco-friendliness in computing.

REFERENCES

[1] R. Wang, B. Li, Y. Lu, *et al.*, "Noisy positive-unlabeled learning with self-training for speculative knowledge graph reasoning," in *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 2440–2457.

[2] R. Wang, Z. Huang, S. Liu, *et al.*, "Dydiff-vae: A dynamic variational framework for information diffusion prediction," in *SIGIR'21*, 2021.

[3] R. Wang, Z. Li, D. Zhang, *et al.*, "Rete: Retrieval-enhanced temporal event forecasting on unified query product evolutionary graph," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22, Virtual Event, Lyon, France, 2022, pp. 462–472.

[4] R. Wang, Z. Li, J. Yang, *et al.*, "Mutually-paced knowledge distillation for cross-lingual temporal knowledge graph reasoning," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23, Austin, TX, USA: Association for Computing Machinery, 2023, pp. 2621–2632, ISBN: 9781450394161.

[5] R. Wang, zheng li, D. Sun, *et al.*, "Learning to sample and aggregate: Few-shot reasoning over temporal knowledge graphs," in *Advances in Neural Information Processing Systems*, 2022.

[6] C. Yang, J. Li, R. Wang, *et al.*, "Hierarchical overlapping belief estimation by structured matrix factorization," in *ASONAM 2020*, 2020.

[7] J. Li, H. Shao, D. Sun, *et al.*, "Unsupervised belief representation learning with information-theoretic variational graph auto-encoders," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '22, 2022, pp. 1728–1738.

[8] R. Wang, Y. Yan, J. Wang, *et al.*, "Acekg: A large-scale knowledge graph for academic data mining," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18, Torino, Italy, 2018.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, 2013.

[10] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias.," in *CIDR*, 2015.

[11] J. Leblay and M. W. Chekol, "Deriving validity time in knowledge graph," in *WWW'18*, 2018.

[12] Z. Tang, S. Pei, Z. Zhang, *et al.*, "Positive-unlabeled learning with adversarial data augmentation for knowledge graph completion," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed., Main Track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 2248–2254.

[13] A. Saxena, S. Chakrabarti, and P. Talukdar, "Question answering over temporal knowledge graphs," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Aug. 2021.

[14] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI'15*, 2015.

[15] B. Yang, W.-t. Yih, X. He, J. Gao, and l. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR'14*.

[16] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard, "Knowledge graph completion via complex tensor factorization," *J. Mach. Learn. Res.*, 2017.

[17] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *International Conference on Learning Representations*, 2019.

[18] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks.," in *ESWC*, 2017, pp. 593–607.

[19] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18, 2018.

[20] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018, pp. 327–333.

[21] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *International Conference on Learning Representations*, 2020.

[22] Z. Li, X. Jin, W. Li, *et al.*, "Temporal knowledge graph reasoning based on evolutional representation learning," in *SIGIR*, 2021.

[23] X. Chen, M. Chen, W. Shi, Y. Sun, and C. Zaniolo, "Embedding uncertain knowledge graphs," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19, 2019.

[24] N. Kertkeidkachorn, X. Liu, and R. Ichise, "Gtranse: Generalizing translation-based model on uncertain knowledge graph embedding," in *Advances in Artificial Intelligence - Selected Papers from the Annual Conference of Japanese Society of Artificial Intelligence (JSAI 2019), Niigata, Japan, 4-7 June 2019*, ser. Advances in Intelligent Systems and Computing, vol. 1128, 2019, pp. 170–178.

[25] M. D. Plessis, G. Niu, and M. Sugiyama, "Convex formulation for learning from positive and unlabeled data," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France, Jul. 2015, pp. 1386–1394.

[26] S. Jain, M. White, and P. Radivojac, "Estimating the class prior and posterior from noisy positives and unlabeled data," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, Barcelona, Spain: Curran Associates Inc., 2016, pp. 2693–2701, ISBN: 9781510838819.

[27] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[28] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1499–1509.

[29] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[30] Y. Zhang, Q. Yao, Y. Shao, and L. Chen, "Nscaching: Simple and efficient negative sampling for knowledge graph embedding," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, IEEE, 2019, pp. 614–625.

[31] K. Ahrabian, A. Feizi, Y. Salehi, W. L. Hamilton, and A. J. Bose, "Structure aware negative sampling in knowledge graphs," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 6093–6101.

[32] J. Qian, X.-Y. Li, C. Zhang, L. Chen, T. Jung, and J. Han, "Social network deanonymization and privacy inference with knowledge graph model," *IEEE Transactions on Dependable and Secure Computing*, pp. 679–692, 2019.

[33] A. Garcia-Duran, S. Dumancic, and M. Niepert, "Learning sequence encoders for temporal knowledge graph completion," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 4816–4821.

[34] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, "Semi-supervised hypergraph node classification on hypergraph line expansion," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, ser. CIKM '22, Atlanta, GA, USA: Association for Computing Machinery, 2022, pp. 2352–2361, ISBN: 9781450392365.

[35] J. Pujara, E. Augustine, and L. Getoor, "Sparsity and noise: Where knowledge graph embeddings fall short," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1751–1756.

[36] X. Chen, M. Boratko, M. Chen, S. S. Dasgupta, X. L. Li, and A. McCallum, "Probabilistic box embeddings for uncertain knowledge graph reasoning," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Jun. 2021.

[37] L. Cai and W. Y. Wang, "KBGAN: Adversarial learning for knowledge graph embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1470–1480.

[38] J. Bekker and J. Davis, "Learning from positive and unlabeled data: A survey," *CoRR*, vol. abs/1811.04820, 2018.

[39] F. He, T. Liu, G. I. Webb, and D. Tao, "Instance-dependent PU learning by bayesian optimal relabeling," *CoRR*, vol. abs/1808.02180, 2018.

[40] H. Shi, S. Pan, J. Yang, and C. Gong, "Positive and unlabeled learning via loss decomposition and centroid estimation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18, Stockholm, Sweden: AAAI Press, 2018, pp. 2689–2695, ISBN: 9780999241127.

[41] W. Fan, Y. Ma, Q. Li, *et al.*, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, ACM, 2019.

[42] A. Javari, Z. He, Z. Huang, R. Jeetu, and K. Chen-Chuan Chang, "Weakly supervised attention for hashtag recommendation using graph data," in *Proceedings of The Web Conference 2020*, ser. WWW '20, 2020.

[43] S. Liu, S. Yao, D. Liu, *et al.*, "A latent hawkes process model for event clustering and temporal dynamics learning with applications in github," in *ICDCS'19*, 2019.

[44] H. Shao, S. Yao, A. Jing, *et al.*, "Misinformation detection and adversarial attack cost analysis in directional social networks," in *ICCCN'20*, 2020.

[45] C. Xu, J. Li, D. Sun, *et al.*, "On polarization dynamics in the age of information overload," in *Special Edition on Information Operations on Social Media, Workshop Proc. of the 15th International AAAI Conference on Web and Social Media*, AAAI Press, 2021.

[46] D. Sun, C. Yang, J. Li, *et al.*, "Computational modeling of hierarchically polarized groups by structured matrix factorization," *Frontiers in Big Data*, vol. 4, 2021.

[47] X. Liu, J. Li, D. Sun, *et al.*, *Unsupervised image classification by ideological affiliation from user-content interaction patterns*, 2023.

[48] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *WWW '15*, 2015.

[49] H. Wang, R. Wang, C. Wen, *et al.*, "Author name disambiguation on heterogeneous information network with adversarial representation learning," in *AAAI '20*, 2020.

[50] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, 2001.

[51] C. Yang, M. Sun, H. Liu, S. Han, Z. Liu, and H. Luan, "Neural diffusion model for microscopic cascade study," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[52] J. Wang, V. W. Zheng, Z. Liu, and K. C. Chang, "Topological recurrent neural network for diffusion prediction," *CoRR*, vol. abs/1711.10162, 2017.

[53] M. R. Islam, S. Muthiah, B. Adhikari, B. A. Prakash, and N. Ramakrishnan, "Deepdiffuse: Predicting the "who" and "when" in cascades," *2018 IEEE International Conference on Data Mining (ICDM)*, 2018.

[54] A. Sankar, X. Zhang, A. Krishnan, and J. Han, "Inf-vae: A variational autoencoder framework to integrate homophily and influence in diffusion prediction," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, ser. WSDM '20, 2020.

[55] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03, 2003.

[56] M. Granovetter, "Threshold models of collective behavior," *American Journal of Sociology*, 1978.

[57] S. Bourigault, S. Lamprier, and P. Gallinari, "Representation learning for information diffusion through social networks: An embedded cascade model," in *WSDM '16*, 2016.

[58] J. Yang and J. Leskovec, "Modeling information diffusion in implicit networks," in *ICDM '10*, 2010.

[59] Y. Gu, Y. Sun, and J. Gao, "The co-evolution model for social network evolving and opinion migration," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[60] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proceedings of the 5th International Conference on Learning Representations*, (ICLR), ser. ICLR '17, 2017.

[61] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[62] Z. Wang, C. Chen, and W. LI, "A sequential neural information diffusion model with structure attention," in *CIKM '18*, 2018.

[63] Y. Wang, H. Shen, S. Liu, J. Gao, and X. Cheng, "Cascade dynamics modeling with attention-based recurrent neural network," in *IJCAI'17*, 2017.

[64] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, 2016.

[65] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[66] E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," in *NIPS'19*, 2019.

[67] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer, "Outtweeting the twitterers - predicting information cascades in microblogs," in *WOSN'10*, 2010.

[68] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR'14*, 2014.

[69] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML'14*, 2014.

[70] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP'14*, 2014.

[71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL'19*, 2019.

[72] A. Enders, H. Hungenberg, H.-P. Denker, and S. Mauch, "The long tail of social networking.," *European Management Journal*, 2008.

[73] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[74] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *IJCAI'11*, 2011.

[75] C. Yang, R. Wang, S. Yao, S. Liu, and T. F. Abdelzaher, "Revisiting "over-smoothing" in deep gcns," *CoRR*, vol. abs/2003.13663, 2020.

[76] G. Li, M. Müller, A. K. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" In *ICCV'19*, 2019.

[77] K. Saito, M. Kimura, K. Ohara, and H. Motoda, "Learning continuous-time information diffusion model for social behavioral data analysis," in *Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning*, ser. ACML '09, Springer-Verlag, 2009.

[78] S. Myers and J. Leskovec, "On the convexity of latent social network inference," in *Advances in Neural Information Processing Systems 23*, 2010.

[79] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," *ACM Trans. Knowl. Discov. Data*, 2012.

[80] M. Gomez Rodriguez, J. Leskovec, D. Balduzzi, and B. Schölkopf, "Uncovering the structure and temporal dynamics of information propagation," *Network Science*, 2014.

[81] M. Gomez Rodriguez, J. Leskovec, and B. Schölkopf, "Structure and dynamics of information pathways in online media," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13, Association for Computing Machinery, 2013.

[82] S. Wang, X. Hu, P. S. Yu, and Z. Li, "Mmrate: Inferring multi-aspect diffusion networks with multi-pattern cascades," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, 2014.

[83] C. Yang, J. Tang, M. Sun, G. Cui, and Z. Liu, "Multi-scale information diffusion prediction with reinforced recurrent networks," in *IJCAI'19*, 2019.

[84] Z. Wang and W. Li, "Hierarchical diffusion attention network," in *IJCAI'19*, 2019.

[85] C. Li, X. Guo, and Q. Mei, "Joint modeling of text and networks for cascade prediction," in *ICWSM'18*, 2018.

[86] I. Higgins, L. Matthey, A. Pal, *et al.*, "Beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[87] H. Shao, S. Yao, D. Sun, *et al.*, "ControlVAE: Controllable variational autoencoder," in *ICML'20*, 2020.

[88] Z. Huang, Y. Sun, and W. Wang, "Learning continuous system dynamics from irregularly-sampled partial observations," in *Advances in Neural Information Processing Systems*, 2020.

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, 1997.

[90] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL'15*, 2015.

[91] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of Machine Learning Research*, 2017.

[92] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "Dysat: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 519–527.

[93] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *KDD '16*, 2016.

[94] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation.," in *KDD*, 2019, pp. 950–958.

[95] H. Wang, F. Zhang, J. Wang, *et al.*, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *CIKM '18*, 2018.

[96] Q. Ai, V. Azizi, X. Chen, and Y. Zhang, "Learning heterogeneous knowledge base embeddings for explainable recommendation," *CoRR*, vol. abs/1805.03352, 2018.

[97] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[98] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, "Interest-aware message-passing gcn for recommendation," in *Proceedings of the Web Conference 2021*, 2021.

[99] H. Zeng, M. Zhang, Y. Xia, *et al.*, "Deep graph neural networks with shallow subgraph samplers," in *NeurIPS'21*, 2021.

[100] G. Jeh and J. Widom, "Scaling personalized web search," in *WWW '03*, 2003.

[101] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *ICLR'19*, 2019.

[102] A. Bojchevski, J. Klicpera, B. Perozzi, *et al.*, "Scaling graph neural networks with approximate pagerank," in *KDD'20*, 2020.

[103] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," in *SIGMOD'15*, 2017.

[104] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *SIGIR '11*, 2011.

[105] X. He and T. Chua, "Neural factorization machines for sparse predictive analytics," in *SIGIR '17*, 2017.

[106] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," *CoRR*, vol. abs/1706.03847, 2017.

[107] F. Sun, J. Liu, J. Wu, *et al.*, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *CIKM '19*, 2019.

[108] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *KDD '19*, 2019.

[109] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, ser. FOCS '06, 2006, pp. 475–486.

[110] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *RecSys '17*, 2017.

[111] L. Rakkappan and V. Rajan, "Context-aware sequential recommendations withstacked recurrent neural networks," in *WWW '19*, 2019.

[112] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *WSDM '19*, 2019.

[113] B. Adhikari, P. Sondhi, W. Zhang, M. Sharma, and B. A. Prakash, "Mining e-commerce query relations using customer interaction networks," in *WWW '18*, 2018.

[114] I. A. Adeyanju, D. Song, M.-D. Albakour, U. Kruschwitz, A. De Roeck, and M. Fasli, "Adaptation of the concept hierarchy model with search logs for query recommendation on intranets," in *SIGIR '12*, 2012.

[115] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Recurrent coevolutionary latent feature processes for continuous-time recommendation," in *DLRS 2016*, 2016.

[116] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G.-z. Sun, "Xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *KDD'18*, 2018.

[117] J. Qin, K. Ren, Y. Fang, W. Zhang, and Y. Yu, "Sequential recommendation with dual side neighbor-based collaborative relation modeling," in *WSDM '20*, 2020.

[118] J. Hao, T. Zhao, J. Li, *et al.*, "P-companion: A principled framework for diversified complementary product recommendation," in *CIKM '20*, 2020.

[119] S. Zhou, X. Dai, H. Chen, *et al.*, "Interactive recommender system via knowledge graph-enhanced reinforcement learning," in *SIGIR'20*, 2020.

[120] I. Antonellis, H. G. Molina, and C. C. Chang, "Simrank++: Query rewriting through link analysis of the click graph," *Proc. VLDB Endow.*, 2008.

[121] Y. He, J. Tang, H. Ouyang, C. Kang, D. Yin, and Y. Chang, "Learning to rewrite queries," in *CIKM '16*, 2016.

[122] B. Mitra and N. Craswell, "Query auto-completion for rare prefixes," in *CIKM '15*, 2015.

[123] D. H. Park and R. Chiba, "A neural language model for query auto-completion," in *SIGIR '17*, 2017.

[124] D. Ceccarelli, S. Gordea, C. Lucchese, F. M. Nardini, and R. Perego, "When entities meet query recommender systems: Semantic search shortcuts," in *SAC '13*, 2013.

[125] W. U. Ahmad, K.-W. Chang, and H. Wang, "Context attentive document ranking and query suggestion," in *SIGIR'19*, 2019.

[126] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, 2014.

[127] S. Deng, H. Rangwala, and Y. Ning, "Learning dynamic context graphs for predicting social events," ser. KDD '19, 2019.

[128] C. Yang, R. Wang, S. Yao, and T. F. Abdelzaher, "Hypergraph learning with line expansion," *CoRR*, vol. abs/2005.04843, 2020.

[129] Y. Yan, S. Zhang, and H. Tong, "Bright: A bridging algorithm for network alignment," in *Proceedings of the Web Conference 2021*, 2021.

[130] B. Jing, C. Park, and H. Tong, "Hdmi: High-order deep multiplex infomax," in *Proceedings of the Web Conference 2021*, 2021, pp. 2414–2424.

[131] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[132] H. Wang, R. Wang, C. Wen, *et al.*, "Author name disambiguation on heterogeneous information network with adversarial representation learning," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 238–245.

[133] H. Shao, T. Abdelzaher, J. Han, *et al.*, "Simulating online social response: A stimulus/response perspective," in *2021 Winter Simulation Conference (WSC)*, 2021, pp. 1–12.

[134] X. Liu, R. Wang, D. Sun, *et al.*, *Influence pathway discovery on social media*, 2023.

[135] B. Jing, H. Tong, and Y. Zhu, "Network of tensor time series," in *Proceedings of the Web Conference 2021*, 2021, pp. 2425–2437.

[136] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *ICML'17*.

[137] W. Jin, M. Qu, X. Jin, and X. Ren, "Renet: Autoregressive structure inference over temporal knowledge graphs," in *EMNLP*, 2020.

[138] X. Chen, M. Chen, C. Fan, A. Uppunda, Y. Sun, and C. Zaniolo, "Multilingual knowledge graph completion via ensemble knowledge transfer," in *EMNLP*, 2020, pp. 3227–3238.

[139] H. Singh, S. Chakrabarti, P. JAIN, S. R. Choudhury, and Mausam., "Multilingual knowledge graph completion with joint relation and entity alignment," in *Conference on Automated Knowledge Base Construction*, 2021.

[140] Z. Huang, Z. Li, H. Jiang, *et al.*, "Multilingual knowledge graph completion with self-supervised adaptive graph alignment," in *ACL'22*, 2022.

[141] M. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *NIPS'10*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010.

[142] M. Xu, B. Li, G. Niu, B. Han, and M. Sugiyama, "Revisiting sample selection approach to positive-unlabeled learning: Turning unlabeled data into positive rather than negative," *CoRR*, vol. abs/1901.10155, 2019.

[143] S. Gottschalk and E. Demidova, "Eventkg: The hub of event knowledge on the web and biographical timeline generation.," *Semantic Web*, 2019.

[144] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan, "Inductive representation learning on temporal graphs," in *ICLR'20*, 2020.

[145] Z. Sun, W. Hu, Q. Zhang, and Y. Qu, "Bootstrapping entity alignment with knowledge graph embedding," in *IJCAI'18*, Stockholm, Sweden, 2018, pp. 4396–4402.

[146] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, Y. Teh and M. Titterington, Eds., ser. JMLR W & CP, vol. 9, 2010, pp. 297–304.

[147] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," *CoRR*, vol. abs/2005.10242, 2020.

[148] X. Liu, H. Hong, X. Wang, *et al.*, "Selfkg: Self-supervised entity alignment in knowledge graphs," *Proceedings of the ACM Web Conference 2022*, 2022.

[149] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, and M. Ward, "Icews coded event data," 2015.

[150] J. Lehmann, R. Isele, M. Jakob, *et al.*, "DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web Journal*, vol. 6, no. 2, pp. 167–195, 2015.

[151] L. Liu, B. Du, J. Xu, Y. Xia, and H. Tong, "Joint knowledge graph completion and question answering," in *KDD '22*, Washington DC, USA: Association for Computing Machinery, 2022, pp. 1098–1108, ISBN: 9781450393850.

[152] C. Xu, M. Nayyeri, F. Alkhoury, H. Yazdi, and J. Lehmann, "Temporal knowledge graph completion based on time series gaussian embedding," in *The Semantic Web – ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part I*, 2020, pp. 654–671.

[153] R. Goel, S. M. Kazemi, M. A. Brubaker, and P. Poupart, "Diachronic Embedding for Temporal Knowledge Graph Completion," in *AAAI'20*, 2020.

[154] Z. Han, Z. Ding, Y. Ma, Y. Gu, and V. Tresp, "Learning neural ordinary equations for forecasting future links on temporal knowledge graphs," in *NeurIPS'21*, 2021.

[155] S. Liao, S. Liang, Z. Meng, and Q. Zhang, "Learning dynamic embeddings for temporal knowledge graphs," in *WSDM '21*, 2021.

[156] M. Chen, Y. Tian, M. Yang, and C. Zaniolo, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," in *IJCAI'17*.

[157] Y. Yan, L. Liu, Y. Ban, B. Jing, and H. Tong, "Dynamic knowledge graph alignment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021.

[158] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[159] B. Shi and T. Weninger, "Open-world knowledge graph completion," *CoRR*, 2017.

[160] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," in *IJCAI'17*, 2017, pp. 1802–1808.

[161] P. Wang, J. Han, C. Li, and R. Pan, "Logic attention based neighborhood aggregation for inductive knowledge graph embedding," in *AAAI'19*, 2019.

[162] J. Baek, D. B. Lee, and S. J. Hwang, "Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction," in *Advances in Neural Information Processing Systems*, 2020.

[163] L. Cui, Y. Wu, S. Liu, and Y. Zhang, "Knowledge enhanced fine-tuning for better handling unseen entities in dialogue generation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

[164] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[165] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[166] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017, pp. 1025–1035.

[167] H. Zeng, M. Zhang, Y. Xia, *et al.*, "Decoupling the depth and scope of graph neural networks," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[168] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *International Conference on Learning Representations*, 2018.

[169] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International Conference on Learning Representations*, 2019.

[170] C. Zhang, H. Yao, C. Huang, M. Jiang, Z. Li, and N. V. Chawla, "Few-shot knowledge graph completion," in *AAAI*, 2019.

[171] K. Huang and M. Zitnik, "Graph meta learning via local subgraphs," *NeurIPS*, 2020.

[172] A. J. Bose, A. Jain, P. Molino, and W. L. Hamilton, "Meta-graph: Few shot link prediction via meta learning," *CoRR*, 2019.

[173] D. A. McAllester, "Pac-bayesian model averaging," in *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, ser. COLT'99, ACM, 1999, pp. 164–170.

[174] R. C. Wilson, E. R. Hancock, E. Pekalska, and R. P. Duin, "Spherical and hyperbolic embeddings of data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2255–2269, 2014.

[175] T. R. Davidson, L. Falorsi, N. D. Cao, T. Kipf, and J. M. Tomczak, *Hyperspherical variational auto-encoders*, 2022.

[176] J. Xu and G. Durrett, "Spherical latent spaces for stable variational autoencoders," *CoRR*, vol. abs/1808.10805, 2018.

[177] Y. Meng, J. Huang, G. Wang, *et al.*, "Spherical text embedding," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[178] C. Gulcehre, M. Denil, M. Malinowski, *et al.*, "Hyperbolic attention networks," *arXiv preprint arXiv:1805.09786*, 2018.

[179] A. Ermolov, L. Mirvakhabova, V. Khrulkov, N. Sebe, and I. Oseledets, "Hyperbolic vision transformers: Combining improvements in metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7409–7419.

[180] S. Ge, S. Mishra, S. Kornblith, C.-L. Li, and D. Jacobs, "Hyperbolic contrastive learning for visual representations beyond objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6840–6849.

[181] A. Bdeir, K. Schwethelm, and N. Landwehr, "Hyperbolic geometry in computer vision: A novel framework for convolutional neural networks," *arXiv preprint arXiv:2303.15919*, 2023.

[182] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," *Advances in neural information processing systems*, vol. 30, 2017.

[183] M. Nickel and D. Kiela, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," in *International conference on machine learning*, PMLR, 2018, pp. 3779–3788.

[184] F. Sala, C. De Sa, A. Gu, and C. Ré, "Representation tradeoffs for hyperbolic embeddings," in *International conference on machine learning*, PMLR, 2018, pp. 4460–4469.

[185] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[186] L. Sun, Z. Zhang, J. Zhang, *et al.*, "Hyperbolic variational graph neural network for modeling dynamic graphs," in *AAAI '21*, 2021.

[187] Q. Bai, C. Nie, H. Zhang, D. Zhao, and X. Yuan, "HGWaveNet: A hyperbolic graph neural network for temporal link prediction," in *Proceedings of the ACM Web Conference 2023*, ACM, Apr. 2023.

[188] Q. Liu, M. Nickel, and D. Kiela, "Hyperbolic graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[189] Y. Zhang, X. Wang, C. Shi, X. Jiang, and Y. Ye, "Hyperbolic graph attention network," *IEEE Transacti ons on Big Data*, vol. 8, no. 6, pp. 1690–1701, 2021.

[190] S. Wang, X. Wei, C. N. dos Santos, *et al.*, "Mixed-curvature multi-relational graph neural network for knowledge graph completion," in *The Web Conference 2021*, 2021.

[191] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014, pp. I–XVI, 1–397.

[192] C. Yang, C. Wang, Y. Lu, *et al.*, "Few-shot link prediction in dynamic networks," in *WSDM '22*, 2022.

[193] L. Sun, Z. Zhang, J. Zhang, *et al.*, "Hyperbolic variational graph neural network for modeling dynamic graphs," in *AAAI Conference on Artificial Intelligence*, 2021.

[194] S. Montella, L. M. Rojas Barahona, and J. Heinecke, "Hyperbolic temporal knowledge graph embeddings with relational and time curvatures," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Online: Association for Computational Linguistics, Aug. 2021, pp. 3296–3308.

[195] Z. Han, P. Chen, Y. Ma, and V. Tresp, "Dyernie: Dynamic evolution of riemannian manifold embeddings for temporal knowledge graph completion," in *EMNLP '20'*, Jan. 2020, pp. 7301–7316.

[196] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[197] I. Chami, A. Wolf, D.-C. Juan, F. Sala, S. Ravi, and C. Ré, "Low-dimensional hyperbolic knowledge graph embeddings," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[198] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, 2020.

[199] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[200] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu koray, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, 2016.

[201] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017.

[202] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, 2018.

[203] J. Sheng, S. Guo, Z. Chen, *et al.*, "Adaptive attentional network for few-shot knowledge graph completion," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, Association for Computational Linguistics, 2020, pp. 1681–1691.

[204] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[205] M. Chen, W. Zhang, W. Zhang, Q. Chen, and H. Chen, "Meta relational learning for few-shot link prediction in knowledge graphs," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[206] G. Niu, Y. Li, C. Tang, *et al.*, "Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[207] J. Wu, M. Cao, J. C. K. Cheung, and W. L. Hamilton, *Temp: Temporal message passing for temporal knowledge graph completion*, 2020.

[208] Z. Li, X. Jin, W. Li, *et al.*, "Temporal knowledge graph reasoning based on evolutional representation learning," in *SIGIR*, 2021.

[209] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, 2012.

[210] D. Greene, D. Doyle, and P. Cunningham, "Tracking the evolution of communities in dynamic social networks," in *2010 international conference on advances in social networks analysis and mining*, IEEE, 2010, pp. 176–183.

[211] T. N. Kipf, E. Fetaya, K. Wang, M. Welling, and R. S. Zemel, "Neural relational inference for interacting systems," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, 2018, pp. 2693–2702.

[212] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[213] Y. Zhang, F. Feng, C. Wang, *et al.*, "How to retrain recommender system? a sequential meta-learning method," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20, 2020, pp. 1479–1488.

[214] Q. Yuan, S.-U. Guan, P. Ni, *et al.*, "Continual graph learning: A survey," *arXiv preprint arXiv:2301.12230*, 2023.

[215] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS 2014 Deep Learning Workshop*, 2014.

[216] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," in *International Conference on Learning Representations*, 2021.

[217] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, and N. Shah, "Graph condensation for graph neural networks," in *International Conference on Learning Representations*, 2022.

[218] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen, "Streaming ranking based recommender systems," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 525–534.

[219] S. Ding, F. Feng, X. He, Y. Liao, J. Shi, and Y. Zhang, "Causal incremental graph convolution for recommender system retraining," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[220] X. Kou, Y. Lin, S. Liu, P. Li, J. Zhou, and Y. Zhang, "Disentangle-based continual graph representation learning," *arXiv preprint arXiv:2010.02565*, 2020.

[221] H. Liu, Y. Yang, and X. Wang, "Overcoming catastrophic forgetting in graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 8653–8661.

[222] F. Zhou and C. Cao, "Overcoming catastrophic forgetting in graph neural networks with experience replay," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 4714–4722.

[223] X. Chen, J. Wang, and K. Xie, "Trafficstream: A streaming traffic flow forecasting framework based on graph neural networks and continual learning," *arXiv preprint arXiv:2106.06273*, 2021.

[224] J. Wang, W. Zhu, G. Song, and L. Wang, "Streaming graph neural networks with generative replay," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1878–1888.

[225] C. Wang, Y. Qiu, D. Gao, and S. Scherer, "Lifelong graph learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 13 719–13 728.

[226] H. Yao, Y. Wei, J. Huang, and Z. Li, "Hierarchically structured meta-learning," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 7045–7054.

[227] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[228] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proceedings of the 25th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2019.

[229] S. Rendle and L. Schmidt-Thieme, "Online-updating regularized kernel matrix factorization models for large-scale recommender systems," in *Proceedings of the 2008 ACM conference on Recommender systems*, 2008, pp. 251–258.

[230] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[231] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 501–14 515, 2022.

[232] A. Pareja, G. Domeniconi, J. Chen, *et al.*, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 5363–5370.

[233] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Transactions on Privacy and Security*, 2022.

[234] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, 2018, pp. 2857–2866, ISBN: 9781450355520.

[235] Y. Wang, Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," *CoRR*, 2021.

[236] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18, 2018, pp. 969–976.

[237] H. Lium, Y. Yang, and X. Wang, "Overcoming catastrophic forgetting in graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 8653–8661.

[238] Y. Xu, Y. Zhang, W. Guo, H. Guo, R. Tang, and M. Coates, "Graphsail: Graph structure aware incremental learning for recommender systems," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, ser. CIKM '20, 2020, pp. 2861–2868.

[239] J. Wang, G. Song, Y. Wu, and L. Wang, "Streaming graph neural networks via continual learning," in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 1515–1524.