LEARNING DYNAMIC PROCESSES OVER GRAPHS

A Dissertation
Presented to
The Academic Faculty

By

Rakshit S. Trivedi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computer Science

Georgia Institute of Technology

August 2020

LEARNING DYNAMIC PROCESSES OVER GRAPHS

Approved by:

Dr. Hongyuan Zha, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Duen Horng Chau
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Umit Catalyurek
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Peter Battaglia
DeepMind

Dr. Xin Luna Dong
Amazon

Date Approved: June 25, 2020

I was born not knowing and have had only a little time to change that here and there.

Richard P. Feynman

To my lovely spouse Hazel and my beloved parents.

ACKNOWLEDGEMENTS

I wish to express my sincerest appreciation and gratitude to my advisor, Professor Hongyuan Zha, for his immense support, motivation and patience that made this long journey, both enjoyable and fruitful for me and set the course of what has been achieved through my PhD study. Professor Zha's breadth and depth of knowledge in mathematics and machine learning and visionary insights in formulating problems and tackling them with innovative solution approaches never ceases to amaze me. I still remember the first time I met him to discuss my PhD journey when I was a bit lost and he provided me with invaluable guidance and considerate advising to help me choose the path and topics that were both interesting and challenging for me. His mentorship directed towards making high standard contributions has greatly inspired me to become an independent and productive researcher.

I would like to thank Duen Horng (Polo) Chau and Umit Catalyurek for being on my Ph.D. dissertation committee and for their their time to attend my thesis and proposal defense. Their insightful suggestions and comments on my research and guidance related to various processes throughout my PhD has certainly helped to make it a smooth ride. I would like to express my deepest appreciation to Peter Battaglia for being on my Ph.D. dissertation committee and for his time and patience in providing me with invaluable feedback on my research. His work has greatly inspired my research directions and I have thoroughly cherished participating with him in hours long discussions on variety of technical topics during the conferences and otherwise. I am very grateful to Xin Luna Dong for being on my Ph.D. dissertation committee and more importantly providing me with a platform to gain experience in industrial-scale research during my internship at Amazon. I really enjoy my close collaboration with her as she has been a great mentor who helped me to develop skills in approaching and disseminating complex technical materials in a simplified manner so as to make it easily understandable and consumable.

My collaboration with Le Song inspired me to strive for state-of-the-art research with

perseverance and I am grateful for his time and support towards my research pursuit. I also wish to express special thanks to Christos Faloutsos for helping me to refine my problem solving skills and providing me with invaluable feedback on various aspects of my research including this dissertation. I would like to thank H. Venkateswaran for providing me with much needed academic advising and serving as a guiding light during my tough times. Last but not the least, I am forever indebted to Jayant Harista, my research mentor at IISC (before my time at Georgia Tech), who instilled the seeds of pursuing research as a career in me and played a major role in kickstarting my journey to where I have reached today.

This journey would not have been possible without the strong support and camaraderie of marvelous groups of friends, colleagues and collaborators that I had the privilege and honor to be involved with. At start of my PhD, Mehrdad Farajtabar introduced me to various topics and guided me through the research process. I highly enjoyed various technical conversations and every day collaboration with him which eventually turned into wonderful friendship over time. Elias Khalil and I have become best friends as we navigated various ebbs and tides of PhD life together, whether it is sprinting through sleepless nights to catch conference deadlines or just hanging out and contemplating the world around us. Elias has provided me with a constant platform to sound out my ideas and offered invaluable suggestions to tackle various problems. Finally, I couldn't have asked for a better academic brother than Jiachen Yang, whose sharp knowledge, discipline and straightforward nature constantly inspired me. He has become a great friend and in addition to our continued research collaboration, I highly cherish various discussions and debates with him surrounding philosophy, physics, marathons and joint criticism of various pitfalls in the academic and research practices.

Additionally, several collaborators and colleagues played a vital role in benefiting my research: Yunfei Bai, Prasenjeet Biswal, Sisman Bunyamin, Bo Dai, Hanjun Dai, Nan Du, Chelsea Finn, Ben London, Jun Ma, Apurv Verma, Yichen Wang, Bo Xie, Huan Xu, Xiaojing Ye among others. Further, memorable moments such as election and current

affairs discussions with labmates Caleb and Amrita, sob stories on paper rejections with Yujia Xie, pool time with Harsh Shrivastava, coding practice with Lansie Ma, late night drives and conversations with Yuyu Zhang and Zhi Zhang and many more made my PhD life smooth and enjoyable. I am particularly thankful to my lifetime friends Anshuman Dutt, Pranay Kolakkar, Surabhi Potnis and Pratik Pattani who have been there providing me with warmth and a sense of security despite being at farther distance, thereby continuously supporting me in my pursuit.

I will always be indebted to my parents who showered me with uncountable support and love while making several sacrifices to contribute towards my progress and development. They taught me to persevere under hardest circumstances while providing me with an environment of fostering good values that guides me to always work towards becoming a better human being. I am greatly inspired by my younger and cheerful sister, Shradhdha, who has given me a lot of love and strength to pursue my goals. I am also thankful to my extended family, grand parents, in-laws, uncles, aunts and cousins for all their support and trust in me. Lastly, but most importantly, I am extremely fortunate to have my wife, Hazel, stand by my side as a rock in this arduous journey. Hazel is my best friend, lover and lifetime companion who has always created wonderful moments in our life, motivated me through tough times and encouraged me to never give up on my dreams. Her unwavering support for an always busy, stressed and deadline stretched husband, while pursuing her own challenging career goals speaks of her unparalleled strength and unconditional strong belief in us and our future. Thank you Hazel for everything, I am very proud of you and I look forward to a wonderful future with you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Graphs appear as a versatile representation of information across domains spanning social networks, biological networks, transportation networks, molecular structures, knowledge networks, web information network and many more. Graphs represent heterogeneous information about the real-world entities and complex relationships between them in a very succinct manner. At the same time, graphs exhibit combinatorial, discrete and non-Euclidean properties in addition to being inherently sparse and incomplete which poses several challenges to techniques that analyze and study these graph structures.

There exist various approaches across different fields spanning network science, game theory, stochastic process and others that provide excellent theoretical and analytical tools with interpretability benefits to analyze these networks. However, such approaches do not learn from data and make assumptions about real-world that capture only subset of properties. More importantly, they do not support predictive capabilities critical for decision making applications. In this thesis, we develop novel data driven learning approaches that incorporate useful inductive biases inspired from these classical approaches. The resulting learning approaches exhibit more general properties that go beyond conventional probabilistic assumptions and allow for building transferable and interpretable modules. We build these approaches anchored around two fundamental questions: (i) (Formation Process) How do these networks come into existence? and (ii) (Temporal Evolution Process) How do real-world networks evolve over time?

First, we focus on the challenge of learning in a setting with highly sparse and incomplete knowledge graphs, where it is important to leverage multiple input graphs to support accurate performance for variety of downstream applications such as recommendation, search and question-answering systems. Specifically, we develop a large-scale multi-graph deep relational learning framework that identifies entity linkage as a vital component of data fusion and learns to jointly perform representation learning and graph linkage across

multiple graphs with applications to relational reasoning and knowledge construction.

Next, we consider networks that evolve over time and propose a generative model of dynamic graphs that is useful to encode evolving network information into low-dimensional representations that facilitate accurate downstream event prediction tasks. Our approach relies on the coevolution principle of network structure evolution and network activities being tightly couple processes and develops a multi time scale temporal point process formulation parameterized by a recurrent architecture comprising of a novel Temporal Attention mechanism. Representation learning is posed as a latent mediation process – observed network processes evolve the state of nodes, while this node evolution governs future dynamics of observed processes and applied to downstream dynamic link prediction tasks and time prediction of future realizations (events) of both observed processes.

Finally, we investigate the implication of adopting the optimization perspective of network formation mechanisms for building learning approaches for graph structured data. In this work, we first focus on global mechanisms that govern the formation of links in the network and build an inverse reinforcement learning based algorithm to jointly discover latent mechanisms directly from observed data, optimization of which enables a graph construction procedure capable of producing graphs with properties similar to observed data. Such an approach facilitates transfer and generalization properties and has been applied to variety of real-world graphs. In the last part, we consider the settings where the agents forming links are strategic and build a learnable model of network emergence games that jointly discovers the underlying payoff mechanisms and strategic profiles of agents from the data. This approach enables learning interpretable and transferable payoffs while the learned game as a model facilitates strategic prediction tasks, both of which are applied to several real world networks.

CHAPTER 1
INTRODUCTION

Graphs serve as natural representations of information in many complex system domains such as social networks, knowledge graphs, financial systems, protein-protein networks and many more. While being a versatile representation of information, graphs exhibit discrete and combinatorial characteristics that give rise to significant challenges in leveraging the underlying information for downstream applications. The ubiquitous use of graphs for representing information across domains and applications has inspired concentrated efforts in building scalable machine learning techniques to address these graph learning challenges. Applications of such machine learning models seek to perform new predictions, inference, discover new patterns via generative reasoning and model behaviors. However, graphs that represent complex real-world domains exhibit intricate characteristics beyond discrete and combinatorial structure and raises new learning challenges. Below we briefly outline some of these challenges:

- **Incompleteness.** Real-world data represented as graphs often exhibit sparsity and incompleteness which becomes a big challenge when the downstream applications are dependent on the heterogeneous information contained in the missing edges. The problem is exacerbated large scale of graphs consisting billions of nodes. Hence the learning techniques need to perform powerful reasoning to infer missing information while meeting scalability demands.

- **Dynamics.** Most real-world networks exhibit temporal dynamic properties in which the entities and/or their interactions change over time. For instance, both the user behavior and the network structure they form (with other users on social networks or products on e-commerce platform) vary over time, collaboration networks evolve

with changing interests and interactions between authors and so on. Such settings require the learning approach to adapt and evolve corresponding to the network evolution. Traditional graph learning techniques that consider time-independent structures would cease to work effectively in these dynamic settings due to lack of their ability to consume, process and learn over evolving information.

- **Non-probabilistic Generative Mechanisms.** Learning generative mechanisms of graph-structured data is an important approach to build graph constructors for data augmentation [1] and inference modules for downstream network analysis and prediction tasks. Modeling the generative process of discrete structures is a hard problem and learning approaches tackling this challenge mainly rely on the probabilistic assumption of underlying mechanisms that govern this generative process. However, many real world networks form due to non-probabilistic processes such as global cost optimization (e.g. popularity in social networks) or local strategy optimizations (e.g. benefits in financial networks). These underlying processes violate the probabilistic assumptions often made by existing learning approaches. In real-world domains, such generative mechanisms are often unknown and learning them from data is challenging due to limited or no access to the construction process (one often observes only the final outcome structure).

There are several approaches that address the above challenges by building learning or simulation tools and techniques inspired from human intuition and assumptions about the underlying processes. We briefly discuss some of them below:

- For the incompleteness problem, many data drive organizations take the approach of constructing a unified super-graph by integrating data from multiple sources. Such unification has shown to significantly help in various applications, such as search, question answering, and personal assistance. To this end, there exists a rich body of work on linking entities and relations, and conflict resolution (e.g., knowledge fu-

sion [2]). Another typical approach for learning over multiple graphs is to first solve graph alignment problem to merge the graphs and then use existing relational learning methods on merged graph. These approaches leverage on specialized alignment techniques [3, 4, 5, 6] that address its unique challenges.

- In the dynamic network settings, there exist a rich body of works that focus on modeling temporal evolution of networks for network mining and link prediction tasks [7, 8, 9, 10, 11]. These techniques incorporate specific modeling assumptions to capture the observed processes. Further, there have been efforts in building sophisticated learning approaches that use parametric models for modeling fine-grained temporal dynamics. A successful class of these models leverage the mathematical models of temporal point process that model network evolution as continuous time events [12].

- While learning generative mechanisms under non-probabilistic setting is still an under-explored problem, there are several classical modeling approaches in other fields that focus on analysing the generative process of networks in corresponding domains and putting it to use for explaining network formation and building network design simulators. For instance, network science approaches attempt to explain the network formation process using specific functional forms of global objective, optimization of which is considered to be the driving factor of the generative process. [13, 14, 15, 16]. On the other hand, there has been decades of efforts in building game-theoretic approaches, called Network Formation Games [17], that analyze and explain the emergence of strategic networks formed due to actions of non-inanimate agents. These models serve as an elegant theoretical and interpretable framework to characterize the generative process.

While the above approaches have been effective in specific settings and provide interpretability benefits in some instances, they have their own shortcomings. For instance, graph alignment is currently an unsolved problem and faces severe scalability challenges

making it an expensive external step preceding learning. Next, specific models of dynamic networks are often prone to misspecification when the model assumptions do not align with the real-world intricacies involved in the dynamic evolution process. Finally, existing network science and game-theoretic approaches focus on simple but succinct models that are not learned from data, and which tend to model important albeit a subset of the properties of complex networks. Most importantly, all of the above (learning) approaches consider hand-designed node and edge features which often fail to capture complex information available in the graph. On the other hand, deep learning techniques for graph structured data [18] learn powerful graph representations that successfully capture various important properties of complex networks. These techniques have been extended to support complex tasks over graph structured information such as relational reasoning [19], probabilistic graph generation [20], learning simulation models [21], combinatorial optimization [22, 23, 24] and powerful prediction models [25].

Based on the above discussion, a natural question rises: Can we leverage the benefits of aforementioned classical approaches to build deep learning techniques that: (i) model and learn to discover the complex processes governing the properties exhibited by real-world networks and (ii) facilitate transferable modules and powerful predictions, so as to address the learning challenges on complex real-world networks? In this thesis, we take several steps to address this question.

1.1 Learning dynamic processes over graphs

Below, we introduce each one of our approaches and summarize their applications:

1.1.1 Part I. Multi-graph representation learning

In this part, we focus on alleviating the need of an external expensive step of data fusion for learning from multiple graph sources. Specifically, we identify entity linkage as an important component of data fusion and build a deep learning technique [26], learns

to jointly perform representation learning and graph linkage across multiple knowledge graphs. LinkNBed employs a robust multi-task loss function and a novel evaluation scheme to support entity linkage across various learning scenarios including presence of unlabeled instances or only negative instances. We apply this technique to perform relational reasoning tasks for knowledge graphs.

1.1.2 Part II. Modeling and learning dynamic network processes

Network structure evolution (dynamics **of** the network) and network activities (dynamics **on** the network) do not occur in isolation; these two processes are rather interleaved, affecting each other in an intricate manner. We adopt this coevolution principle for modeling evolution process of dynamic networks and leverage this model to learn dynamic graph representations. Specifically, we propose DyRep [27], a continuous time deep representation learning framework that uses a temporal point process formulation parameterized by a recurrent architecture comprising of a novel Temporal Attention mechanism. Representation learning is posed as a latent mediation process – observed network processes evolve the state of nodes, while this node evolution governs future dynamics of observed processes. DyRep is applied to dynamic link prediction and time prediction tasks. In this part, we also discuss an earlier work [28] modeling dynamically evolving knowledge over multi-relational interactions. This is a specialized version of above approach for temporal reasoning over dynamic knowledge graphs.

1.1.3 Part III. Learning graph formation mechanisms

In the above two parts, we focus on settings where the goal is to build deep learning architectures to model observed properties and encode them into representations, so as to perform effective learning for downstream predictions. In this part, we focus on settings where the underlying processes (mechanisms) are not observed and the goal is to discover such mechanisms directly from observed data, that are the final outcome of the latent mech-

anisms. Specifically, we focus on the network formation mechanisms, the process by which network come to assemble.

In real world graphs, these mechanisms are often unknown due to the limited or no access to the construction process of real-world graphs. In the first approach, we focus on the inverse problem setting of learning to discover such mechanisms from observed data and develop an end-to-end learning framework, GraphOpt [29], that adopts the network science view of optimization-based network formation. GraphOpt jointly learns the forward model of graph construction (posed as a sequential decision process) and solves the inverse problem of discovering an underlying optimization mechanism, in the form of a latent objective function (learned using an inverse reinforcement learning algorithm). GraphOpt learns to discover transferable mechanisms; demonstrates compelling generalization properties via its competitive link prediction performance and showcases its ability to serve as a useful graph constructor.

Complementary to the focus of the above approach on global mechanisms, in the second approach focuses on real-world networks, that emerge due to actions of non-inanimate agents (e.g. humans, animals). Such networks are the result of underlying strategic mechanisms aimed at maximizing local individual or collective benefits. Network learning approaches built to capture these strategic insights would gain interpretability and flexibility benefits that are required to generalize beyond observations. we consider a game-theoretic formalism of network emergence that accounts for the underlying strategic mechanisms and take it to data to discover an explanation for the observed real-world networks. We propose MINE, a new learning framework that solves Markov-Perfect network emergence games using multi-agent inverse reinforcement learning. MINE recovers a versatile and robust payoff useful for explaining final network structure and the network emergence game as a learned model facilitates strategic predictions.

1.2 Organization

The rest of the document is organized as follows: In Chapter 2, I present the literature survey on various works related to the above approaches. Then, in part I, I discuss my work on learning multi-source graph representations. Next, In part II, I present the works on modeling dynamic graphs and learning evolving representations over them. In part III, chapter 6, I discuss the work on learning optimization models of graph formation. Finally, in part III, chapter 7, I outline our recent work on learning strategic network emergence mechanisms. I conclude my thesis with a discussion on some limitations of my works and corresponding future directions.

CHAPTER 2
LITERATURE SURVEY

In this chapter, we discuss related works in different settings of our focus.

2.1 Entity Resolution in relational data and learning across multiple graphs

Entity Resolution refers to resolving entities available in knowledge graphs with entity mentions in text. [30] proposed entity disambiguation method for KB population, [31] learns entity embeddings for resolution, [32] propose a sophisticated DNN architecture for resolution, [33] proposes entity resolution across multiple social domains, [34] jointly embeds text and knowledge graph to perform resolution while [35] proposes Attention Mechanism for Collective Entity Resolution.

Recently, learning over multiple graphs have gained traction. [3] divides a multi-relational graph into multiple homogeneous graphs and learns associations across them by employing product operator. Unlike our work, they do not learn across multiple multi-relational graphs. [36] provides logic based insights for cross learning, [4] does pairwise entity matching across multi-relational graphs and is very expensive, [37] learns embeddings to support multi-lingual learning and Big-Align [5] tackles graph alignment problem efficiently for bipartite graphs. None of these methods learn latent representations or jointly train graph alignment and learning which is the goal of our work.

2.2 Dynamic graph representation learning

Preliminary approaches in dynamic representation learning have considered discrete time approach. [38] propose a temporal latent space model for link prediction using nonnegative matrix factorization. [39] uses a warm start method to train across snapshots and employs

a heuristic approach to learn stable embeddings over time but do not model time. [40] focuses on specific structure of triad to model how close triads are formed from open triads in dynamic networks. [41] proposes a deep architecture based on combination of CNN to capture spatial characteristics and an RNN to capture temporal characteristics, to model structured sequences which in graph case will lead to discrete time model. [42] develops extends skip-gram based approaches for network embedding to dynamic setting where the graphs a re observed as discrete time snapshot and the goal is to learn embeddings that can preserve the optimality of skip-gram objective. NetWalk [43] is a discrete-time dynamic embedding approach specifically designed for anomaly detection which uses clique based embedding techniques to learn vertex representations. Recently, [28] proposed Know-Evolve, a deep recurrent architecture to model multi-relational timestamped edges that addresses the communication process. Unlike our approach, Know-Evolve models all edges at a single timescale, works for setting restricted to relational graphs and uses only edge-level structural information with no attention mechanism. DANE [44] proposes a network embedding method in dynamic environment but their dynamics consists of change in node's attributes over time and their current work can be considered orthogonal to our approach. [45] proposes a dynamic network formation model to learn node representations by employing a Hawkes process to model the temporal evolution of neighborhood for nodes. This work only considers association events. [46] proposes a continuous time embedding framework that employs a temporal version of traditional random walks in a simple manner to capture temporally evolving neighborhood information.

2.3 Deep Generative Models of Graph Generation

Recently, there have been significant research efforts in building deep generative models of graph generation as they allow to effectively capture complex structural properties observed in a graph and use that information to output realistic graphs. Most of these works can be broadly categorized into two classes: (i) Methods that learn from collection of graphs (e.g.

DeepGMG [47], GraphRNN [20], GCPN [48]) and (ii) Methods that learn from a single graph (e.g VGAE [49], GraphGan [50], MolGAN [51], NetGan [52]). As discussed in the main paper, our current approach falls into the second category. DeepGMG builds probabilistic model where the partially generated graph is encoded by the graph neural network (GNN) and the representation is used to make decision of constructing next node or edge. GraphRNN proposes an auto-regressive model of graph generation, wherein the focus is on generating sequence of adjacency vectors that be mapped to graph structure. It employs hierarchical recurrent architecture to encode the historical path information. While it can produce arbitrarily large graphs, it has been shown to learn from relatively small sized graphs (input). Finally, it still needs a collection of graphs to learn well. GCPN combines GCN with RL and learns a deep generative model using an objective that is very specific to domain of chemistry. GCPN also requires a collection of molecular structures as input and works only with small graphs. In the second category, methods like GraphGan and GVAE are implicit models but their main focus is to learn graph representations and hence perform weakly on generation tasks and have limited scalability. NetGan is a recently proposed implict graph generator model exhibiting generalization properties. However, unlike GraphOpt, NetGAN optimizes a GAN-based objective which converge to an uninformative discriminator, thereby not useful for transfer, which in contrast is a key objective of our approach. Further, our method learns to model graphs using GNN over graph structure, thereby capturing better structural properties, in contrast to Netgan that employs an LSTM architecture to learn information from fixed length random walks. Finally, unlike NetGan, our appraoach is able to serve as an unsupervised framework for learning node representations (comes for free) that are useful for downstream prediction tasks.

2.4 Network Emergence Games

Network emergence games focus on analyzing the construction of equilibrium networks where no agent want to locally change the network [53]. Various equilibrium concepts

(with special focus on network stability (robustness)) have been proposed and studied to analyze the formation process in networks. It has been shown that pure Nash equilibrium is a weak and restrictive concept, for instance empty networks (where no agent announces any link) are always in Nash equilibrium, which further requires that agents' actions in each state to be independent [54]. As a more useful solution concept, [55] proposed pairwise stability that searches for networks that are robust to one link deviation, where link creation is bilateral and under mutual consent of the agents while link severance is unilateral. Building upon two concepts, [55, 56] proposed pairwise-Nash equilibrium that allows for unilateral multi-link severance in addition to mutual one-link creation and thereby effectively model non-cooperative games. [56]'s proposal is a noncooperative linking game in which agents independently announce which bilateral links they would like to see formed and then standard game-theoretic equilibrium concepts apply for making predictions. Finally, [57] proposed another linking game where players can §offer or demand transfers along with the links they suggest, which allows players to subsidize the emergence of particular links. Network emergence games have been generally modeled as one-shot normal form game or an extensive form game and only recently have been investigated by [58, 59] in Markov setting. Finally, couple of approaches[59, 60] have focused on estimating network emergence games where they use data to estimate the parameters of the model. While being computationally inefficient and limited in their practical applications, these approaches demonstrate the promise in investigating the use of observed data to learn the specifications for game theoretic approaches.

Part I

Multi-Graph Representation Learning

CHAPTER 3

RELATIONAL LEARNING OVER MULTI-SOURCE KNOWLEDGE

Knowledge graphs have emerged as an important model for studying complex multi-relational data. This has given rise to the construction of numerous large scale but incomplete knowledge graphs encoding information extracted from various resources. An effective and scalable approach to jointly learn over multiple graphs and eventually construct a unified graph is a crucial next step for the success of knowledge-based inference for many downstream applications. To this end, we propose **LinkNBed**, a deep relational learning framework that learns entity and relationship representations across multiple graphs. We identify **entity linkage** across graphs as a vital component to achieve our goal. We design a novel objective that leverage entity linkage and build an efficient multi-task training procedure. Experiments on link prediction and entity linkage demonstrate substantial improvements over the State-of-the-art relational learning approaches.

3.1 Introduction

Reasoning over multi-relational data is a key concept in Artificial Intelligence and knowledge graphs have appeared at the forefront as an effective tool to model such multi-relational data. Knowledge graphs have found increasing importance due to its wider range of important applications such as information retrieval [61], natural language processing [62], recommender systems [63], question-answering [64] and many more. This has led to the increased efforts in constructing numerous large-scale Knowledge Bases (e.g. Freebase [65], DBpedia [66], Google's Knowledge graph [2], Yago [67] and NELL [68]), that can cater to these applications, by representing information available on the web in relational format.

All knowledge graphs share common drawback of *incompleteness* and *sparsity* and hence most existing relational learning techniques focus on using observed triplets in an

incomplete graph to infer unobserved triplets for that graph [19]. Neural embedding techniques that learn vector space representations of entities and relationships have achieved remarkable success in this task. However, these techniques only focus on learning from a single graph. In addition to incompleteness property, these knowledge graphs also share a set of overlapping entities and relationships with varying information about them. This makes a compelling case to design a technique that can learn over multiple graphs and eventually aid in constructing a unified giant graph out of them. While research on learning representations over single graph has progressed rapidly in recent years [69, 2, 70, 71, 72, 73], there is a conspicuous lack of principled approach to tackle the unique challenges involved in learning across multiple graphs.

One approach to multi-graph representation learning could be to first solve graph alignment problem to merge the graphs and then use existing relational learning methods on merged graph. Unfortunately, graph alignment is an important but still unsolved problem and there exist several techniques addressing its challenges [3, 4, 5, 6] in limited settings. The key challenges for the graph alignment problem emanate from the fact that the real world data are noisy and intricate in nature. The noisy or sparse data make it difficult to learn robust alignment features, and data abundance leads to computational challenges due to the combinatorial permutations needed for alignment. These challenges are compounded in multi-relational settings due to heterogeneous nodes and edges in such graphs.

Recently, deep learning has shown significant impact in learning useful information over noisy, large-scale and heterogeneous graph data [74]. We, therefore, posit that combining graph alignment task with deep representation learning across multi-relational graphs has potential to induce a synergistic effect on both tasks. Specifically, we identify that a key component of graph alignment process—entity linkage—also plays a vital role in learning across graphs. For instance, the embeddings learned over two knowledge graphs for an actor should be closer to one another compared to the embeddings of all the other entities. Similarly, the entities that are already aligned together across the two graphs should pro-

duce better embeddings due to the shared context and data. To model this phenomenon, we propose **LinkNBed**, a novel deep learning framework that jointly performs representation learning and graph linkage task. To achieve this, we identify key challenges involved in the learning process and make the following contributions to address them:

- We propose novel and principled approach towards jointly learning entity representations and entity linkage. The novelty of our framework stems from its ability to support linkage task across heterogeneous types of entities.
- We devise a graph-independent inductive framework that learns functions to capture contextual information for entities and relations. It combines the structural and semantic information in individual graphs for joint inference in a principled manner.
- Labeled instances (specifically positive instances for linkage task) are typically very sparse and hence we design a novel multi-task loss function where entity linkage task is tackled in robust manner across various learning scenarios such as learning only with unlabeled instances or only with negative instances.
- We design an efficient training procedure to perform joint training in linear time in the number of triples. We demonstrate superior performance of our method on two datasets curated from Freebase and IMDB against State-of-the-art neural embedding methods.

3.2 Preliminaries

3.2.1 Knowledge Graph Representation

A knowledge graph $\mathcal{G}$ comprises of set of facts represented as triplets (e^s, r, e^o) denoting the relationship r between subject entity e^s and object entity e^o. Associated to this knowledge graph, we have a set of attributes that describe observed characteristics of an entity. Attributes are represented as set of key-value pairs for each entity and an attribute can have

null (missing) value for an entity. We follow *Open World Assumption - triplets not observed in knowledge graph are considered to be missing but not false.* We assume that there are no duplicate triplets or self-loops.

3.2.2 Multi-Graph Relational Learning

Definition. Given a collection of knowledge graphs $\mathcal{G}$, Multi-Graph Relational Learning refers to the the task of learning information rich representations of entities and relationships across graphs. The learned embeddings can further be used to infer new knowledge in the form of link prediction or learn new labels in the form of entity linkage. We motivate our work with the setting of two knowledge graphs where given two graphs $G_1, G_2 \in \mathcal{G}$, the task is to match an entity $e_{G_1} \in G_1$ to an entity $e_{G_2} \in G_2$ if they represent the same real-world entity. We discuss a straightforward extension of this setting to more than two graphs in Section 7.

Notations. Let X and Y represent realization of two such knowledge graphs extracted from two different sources. Let n_e^X and n_e^Y represent number of entities in X and Y respectively. Similarly, n_r^X and n_r^Y represent number of relations in X and Y. We combine triplets from both X and Y to obtain set of all observed triplets $\mathcal{D} = \{(e^s, r, e^o)_p\}_{p=1}^{P}$ where P is total number of available records across from both graphs. Let $\mathcal{E}$ and $\mathcal{R}$ be the set of all entities and all relations in $\mathcal{D}$ respectively. Let $|\mathcal{E}| = n$ and $|\mathcal{R}| = m$. In addition to $\mathcal{D}$, we also have set of linkage labels $\mathcal{L}$ for entities between X and Y. Each record in $\mathcal{L}$ is represented as triplet ($e^X \in X$, $e^Y \in Y$, $l \in \{0, 1\}$) where $l = 1$ when the entities are matched and $l = 0$ otherwise.

3.3 Proposed Method: LinkNBed

We present a novel *inductive multi-graph* relational learning framework that learns a set of aggregator functions capable of ingesting various contextual information for both entities and relationships in multi-relational graph. These functions encode the ingested structural

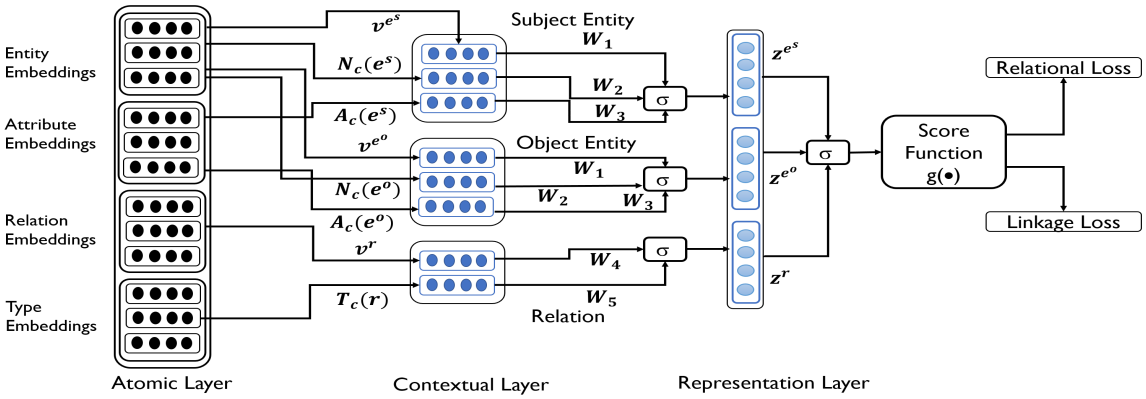


Figure 3.1: LinkNBed Architecture Overview - one step score computation for a given triplet (e^s, r, e^o). The Attribute embeddings are not simple lookups but they are learned as shown in Eq 3.3

and semantic information into low-dimensional entity and relation embeddings. Further, we use these representations to learn a relational score function that computes how two entities are likely to be connected in a particular relationship. The key idea behind this formulation is that when a triplet is observed, the relationship between the two entities can be explained using various contextual information such as local *neighborhood* features of both entities, *attribute* features of both entities and *type* information of the entities which participate in that relationship.

We outline two key insights for establishing the relationships between embeddings of the entities over multiple graphs in our framework:

Insight 1 (Embedding Similarity): If the two entities $e^X \in X$ and $e^Y \in Y$ represent the same real-world entity then their embeddings $\mathbf{e^X}$ and $\mathbf{e^Y}$ will be close to each other.

Insight 2 (Semantic Replacement): For a given triplet $t = (e^s, r, e^o) \in X$, denote $g(t)$ as the function that computes a relational score for t using entity and relation embeddings. If there exists a matching entity $e^{s'} \in Y$ for $e^s \in X$, denote $t' = (e^{s'}, r, e^o)$ obtained after replacing e^s with $e^{s'}$. In this case, $g(t) \sim g(t')$ i.e. score of triplets t and t' will be similar.

For a triplet $(e^s, r, e^o) \in \mathcal{D}$, we describe encoding mechanism of LinkNBed as three-layered architecture that computes the final output representations of $\mathbf{z}^r, \mathbf{z}^{e^s}, \mathbf{z}^{e^o}$ for the given triplet. Figure 3.1 provides an overview of LinkNBed architecture and we describe

the three steps below:

3.3.1 Atomic Layer

Entities, Relations, Types and Attributes are first encoded in its basic vector representations. We use these basic representations to derive more complex contextual embeddings further.

Entities, Relations and Types. The embedding vectors corresponding to these three components are learned as follows:

$$\mathbf{v}^{\mathbf{e^s}} = f(\mathbf{W}^{\mathbf{E}}\mathbf{e}^{\mathbf{s}}) \qquad \mathbf{v}^{\mathbf{e^o}} = f(\mathbf{W}^{\mathbf{E}}\mathbf{e}^{\mathbf{o}}) \tag{3.1}$$

$$\mathbf{v}^{\mathbf{r}} = f(\mathbf{W}^{\mathbf{R}}\mathbf{r}) \qquad \mathbf{v}^{\mathbf{t}} = f(\mathbf{W}^{\mathbf{T}}\mathbf{t}) \tag{3.2}$$

where $\mathbf{v}^{\mathbf{e^s}}$,$\mathbf{v}^{\mathbf{e^o}} \in \mathbb{R}^d$. $\mathbf{e^s}$, $\mathbf{e^o} \in \mathbb{R}^n$ are "one-hot" representations of e^s and e^o respectively. $\mathbf{v}^{\mathbf{r}} \in \mathbb{R}^k$ and $\mathbf{r} \in \mathbb{R}^m$ is "one-hot" representation of r. $\mathbf{v}^{\mathbf{t}} \in \mathbb{R}^q$ and $\mathbf{t} \in \mathbb{R}^z$ is "one-hot" representation of t . $\mathbf{W}^{\mathbf{E}} \in \mathbb{R}^{d\times n}$, $\mathbf{W}^{\mathbf{R}} \in \mathbb{R}^{k\times m}$ and $\mathbf{W}^{\mathbf{T}} \in \mathbb{R}^{q\times z}$ are the entity, relation and type embedding matrices respectively. f is a nonlinear activation function (Relu in our case). $\mathbf{W}^{\mathbf{E}}$, $\mathbf{W}^{\mathbf{R}}$ and $\mathbf{W}^{\mathbf{T}}$ can be initialized randomly or using pre-trained word embeddings or vector compositions based on name phrases of components [75].

Attributes. For a given attribute a represented as key-value pair, we use paragraph2vec [76] type of embedding network to learn attribute embedding. Specifically, we represent attribute embedding vector as:

$$\mathbf{a} = f(\mathbf{W}^{\mathbf{key}}\mathbf{a}_{\mathbf{key}} + \mathbf{W}^{\mathbf{val}}\mathbf{a}_{\mathbf{val}}) \tag{3.3}$$

where $\mathbf{a} \in \mathbb{R}^y$, $\mathbf{a}_{\mathbf{key}} \in \mathbb{R}^u$ and $\mathbf{a}_{\mathbf{val}} \in \mathbb{R}^v$. $\mathbf{W}^{\mathbf{key}} \in \mathbb{R}^{y\times u}$ and $\mathbf{W}^{\mathbf{val}} \in \mathbb{R}^{y\times v}$. $\mathbf{a}_{\mathbf{key}}$ will be "one-hot" vector and $\mathbf{a}_{\mathbf{val}}$ will be feature vector. Note that the dimensions of the embedding vectors do not necessarily need to be the same.

3.3.2 Contextual Layer

While the entity and relationship embeddings described above help to capture very generic latent features, embeddings can be further enriched to capture structural information, attribute information and type information to better explain the existence of a fact. Such information can be modeled as context of nodes and edges in the graph. To this end, we design the following canonical aggregator function that learns various contextual information by aggregating over relevant embedding vectors:

$$\mathbf{c}(z) = \text{AGG}(\{\mathbf{z}', \forall z' \in C(z)\}) \tag{3.4}$$

where $\mathbf{c}(z)$ is the vector representation of the aggregated contextual information for component z. Here, component z can be either an entity or a relation. $C(z)$ is the set of components in the context of z and $\mathbf{z}'$ correspond to the vector embeddings of those components. AGG is the aggregator function which can take many forms such Mean, Max, Pooling or more complex LSTM based aggregators. It is plausible that different components in a context may have varied impact on the component for which the embedding is being learned. To account for this, we employ a soft attention mechanism where we learn attention coefficients to weight components based on their impact before aggregating them. We modify Eq. 3.4 as:

$$\mathbf{c}(z) = \text{AGG}(\mathbf{q}(z) * \{\mathbf{z}', \forall z' \in C(z)\}) \tag{3.5}$$

where

$$\mathbf{q}(z) = \frac{\exp(\theta_z)}{\sum\limits_{z' \in C(z)} \exp(\theta_{z'})} \tag{3.6}$$

and θ_z's are the parameters of attention model.

Following contextual information is modeled in our framework:

Entity Neighborhood Context $\mathbf{N_c}(e) \in \mathbb{R}^d$. Given a triplet (e^s, r, e^o), the neighborhood context for an entity e^s will be the nodes located near e^s other than the node e^o. This will capture the effect of local neighborhood in the graph surrounding e^s that drives e^s to participate in fact (e^s, r, e^o). We use Mean as aggregator function. As there can be large number of neighbors, we collect the neighborhood set for each entity as a pre-processing step using a random walk method. Specifically, given a node e, we run k rounds of random-walks of length l following [18] and create set $\mathcal{N}(e)$ by adding all unique nodes visited across these walks. This context can be similarly computed for object entity.

Entity Attribute Context $\mathbf{A_c}(e) \in \mathbb{R}^y$. For an entity e, we collect all attribute embeddings for e obtained from Atomic Layer and learn aggregated information over them using Max operator given in Eq. 3.4.

Relation Type Context $\mathbf{T_c}(r) \in \mathbb{R}^q$. We use type context for relation embedding i.e. for a given relationship r, this context aims at capturing the effect of type of entities that have participated in this relationship. For a given triplet (e^s, r, e^o), type context for relationship r is computed by aggregation with mean over type embeddings corresponding to the context of r. Appendix C provides specific forms of contextual information.

3.3.3 Representation Layer

Having computed the atomic and contextual embeddings for a triplet (e^s, r, e^o), we obtain the final embedded representations of entities and relation in the triplet using the following formulation:

$$\mathbf{z^{e^s}} = \sigma(\underbrace{\mathbf{W_1 v^{e^s}}}_{\text{Subject Entity Embedding}} + \underbrace{\mathbf{W_2 N_c}(e^s)}_{\text{Neighborhood Context}} + \underbrace{\mathbf{W_3 A_c}(e^s))}_{\text{Subject Entity Attributes}} \tag{3.7}$$

$$\mathbf{z^{e^o}} = \sigma(\underbrace{\mathbf{W_1 v^{e^o}}}_{\text{Object Entity Embedding}} + \underbrace{\mathbf{W_2 N_c}(e^o)}_{\text{Neighborhood Context}} + \underbrace{\mathbf{W_3 A_c}(e^o))}_{\text{Object Entity Attributes}} \tag{3.8}$$

$$\mathbf{z^r} = \sigma(\underbrace{\mathbf{W_4 v^r}}_{\text{Relation Embedding}} + \underbrace{\mathbf{W_5 T_c}(r))}_{\text{Entity Type Context}} \tag{3.9}$$

where $\mathbf{W_1}, \mathbf{W_2} \in \mathbb{R}^{d \times d}$, $\mathbf{W_3} \in \mathbb{R}^{d \times y}$, $\mathbf{W_4} \in \mathbb{R}^{d \times k}$ and $\mathbf{W_5} \in \mathbb{R}^{d \times q}$. σ is nonlinear activation function – generally Tanh or Relu.

Following is the rationale for our formulation: An entity's representation can be enriched by encoding information about the local neighborhood features and attribute information associated with the entity in addition to its own latent features. Parameters $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}$ learn to capture these different aspects and map them into the entity embedding space. Similarly, a relation's representation can be enriched by encoding information about entity types that participate in that relationship in addition to its own latent features. Parameters $\mathbf{W_4}, \mathbf{W_5}$ learn to capture these aspects and map them into the relation embedding space. Further, as the ultimate goal is to jointly learn over multiple graphs, shared parameterization in our model facilitate the propagation of information across graphs thereby making it a graph-independent inductive model. The flexibility of the model stems from the ability to shrink it (to a very simple model considering atomic entity and relation embeddings only) or expand it (to a complex model by adding different contextual information) without affecting any other step in the learning procedure.

3.3.4 Relational Score Function

Having observed a triplet (e^s, r, e^o), we first use Eq. 7, 8 and 9 to compute entity and relation representations. We then use these embeddings to capture relational interaction between two entities using the following score function $g(\cdot)$:

$$g(e^s, r, e^o) = \sigma(\mathbf{z}^{r^T} \cdot (\mathbf{z}^{e^s} \odot \mathbf{z}^{e^o})) \tag{3.10}$$

where $\mathbf{z}^r, \mathbf{z}^{e^s}, \mathbf{z}^{e^o} \in \mathbb{R}^d$ are d-dimensional representations of entity and relationships as described below. σ is the nonlinear activation function and $\odot$ represent element-wise product.

3.4 Efficient Learning Procedure

3.4.1 Objective Function

The complete parameter space of the model can be given by:
$\mathbf{\Omega} = \{\{\mathbf{W_i}\}_{\mathbf{i=1}}^{\mathbf{5}}, \mathbf{W^E}, \mathbf{W^R}, \mathbf{W^{key}}, \mathbf{W^{val}}, \mathbf{W^t}, \mathbf{\Theta}\}$. To learn these parameters, we design a novel multi-task objective function that jointly trains over two graphs. As identified earlier, the goal of our model is to leverage the available linkage information across graphs for optimizing the entity and relation embeddings such that they can explain the observed triplets across the graphs. Further, we want to leverage these optimized embeddings to match entities across graphs and expand the available linkage information. To achieve this goal, we define following two different loss functions catering to each learning task and jointly optimize over them as a multi-task objective to learn model parameters:

Relational Learning Loss. This is conventional loss function used to learn knowledge graph embeddings. Specifically, given a p-th triplet $(e^s, r, e^o)_p$ from training set $\mathcal{D}$, we sample C negative samples by replacing either head or tail entity and define a contrastive

max margin function as shown in [75]:

$$L_{rel} = \sum_{c=1}^{C} \max(0, \gamma - g(e_p^s, r_p, e_p^o) + g'(e_c^s, r_p, e_p^o)) \tag{3.11}$$

where, γ is margin, e_c^s represent corrupted entity and $g'(e_c^s, r_p, e_p^o)$ represent corrupted triplet score.

Linkage Learning Loss: We design a novel loss function to leverage pairwise label set $\mathcal{L}$. Given a triplet (e_X^s, r_X, e_X^o) from knowledge graph X, we first find the entity e_Y^+ from graph Y that represent the same real-world entity as e_X^s. We then replace e_X^s with e_Y^+ and compute score $g(e_Y^+, r_X, e_X^o)$. Next, we find set of all entities E_Y^- from graph Y that has a negative label with entity e_X^s. We consider them analogous to the negative samples we generated for Eq. 3.11. We then propose the label learning loss function as:

$$L_{lab} = \sum_{z=1}^{Z} \max(0, \gamma - g(e_Y^+, r_X, e_X^o) + (g'(e_Y^-, r_X, e_X^o)_z)) \tag{3.12}$$

where, Z is the total number of negative labels for e_X. γ is margin which is usually set to 1 and $e_Y^- \in E_Y^-$ represent entity from graph Y with which entity e_X^s had a negative label. Please note that this applies symmetrically for the triplets that originate from graph Y in the overall dataset. Note that if both entities of a triplet have labels, we will include both cases when computing the loss. Eq. 3.12 is inspired by *Insight 1* and *Insight 2* defined earlier in Section 2. Given a set $\mathcal{D}$ of N observed triplets across two graphs, we define complete multi-task objective as:

$$\mathbf{L}(\mathbf{\Omega}) = \sum_{i=1}^{N} [b \cdot L_{rel} + (1 - b) \cdot L_{lab}] + \lambda \left\| \mathbf{\Omega} \right\|_2^2 \tag{3.13}$$

Algorithm 1 LinkNBed mini-batch Training

Input: Mini-batch $\mathcal{M}$, Negative Sample Size C, Negative Label Size Z, Attribute data att_data, Neighborhood data $nhbr\_data$, Type data $type\_data$, Positive Label Dict pos_dict, Negative Label Dict neg_dict
Output: Mini-batch Loss $\mathcal{L}_{\mathcal{M}}$.
$\mathcal{L}_{\mathcal{M}} = 0$
score_pos = []; score_neg = []; score_pos_lab = []; score_neg_lab = []
for $i = 0$ **to** size($\mathcal{M}$) **do**
    input_tuple = $\mathcal{M}[i] = (e^s, r, e^o)$
    sc = compute_triplet_score(e^s, r, e^o) (Eq. 3.10)
    score_pos.append(sc)
    for $j = 0$ **to** C **do**
        Select e_c^s from entity list such that $e_c^s \neq e^s$ and $e_c^s \neq e^o$ and $(e_c^s, r, e^o) \notin \mathcal{D}$
        sc_neg = compute_triplet_score(e_c^s, r, e^o)
        score_neg.append(sc_neg)
    end for
    if e^s in pos_dict **then**
        e^{s+} = positive label for e^s
        sc_pos_l = compute_triplet_score(e^{s+}, r, e^o)
        score_pos_lab.append(sc_pos_l)
    end if
    for $k = 0$ **to** Z **do**
        Select e^{s-} from neg_dict
        sc_neg_l = compute_triplet_score(e^{s-}, r, e^o)
        score_neg_lab.append(sc_neg_l)
    end for
end for
$\mathcal{L}_{\mathcal{M}}$ += compute_minibatch_loss(score_pos, score_neg, score_pos_lab, score_neg_lab) (Eq. 3.13)
Back-propagate errors and update parameters $\mathbf{\Omega}$
return $\mathcal{L}_{\mathcal{M}}$

where $\mathbf{\Omega}$ is set of all model parameters and λ is regularization hyper-parameter. b is weight hyper-parameter used to attribute importance to each task. We train with mini-batch SGD procedure (Algorithm 7) using Adam Optimizer.

Missing Positive Labels. It is expensive to obtain positive labels across multiple graphs and hence it is highly likely that many entities will not have positive labels available. For those entities, we will modify Eq. 3.12 to use the original triplet (e_X^s, r_X, e_X^o) in place of perturbed triplet $g(e_Y^+, r_X, e_X^o)$ for the positive label. The rationale here again arises from

Insight 2 wherein embeddings of two duplicate entities should be able to replace each other without affecting the score.

Training Time Complexity. Most contextual information is pre-computed and available to all training steps which leads to constant time embedding lookup for those context. But for attribute network, embedding needs to be computed for each attribute separately and hence the complexity to compute score for one triplet is $\mathcal{O}(2a)$ where a is number of attributes. Also for training, we generate C negative samples for relational loss function and use Z negative labels for label loss function. Let $k = C + Z$. Hence, the training time complexity for a set of n triplets will be $\mathcal{O}(2ak*n)$ which is linear in number of triplets with a constant factor as $ak << n$ for real world knowledge graphs. This is desirable as the number of triplets tend to be very large per graph in multi-relational settings.

Memory Complexity. We borrow notations from [19] and describe the parameter complexity of our model in terms of the number of each component and corresponding embedding dimension requirements. Let $H_a = 2 * N_e H_e + N_r H_r + N_t H_t + N_k H_k + N_v H_v$. The parameter complexity of our model is: $H_a * (H_b + 1)$. Here, N_e, N_r, N_t, N_k, N_v signify number of entities, relations, types, attribute keys and vocab size of attribute values across both datasets. Here H_b is the output dimension of the hidden layer.

3.5 Experiments

3.5.1 Datasets

We evaluate LinkNBed and baselines on two real world knowledge graphs: D-IMDB (derived from large scale IMDB data snapshot) and D-FB (derived from large scale Freebase data snapshot). Table 2.1 provides statistics for our final dataset used in the experiments. Appendix B.1 provides complete details about dataset processing.

Table 3.1: Statistics for Datasets: D-IMDB and D-FB

Dataset Name	# Entities	# Relations	# Attributes	# Entity Types	# Available Triples
D-IMDB	378207	38	23	41	143928582
D-FB	39667	146	69	324	22140475

3.5.2 Baselines

We compare the performance of our method against State-of-the-art representation learning baselines that use neural embedding techniques to learn entity and relation representation. Specifically, we consider compositional methods of RESCAL [69] as basic matrix factorization method, DISTMULT [73] as simple multiplicative model good for capturing symmetric relationships, and Complex [70], an upgrade over DISTMULT that can capture asymmetric relationships using complex valued embeddings. We also compare against translational model of STransE that combined original structured embedding with TransE and has shown State-of-art performance in benchmark testing [77]. Finally, we compare with GAKE [78], a model that captures context in entity and relationship representations.

In addition to the above State-of-art models, we analyze the effectiveness of different components of our model by comparing with various versions that use partial information. Specifically, we report results on following variants:

LinkNBed - Embed Only. Only use entity embeddings, **LinkNBed - Attr Only.** Only use Attribute Context, **LinkNBed - Nhbr Only.** Only use Neighborhood Context, **LinkNBed - Embed + Attr.** Use both Entity embeddings and Attribute Context, **LinkNBed - Embed + Nhbr.** Use both Entity embeddings and Neighbor Context and **LinkNBed - Embed All.** Use all three Contexts.

3.5.3 Evaluation Scheme

We evaluate our model using two inference tasks:

Link Prediction. Given a test triplet (e^s, r, e^o), we first score this triplet using Eq. 3.10. We then replace e^o with all other entities in the dataset and filter the resulting set of triplets as shown in [71]. We score the remaining set of perturbed triplets using Eq. 3.10. All the scored triplets are sorted based on the scores and then the rank of the ground truth triplet is used for the evaluation. We use this ranking mechanism to compute HITS@10 (predicted rank ≤ 10) and reciprocal rank ($\frac{1}{rank}$) of each test triplet. We report the mean over all test samples.

Entity Linkage. In alignment with *Insight 2*, we pose a novel evaluation scheme to perform entity linkage. Let there be two ground truth test sample triplets: $(e_X, e_Y^+, 1)$ representing a positive duplicate label and $(e_X, e_Y^-, 0)$ representing a negative duplicate label. Algorithm 6 outlines the procedure to compute linkage probability or score q ($\in [0, 1]$) for the pair (e_X, e_Y). We use $L1$ distance between the two vectors analogous to Mean Absolute Error (MAE). In lieu of hard-labeling test pairs, we use score q to compute Area Under the Precision-Recall Curve (AUPRC).

For the baselines and the unsupervised version (with no labels for entity linkage) of our model, we use second stage multilayer Neural Network as classifier for evaluating entity linkage. Appendix B.2 provides training configuration details.

3.5.4 Predictive Analysis

Link Prediction Results. We train LinkNBed model jointly across two knowledge graphs and then perform inference over individual graphs to report link prediction reports. For baselines, we train each baseline on individual graphs and use parameters specific to the graph to perform link prediction inference over each individual graph. Table 2.2 shows link prediction performance for all methods. Our model variant with attention mechanism

Algorithm 2 Entity Linkage Score Computation

Input: Test pair – $(e_X \in X, e_Y \in Y)$.
Output: Linkage Score – q.

1. Collect all triplets involving e_X from graph X and all triplets involving e_Y from graph Y into a combined set $\mathcal{O}$. Let $|\mathcal{O}| = k$.
2. Construct $S_{orig} \in \mathbb{R}^k$.
For each triplet $o \in \mathcal{O}$, compute score $g(o)$ using Eq. 3.10 and store the score in S_{orig}.
3. Create triplet set $\mathcal{O}'$ as following:
if $o \in \mathcal{O}$ contain $e^X \in X$ **then**
    Replace e^X with e^Y to create perturbed triplet o' and store it in $\mathcal{O}'$
end if
if $o \in \mathcal{O}$ contain $e^Y \in Y$ **then**
    Replace e^Y with e^X to create perturbed triplet o' and store it in $\mathcal{O}'$
end if
4. Construct $S_{repl} \in \mathbb{R}^k$.
For each triplet $o' \in \mathcal{O}'$, compute score $g(o')$ using Eq. 3.10 and store the score in S_{repl}.
5. Compute q.
Elements in S_{orig} and S_{repl} have one-one correspondence so take the mean absolute difference:
q = $|S_{orig}$ - $S_{repl}|_1$
return q

outperforms all the baselines with 4.15% improvement over single graph State-of-the-art Complex model on D-IMDB and 8.23% improvement on D-FB dataset. D-FB is more challenging dataset to learn as it has a large set of sparse relationships, types and attributes and it has an order of magnitude lesser relational evidence (number of triplets) compared to D-IMDB. Hence, LinkNBed's pronounced improvement on D-FB demonstrates the effectiveness of the model. The simplest version of LinkNBed with only entity embeddings resembles DISTMULT model with different objective function. Hence closer performance of those two models aligns with expected outcome. We observed that the Neighborhood context alone provides only marginal improvements while the model benefits more from the use of attributes. Despite being marginal, attention mechanism also improves accuracy for both datasets. Compared to the baselines which are obtained by trained and evaluated on individual graphs, our superior performance demonstrates the effectiveness of multi-graph learning.

Table 3.2: Link Prediction Results on both datasets

Method	D-IMDB-HITS10	D-IMDB-MRR	D-FB-HITS10	D-FB-MRR
RESCAL	75.3	0.592	69.99	0.147
DISTMULT	79.5	0.691	72.34	0.556
Complex	83.2	0.725	75.67	0.629
STransE	80.7	0.421	69.87	0.397
GAKE	69.5	0.114	63.22	0.093
LinkNBed-Embed Only	79.9	0.612	73.2	0.519
LinkNBed-Attr Only	82.2	0.676	74.7	0.588
LinkNBed-Nhbr Only	80.1	0.577	73.4	0.572
LinkNBed-Embed + Attr	84.2	0.673	78.39	0.606
LinkNBed-Embed + Nhbr	81.7	0.544	73.45	0.563
LinkNBed-Embed All	84.3	0.725	80.2	0.632
LinkNBed-Embed All (Attention)	**86.8**	**0.733**	**81.9**	**0.677**
Improvement (%)	**4.15**	**1.10**	**7.61**	**7.09**

Entity Linkage Results. We report entity linkage results for our method in two settings: a.) Supervised case where we train using both the objective functions. b.) Unsupervised case where we learn with only the relational loss function. The latter case resembles the baseline training where each model is trained separately on two graphs in an unsupervised manner. For performing the entity linkage in unsupervised case for all models, we first train a second stage of simple neural network classifier and then perform inference. In the supervised case, we use Algorithm 6 for performing the inference. Table 2.3 demonstrates the performance of all methods on this task. Our method significantly outperforms all the baselines with 33.86% over second best baseline in supervised case and 17.35% better performance in unsupervised case. The difference in the performance of our method in two cases demonstrate that the two training objectives are helping one another by learning across the graphs. GAKE's superior performance on this task compared to the other State-of-the-art relational baselines shows the importance of using contextual information for entity linkage. Performance of other variants of our model again demonstrate that attribute information is more helpful than neighborhood context and attention provides marginal improvements. We provide further insights with examples and detailed discussion on entity linkage task in Appendix A.

Table 3.3: Entity Linkage Results - Unsupervised case uses classifier at second step

Method	AUPRC (Supervised)	AUPRC (Unsupervised)
RESCAL	-	0.327
DISTMULT	-	0.292
Complex	-	0.359
STransE	-	0.231
GAKE	-	**0.457**
LinkNBed-Embed Only	0.376	0.304
LinkNBed-Attr Only	0.451	0.397
LinkNBed-Nhbr Only	0.388	0.322
LinkNBed-Embed + Attr	0.512	0.414
LinkNBed-Embed + Nhbr	0.429	0.356
LinkNBed-Embed All	0.686	0.512
LinkNBed-Embed All (Attention)	**0.691**	**0.553**
Improvement (%)	**33.86**	**17.35**

3.6 Related Work

3.6.1 Neural Embedding Methods for Relational Learning

Compositional Models learn representations by various composition operators on entity and relational embeddings. These models are multiplicative in nature and highly expressive but often suffer from scalability issues. Initial models include RESCAL [69] that uses a relation specific weight matrix to explain triplets via pairwise interactions of latent features, Neural Tensor Network [75], more expressive model that combines a standard NN layer with a bilinear tensor layer and [2] that employs a concatenation-projection method to project entities and relations to lower dimensional space. Later, many sophisticated models (Neural Association Model [79], HoLE [80]) have been proposed. Path based composition models [81] and contextual models GAKE [78] have been recently studied to capture more information from graphs. Recently, model like Complex [70] and Analogy [82] have demonstrated State-of-the art performance on relational learning tasks. **Translational Models** ([83], [84], [71], [85], [86], [72]) learn representation by employing translational

operators on the embeddings and optimizing based on their score. They offer an additive and efficient alternative to expensive multiplicative models. Due to their simplicity, they often loose expressive power. For a comprehensive survey of relational learning methods and empirical comparisons, we refer the readers to [19], [77], [87] and [73]. None of these methods address multi-graph relational learning and cannot be adapted to tasks like entity linkage in straightforward manner.

3.6.2 Entity Resolution in Relational Data

Entity Resolution refers to resolving entities available in knowledge graphs with entity mentions in text. [30] proposed entity disambiguation method for KB population, [31] learns entity embeddings for resolution, [32] propose a sophisticated DNN architecture for resolution, [33] proposes entity resolution across multiple social domains, [34] jointly embeds text and knowledge graph to perform resolution while [35] proposes Attention Mechanism for Collective Entity Resolution.

3.6.3 Learning across multiple graphs

Recently, learning over multiple graphs have gained traction. [3] divides a multi-relational graph into multiple homogeneous graphs and learns associations across them by employing product operator. Unlike our work, they do not learn across multiple multi-relational graphs. [36] provides logic based insights for cross learning, [4] does pairwise entity matching across multi-relational graphs and is very expensive, [37] learns embeddings to support multi-lingual learning and Big-Align [5] tackles graph alignment problem efficiently for bipartite graphs. None of these methods learn latent representations or jointly train graph alignment and learning which is the goal of our work.

3.7 Summary

We present a novel relational learning framework that learns entity and relationship embeddings across multiple graphs. The proposed representation learning framework leverage an efficient learning and inference procedure which takes into account the duplicate entities representing the same real-world entity in a multi-graph setting. We demonstrate superior accuracies on link prediction and entity linkage tasks compared to the existing approaches that are trained only on individual graphs. We believe that this work opens a new research direction in joint representation learning over multiple knowledge graphs.

Many data driven organizations such as Google and Microsoft take the approach of constructing a unified super-graph by integrating data from multiple sources. Such unification has shown to significantly help in various applications, such as search, question answering, and personal assistance. To this end, there exists a rich body of work on linking entities and relations, and conflict resolution (e.g., knowledge fusion [2]. Still, the problem remains challenging for large scale knowledge graphs and this paper proposes a deep learning solution that can play a vital role in this construction process. In real-world setting, we envision our method to be integrated in a large scale system that would include various other components for tasks like conflict resolution, active learning and human-in-loop learning to ensure quality of constructed super-graph. However, we point out that our method is not restricted to such use cases—one can readily apply our method to directly make inference over multiple graphs to support applications like question answering and conversations.

For future work, we would like to extend the current evaluation of our work from a two-graph setting to multiple graphs. A straightforward approach is to create a unified dataset out of more than two graphs by combining set of triplets as described in Section 2, and apply learning and inference on the unified graph without any major change in the methodology. Our inductive framework learns functions to encode contextual information and hence is graph independent. Alternatively, one can develop sophisticated approaches

with iterative merging and learning over pairs of graphs until exhausting all graphs in an input collection.

3.7.1 Discussion and Insights on Entity Linkage Task

Entity linkage task is novel in the space of multi-graph learning and yet has not been tackled by any existing relational learning approaches. Hence we analyze our performance on the task in more detail here. We acknowledge that baseline methods are not tailored to the task of entity linkage and hence their low performance is natural. But we observe that our model performs well even in the unsupervised scenario where essentially the linkage loss function is switched off and our model becomes a relational learning baseline. We believe that the inductive ability of our model and shared parameterization helps to capture knowledge across graphs and allows for better linkage performance. This outcome demonstrates the merit in multi-graph learning for different inference tasks. Having said that, we admit that our results are far from comparable to State-of-the-art linkage results (Das et al., 2017) and much work needs to be done to advance representation and relational learning methods to support effective entity linkage. But we note that our model works for multiple types of entities in a very heterogeneous environment with some promising results which serves as an evidence to pursue this direction for entity linkage task.

We now discuss several use-case scenarios where our model did not perform well to gain insights on what further steps can be pursued to improve over this initial model:

Han Solo with many attributes (False-negative example). Han Solo is a fictional character in Star Wars and appears in both D-IMDB and D-FB records. We have a positive label for this sample but we do not predict it correctly. Our model combines multiple components to effectively learn across graphs. Hence we investigated all the components to check for the failures. One observation we have is the mismatch in the amount of attributes across

the two datasets. Further, this is compounded by multi-value attributes. As described, we use paragraph2vec like model to learn attribute embeddings where for each attribute, we aggregate over all its values. This seems to be computing embeddings that are very noisy. As we have seen attributes are affecting the final result with high impact and hence learning very noisy attributes is not helping. Further, the mismatch in number of types is also an issue. Even after filtering the types, the difference is pretty large. Types are also included as attributes and they contribute context to relation embeddings. We believe that the skew in type difference is making the model learn bad embeddings. Specifically this happens in cases where lot of information is available like Han Solo as it lead to the scenario of abundant noisy data. With our investigation, we believe that contextual embeddings need further sophistication to handle such scenarios. Further, as we already learn relation, type and attribute embeddings in addition to entity embeddings, aligning relations, types and attributes as integral task could also be an important future direction.

Alfred Pennyworth is never the subject of matter (False-negative example). In this case, we observe a new pattern which was found in many other examples. While there are many triples available for this character in D-IMDB, very few triplets are available in D-FB. This skew in availability of data hampers the learning of deep network which ends up learning very different embeddings for two realizations. Further, we observe another patter where Alfred Pennyworth appears only as an object in all those few triplets of D-FB while it appears as both subject and object in D-IMDB. Accounting for asymmetric relationships in an explicit manner may become helpful for this scenario.

Thomas Wayne is Martha Wayne! (False-positive example). This is the case of abundance of similar contextual information as our model predicts Thomas Wayne and Martha Wayne to be same entity. Both the characters share a lot of context and hence many triples and attributes, neighborhood etc. are similar for of them eventually learning very similar

embeddings. Further as we have seen before, neighborhood has shown to be a weak context which seems to hamper the learning in this case. Finally, the key insight here is to be able to attend to the very few discriminative features for the entities in both datasets (e.g. male vs female) and hence a more sophisticated attention mechanism would help.

In addition to the above specific use cases, we would like to discuss insights on following general concepts that naturally occur when learning over multiple graphs:

- **Entity Overlap Across Graphs.** In terms of overlap, one needs to distinguish between *real* and *known* overlap between entities. For the known overlap between entities, we use that knowledge for linkage loss function L_{lab}. But our method does not need to assume either types of overlap. In case there is no real overlap, the model will learn embeddings as if they were on two separate graphs and hence will only provide marginal (if any) improvement over State-of-art embedding methods for single graphs. If there is real overlap but no known overlap (i.e., no linked entity labels), the only change is that Equation (13) will ignore the term $(1-b) \cdot L_{lab}$. Table 3 shows that in this case (corresponding to AUPRC (Unsupervised)), we are still able to learn similar embeddings for graph entities corresponding to the same real-world entity.

- **Disproportionate Evidence for entities across graphs.** While higher proportion of occurrences help to provide more evidence for training an entity embedding, the overall quality of embedding will also be affected by all other contexts and hence we expect to have varied entity-specific behavior when they occur in different proportions across two graphs

- **Ambiguity vs. Accuracy.** The effect of ambiguity on accuracy is dependent on the type of semantic differences. For example, it is observed that similar entities with major difference in attributes across graphs hurts the accuracy while the impact is not so prominent for similar entities when only their neighborhood is different.

3.7.2 Implementation Details

Additional Dataset Details

We perform light pre-processing on the dataset to remove self-loops from triples, clean the attributes to remove garbage characters and collapse CVT (Compound Value Types) entities into single triplets. Further we observe that there is big skew in the number of types between D-IMDB and D-FB. D-FB contains many non-informative type information such as $\#base.*$. We remove all such non-informative types from both datasets which retains 41 types in D-IMDB and 324 types in D-FB. This filtering does not reduce the number of entities or triples by significant number (less than 1000 entities filtered)

For comparing at scale with baselines, we further reduce dataset using similar techniques adopted in producing widely accepted FB-15K or FB-237K. Specifically, we filter relational triples such that both entities in a triple contained in our dataset must appear in more than k triples. We use $k = 50$ for D-FB and $k = 100$ for D-IMDB as D-IMDB has orders of magnitude more triples compared to D-FB in our curated datasets. We still maintain the overall ratio of the number of triples between the two datasets.

Positive and Negative Labels. We obtain 500662 positive labels using the existing links between the two datasets. Note that any entity can have only one positive label. We also generate 20 negative labels for each entity using the following method: (i) randomly select 10 entities from the other graph such that both entities belong to the same type and there exist no positive label between entities (ii) randomly select 10 entities from the other graph such that both entities belong to different types.

Training Configurations

We performed hyper-parameter grid search to obtain the best performance of our method and finally used the following configuration to obtain the reported results:

– Entity Embedding Size: 256, Relation Embedding Size=64, Attribute Embedding Size =

16, Type Embedding Size = 16, Attribute Value Embedding Size = 512. We tried multiple batch sizes with very minor difference in performance and finally used size of 2000. For hidden units per layer, we use size = 64. We used $C = 50$ negative samples and $Z = 20$ negative labels. The learning rate was initialized as 0.01 and then decayed over epochs. We ran our experiments for 5 epochs after which the training starts to convert as the dataset is very large. We use loss weights b as 0.6 and margin as 1. Further, we use $K = 50$ random walks of length $l = 3$ for each entity We used a train/test split of 60%/40% for both the triples set and labels set. For baselines, we used the implementations provided by the respective authors and performed grid search for all methods according to their requirements.

3.7.3 Contextual Information Formulations

Here we describe exact formulation of each context that we used in our work.

Neighborhood Context: Given a triplet (e^s, r, e^o), the neighborhood context for an entity e^s will be all the nodes at 1-hop distance from e^s other than the node e^o. This will capture the effect of other nodes in the graph surrounding e^s that drives e^s to participate in fact (e^s, r, e^o). Concretely, we define the neighborhood context of e^s as follows:

$$\mathbf{N_c}(e^s) = \frac{1}{n_{e'}} \sum_{\substack{e' \in \mathcal{N}(e^s) \\ e' \neq e^o}} \mathbf{v}^{\mathbf{e}'} \tag{3.14}$$

where $\mathcal{N}(e^s)$ is the set of all entities in neighborhood of e^s other than e^o. We collect the neighborhood set for each entity as a pre-processing step using a random walk method. Specifically, given a node e, we run k rounds of random-walks of length l and create the neighborhood set $\mathcal{N}(e)$ by adding all unique nodes visited across these walks.

Please note that we can also use $\max$ function in (A.1) instead of sum. $\mathbf{N_c}(e^s) \in \mathbb{R}^d$ and the context can be similarly computed for object entity.

Attribute Context. For an entity e^s, the corresponding attribute context is defined as

$$\mathbf{A_c}(e^s) = \frac{1}{n_a} \sum_{i=1}^{n_a} \mathbf{a_i^{e^s}} \tag{3.15}$$

where n_a is the number of attributes. $\mathbf{a_i^{e^s}}$ is the embedding for attribute i. $\mathbf{A_c}(e^s) \in \mathbb{R}^y$.

Type Context. We use type context mainly for relationships i.e. for a given relationship r, this context aims at capturing the effect of type of entities that have participated in this relationship. For a given triplet (e^s, r, e^o), we define type context for relationship r as:

$$\mathbf{T_c}(r) = \frac{1}{n_t^r} \sum_{i=1}^{n_t^r} \mathbf{v_i^{t'}} \tag{3.16}$$

where, n_t^r is the total number of types of entities that has participated in relationship r and $\mathbf{v_i^{t'}}$ is the type embedding that corresponds to type t. $\mathbf{T_c}(r) \in \mathbb{R}^q$.

Part II

Modeling and Learning Dynamic Network Processes

CHAPTER 4

REPRESENTATION LEARNING OVER DYNAMIC GRAPHS

Representation Learning over graph structured data has received significant attention recently due to its ubiquitous applicability. However, most advancements have been made in static graph settings while efforts for jointly learning dynamic of the graph and dynamic on the graph are still in an infant stage. Two fundamental questions arise in learning over dynamic graphs: (i) How to elegantly model dynamical processes over graphs? (ii) How to leverage such a model to effectively encode evolving graph information into low-dimensional representations? We present **DyRep** - a novel modeling framework for dynamic graphs that posits representation learning as a *latent mediation process* bridging two observed processes namely – dynamics *of* the network (realized as topological evolution) and dynamics *on* the network (realized as activities between nodes). Concretely, we propose a two-time scale deep temporal point process model that captures the interleaved dynamics of the observed processes. This model is further parameterized by a temporal-attentive representation network that encodes temporally evolving structural information into node representations which in turn drives the nonlinear evolution of the observed graph dynamics. Our unified framework is trained using an efficient unsupervised procedure and has capability to generalize over unseen nodes. We demonstrate that DyRep outperforms state-of-the-art baselines for dynamic link prediction and time prediction tasks and present extensive qualitative insights into our framework.

4.1 Introduction

Representation learning over graph structured data has emerged as a keystone machine learning task due to its ubiquitous applicability in variety of domains such as social networks, bioinformatics, natural language processing, and relational knowledge bases. Learn-

ing node representations to effectively encode high-dimensional and non-Euclidean graph information is a challenging problem but recent advances in deep learning has helped important progress towards addressing it [88, 89, 90, 91, 92, 93, 94], with majority of the approaches focusing on advancing the state-of-the-art in static graph setting. However, several domains now present highly dynamic data that exhibit complex temporal properties in addition to earlier cited challenges. For instance, social network communications, financial transaction graphs or longitudinal citation data contain fine-grained temporal information on nodes and edges that characterize the dynamic evolution of a graph and its properties over time.

These recent developments have created a conspicuous need for principled approaches to advance graph embedding techniques for dynamic graphs [95]. We focus on two pertinent questions fundamental to representation learning over dynamic graphs: (i) *What can serve as an elegant model for dynamic processes over graphs?* — A key modeling choice in existing representation learning techniques for dynamic graphs [39, 40, 28, 46, 43] assume that graph dynamics evolve as a single time scale process. In contrast to these approaches, we observe that most real-world graphs exhibit at least two distinct dynamic processes that evolve at different time scales — *Topological Evolution*: where the number of nodes and edges are expected to grow (or shrink) over time leading to structural changes in the graph; and *Node Interactions*: which relates to activities between nodes that may or may not be structurally connected. Modeling interleaved dependencies between these non-linearly evolving dynamic processes is a crucial next step for advancing the formal models of dynamic graphs.

(ii) *How can one leverage such a model to learn dynamic node representations that are effectively able to capture evolving graph information over time?* — Existing techniques in this direction can be divided into two approaches: a.) Discrete-Time Approach, where the evolution of a dynamic graph is observed as collection of static graph snapshots over time [38, 39, 40]. These approaches tend to preserve (encode) very limited structural

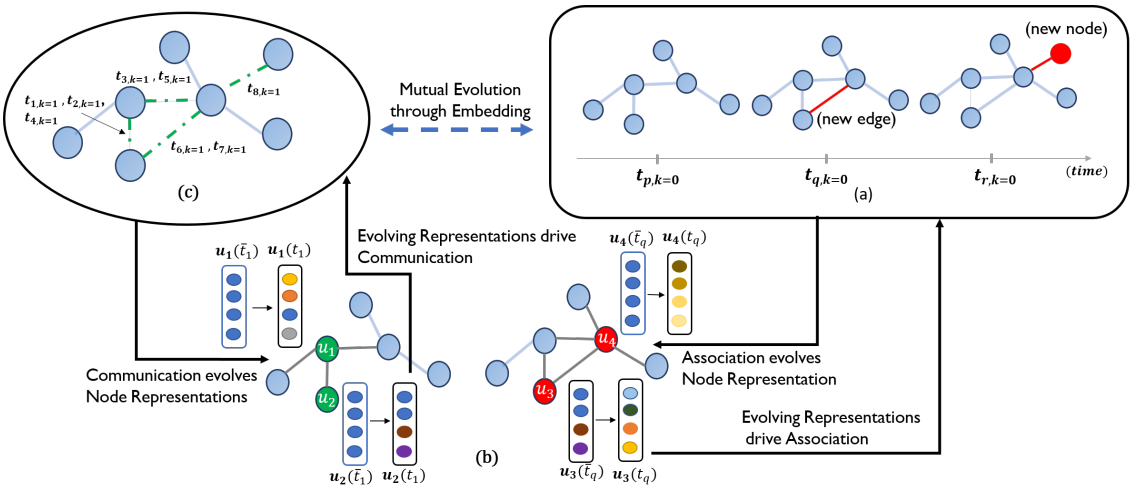


Figure 4.1: Evolution Through Mediation. **(a)** Association events (k=0) where the node or edge grows. **(c)** Communication Events (k=1) where nodes interact with each other. For both these processes, $t_{p,k=0} < (t_1, t_2, t_3, t_4, t_5)_{k=1} < t_{q,k=0} < (t_6, t_7)_{k=1} < t_{r,k=0}$. **(b)** Evolving Representations.

information and capture temporal information at a very coarse level which leads to loss of information between snapshots and lack of ability to capture fine-grained temporal dynamics. Another challenge in such approaches is the selection of appropriate aggregation granularity which is often misspecified. b.) Continuous-Time Approach, where evolution is modeled at finer time granularity in order to address the above challenges. While existing approaches have demonstrated to be very effective in specific settings, they either model simple structural and complex temporal properties in a decoupled fashion [28] or use simple temporal models (exponential family in [46]). But several domains exhibit highly nonlinear evolution of structural properties coupled with complex temporal dynamics and it remains an open problem to effectively model and learn informative representations capturing various dynamical properties of such complex systems.

As noted in [96], an important requirement to effectively learn over such dynamical systems is the ability to express the dynamical processes at different scales. We propose that any dynamic graph must be minimally expressed as a result of two fundamental processes evolving at different time scales: **Association Process** (dynamics *of* the network), that brings change in the graph structure and leads to long lasting information exchange

between nodes; and **Communication Process** (dynamics *on* the network), that relates to activities between (not necessarily connected) nodes which leads to temporary information flow between them [97, 98]. We, then, posit our goal of learning node representations as modeling a *latent mediation process* that bridges the above two observed processes such that learned representations drive the complex temporal dynamics of both processes and these processes subsequently lead to the nonlinear evolution of node representations. Further, the information propagated across the graph is governed by the temporal dynamics of communication and association histories of nodes with its neighborhood. For instance, in a social network, when a node's neighborhood grows, it changes that node's representation which in turn affects her social interactions (*association* $\rightarrow$ **embedding** $\rightarrow$ *communication*). Similarly, when node's interaction behavior changes, it affects the representation of her neighbors and herself which in turn changes the structure and strength of her connections due to link addition or deletion (*communication* $\rightarrow$ **embedding** $\rightarrow$ *association*). We call this phenomenon — ***evolution through mediation*** and illustrate it graphically in Figure 4.1.

In this work, we propose a novel representation learning framework for dynamic graphs, **DyRep**, to model interleaved evolution of two observed processes through latent mediation process expressed above and effectively learn richer node representations over time. Our framework ingests dynamic graph information in the form of association and communication events over time and updates the node representations as they appear in these events. We build a two-time scale deep temporal point process approach to capture the continuous-time fine-grained temporal dynamics of the two observed processes. We further parameterize the conditional intensity function of the temporal point process with a deep inductive representation network that learns functions to compute node representations. Finally, we couple the structural and temporal components of our framework by designing a novel *Temporal Attention Mechanism*, which induces temporal attentiveness over neighborhood nodes using the learned intensity function. This allows to capture highly interleaved and

nonlinear dynamics governing node representations over time. We design an efficient unsupervised training procedure for end-to-end training of our framework. We demonstrate consistent and significant improvement over state-of-the-art representative baselines on two real-world dynamic graphs for the tasks of dynamic link prediction and time prediction. We further present an extensive qualitative analysis through embedding visualization and ablation studies to discern the effectiveness of our framework.

4.2 Background and Preliminaries

4.2.1 Related Work

Representation Learning approaches for static graphs either perform node embedding [88, 89, 90, 91, 92, 93, 94] or sub-graph embedding [99, 100, 101] which can also utilize convolutional neural networks [102, 103, 104]. Among them, GraphSage [18] is an inductive method for learning functions to compute node representations that can be generalized to unseen nodes. Most of these approaches only work with static graphs or can model evolving graphs without temporal information. Dynamic network embedding is pursued through various techniques such as matrix factorization [38], structural properties [40], CNN-based approaches [41], deep recurrent models [28], and random walks [46]. There exists a rich body of literature on temporal modeling of dynamic networks [7], that focus on link prediction tasks but their goal is orthogonal to our work as they build task specific methods and do not focus on representation learning. Authors in [105, 106] proposed models of learning dynamic embeddings but none of them consider time at finer level and do not capture both topological evolution and interactions simultaneously. In parallel, research on deep point process models include parametric approaches to learn intensity [107, 108] using recurrent neural networks and GAN based approaches to learn intensity functions [109]. More detailed related works are provided in **Appendix F**.

4.2.2 Temporal Point Processes

Stochastic point processes [110] are random processes whose realization comprises of discrete events in time, $t_1, t_2, \ldots$. A temporal point process is one such stochastic process that can be equivalently represented as a counting process, $N(t)$, which contains the number of events up to time t. The common way to characterize temporal point processes is via the conditional intensity function $\lambda(t)$, a stochastic model of rate of happening events given the previous events. Formally, $\lambda(t)\mathrm{d}t$ is the conditional probability of observing an event in the tiny window $[t, t + \mathrm{d}t)$, $\lambda(t)\mathrm{d}t := \mathbb{P}[\text{event in } [t, t + \mathrm{d}t)|\mathcal{T}(t)] = \mathbb{E}[\mathrm{d}N(t)|\mathcal{T}(t)]$, where $\mathcal{T}(t) = t_k|t_k < t$ is history until t. Similarly, for $t > t_n$ and given history $\mathcal{T} = t_1, \ldots, t_n$, we characterize the conditional probability that no event happens during $[t_n, t)$ as $S(t|\mathcal{T}) = \exp\left(-\int_{t_n}^{t} \lambda(\tau)\,\mathrm{d}\tau\right)$, which is called survival function of the process [111]. Moreover, the conditional density that an event occurs at time t is defined as $f(t) = \lambda(t)\,S(t)$. The intensity $\lambda(t)$ is often designed to capture phenomena of interests – common forms include Poisson Process, Hawkes processes [112, 113, 114, 115], Self-Correcting Process [116]. Temporal Point Processes have previously been used to model both – dynamics on the network [117, 118, 119] and dynamics of the network [120, 12].

4.2.3 Notations and Dynamic Graph Setting

Notations. Let $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ denote graph $\mathcal{G}$ at time t, where $\mathcal{V}_t$ is the set of nodes and $\mathcal{E}_t$ is the set of edges in $\mathcal{G}_t$ and the edges are undirected. ***Event Observation –*** Both communication and association processes are realized in the form of dyadic events observed between nodes on graph $\mathcal{G}$ over a temporal window $[t_0, T]$ and ordered by time. We use the following canonical tuple representation for any type of event at time t of the form $e = (u, v, t, k)$, where u, v are the two nodes involved in an event. t represents time of the event. $k \in \{0, 1\}$ and we use $k = 0$ to signify events from the topological evolution process (association) and $k = 1$ to signify events from node interaction process (communication). Persistent edges in the graph only appear through topological events while interaction events do not contribute

them. Hence, k represents an abstraction of scale (evolution rate) associated with processes that generate topological (dynamic of the network) and interaction events (dynamic on the network) respectively. We then represent complete set of P observed events ordered by time in window $[0, T]$ as $\mathcal{O} = \{(u, v, t, k)_p\}_{p=1}^{P}$. Here, $t_p \in \mathbb{R}^+$, $0 \leq t_p \leq T$. **Appendix B** discusses a marked point process view of such an event set. ***Node Representation—*** Let $\mathbf{z}^v \in \mathbb{R}^d$ represent d-dimensional representation of node v. As the representation evolve over time, we qualify them as function of time: $\mathbf{z}^v(t)$ — the representation of node v being updated after an event involving v at time t. We use $\mathbf{z}^v(\bar{t})$ for most recently updated embedding of node v just before t.

Dynamic Graph Setting. Let $\mathcal{G}_{t_0} = (\mathcal{V}_{t_0}, \mathcal{E}_{t_0})$ be the initial snapshot of a graph at time t_0. Please note that $\mathcal{G}_{t_0}$ may be empty or it may contain an initial structure (association edges) but it will not have any communication history. Our framework observes evolution of graph as a stream of events $\mathcal{O}$ and hence any new node will always be observed as a part of such an event. This will induce a natural ordering over nodes as available from the data. As our method is inductive, we never learn node-specific representations and rather learn functions to compute node representations. In this work, we only support growth of network i.e. we only model addition of nodes and structural edges and leave deletion as future work. Further, for general description of the model, we will assume that an edge in the graph do not have types and nodes do not have attributes but we discuss the details on how to use our model to accommodate these features in Appendix B.

4.3 Proposed Method: DyRep

The key idea of DyRep is to build a unified architecture that can ingest evolving information over graphs and effectively model the *evolution through mediation* phenomenon described in Section 1. To achieve this, we design a two-time scale temporal point process model of observed processes and parameterize it with an inductive representation network which subsequently models the latent mediation process of learning node representations. The

rationale behind our framework is that the observed set of events are the realizations of the nonlinear dynamic processes governing the changes in topological structure of graph and interactions between the nodes in the graph. Now, when an event is observed between two nodes, information flows from the neighborhood of one node to the other and affects the representations of the nodes accordingly. While a communication event (interaction) only propagates local information across two nodes, an association event changes the topology and thereby has more global effect. The goal is to learn node representations that encode information evolving due to such local and global effects and further drive the dynamics of the observed events.

4.3.1 Modeling Two-Time Scale Observed Graph Dynamics

The observations over dynamic graph contain temporal point patterns of two interleaved complex processes in the form of communication and association events respectively. At any time t, the occurrence of an event, from either of these processes, is dependent on the most recent state of the graph, *i.e.*, two nodes will participate in any event based on their most current representations. Given an observed event $p = (u, v, t, k)$, we define a continuous-time deep model of temporal point process using the conditional intensity function $\lambda_k^{u,v}(t)$ that models the occurrence of event p between nodes u and v at time t:

$$\lambda_k^{u,v}(t) = f_k(g_k^{u,v}(\bar{t})) \tag{4.1}$$

where $\bar{t}$ signifies the timepoint just before current event. The inner function $g_k(\bar{t})$ computes the compatibility of the most recently updated representations of two nodes, $\mathbf{z}^u(\bar{t})$ and $\mathbf{z}^v(\bar{t})$ as follows:

$$g_k^{u,v}(\bar{t}) = \boldsymbol{\omega}_k^T \cdot [\mathbf{z}^u(\bar{t}); \mathbf{z}^v(\bar{t})] \tag{4.2}$$

[;] signifies concatenation and $\boldsymbol{\omega}_k \in \mathbb{R}^{2d}$ serves as the model parameter that learns time-scale specific compatibility. $g_k(\bar{t})$ is a function of node representations learned through a

representation network described in Section 3.2. This network parameterizes the intensity function of the point process model which serves as a unifying factor. Note that the dynamics are not two simple point processes dependent on each other, but, they are related through the mediation process and in the embedding space. Further, a well curated attention mechanism is employed to learn how the past drives future.

The choice of outer function f_k needs to account for two critical criteria: 1) Intensity needs to be positive. 2) As mentioned before, the dynamics corresponding to communication and association processes evolve at different time scales. To account for this, we use a modified version of softplus function parameterized by a dynamics parameter ψ_k to capture this *timescale* dependence:

$$f_k(x) = \psi_k \log(1 + \exp(x/\psi_k)) \tag{4.3}$$

where, $x = g(\bar{t})$ in our case and $\psi_k (> 0)$ is scalar time-scale parameter learned as part of training. ψ_k corresponds to the rate of events arising from a corresponding process. In 1D event sequences, the formulation in (4.3) corresponds to the nonlinear transfer function in [108].

4.3.2 Learning latent Mediation Process Via Temporally Attentive Representation Network

We build a deep recurrent architecture that parameterizes the intensity function in Eq. (4.1) and learns functions to compute node representations. Specifically, after an event has occurred, the representation of both the participating nodes need to be updated to capture the effect of the observed event based on the principles of:

Self-Propagation. Self-propagation can be considered as a minimal component of the dynamics governing an individual node's evolution. A node evolves in the embedded space with respect to its previous position (e.g. set of features) and not in a random fashion.

Exogenous Drive. Some exogenous force may smoothly update the node's current features during the time interval (e.g. between two global events involving that node).

Localized Embedding Propagation. Two nodes involved in an event form a temporary (communication) or a permanent (association) pathway for the information to propagate from the neighborhood of one node to the other node. This corresponds to the influence of the nodes at second-order proximity passing through the other node participating in the event (See **Appendix A** for pictorial depiction).

To realize the above processes in our setting, we first describe an example setup: Consider nodes u and v participating in any type of event at time t. Let $\mathcal{N}_u$ and $\mathcal{N}_v$ denote the neighborhood of nodes u and v respectively. We discuss two key points here: 1) Node u serves as a bridge passing information from $\mathcal{N}_u$ to node v and hence v receives the information in an aggregated form through u. 2) While each neighbor of u passes its information to v, the information that node u relays is governed by an aggregate function parametrized by u's communication and association history with its neighbors.

With this setup, for any event at time t, we update the embeddings for both nodes involved in the event using a recurrent architecture. Specifically, for p-th event of node v, we evolve $\mathbf{z}^v$ as:

$$\mathbf{z}^v(t_p) = \sigma(\underbrace{\mathbf{W}^{struct}\mathbf{h}^u_{struct}(\bar{t_p})}_{\text{Localized Embedding Propagation}} + \underbrace{\mathbf{W}^{rec}\mathbf{z}^v(\bar{t^v_p})}_{\text{Self-Propagation}} + \underbrace{\mathbf{W}^t(t_p - \bar{t^v_p})}_{\text{Exogenous Drive}}), \tag{4.4}$$

where, $\mathbf{h}^u_{struct} \in \mathbb{R}^d$ is the output representation vectors obtained from aggregator function on node u's neighborhood and $\mathbf{z}^v(\bar{t^v_p})$ is the recurrent state obtained from the previous representation of node v. t_p is time point of current event, $\bar{t_p}$ signifies the timepoint just before current event and $\bar{t^v_p}$ represent time point of previous event for node v. $\mathbf{z}^v(\bar{t^v_p} = 0)$, the initial representation of a node v may be initialized either using input node features from dataset or random vector as per the setting. Eq. 4 is a neural network based functional form parameterized by $\mathbf{W}^{struct}, \mathbf{W}^{rec} \in \mathbb{R}^{d\times d}$ and $\mathbf{W}^t \in \mathbb{R}^d$ that govern the aggregate effect of all

the three inputs (graph structure, previous embedding and exogenous feature) respectively to compute representations. The above formulation is inductive (supports unseen nodes) and flexible (supports node and edge types) as discussed in **Appendix B**.

Temporally Attentive Aggregation

The Localized Embedding Propagation principle above captures rich structural properties based on neighborhood structure which is a key to any representation learning task over graphs. However, for a given node, not all of its neighbors are uniformly important and hence it becomes extremely important to capture information from each neighbor in some weighted fashion. Recently proposed attention mechanisms have shown great success in dealing with variable sized inputs, focusing on the most relevant parts of the input to make decisions. However, existing approaches consider attention as a static quantity. In dynamic graphs, changing neighborhood structure and interaction activities between nodes evolves importance of each neighbor to a node over time, thereby making attention itself a temporally evolving quantity. Further this quantity is dependent on the temporal history of association and communication of neighboring nodes through evolving representations. To this end, we propose a novel *Temporal Point Process based Attention Mechanism* that uses temporal information to compute the attention coefficient for a structural edge between nodes. These coefficient are then used to compute the aggregate quantity ($\mathbf{h}_{\mathbf{struct}}$) required for embedding propagation.

Let $\mathbf{A}(t) \in \mathbb{R}^{n \times n}$ be the adjacency matrix for graph $\mathcal{G}_t$ at time t. Let $\mathcal{S}(t) \in \mathbb{R}^{n \times n}$ be a stochastic matrix capturing the strength between pair of vertices at time t. One can consider $\mathcal{S}$ as a selection matrix that induces a ***natural selection*** process for a node – it would tend to communicate more with other nodes that it wants to associate with or has recently associated with. And it would want to attend less to non-interesting nodes. We start with following implication required for the construction of $\mathbf{h}^u_{struct}$ in (4.4): For any two nodes u and v at time t, $\mathcal{S}_{uv}(t) \in [0, 1]$ if $\mathbf{A}_{uv}(t) = 1$ and $\mathcal{S}_{uv}(t) = 0$ if $\mathbf{A}_{uv}(t) = 0$.

Denote $\mathcal{N}_u(t) = \{i : \mathbf{A}_{iu}(t) = 1\}$ as the 1-hop neighborhood of node u at time t.

To formally capture the difference in the influence of different neighbors, we propose a novel *conditional intensity based attention layer* that uses the matrix $\mathcal{S}$ to induce a shared attention mechanism to compute attention coefficients over neighborhood. Specifically, we perform *localized attention* for a given node u and compute the coefficients pertaining to the 1-hop neighbors i of node u as: $q_{ui}(t) = \frac{\exp(\mathcal{S}_{ui}(\bar{t}))}{\sum_{i' \in \mathcal{N}_u(t)} \exp(\mathcal{S}_{ui'}(\bar{t}))}$, where q_{ui} signifies the attention weight for the neighbor i at time t and hence it is a temporally evolving quantity. These attention coefficients are then used to compute the aggregate information $\mathbf{h}^u_{struct}(\bar{t})$ for node u by employing an attended aggregation mechanism across neighbors as follows: $\mathbf{h}^u_{struct}(\bar{t}) = \max\left(\{\sigma\left(q_{ui}(t) \cdot \mathbf{h}^i(\bar{t})\right), \forall i \in \mathcal{N}_u(\bar{t})\}\right)$, where, $\mathbf{h}^i(\bar{t}) = \mathbf{W}^h \mathbf{z}^i(\bar{t}) + \mathbf{b}^h$ and $\mathbf{W}^h \in \mathbb{R}^{d \times d}$ and $\mathbf{b}^h \in \mathbb{R}^d$ are parameters governing the information propagated by each neighbor of u. $\mathbf{z}^i(\bar{t}) \in \mathbb{R}^d$ is the most recent embedding for node i. The use of $\max$ operator is inspired from learning on general point sets [121]. By applying max-pooling operator element-wise, the model effectively captures different aspects of the neighborhood. We found $\max$ to work slightly better as it considers temporal aspect of neighborhood which would be amortized if $mean$ is used instead.

Connection to Neural Attention over Graphs. Our proposed temporal attention layer shares the motivation of recently proposed Graph Attention Networks (GAT) [122] and Gated Attention Networks (GaAN) [123] in the spirit of applying non-uniform attention over neighborhood. Both GAT and GaAN have demonstrated significant success in static graph setting. GAT advances GraphSage [18] by employing multi-head non-uniform attention over neighborhood and GaAN advances GAT by applying different weights to different heads in the multi-head attention formulation. The key innovation in our model is the parameterization of attention mechanism by a point process based temporal quantity $\mathcal{S}$ that is evolving and drives the impact that each neighbor has on the given node. Further, unlike static methods, we use these attention coefficients as input to the aggregator function

Algorithm 3 Update Algorithm for $\mathcal{S}$ and $\mathbf{A}$

Input: Event record $o = (u, v, t, k)$, Event Intensity $\lambda_k^{u,v}(t)$ computed in (4.1), most recently updated $\mathbf{A}(\bar{t})$ and $\mathcal{S}(\bar{t})$. **Output:** $\mathbf{A}(t)$ and $\mathcal{S}(t)$

1. Update $\mathbf{A} : \mathbf{A}(t) = \mathbf{A}(\bar{t})$
if $k = 0$ **then** $\mathbf{A}_{uv}(t) = \mathbf{A}_{vu}(t) = 1$ ▷ *Association event*

2. Update $\mathcal{S} : \mathcal{S}(t) = \mathcal{S}(\bar{t})$
if $k = 1$ and $\mathbf{A}_{uv}(t) = 0$ **return** $\mathcal{S}(t), \mathbf{A}(t)$ ▷ *Communication event, no Association exists*
for $j \in \{u, v\}$ **do**
  $b = \frac{1}{|\mathcal{N}_j(t)|}$ where $|\mathcal{N}_j(t)|$ is the size of $\mathcal{N}_j(t) = \{i : \mathbf{A}_{ij}(t) = 1\}$
  $\mathbf{y} \leftarrow \mathcal{S}_j(t)$
  if $k = 1$ and $\mathbf{A}_{uv}(t) = 1$ **then** ▷ *Communication event, Association exists*
    $\mathbf{y}_i = b + \lambda_k^{ji}(t)$ where i is the other node involved in the event. ▷ λ *from Eq. 2*
  else if $k = 0$ and $\mathbf{A}_{uv}(t) = 0$ **then** ▷ *Association event*
    $b' = \frac{1}{|\mathcal{N}_j(\bar{t})|}$ where $|\mathcal{N}_j(\bar{t})|$ is the size of $\mathcal{N}_j(\bar{t}) = \{i : \mathbf{A}_{ij}(\bar{t}) = 1\}$
    $x = b' - b$
    $\mathbf{y}_i = b + \lambda_k^{ji}(t)$ where i is the other node involved in the event ▷ λ *from Eq. 2*
    $\mathbf{y}_w = \mathbf{y}_w - x; \ \ \forall w \neq i, \ \mathbf{y}_w \neq 0$
  end if
  Normalize $\mathbf{y}$ and set $\mathcal{S}_j(t) \leftarrow \mathbf{y}$
end for
return $\mathcal{S}(t), \mathbf{A}(t)$

for computing the temporal-structural effect of neighborhood. Finally, static methods use multi-head attention to stabilize learning by capturing multiple representation spaces but this is an inherent property in our layer as representations and event intensities update over time and hence new events help capture multiple representation spaces.

Construction and Update of $\mathcal{S}$. We construct a single stochastic matrix $\mathcal{S}$ (used to parameterize attention in the earlier section) to capture complex temporal information. At the initial timepoint $t = t_0$, we construct $\mathcal{S}(t_0)$ directly from $\mathbf{A}(t_0)$. Specifically, for a given node v, we initialize the elements of corresponding row vector $\mathcal{S}_v(t_0)$ as: $\mathcal{S}_{vu}(t_0) = 0$ if ($v = u$ or $\mathbf{A}_{vu}(t_0) = 0$) and $\mathcal{S}_{vu}(t_0) = \frac{1}{|\mathcal{N}_v(t_0)|}$ if $\mathcal{N}_v(t_0) = \{u : \mathbf{A}_{uv}(t_0) = 1\}$.

After observing an event $o = (u, v, t, k)$ at time $t > t_0$, we make updates to $\mathbf{A}$ and $\mathcal{S}$ as per the observation of k. Specifically, $\mathbf{A}$ only gets updated for association events (k=0,

change in structure). Note that $\mathcal{S}$ is parameter for a structural temporal attention which means temporal attention is only applied on structural neighborhood of a node. Hence, the values of $\mathcal{S}$ are only updated/active in two scenarios: a) the current event is an interaction between nodes which already has structural edge ($\mathbf{A}_{uv}(t) = 1$ and $k = 1$) and b) the current event is an association event ($k = 0$). Given a neighborhood of node u, b represents background (base) attention for each edge which is uniform attention based on neighborhood size. Whenever an event involving u occurs, this attention changes in following ways: For case (a), the attention value for corresponding $\mathcal{S}$ entries are updated using the intensity of the event. For case (b), repeat same as (a) but also adjust the background attention (by $b - b'$, b and b' being the new and old background attention respectively) for edge with other neighbors as the neighborhood size grows in this case. From mathematical viewpoint, this update resembles a standard temporal point process formulation where the term coming from b serves as background attention while λ can be viewed as endogenous intensity based attention. Algorithm 7 outlines complete update scenarios. In the directed graph case, updates to $\mathbf{A}$ will not be symmetric, which will subsequently affect the neighborhood structure and attention flow for a node. **Appendix A** provides a pictorial depiction of the complete DyRep framework discussed in this section. We provide an extensive ablation study in **Appendix C** that can help discern the contribution of all the above components in achieving our goal.

4.4 Efficient Learning Procedure

The complete parameter space for the current model is $\mathbf{\Omega} = \{\mathbf{W}^{struct}, \mathbf{W}^{rec}, \mathbf{W}^{t}, \mathbf{W}^{h}, \mathbf{b}^{h}, \{\boldsymbol{\omega}_k\}_{k=0,1}, \{\psi_k\}_{k=0,1}\}$. For a set $\mathcal{O}$ of P observed events, we learn these parameters by minimizing the negative log likelihood: $\mathcal{L} = -\sum_{p=1}^{P} \log\left(\lambda_p(t)\right) + \int_0^T \Lambda(\tau) d\tau$, where $\lambda_p(t) = \lambda_{k_p}^{u_p, v_p}(t)$ represent the intensity of event at time t and $\Lambda(\tau) = \sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{k \in \{0,1\}} \lambda_k^{u,v}(\tau)$ represent total survival probability for events that do not happen. While it is intractable (will require $\mathcal{O}(n^2 k)$ time) and unnecessary to compute the integral in the log-likelihood

equation for all possible non-events in a stochastic setting, we can locally optimize $\mathcal{L}$ using mini-batch stochastic gradient descent where we estimate the integral using novel sampling technique. Algorithm 6 in **Appendix H** adopts a simple variant of Monte Carlo trick to compute the survival term of log-likelihood equation. Specifically, in each mini-batch, we sample non-events instead of considering all pairs of non-events (which can be millions). Let m be the mini-batch size and N be the number of samples. The complexity of Algorithm 6 will then be $\mathcal{O}(2mkN)$ for the batch where the factor of 2 accounts for the update happening for two nodes per event which demonstrates linear scalability in number of events which is desired to tackle web-scale dynamic networks [124]. The overall training procedure is adopted from [28] where the Backpropagation Through Time (BPTT) training is conducted over a global sequence, thereby maintaining the dependencies between events across sequences while avoiding gradient related issues. Implementation details are left to **Appendix G**.

4.5 Experiments

4.5.1 Datasets

We evaluate DyRep and baselines on two real world datasets: **Social Evolution Dataset** released by MIT Human Dynamics Lab — #nodes: 83, #Initial Associations: 376, #Final Associations: 791, #Communications: 2016339 and Clustering Coefficient: 0.548. **Github Dataset** available at Github Archive — #nodes: 12328, #Initial Associations: 70640, #Final Associations: 166565, #Communications: 604649 and Clustering Coefficient: 0.087. These datasets cover a range of configurations as Social Dataset is a small network with high clustering coefficient and over 2M events. In contrast, Github dataset forms a large network with low clustering coefficient and sparse events thus allowing us to test the robustness of our model. Further, Github dataset contains several unseen nodes which were never encountered during training.

4.5.2 Tasks and Metrics

We study the effectiveness of DyRep by evaluating our model on tasks of dynamic link prediction and event time prediction tasks:

Dynamic Link Prediction. When any two nodes in a graph has increased rate of interaction events, they are more likely to get involved in further interactions and eventually these interactions may lead to the formation of structural link between them. Similarly, formation of the structural link may lead to increased likelihood of interactions between newly connected nodes. To understand, how well our model captures these phenomenon, we ask questions like: Which is the most likely node u that would undergo an event with a given node v governed by dynamics k at time t? The conditional density of such and event at time t can be computed: $f_k^{u,v}(t) = \lambda_k^{u,v}(t) \cdot \exp\left(\int_{\bar{t}}^{t} \lambda(s)ds\right)$, where $\bar{t}$ is the time of the most recent event on either dimension u or v. We use this conditional density to find most likely node.

For a given test record (u, v, t, k), we replace v with other entities in the graph and compute the density as above. We then rank all the entities in descending order of the density and report the rank of the ground truth entity. Please note that the latest embeddings of the nodes update even during the test while the parameters of the model remaining fixed. Hence, when ranking the entities, we remove any entities that creates a pair already seen in the test. We report Mean Average Rank (MAR) and HITS(@10) metric for dynamic link prediction.

Event Time Prediction. This is a relatively novel application where the aim is to compute the next time point when a particular type of event (structural or interaction) can occur. Given a pair of nodes (u, v) and event type k at time t, we use the above density formulation to compute conditional density at time t. The next time point $\hat{t}$ for the event can then be computed as: $\hat{t} = \int_t^\infty t f_k^{u,v}(t)dt$ where the integral does not have an analytic form and hence we estimate it using Monte Carlo trick. For a given test record (u, v, t, k), we compute the next time this communication event may occur and report Mean Absolute

Table 4.1: Comparison of DyRep with state-of-the-art approaches

Key Properties	DyRep (Our Method)	Know-Evolve (Dynamic)	DynGem (Dynamic)	GraphSage (Static)	GAT (Static)
Models Association	✓	X	✓	✓	✓
Models Communication	✓	✓	X	X	X
Models Time	✓	✓	X	X	X
Learns Representation	✓	✓	✓	✓	✓
Predicts Time	✓	✓	X	X	X
Graph Information	2nd-order Neighborhood	Single Edge	1st and 2nd-order Neighborhood	2nd-order Neighborhood	1st-order Neighborhood
Attention Mechanism	Temporal Point Process (Non-Uniform)	None	None	Sampling (Uniform)	Multi-head (Non-Uniform)
Learning	Unsupervised	Unsupervised	Semi-Supervised	Unsupervised	Supervised

Error (MAE) against the ground truth.

4.5.3 Baselines

Dynamic Link Prediction. We compare the performance of our model against multiple representation learning baselines, four of which has capability to model evolving graphs. Specifically, we compare with **Know-Evolve** [28]— a state-of-the-art model for multi-relational dynamic graphs where each edge has time-stamp and type (communication events), **DynGem** [39]—divides timeline into discrete time points and learns embedding for the graph snapshots at these time points. **DynTrd** [40] focuses on specific structure of triad to model how close triads are formed from open triads in dynamic networks. **GraphSage** [18]— an inductive representation learning method that learns sample and aggregation functions to learn representations instead of training for individual node. **Node2Vec** [89]—simple transductive baseline to learn graph embeddings over static graphs. Table 4.1 provides qualitative comparison between state-of-the-art methods and our framework. In our experiments, we compare with GraphSage instead of GAT as we share the unsupervised setting with GraphSage while GAT is designed for supervised learning. In Appendix A (Ablation studies), we show results on one version where we only update attention based on Association events which is temporal analogous to GAT.

Event Time Prediction. We compare our model against (i) Know-Evolve which has the ability to predict time in a multi-relational dynamic graphs (II) Multi-dimensional Hawkes

Process (MHP) [42] model where all events in graph are considered as dyadic.

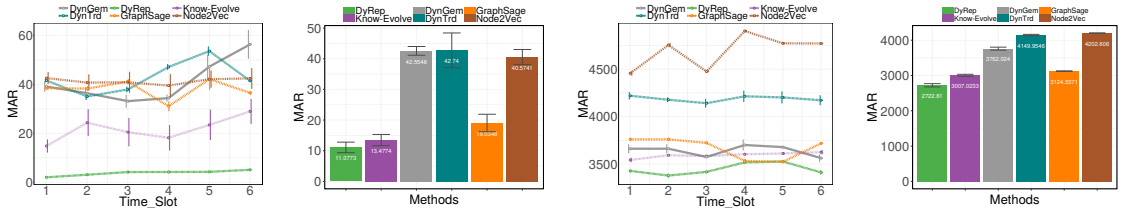
4.5.4 Evaluation Scheme

We divide our test sets into $n(= 6)$ slots based on time and report the performance for each time slot, thus providing comprehensive temporal evaluation of different methods. This method of reporting is expected to provide fine-grained insights on how various methods perform over time as they move farther from the learned training history. For dynamic baselines that do not explicitly model time (DynGem, DynTrd, GraphSage) and static baselines (Node2Vec), we adopt a sliding window training approach with warm-start method where we learn on initial train set and test for the first slot. Then we add the data from first slot in the train set and remove equal amount of data from start of train set and retrain the model using the embeddings from previous train.

4.5.5 Experimental Results

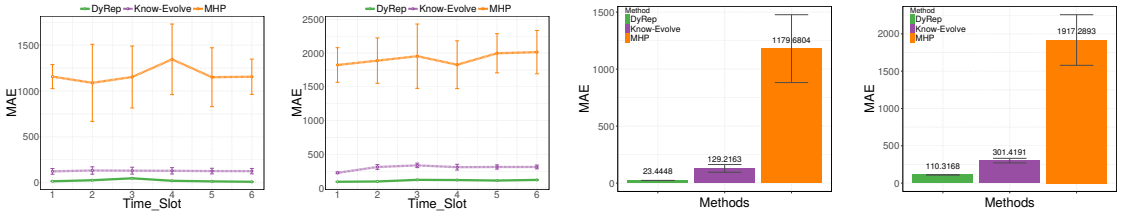
Communication Event Prediction Performance. We first consider the task of predicting communication events between nodes which may or may not have a permanent edge (association) between them. Figure 4.2 **(a-b)** shows corresponding results.

Social Evolution. Our method significantly and consistently outperforms all the baselines on both metrics. While the performance of our method drops a little over time, it is expected due to the temporal recency affect on node's evolution. Know-Evolve can capture event dynamics well and shows consistently better rank than others but its performance deteriorates significantly in HITS@10 metric over time. We conjecture that features learned through edge-level modeling limits the predictive capacity of the method over time. The inability of DynGem (snapshot based dynamic), DynTrd and GraphSage (inductive) to significantly outperform Node2vec (transductive static baseline) demonstrate that discrete time snapshot based models fail to capture fine-grained dynamics of communication events.



(a) Social (Communication) (b) Social (Association) (c) Github (Communication) (c) Github (Association)

Figure 4.2: Dynamic Link Prediction Performance for **(a-b)** Social Evolution Dataset **(c-d)** Github Dataset. We report HITS@10 results and zoomed versions in **Appendix E**. Best viewed in pdf.

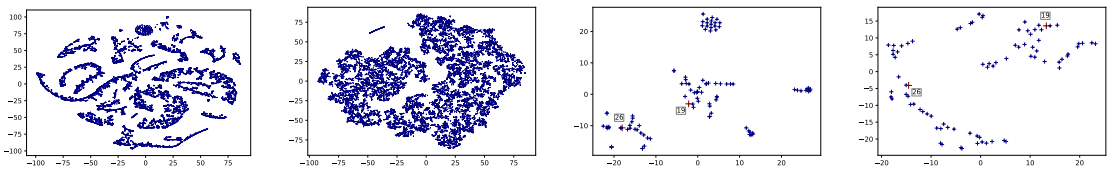


(a) Social Evolution (b) Github (c) Social Evolution (d) Github

Figure 4.3: Time Prediction Performance (unit is hrs). Figure best viewed in pdf or colored print.

Github dataset. We demonstrate comparable performance with both Know-Evolve and GraphSage on Rank metric. We would like to note that overall performance for all methods on rank metric is low. As we reported earlier, Github dataset is very sparse with very low clustering coefficient which makes it a challenging dataset to learn. It is expected that for a large number of nodes with no communication history, most of the methods will show comparable performance but our method outperforms all others when there is some history available. This is demonstrated by our significantly better performance for HITS@10 metric where we are able to do highly accurate prediction for nodes where we learn better history. This can also be attributed to our model's ability to capture the effect of evolving topology which is missed by Know-Evolve. Finally, we do not see significant decrease in performance of any method over time in this case which can again be attributed to roughly uniform distribution of nodes with no communication history across time slots.

Association Event Prediction Performance. Association events are not available for all time slots so Figure 4.2 **(c-d)** report the aggregate number for this task. For both the datasets, our model significantly outperforms the baselines for this task. Specifically, our



(a) DyRep Embeddings (b) GraphSage Embeddings (c) DyRep Embeddings (d) GraphSage Embeddings

Figure 4.4: tSNE for learned embeddings after training. Figure best viewed in color.

model's strong performance on HITS@10 metric across both datasets demonstrates its robustness in accurate learning from various properties of data. On Social evolution dataset, the number of association events are very small (only 485) and hence our strong performance shows that the model is able to capture the influence of communication events on the association events through the learned representations (mediation). On the Github dataset, the network grows through new nodes and our model's strong performance across both metric demonstrates its inductive ability to generalize across new nodes across time. An interesting observation was poor performance of DynTrd which seems to be due to its objective to complete triangles. Github dataset is very sparse and has very few possibilities for triadic closure.

Time Prediction Performance. Figure 4.3 demonstrates consistently better performance than state-of-the-art baseline for event time prediction on both datasets. While Know-Evolve models both processes as two different relations between entities, it does not explicitly capture the variance in the time scales of two processes. Further, Know-Evolve does not consider influence of neighborhood which may lead to capturing weaker temporal-structural dynamics across the graph. MHP uses specific parametric intensity function which fails to account for intricate dependencies across graph.

Qualitative Performance. We conducted a series of qualitative analysis to understand the discriminative power of evolving embeddings learned by DyRep. We compare our embeddings against GraphSage embeddings as it is state-of-the-art embedding method that is also inductive. Figure 4.4 **(a-b)** shows the tSNE embeddings learned by Dyrep (left) and GraphSage (right) respectively. The visualization demonstrates that DyRep embeddings

have more discriminative power as it can effectively capture the distinctive and evolving structural features over time as aligned with empirical evidence. Figure 4.4 **(c-d)** shows use case of two associated nodes (19 and 26) that has persistent edge but less communication for above two methods. DyRep keeps the embeddings nearby although not in same cluster (cos. dist. - 0.649) which demonstrates its ability to learn the association and less communication dynamics between two nodes. For GraphSage the embeddings are on opposite ends of cluster with (cos. dist. - 1.964). We provide more analysis in **Appendix D**.

4.6 Summary

We introduced a novel modeling framework for dynamic graphs that effectively and efficiently learns node representations by posing representation learning as latent mediation process bridging dynamic processes of topological evolution and node interactions. We proposed a deep temporal point process model parameterized by temporally attentive representation network that models these complex and nonlinearly evolving dynamic processes and learns to encode structural-temporal information over graph into low dimensional representations. Our superior evaluation performance demonstrates the effectiveness of our approach compared to state-of-the-art methods. We present this work as the first generic and unified representation learning framework that adopts a novel modeling paradigm for dynamic graphs and support wide range of dynamic graph characteristics which can potentially have many exciting adaptations. As a part of our framework, we also propose a novel temporal point process based attention mechanism that can attend over neighborhood based on the history of communications and association events in the graph. Currently, DyRep does not support network shrinkage due to following reasons: (i) It is difficult to procure data with fine grained deletion time stamps and (ii) The temporal point process model requires more sophistication to support deletion. For example, one can augment the model with a survival process formulation to account for lack of node/edge at future time. Another interesting future direction could be to support encoding higher order dynamic structures.

CHAPTER 5

TEMPORAL REASONING OVER DYNAMIC KNOWLEDGE

The availability of large scale event data with time stamps has given rise to *dynamically evolving* knowledge graphs that contain temporal information for each edge. Reasoning over time in such dynamic knowledge graphs is not yet well understood. To this end, we present **Know-Evolve**, a novel deep evolutionary knowledge network that learns non-linearly evolving entity representations over time. The occurrence of a fact (edge) is modeled as a multivariate point process whose intensity function is modulated by the score for that fact computed based on the learned entity embeddings. We demonstrate significantly improved performance over various relational learning approaches on two large scale real-world datasets. Further, our method effectively predicts occurrence or recurrence time of a fact which is novel compared to prior reasoning approaches in multi-relational setting.

5.1 Introduction

Reasoning is a key concept in artificial intelligence. A host of applications such as search engines, question-answering systems, conversational dialogue systems, and social networks require reasoning over underlying structured knowledge. Effective representation and learning over such knowledge has come to the fore as a very important task. In particular, Knowledge Graphs have gained much attention as an important model for studying complex multi-relational settings. Traditionally, knowledge graphs are considered to be static snapshot of multi-relational data. However, recent availability of large amount of event based interaction data that exhibits complex temporal dynamics in addition to its multi-relational nature has created the need for approaches that can characterize and reason over temporally evolving systems. For instance, GDELT [125] and ICEWS [126] are two popular event based data repository that contains evolving knowledge about entity

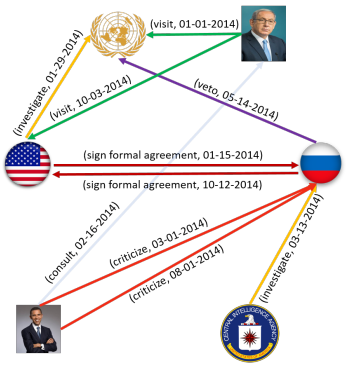


Figure 5.1: Sample temporal knowledge subgraph between persons, organizations and countries.

interactions across the globe.

Thus traditional knowledge graphs need to be augmented into *Temporal Knowledge Graphs*, where facts occur, recur or evolve over time in these graphs, and each edge in the graphs have temporal information associated with it. Figure 5.1 shows a subgraph snapshot of such temporal knowledge graph. Static knowledge graphs suffer from incompleteness resulting in their limited reasoning ability. Most work on static graphs have therefore focussed on advancing entity-relationship representation learning to infer missing facts based on available knowledge. But these methods lack ability to use rich temporal dynamics available in underlying data represented by temporal knowledge graphs.

Effectively capturing temporal dependencies across facts in addition to the relational (structural) dependencies can help improve the understanding on behavior of entities and how they contribute to generation of facts over time. For example, one can precisely answer questions like:

- **Object prediction**. (Who) will Donald Trump mention next?
- **Subject prediction**. (Which country) will provide material support to US next month?
- **Time prediction**. (When) will Bob visit Burger King?

"People (entities) change over time and so do relationships." When two entities forge a relationship, the newly formed edge drives their preferences and behavior. This change

is effected by combination of their own historical factors (**temporal evolution**) and their compatibility with the historical factors of the other entity (**mutual evolution**).

For instance, if two countries have tense relationships, they are more likely to engage in conflicts. On the other hand, two countries forging an alliance are most likely to take confrontational stands against enemies of each other. Finally, time plays a vital role in this process. A country that was once peaceful may not have same characteristics 10 years in future due to various facts (events) that may occur during that period. Being able to capture this temporal and evolutionary effects can help us reason better about future relationship of an entity. We term this combined phenomenon of evolving entities and their dynamically changing relationships over time as **"knowledge evolution"**.

In this paper, we propose an elegant framework to model knowledge evolution and reason over complex non-linear interactions between entities in a multi-relational setting. The key idea of our work is to model the occurrence of a fact as multidimensional temporal point process whose conditional intensity function is modulated by the relationship score for that fact. The relationship score further depends on the dynamically evolving entity embeddings. Specifically, our work makes the following contributions:

- We propose a novel deep learning architecture that evolves over time based on availability of new facts. The dynamically evolving network will ingest the incoming new facts, learn from them and update the embeddings of involved entities based on their recent relationships and temporal behavior.

- Besides predicting the occurrence of a fact, our architecture has ability to predict time when the fact may potentially occur which is not possible by any prior relational learning approaches to the best of our knowledge.

- Our model supports *Open World Assumption* as missing links are not considered to be false and may potentially occur in future. It further supports prediction over unseen entities due to its novel dynamic embedding process.

- The large-scale experiments on two real world datasets show that our framework has consistently and significantly better performance for link prediction than state-of-arts that do not account for temporal and evolving non-linear dynamics.

- Our work aims to introduce the use of powerful mathematical tool of temporal point process framework for temporal reasoning over dynamically evolving knowledge graphs. It has potential to open a new research direction in reasoning over time for various multi-relational settings with underlying spatio-temporal dynamics.

5.2 Preliminaries

5.2.1 Temporal Point Process

A temporal point process [127] is a random process whose realization consists of a list of events localized in time, $\{t_i\}$ with $t_i \in \mathbb{R}^+$. Equivalently, a given temporal point process can be represented as a counting process, $N(t)$, which records the number of events before time t.

An important way to characterize temporal point processes is via the conditional intensity function $\lambda(t)$, a stochastic model for the time of the next event given all the previous events. Formally, $\lambda(t)\mathrm{d}t$ is the conditional probability of observing an event in a small window $[t, t + \mathrm{d}t)$ given the history $\mathcal{T}(t) := \{t_k | t_k < t\}$ up to t, *i.e.*,

$$\begin{aligned} \lambda(t)\mathrm{d}t &:= \mathbb{P}\left\{\text{event in } [t, t + \mathrm{d}t) | \mathcal{T}(t)\right\} \\ &= \mathbb{E}[\mathrm{d}N(t) | \mathcal{T}(t)] \end{aligned} \tag{5.1}$$

where one typically assumes that only one event can happen in a small window of size $\mathrm{d}t$, *i.e.*, $\mathrm{d}N(t) \in \{0, 1\}$.

From the survival analysis theory [111], given the history $\mathcal{T} = \{t_1, \ldots, t_n\}$, for any $t > t_n$, we characterize the conditional probability that no event happens during $[t_n, t)$ as $S(t|\mathcal{T}) = \exp\big(-\int_{t_n}^{t} \lambda(\tau)\,\mathrm{d}\tau\big)$. Moreover, the conditional density that an event occurs at

time t is defined as :

$$f(t) = \lambda(t)\, S(t) \tag{5.2}$$

The functional form of the intensity $\lambda(t)$ is often designed to capture the phenomena of interests. Some Common forms include: Poisson Process, Hawkes processes [113], Self-Correcting Process [116], Power Law and Rayleigh Process.

Rayleigh Process is a non-monotonic process and is well-adapted to modeling fads, where event likelihood drops rapidly after rising to a peak. Its intensity function is $\lambda(t) = \alpha \cdot (t)$, where $\alpha > 0$ is the weight parameter, and the log survival function is $\log S(t|\alpha) = -\alpha \cdot (t)^2/2$.

5.2.2 Temporal Knowledge Graph representation

We define a *Temporal Knowledge Graph (TKG)* as a multi-relational directed graph with timestamped edges between any pair of nodes. In a *TKG*, each edge between two nodes represent an event in the real world and edge type (relationship) represent the corresponding event type. Further an edge may be available multiple times (recurrence). We do not allow duplicate edges and self-loops in graph. Hence, all recurrent edges will have different time points and every edge will have distinct subject and object entities.

Given n_e entities and n_r relationships, we extend traditional triplet representation for knowledge graphs to introduce time dimension and represent each fact in *TKG* as a quadruplet (e^s, r, e^o, t), where $e^s, e^o \in \{1, \ldots, n_e\}$, $e^s \neq e^o$, $r \in \{1, \ldots, n_r\}$, $t \in \mathbb{R}^+$. It represents the creation of relationship edge r between subject entity e^s, and object entity e^o at time t. The complete TKG can therefore be represented as an $n_e \times n_e \times n_r \times \mathcal{T}$ - dimensional tensor where $\mathcal{T}$ is the total number of available time points. Consider a TKG comprising of N edges and denote the globally ordered set of corresponding N observed events as $\mathcal{D} = \{(e^s, r, e^o, t)_n\}_{n=1}^{N}$, where $0 \leq t_1 \leq t_2 \ldots \leq T$.

5.3 Evolutionary Knowledge Network

We present our unified knowledge evolution framework (Know-Evolve) for reasoning over temporal knowledge graphs. The reasoning power of Know-Evolve stems from the following three major components:

1. A powerful mathematical tool of temporal point process that models occurrence of a fact.

2. A bilinear relationship score that captures multi-relational interactions between entities and modulates the intensity function of above point process.

3. A novel deep recurrent network that learns non-linearly and mutually evolving latent representations of entities based on their interactions with other entities in multi-relational space over time.

5.3.1 Temporal Process

Large scale temporal knowledge graphs exhibit highly heterogeneous temporal patterns of events between entities. Discrete epoch based methods to model such temporal behavior fail to capture the underlying intricate temporal dependencies. We therefore model time as a random variable and use temporal point process to model occurrence of fact.

More concretely, given a set of observed events $\mathcal{O}$ corresponding to a *TKG*, we construct a relationship-modulated multidimensional point process to model occurrence of these events. We characterize this point process with the following conditional intensity function:

$$\lambda_r^{e^s,e^o}(t|\bar{t}) = f(g_r^{e^s,e^o}(\bar{t})) * (t - \bar{t}) \tag{5.3}$$

where $t > \bar{t}$, t is the time of the current event and $\bar{t} = max(t^{e^s}-, t^{e^o}-)$ is the most recent time point when either subject or object entity was involved in an event before time t. Thus, $\lambda_r^{e^s,e^o}(t|\bar{t})$ represents intensity of event involving triplet (e^s, r, e^j) at time t given previous

time point $\bar{t}$ when either e^s or e^o was involved in an event. This modulates the intensity of current event based on most recent activity on either entities' timeline and allows to capture scenarios like non-periodic events and previously unseen events. $f(\cdot) = \exp(\cdot)$ ensures that intensity is positive and well defined.

5.3.2 Relational Score Function

The first term in (5.3) modulates the intensity function by the relational compatibility score between the involved entities in that specific relationship. Specifically, for an event $(e^s, r, e^o, t) \in \mathcal{D}$ occurring at time t, the score term $g_r^{e^s,e^o}$ is computed using a bilinear formulation as follows:

$$g_r^{e^s,e^o}(t) = \mathbf{v}^{\mathbf{e^s}}(t-)^T \cdot \mathbf{R_r} \cdot \mathbf{v}^{\mathbf{e^o}}(t-) \tag{5.4}$$

where $\mathbf{v}^{\mathbf{e^s}}$, $\mathbf{v}^{\mathbf{e^s}} \in \mathbb{R}^d$ represent latent feature embeddings of entities appearing in subject and object position respectively. $\mathbf{R_r} \in \mathbb{R}^{d \times d}$ represents relationship weight matrix which attempts to capture interaction between two entities in the specific relationship space r. This matrix is unique for each relation in dataset and is learned during training. t is time of current event and $t-$ represent time point just before time t. $\mathbf{v}^{\mathbf{e^s}}(t-)$ and $\mathbf{v}^{\mathbf{e^o}}(t-)$, therefore represent most recently updated vector embeddings of subject and object entities respectively before time t. As these entity embeddings evolve and update over time, $g_r^{e^s,e^o}(t)$ is able to capture cumulative knowledge learned about the entities over the history of events that have affected their embeddings.

5.3.3 Dynamically Evolving Entity Representations

We represent latent feature embedding of an entity e at time t with a low-dimensional vector $\mathbf{v}^{\mathbf{e}}(t)$. We add superscript $\mathbf{s}$ and $\mathbf{o}$ as shown in Eq. (5.4) to indicate if the embedding corresponds to entity in subject or object position respectively. We also use relationship-

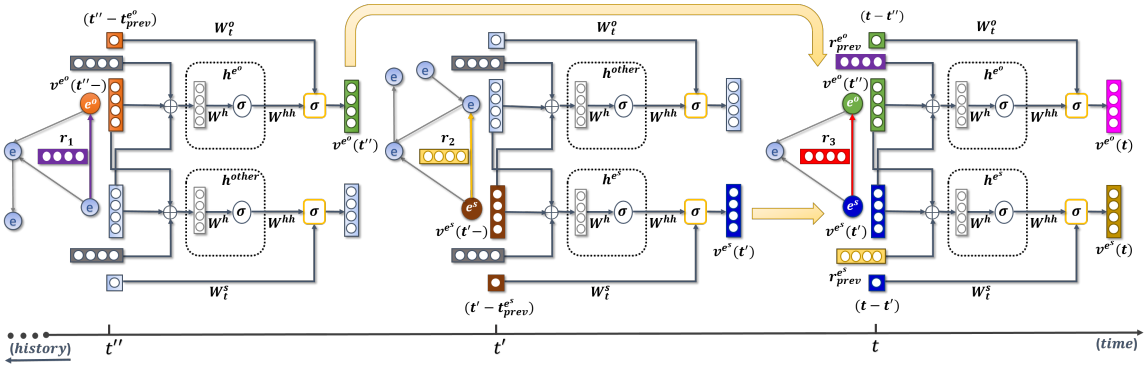
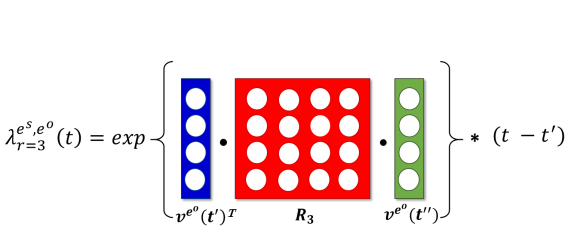
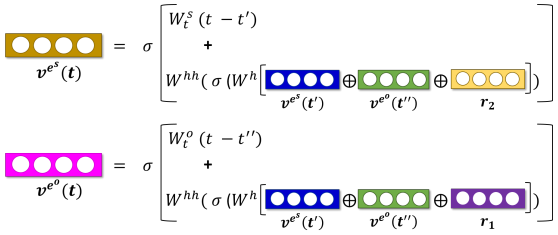


Figure 5.2: Realization of Evolutionary Knowledge Network Architecture over a timeline. Here t'', t' and t may or may not be consecutive time points. We focus on the event at time point t and show how previous events affected the embeddings of entities involved in this event. From Eq. (5.5) and (5.6), $t_{p-1} = t'$ and $t_{q-1} = t''$ respectively. $t^{e^s}_{prev}$, $t^{e^o}_{prev}$ represent previous time points in history before t', t''. $\mathbf{h}^{\mathbf{other}}$ stands for hidden layer for the entities (other than the ones in focus) involved in events at t' and t''. $r^{e^s}_{prev} = r_2$ and $r^{e^o}_{prev} = r_1$. All other notations mean exactly as defined in text. We only label nodes, edges and embeddings directly relevant to event at time t for clarity.



(a) Intensity Computation at time t



(c) Entity Embedding update after event observed at time t

Figure 5.3: One step visualization of Know-Evolve computations done in Figure 5.2 after observing an event at time t. (Best viewed in color)

specific low-dimensional representation for each relation type.

The latent representations of entities change over time as entities forge relationships with each other. We design novel deep recurrent neural network based update functions to capture mutually evolving and nonlinear dynamics of entities in their vector space representations. We consider an event $m = (e^s, r, e^o, t)_m \in \mathcal{D}$ occurring at time t. Also, consider that event m is entity e^s's p-th event while it is entity e^o's q-th event. As entities participate in events in a heterogeneous pattern, it is less likely that $p = q$ although not impossible. Having observed this event, we update the embeddings of two involved entities as follows:

Subject Embedding:

$$\begin{aligned}\mathbf{v^{e^s}}(t_p) &= \sigma(\mathbf{W_t^s}(t_p - t_{p-1}) + \mathbf{W^{hh}} \cdot \mathbf{h^{e^s}}(t_p-)) \\ \mathbf{h^{e^s}}(t_p-) &= \sigma(\mathbf{W^h} \cdot [\mathbf{v^{e^s}}(t_{p-1}) \oplus \mathbf{v^{e^o}}(t_p-) \oplus \mathbf{r_{p-1}^{e^s}}])\end{aligned} \tag{5.5}$$

Object Embedding:

$$\begin{aligned}\mathbf{v^{e^o}}(t_q) &= \sigma(\mathbf{W_t^o}(t_q - t_{q-1}) + \mathbf{W^{hh}} \cdot \mathbf{h^{e^o}}(t_q-)) \\ \mathbf{h^{e^o}}(t_q-) &= \sigma(\mathbf{W^h} \cdot [\mathbf{v^{e^o}}(t_{q-1}) \oplus \mathbf{v^{e^s}}(t_q-) \oplus \mathbf{r_{q-1}^{e^o}}])\end{aligned} \tag{5.6}$$

where, $\mathbf{v^{e^s}}$, $\mathbf{v^{e^o}} \in \mathbb{R}^d$. $t_p = t_q = t_m$ is the time of observed event. For subject embedding update in Eq. (5.5), t_{p-1} is the time point of the previous event in which entity e^s was involved. t_p- is the timepoint just before time t_p. Hence, $\mathbf{v^{e^s}}(t_{p-1})$ represents latest embedding for entity e^s that was updated after $(p-1)$-th event for that entity. $\mathbf{v^{e^o}}(t_p-)$ represents latest embedding for entity e^o that was updated any time just before $t_p = t_m$. This accounts for the fact that entity e^o may have been involved in some other event during the interval between current (p) and previous ($p-1$) event of entity e^s. $\mathbf{r_{p-1}^{e^s}} \in \mathbb{R}^c$ represent relationship embedding that corresponds to relationship type of the $(p-1)$-th event of entity e^s. Note that the relationship vectors are static and do not evolve over time. $\mathbf{h^{e^s}}(t_p-) \in \mathbb{R}^d$ is the hidden layer. The semantics of notations apply similarly to object embedding update in Eq. (5.6).

$\mathbf{W_t^s}, \mathbf{W_t^o} \in \mathbb{R}^{d \times 1}$, $\mathbf{W^{hh}} \in \mathbb{R}^{d \times l}$ and $\mathbf{W^h} \in \mathbb{R}^{l \times (2d+c)}$ are weight parameters in network learned during training. $\mathbf{W_t^s}, \mathbf{W_t^o}$ captures variation in temporal drift for subject and object respectively. $\mathbf{W^{hh}}$ is shared parameter that captures recurrent participation effect for each entity. $\mathbf{W^h}$ is a shared projection matrix applied to consider the compatibility of entities in their previous relationships. $\oplus$ represent simple concatenation operator. $\sigma(\cdot)$ denotes nonlinear activation function ($tanh$ in our case). Our formulations use simple RNN units but it can be replaced with more expressive units like LSTM or GRU in straightforward

manner. In our experiments, we choose $d = l$ and $d \neq c$ but they can be chosen differently. Below we explain the rationales of our deep recurrent architecture that captures nonlinear evolutionary dynamics of entities over time.

Reasoning Based on Structural Dependency: The hidden layer ($\mathbf{h}^{\mathbf{e^s}}$) reasons for an event by capturing the compatibility of most recent subject embedding with most recent object embedding in previous relationship of subject entity. This accounts for the behavior that within a short period of time, entities tend to form relationships with other entities that have similar recent actions and goals. This layer thereby uses historical information of the two nodes involved in current event and the edges they both created before this event. This holds symmetrically for hidden layer ($\mathbf{h}^{\mathbf{e^o}}$).

Reasoning based on Temporal Dependency: The recurrent layer uses hidden layer information to model the intertwined evolution of entity embeddings over time. Specifically this layer has two main components:

- **Drift over time:** The first term captures the temporal difference between consecutive events on respective dimension of each entity. This captures the external influences that entities may have experienced between events and allows to smoothly drift their features over time. This term will not contribute anything in case when multiple events happen for an entity at same time point (e.g. within a day in our dataset). While $t_p - t_{p-1}$ may exhibit high variation, the corresponding weight parameter will capture these variations and along with the second recurrent term, it will prevent $\mathbf{v}^{\mathbf{e^s}}(t_p)$ to collapse.

- **Relation-specific Mutual Evolution:** The latent features of both subject and object entities influence each other. In multi-relational setting, this is further affected by the relationship they form. Recurrent update to entity embedding with the information from the hidden layer allows to capture the intricate non-linear and evolutionary dynamics of an entity with respect to itself and the other entity in a specific relationship

space.

5.3.4 Understanding Unified View of Know-Evolve

Figure (5.2) and Figure (5.3) shows the architecture of knowledge evolution framework and one step of our model.

The updates to the entity representations in Eq. (5.5) and (5.6) are driven by the events involving those entities which makes the embeddings piecewise constant i.e. an entity embedding remains unchanged in the duration between two events involving that entity and updates only when an event happens on its dimension. This is justifiable as an entity's features may update only when it forges a relationship with other entity within the graph. Note that the first term in Eq. (5.5) and (5.6) already accounts for any external influences.

Having observed an event at time t, Know-Evolve considers it as an incoming fact that brings new knowledge about the entities involved in that event. It computes the intensity of that event in Eq. (5.3) which is based on relational compatibility score in Eq. (5.4) between most recent latent embeddings of involved entities. As these embeddings are piecewise constant, we use time interval term $(t - \bar{t})$ in Eq. (5.3) to make the overall intensity piecewise linear which is standard mathematical choice for efficient computation in point process framework. This formulation naturally leads to Rayleigh distribution which models time interval between current event and most recent event on either entities' dimension. Rayleigh distribution has an added benefit of having a simple analytic form of likelihood which can be further used to find entity for which the likelihood reaches maximum value and thereby make precise entity predictions.

5.4 Efficient Training Procedure

The complete parameter space for the above model is:

$\mathbf{\Omega} = \{\{\mathbf{V^e}\}_{e=1:n_e}, \{\mathbf{R_r}\}_{r=1:n_r}, \mathbf{W_e}, \mathbf{W_t^s}, \mathbf{W_t^o}, \mathbf{W^h},$

$\mathbf{W^{hh}}, \mathbf{W_r}\}$. Although Know-Evolve gains expressive power from deep architecture, Table

4 (Appendix D) shows that the memory footprint of our model is comparable to simpler relational models. The intensity function in (5.3) allows to use maximum likelihood estimation over all the facts as our objective function. Concretely, given a collection of facts recorded in a temporal window $[0, T)$, we learn the model by minimizing the joint negative log likelihood of intensity function [128] written as:

$$\mathcal{L} = \underbrace{-\sum_{p=1}^{N} \log\left(\lambda_r^{e^s,e^o}(t_p|\bar{t_p})\right)}_{\text{happened events}} + \underbrace{\sum_{r=1}^{n_r}\sum_{e^s=1}^{n_e}\sum_{e^o=1}^{n_e}\int_0^T \lambda_r^{e^s,e^o}(\tau|\bar{\tau})\,d\tau}_{\text{survival term}} \tag{5.7}$$

The first term maximizes the probability of specific type of event between two entities; the second term penalizes non-presence of all possible types of events between all possible entity pairs in a given observation window. We use Back Propagation Through Time (BPTT) algorithm to train our model. Previous techniques [107, 129] that use BPTT algorithm decompose data into independent sequences and train on mini-batches of those sequences. But there exists intricate relational and temporal dependencies between data points in our setting which limits our ability to efficiently train by decomposing events into independent sequences. To address this challenge, we design an efficient Global BPTT algorithm (Algorithm 2, Appendix A) that creates mini-batches of events over global timeline in sliding window fashion and allows to capture dependencies across batches while retaining efficiency.

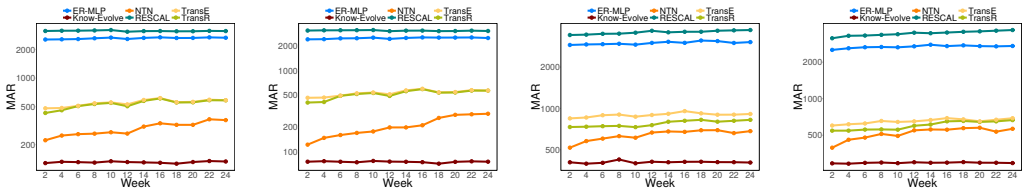
Intractable Survival Term. To compute the second survival term in (5.7), since our intensity function is modulated by relation-specific parameter, for each relationship we need to compute survival probability over all pairs of entities. Next, given a relation r and entity pair (e^s, e^o), we denote $P_{(e^s,e^o)}$ as total number of events of type r involving either e^s or e^o in window $[T_0, T)$. As our intensity function is piecewise-linear, we can decompose the

integration term $-\int_{T_0}^{T} \lambda_r^{e^s,e^o}(\tau|\bar{\tau})d\tau$ into multiple time intervals where intensity is constant:

$$\begin{aligned}
&\int_{T_0}^{T} \lambda_r^{e^s,e^o}(\tau|\bar{\tau})d\tau \\
&= \sum_{p=1}^{P_{(e^s,e^o)}-1} \int_{t_p}^{t_{p+1}} \lambda_r^{e^s,e^o}(\tau|\bar{\tau})d\tau \\
&= \sum_{p=1}^{P_{(e^s,e^o)}-1} (t_{p+1}^2 - t_p^2) \cdot exp(\mathbf{v}^{\mathbf{e^s}}(t_p)^T \cdot \mathbf{R_r} \cdot \mathbf{v}^{\mathbf{e^o}}(t_p))
\end{aligned} \tag{5.8}$$

The integral calculations in (5.8) for all possible triplets requires $\mathcal{O}(n^2 r)$ computations (n is number of entities and r is the number of relations). This is computationally intractable and also unnecessary. Knowledge tensors are inherently sparse and hence it is plausible to approximate the survival loss in a stochastic setting. We take inspiration from techniques like noise contrastive [130] estimation and adopt a random sampling strategy to compute survival loss: Given a mini-batch of events, for each relation in the mini-batch, we compute dyadic survival term across all entities in that batch. Algorithm 6 presents the survival loss computation procedure. While this procedure may randomly avoid penalizing some dimensions in a relationship, it still includes all dimensions that had events on them. The computational complexity for this procedure will be $\mathcal{O}(2n'r'm)$ where m is size of mini-batch and n' and r' represent number of entities and relations in the mini-batch.

5.5 Experiments

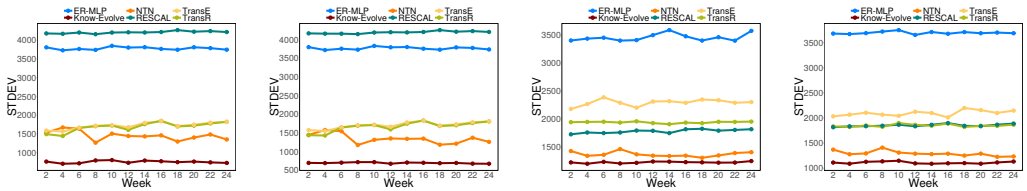


(a) ICEWS-raw (b) ICEWS-filtered (c) GDELT-raw (d) GDELT-filtered

Figure 5.4: Mean Average Rank (MAR) for Entity Prediction on both datasets.

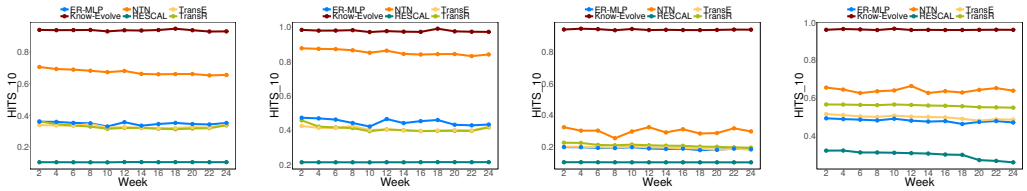
Algorithm 4 Survival Loss Computation in mini-batch

Input: Minibatch $\mathcal{E}$, size s, Batch Entity List bl
$loss = 0.0$
for $p = 0$ **to** $s - 1$ **do**
  subj_feat = $\mathcal{E}_p \rightarrow \mathbf{v}^{\mathbf{e^s}}(t-)$
  obj_feat = $\mathcal{E}_p \rightarrow \mathbf{v}^{\mathbf{e^o}}(t-)$
  rel_weight = $\mathcal{E}_p \rightarrow \mathbf{R_r}$
  t_end = $\mathcal{E}_p \rightarrow t$
  $subj\_surv = 0$, $obj\_surv = 0$, $total\_surv = 0$
  for $i = 0$ **to** $bl.size$ **do**
    obj_other = $bl[i]$
    if obj_other $==$ $\mathcal{E}_p \rightarrow e^s$ **then**
      continue
    end if
    $\bar{t} = max(t^{e^s}-, t^{e^o}-)$
    $subj\_surv \mathrel{+}= (t\_end^2 - \bar{t}^2) \cdot exp(subj\_feat^T \cdot rel\_weight \cdot obj\_other\_feat)$
  end for
  for $j = 0$ **to** $bl.size$ **do**
    subj_other = $bl[i]$
    if subj_other $==$ $\mathcal{E}_p \rightarrow e^o$ **then**
      continue
    end if
    $\bar{t} = max(t^{e^s}-, t^{e^o}-)$
    $obj\_surv \mathrel{+}= (t\_end^2 - \bar{t}^2) \cdot exp(subj\_other\_feat^T \cdot rel\_weight \cdot obj\_feat)$
  end for
  $loss \mathrel{+}= subj\_surv + obj\_surv$
end for



(a) ICEWS-raw (b) ICEWS-filtered (c) GDELT-raw (d) GDELT-filtered

Figure 5.5: Standard Deviation (STD) in MAR for Entity Prediction on both datasets.



(a) ICEWS-raw (b) ICEWS-filtered (c) GDELT-raw (d) GDELT-filtered

Figure 5.6: HITS@10 for Entity Prediction on both datasets.

5.5.1 Temporal Knowledge Graph Data

We use two datasets: Global Database of Events, Language, and Tone (GDELT) [125] and Integrated Crisis Early Warning System (ICEWS) [126] which has recently gained attention in learning community [131] as useful temporal KGs. GDELT data is collected from April 1, 2015 to Mar 31, 2016 (temporal granularity of 15 mins). ICEWS dataset is collected from Jan 1, 2014 to Dec 31, 2014 (temporal granularity of 24 hrs). Both datasets contain records of events that include two actors, action type and timestamp of event. We use different hierarchy of actions in two datasets - (top level 20 relations for GDELT while last level 260 relations for ICEWS) - to test on variety of knowledge tensor configurations. Note that this does not filter any record from the dataset. We process both datasets to remove any duplicate quadruples, any mono-actor events (*i.e.*, we use only dyadic events), and self-loops. We report our main results on full versions of each dataset. We create smaller version of both datasets for exploration purposes. Table 1 (Appendix B) provide statistics about the data and Table 2 (Appendix B) demonstrates the sparsity of knowledge tensor.

5.5.2 Competitors

We compare the performance of our method[1] with following relational learning methods: RESCAL, Neural Tensor Network (NTN), Multiway Neural Network (ER-MLP), TransE and TransR. To the best of our knowledge, there are no existing relational learning approaches that can predict time for a new fact. Hence we devised two baseline methods for evaluating time prediction performance — *(i) Multi-dimensional Hawkes process (MHP):* We model dyadic entity interactions as multi-dimensional Hawkes process similar to [42]. Here, an entity pair constitutes a dimension and for each pair we collect sequence of events on its dimension and train and test on that sequence. Relationship is not modeled in this setup. *(ii) Recurrent Temporal Point Process (RTPP):* We implement a simplified version

[1]Code is available at: https://github.com/rstriv/Know-Evolve

of RMTPP [107] where we do not predict the marker. For training, we concatenate static entity and relationship embeddings and augment the resulting vector with temporal feature. This augmented unit is used as input to global RNN which produces output vector $\mathbf{h_t}$. During test time, for a given triplet, we use this vector $\mathbf{h_t}$ to compute conditional intensity of the event given history which is further used to predict next event time. Appendix C provides implementation details of our method and competitors.

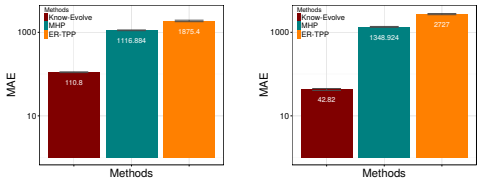
5.5.3 Evaluation Protocol

We report experimental results on two tasks: *Link prediction* and *Time prediction*.

Link prediction: Given a test quadruplet (e^s, r, e^o, t), we replace e^o with all the entities in the dataset and compute the conditional density $d_r^{e^s,e^o} = \lambda_r^{e^s,e^o}(t) S_r^{e^s,e^o}(t)$ for the resulting quadruplets including the ground truth. We then sort all the quadruplets in the descending order of this density to rank the correct entity for object position. We also conduct testing after applying the filtering techniques described in [71] - we only rank against the entities that do not generate a true triplet (seen in train) when it replaces ground truth object. We report Mean Absolute Rank (MAR), Standard Deviation for MAR and HITS@10 (correct entity in top 10 predictions) for both Raw and Filtered Versions.

Time prediction: Give a test triplet (e^s, r, e^o), we predict the expected value of next time the fact (e^s, r, e^o) can occur. This expectation is defined by: $\mathbb{E}_r^{e^s,e^o}(t) = \sqrt{\frac{\pi}{2\exp\left(g_r^{e^s,e^o}(t)\right)}}$, where $g_r^{e^s,e^o}(t)$ is computed using equation (5.4). We report Mean Absolute Error (MAE) between the predicted time and true time in hours.

Sliding Window Evaluation. As our work concentrates on temporal knowledge graphs, it is more interesting to see the performance of methods over time span of test set as compared to single rank value. This evaluation method can help to realize the effect of modeling temporal and evolutionary knowledge. We therefore partition our test set in 12 different slides and report results in each window. For both datasets, each slide included 2 weeks of time.



(a) GDELT-500 (b) ICEWS-500

Figure 5.7: Time prediction performance (Unit is hours).

5.5.4 Quantitative Analysis

Link Prediction Results. Figure (5.4, 5.5, 5.6) demonstrate link prediction performance comparison on both datasets. Know-Evolve significantly and consistently outperforms all competitors in terms of prediction rank without any deterioration over time. Neural Tensor Network's second best performance compared to other baselines demonstrate its rich expressive power but it fails to capture the evolving dynamics of intricate dependencies over time. This is further substantiated by its decreasing performance as we move test window further in time.

The second row represents deviation error for MAR across samples in a given test window. Our method achieves significantly low deviation error compared to competitors making it most stable. Finally, high performance on HITS@10 metric demonstrates extensive discriminative ability of Know-Evolve. For instance, GDELT has only 20 relations but 32M events where many entities interact with each other in multiple relationships. In this complex setting, other methods depend only on static entity embeddings to perform prediction unlike our method which does effectively infers new knowledge using powerful evolutionary network and provides accurate prediction results.

Time Prediction Results. Figure 5.7 demonstrates that Know-Evolve performs significantly better than other point process based methods for predicting time. MHP uses a specific parametric form of the intensity function which limits its expressiveness. Further, each entity pair interaction is modeled as an independent dimension and does not take into account relational feature which fails to capture the intricate influence of different entities on each other. On the other hand, RTPP uses relational features as part of input, but it

sees all events globally and cannot model the intricate evolutionary dependencies on past events. We observe that our method effectively captures such non-linear relational and temporal dynamics.

In addition to the superior quantitative performance, we demonstrate the effectiveness of our method by providing extensive exploratory analysis in Appendix E.

5.6 Related Work

In this section, we discuss relevant works in relational learning and temporal modeling techniques.

5.6.1 Relational Learning

Among various relational learning techniques, neural embedding models that focus on learning low-dimensional representations of entities and relations have shown state-of-the-art performance. These methods compute a score for the fact based on different operations on these latent representations. Such models can be mainly categorized into two variants:

Compositional Models. RESCAL [69] uses a relation specific weight matrix to explain triplets via pairwise interactions of latent features. Neural Tensor Network (NTN) [75] is more expressive model as it combines a standard NN layer with a bilinear tensor layer. [2] employs a concatenation-projection method to project entities and relations to lower dimensional space. Other sophisticated models include Holographic Embeddings (HoLE) [80] that employs circular correlation on entity embeddings and Neural Association Models (NAM) [79], a deep network used for probabilistic reasoning.

Translation Based Models. [84] uses two relation-specific matrices to project subject and object entities and computes L_1 distance to score a fact between two entity vectors. [71] proposed TransE model that computes score as a distance between relation-specific translations of entity embeddings. [85] improved TransE by allowing entities to have distributed representations on relation specific hyperplane where distance between them is

computed. TransR [86] extends this model to use separate semantic spaces for entities and relations and does translation in the relationship space.

[19] and [73, 87] contains comprehensive reviews and empirical comparison of relational learning techniques respectively. All these methods consider knowledge graphs as static models and lack ability to capture temporally evolving dynamics.

5.6.2 Temporal Modeling

Temporal point processes have been shown as very effective tool to model various intricate temporal behaviors in networks [132, 133, 12, 42, 107, 134, 135, 114, 136, 137]. Recently, [134, 138] proposed novel co-evolutionary feature embedding process that captures self-evolution and co-evolution dynamics of users and items interacting in a recommendation system. In relational setting, [9] proposed relational mining approach to discover changes in structure of dynamic network over time. [8] proposes method to capture temporal autocorrelation in data to improve predictive performance. [139] proposes summarization techniques to model evolving relational-temporal domains. Recently, [140] proposed multiway neural network architecture for modeling event based relational graph. The authors draw a synergistic relation between a static knowledge graph and an event set wherein the knowledge graph provide information about entities participating in events and new events in turn contribute to enhancement of knowledge graph. They do not capture the evolving dynamics of entities and model time as discrete points which limits its capacity to model complex temporal dynamics. [141] models dependence of relationship on time to facilitate time-aware link prediction but do not capture evolving entity dynamics.

5.7 Summary

We propose a novel deep evolutionary knowledge network that efficiently learns non-linearly evolving entity representations over time in multi-relational setting. Evolutionary dynamics of both subject and object entities are captured by deep recurrent architecture

that models historical evolution of entity embeddings in a specific relationship space. The occurrence of a fact is then modeled by multivariate point process that captures temporal dependencies across facts. The superior performance and high scalability of our method on large real-world temporal knowledge graphs demonstrate the importance of supporting temporal reasoning in dynamically evolving relational systems. Our work establishes previously unexplored connection between relational processes and temporal point processes with a potential to open a new direction of research on reasoning over time.

Part III

Learning Graph Formation Mechanisms

CHAPTER 6

LEARNING OPTIMIZATION MODELS OF GRAPHS

Formation mechanisms are fundamental to the study of complex networks, but learning them from observations is challenging. In real-world domains, one often has access only to the final constructed graph, instead of the full construction process, and observed graphs exhibit complex structural properties. In this work, we propose GraphOpt, an end-to-end framework that jointly learns an implicit model of graph structure formation and discovers an *underlying optimization mechanism* in the form of a *latent* objective function. The learned objective can serve as an explanation for the observed graph properties, thereby lending itself to transfer across different graphs within a domain. GraphOpt poses link formation in graphs as a sequential decision-making process and solves it using maximum entropy inverse reinforcement learning algorithm. Further, it employs a novel continuous *latent action space* that aids scalability. Empirically, we demonstrate that GraphOpt discovers a latent objective transferable across graphs with different characteristics. GraphOpt also learns a robust stochastic policy that achieves competitive link prediction performance without being explicitly trained on this task and further enables construction of graphs with properties similar to those of the observed graph.

6.1 Introduction and Related Work

Learning generative mechanisms of graph-structured data is an important approach to build graph constructors for data augmentation [1] and inference modules for downstream network analysis and prediction tasks. Such models have wide-ranging applications in several domains spanning recommendation systems [142], biological networks [143], knowledge graphs [72], social networks [144] and many more. Formation mechanisms play a fundamental role in driving the generative process of structures observed in the real-world [145].

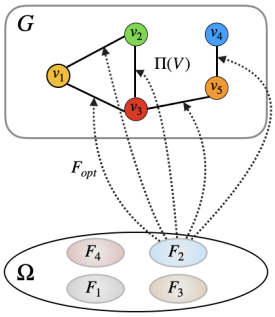


Figure 6.1: Ω is set of latent objective functions $\{\mathcal{F}_i\}$, any of which could lead to observed graph $\mathcal{G}$ when optimised. Our goal is to discover one such latent objective $\mathcal{F}_{\text{opt}}$ that could serve as an explanation of the observed graph properties, and optimise it to learn a graph construction procedure Π such that $\Pi(\mathcal{V})$, given node set $\mathcal{V}$, mimics the network patterns observed in $\mathcal{G}$. While $\mathcal{F}_{\text{opt}}$ may not match the unknown ground truth mechanism when one exists, it can produce an accurate Π and hence can be operationally equivalent to the true mechanism.

Modeling these mechanisms is important as it could facilitate synthesis of novel graph structures for various subsequent studies such as analysing disease progression. Moreover, access to these mechanisms could help to build intelligent systems that *generalize* beyond the task of structure generation and support *transfer* to graphs beyond available observations. However, in real-world domains, such formation mechanisms are often unknown and learning them from data is challenging due to: limited or no access to the construction process (one often observes only the final graph); complex discrete nature of graph and rare availability of large collection of graph samples for learning.

In this paper, we investigate the novel problem setting of discovering an underlying formation mechanism of the observed graph structure. Concretely, we propose **GraphOpt**, an end-to-end learning framework that jointly learns the forward model of graph construction and solves the inverse problem of discovering an *underlying optimization mechanism*, in the form of a *latent* objective function, that serves as an explanation for the existence of the observed graph structure. From a broader perspective of network science, GraphOpt naturally aligns more with the "optimization" viewpoint of graph formation [146, 13, 145]—link formation is viewed as the outcome of an underlying optimization mechanism whereby decisions are based on current global state of the network. For instance, links in transportation

networks appear as a result of optimizing some underlying cost function [147]. In contrast, most existing learning approaches for graphs are implicitly rooted in the ”probabilistic” viewpoint, which models link formation in networks as random events that depend largely on local structural properties [148, 149]. 6.1 provides an overview of our proposition.

Formally, GraphOpt is realised as an efficient maximum entropy based reinforcement learning [150, 151] framework that models graph formation as a sequential decision-making process. It trains a novel structured policy network such that the learned stochastic policy constructs edges in a sequential manner to produce a graph with minimal deviation in a set of graph properties from an observed graph. This policy network uses a graph neural network to capture the complex information of a partially constructed graph into a continuous state representation. As the true graph formation objective function is unknown, the policy optimizes a latent reward function learned via inverse reinforcement learning (IRL) [151, 152], which amounts to learning an optimization-based model of graph formation. Further, we propose a novel *continuous latent action space* that is independent of the size of graph, thereby allowing GraphOpt to learn over large graphs.

Traditional generative approaches include explicit probabilistic models that are carefully hand-designed to incorporate assumptions on structural properties [153, 154, 155, 156, 157]. Such intuitive model specifications produce graphs that often exhibit disagreement with real-world graphs [158, 159]. Recent advances in deep generative models for graphs [47, 160, 20, 49] address this issue by directly learning from data to be able to mimic the observed properties and produce realistic graphs. However, these techniques are either limited to learn over small graphs or require a large collection of graphs from the same distribution to achieve a desired fidelity, both of which pose great limitations on learning over real-world graphs. Further, the above techniques only facilitate graph generation but does not directly allow downstream inference tasks, which further limits their usability. A recently proposed deep generative model NetGAN [52], resembles GraphOpt in being implicit model for graph construction, however, the two approaches are fundamen-

tally different as NetGAN builds a probabilistic model of random walks over graphs and avoids learning an objective function which stands in contrast to our optimization-based framework.

We perform extensive experiments on graphs with varying properties to gauge the efficacy of GraphOpt on the following measures: (i) Can GraphOpt discover a useful and transferable latent objective for a given domain? (ii) How well does GraphOpt's construction policy generalize to downstream inference tasks? (iii) Can GraphOpt serve as an effective graph constructor useful to synthesize new graphs that exhibit structural patterns similar to the ones found in an observed graph? We comprehensively answer all three questions in the positive via experiments demonstrating effective transfer in the domain of citation graphs; competitive link prediction performance against dedicated baselines demonstrating compelling generalization properties; and consistently superior performance on graph construction experiments against strong baselines that learn from single input graph. We discuss more related works in D.2.

6.2 Proposed Approach: GraphOpt

We first elaborate on the optimization viewpoint of graph formation and how it motivates our modeling approach. Next, we formally define the problem we tackle and define the corresponding sequential decision-making process. Finally, we present architecture details of the GraphOpt framework.

6.2.1 Optimization Models of Graph Formation

A reasonable graph formation model can help to determine how networks come into existence, which can be fundamentally important in various applications where the network structure often influences decision making [161]. While networks in certain domains (e.g. transportation [147]) can be explained by an underlying optimization mechanism with a known functional form, most general networks often exhibit properties that have given rise

to long-standing debates in the network science community on the true mechanisms underlying their emergence [16]. For instance, power laws observed in social and biological networks can be explained by the probabilistic model of preferential attachment [162], but they can also be the result of an underlying optimization process in which nodes optimize between popularity and similarity when forming connections [13, 14, 15]. For complex networks, the functional form of the objective being optimized in this process is often unknown or difficult to specify in closed form.

In this work, we rigorously investigate the optimization viewpoint and its implications on developing learning approaches for graphs. As the true underlying mechanisms are unknown, we design an algorithm that *discovers* a latent objective that is operationally equivalent to the true mechanism, in the sense that the discovered objective, when optimised, enables a suitable graph construction procedure to produce graphs with similar properties as the observed one. As both the construction procedure and the objective depend on the global information, our approach is naturally aligned with the optimization viewpoint of graph formation.

6.2.2 Problem Definition

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we propose a graph formation model in which the optimization of some latent objective function $\mathcal{F}_{\text{opt}} \colon \mathcal{G} \to \mathbb{R}$ drives the formation of edges in $\mathcal{E}$. $\mathcal{F}_{\text{opt}}$ may correspond to any domain-specific or generic graph property but is unknown in general. Our primary aim is to learn a graph construction procedure Π^* and discover a latent objective $\mathcal{F}_{\text{opt}}$, such that the optimization of $\mathcal{F}_{\text{opt}}$ by Π^* leads to the construction of a graph $\mathcal{G}' = \Pi^*(\mathcal{V})$ with structural properties similar to the observed graph $\mathcal{G}$, given node set $\mathcal{V}$ and initially empty edge set $\mathcal{E}_0$. Learning the construction procedure is formalized as:

$$\Pi^* = \operatorname*{argmax}_{\Pi} \mathcal{F}_{\text{opt}}(\Pi(\mathcal{V})), \tag{6.1}$$

$\mathcal{F}_{\text{opt}}$ is often unknown as we only observe the final graph. To this end, we draw inspiration from inverse reinforcement learning [163] and formulate our objective as the following minmax optimization problem:

$$\begin{aligned} \Pi^* &= \operatorname*{argmin}_{\Pi} \max_{\mathcal{F}} \left[\mathcal{F}(\mathcal{G}) - \mathcal{F}(\Pi(\mathcal{V}))\right] \\ \mathcal{F}_{\text{opt}} &= \operatorname*{argmax}_{\mathcal{F}} \left[\mathcal{F}(\mathcal{G}) - \mathcal{F}(\Pi^*(\mathcal{V}))\right] \end{aligned} \tag{6.2}$$

where the goal is to learn jointly: (i) a latent objective $\mathcal{F}$, defined via a reward function, that assigns higher value to $\mathcal{G}$ than to all other graphs with different structural properties; (ii) a construction procedure Π, defined as the sequential execution of a policy, that constructs $\mathcal{G}'$ with minimal difference from $\mathcal{G}$ in structural properties measured by $\mathcal{F}$.

6.2.3 Graph Formation as a Markov Decision Process

The graph formation mechanism is the central focus of our work. Formation of real-world graphs in general is not confined to result in a connected graph. Therefore, we propose a mechanism for link formation without this constraint. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{Y}, \mathcal{X})$ denote a graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{Y}$ is the set of edge types and $\mathcal{X}$ is the set of node features. We define a Graph Formation Markov decision process (GF-MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, P)$ as follows:

State $s_t \in \mathcal{S}$. The state of the environment s_t at time t is the partially constructed graph $G_t = (\mathcal{V}, \mathcal{E}_t, \mathcal{Y}, \mathcal{X})$. Initial state $s_0 = G_0$ is a graph with all nodes but no edges, i.e. $\mathcal{E}_t = \emptyset$. Node features $\mathcal{X}$ and edge types $\mathcal{Y}$ are optional. This definition is sufficient to describe the graph at any time t and allows for sequential construction of edges without enforcing connectivity. For ease of exposition and w.l.o.g., we let $s_t = (\mathcal{V}, \mathcal{E}_t)$ represent a state in this paper.

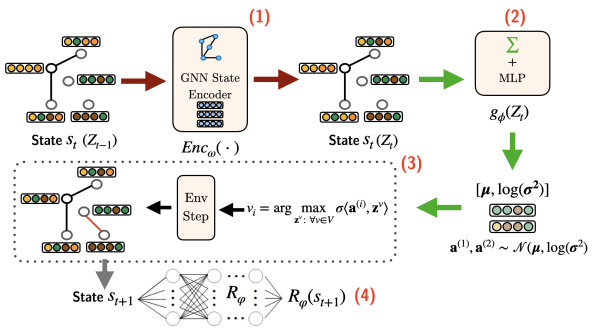
Action $a_t \in \mathcal{A}$. Each step in procedure Π involves the creation of an edge between two nodes in $\mathcal{V}$. For any state s_t, information of nodes in $\mathcal{V}$, as encoded in their representations,

is vital in determining the compatibility of two nodes for next edge creation. To capture this insight, we propose a novel *continuous latent action space*, whereby an action is mapped to the creation of an edge between two nodes with feature representation most similar to the action vector (6.2.4). In contrast to previous RL approaches to modeling graph structured data that define a discrete action space [24, 164, 48], our continuous latent action is independent of the size of graph, thereby facilitating scalable learning.

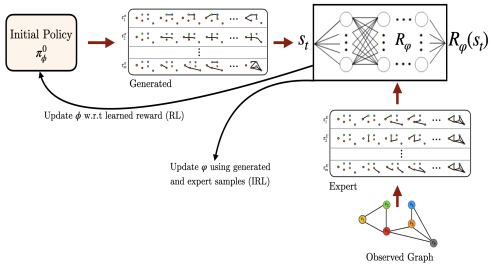
Transition Dynamics. The transition function $P(s_{t+1}|s_t, a_t)$ is defined such that an action mapped to edge (v_i, v_j) chosen at a state $s_t = (\mathcal{V}, \mathcal{E}_t)$ produces a next state $s_{t+1} = (\mathcal{V}, \mathcal{E}_t \cup (v_i, v_j))$. All edges are allowed for selection except when transition is a self-loop—i.e., both action components map to same node—which is rejected with no change in state.

Reward $\mathcal{R}$. The GF-MDP perspective gives a concrete instantiation of the key component of our model – an underlying (latent) optimization objective $\mathcal{F}_{\text{opt}}$ in (6.2): $\mathcal{F}_{\text{opt}}$ is exactly the expected return $\mathbb{E}_{\pi_\phi, P}[\sum_{t=0}^{T} \mathcal{R}(s_t)]$ for executing a policy $\pi_\phi(a_t|s_t)$ with transition function $P(s_{t+1}|s_t, a_t)$ under a latent reward function $\mathcal{R}$ evaluated at every state. In contrast to existing RL frameworks for modeling graph structured data [165, 48], which use specific forms on the reward function, we propose to learn $\mathcal{R}$ directly from the observed graph.

Optimization over $\mathcal{F}$ in (6.2) is the search for a reward function $\mathcal{R}$ that assigns greater value to the observed graph than all other generated graphs, which reflects the assumption of optimality of the observed graph. The optimization over Π in (6.2) is an optimization over π_ϕ to maximize the expected return of $\mathcal{R}$ over the formation process, which serves to construct a graph with minimal difference in structural properties, as measured by the reward, from the observed graph.



(a) GraphOpt Neural Policy Architecture



(b) GraphOpt Learning Loop

Figure 6.2: Overview of GraphOpt Framework. (a) A GNN encoder maps a graph state s_t into a representation Z_t (1), which is aggregated and passed through an MLP (2), and interpreted as the mean and standard deviation of a Gaussian policy. A latent continuous action $(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ is sampled and mapped to two nodes with most similar embeddings (3). States are evaluated by reward function R_φ (4). (b) GraphOpt interleaves policy improvement using the current reward function and reward updates using generated and expert trajectories.

6.2.4 GraphOpt's Neural Policy Architecture

As GraphOpt operates in a graph-structured environment, we build a graph neural network (GNN) based structured policy network to effectively utilize the structural information. At time step t, GNN encodes graph state s_t into low dimensional representation for the policy to compute a corresponding action a_t. We first describe the action selection procedure followed by the state encoder architecture. Figure 6.2 (a) provides an overview of the policy network.

Stochastic Action Selection

We design a stochastic policy that takes as input state s_t and outputs a link formation action a_t. As outlined in GF-MDP, we introduce a novel *continuous latent action space* which induces action over node representations learned from data. Specifically, action a_t is a 2-tuple $(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ whose components $\mathbf{a}^{(i)} \in \mathbb{R}^d$ are mapped to the node representations so as to select two nodes to construct an edge. Let $v \in \mathcal{V}$ denote a node and let $\mathbf{z}^v \in \mathbb{R}^d$ denote its embedding (learned in 6.2.4). Under a Gaussian policy π_ϕ, the next action is computed

as follows:

$$[\boldsymbol{\mu}, \log(\boldsymbol{\sigma^2})] = \pi(s_t) = g_\phi(\text{Enc}_\omega(s_t))$$
$$\mathbf{a}^{(1)}, \mathbf{a}^{(2)} \sim \mathcal{N}(\boldsymbol{\mu}, \log(\boldsymbol{\sigma}^2)) \tag{6.3}$$

where g_ϕ is a two layer MLP with the policy parameters ϕ. $\text{Enc}_\omega(\cdot)$ is a state encoder that computes low-dimensional representations of graph states. For an effective encoding of state information, we employ a GNN architecture with parameters ω (6.2.4). Then we select two nodes to construct an edge using a similarity criterion:

$$v_i = \underset{\mathbf{z}^v\colon \forall v \in V}{\text{argmax}}\, \sigma\langle \mathbf{a}^{(i)}, \mathbf{z}^v \rangle \quad \text{for } i = 1, 2 \tag{6.4}$$

where $\langle \cdot, \cdot \rangle$ is a dot product and σ is the sigmoid function. As the mapping from continuous $\mathbf{a}^{(i)}$'s to node indices is external to policy network, GraphOpt is fully differentiable.

Structured State Encoder

During the graph formation process, the present structure of the graph may be a crucial factor that determines a new edge creation. Structural information of graphs are often encoded into low-dimensional representations and input to the policy network [166]. To achieve this, each state s_t is represented by a node embedding matrix $\mathbf{Z}_t \in \mathbb{R}^{n \times d}$ (where $n = |\mathcal{V}|$), computed using a GNN [167] via a p-step message propagation architecture. At initial state s_0 when $\mathcal{E} = \emptyset$, $\mathbf{Z}_0$ is initialized with node features. After adding an edge at time t, we perform p iterations of message passing across the node set to obtain $\mathbf{Z}_{t+1}$. For each iteration p, we update representation of each node as per the following equations:

Aggregate messages from the neighborhood of v:

$$\mathbf{m}_v^p \leftarrow \mathit{AGG}(M(\mathbf{H}_u^{p-1})), \;\; \forall u\colon \mathbf{A}_t(u, v) = 1$$

and then compute representation update for v using:

$$\mathbf{H}_v^p \leftarrow \mathit{U}(\mathbf{H}_v^{p-1}, \mathbf{m}_v^p)$$

where $\mathbf{A}_t$ is the adjacency matrix. We use max pooling as AGG aggregation function due to its better empirical performance. Both the message function M and the update function U are MLP. At the end of each training episode, we reset $\mathbf{Z}_t$ to initial state when resetting the environment.

6.3 Maximum Entropy Learning Procedure

GraphOpt contains three modules: a graph construction policy π, a latent reward function $\mathcal{R}$, and a state encoder network[1]. As the policy and latent reward are learned simultaneously in a graph structured environment, we require both stability and efficiency, which are difficult to satisfy simultaneously by off-policy methods such as DDPG [168] and on-policy methods such as PPO [169]. To this end, we adopt Soft-Actor-Critic (SAC) [150], a maximum entropy variant of the actor-critic framework [170], and combine it with maximum entropy based Inverse Optimal Control (IOC) objective [152]. We build a unified training pipeline that optimizes following objectives:

(a) **Soft Q-function.** SAC trains a function $Q_\theta(s_t, a_t)$ on off-policy experiences by minimizing the Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim B}\big[\big(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t)\big)^2/2\big]$$

where $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\psi}}(s_{t+1})]$. Value function $V_{\bar{\psi}}$ is implicitly defined by parameters of $Q_\theta(s, a)$[150, Equation 3].

(b) **Policy Network.** The policy network $\pi_\phi(a_t|s_t)$ is trained using the following objective function:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim B,\epsilon_t\sim\mathcal{N}}[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)].$$

Following [150], we also use a reparameterization trick with a neural network transforma-

[1]The state encoder network is trained by back-propagating the policy gradients to GNN parameters in an end-to-end manner.

Algorithm 5 GraphOpt Algorithm

procedure GRAPHOPT
  Input: Empty trajectories list $\mathcal{T}_{gen}$, replay buffer B
  node representation matrix $\mathbf{Z}_0$, parameters
  $\psi, \phi, \theta, \omega, \varphi$.
  for each epoch **do**
    Reset adj. matrix $\mathbf{A}_0 = \mathbf{0}$
    # Using state encoder,
    Reset state to $s_0 = \text{Enc}_\omega(\mathbf{Z}_0, \mathbf{A}_0)$
    Initialize trajectory $\tau = \{s_0\}$
    for each environment step **do**
      $(v_1, v_2) \leftarrow a_t \sim \pi_\phi(a_t|s_t)$ using Eq 6.3, 6.4
      Update $\mathbf{A}_{t+1} \leftarrow \mathbf{A}_t[v_1, v_2] = 1$.
      Update $s_{t+1} = \text{Enc}_\omega(s_t, A_{t+1})$
      Compute $r_t = R_\varphi(s_{t+1})$
      $B \leftarrow B \cup \{(s_t, a_t, r_t, s_{t+1}, A_{t+1})\}$
      Update trajectory $\tau \leftarrow \text{concat}(\tau, s_{t+1})$
      Train_Policy $(B, \psi, \phi, \theta, \bar{\psi}, \omega)$ *# Alg 8*
      If each edge in G_t is repeated k times or
      max_path_length reached **then**
      reset episode, store τ in $\mathcal{T}_{gen}$,
      and start new trajectory $\tau = \{s_0\}$
    end for
    Collect trajectories $\mathcal{T}_{meas}$ (expert)
    Use $\mathcal{T}_{meas}$ and $\mathcal{T}_{gen}$ to update reward
    estimator R_φ
  end for
end procedure

tion as: $a_t = f_\phi(\epsilon_t; s_t)$ that results in low variance estimator.

(c) **Reward function.** $R_\varphi(s_t)$ is learned using the inverse optimal control objective: $J_R(\varphi) := -\frac{1}{N}\sum_{\tau_i \in \mathcal{T}_{\text{meas}}} \mathcal{R}_\varphi(\tau_i) + \log\big(\frac{1}{M}\sum_{\tau_j \in \mathcal{T}_{\text{gen}}} z_j \exp(\mathcal{R}_\varphi(\tau_j))\big)$ where $\mathcal{T}_{gen}$ is the set of link formation trajectories obtained from the learned policy and $\mathcal{T}_{meas}$ is the set of link formation trajectories obtained from the observed graph. These measured trajectories are collected by accumulating edges from different permutations over the ordering of edges in the original graph. All permutations can be considered "expert" trajectories as each starts from same initial state ($\mathcal{E}_0 = \emptyset$) and contains only true edges seen in the observed graph.

5 outlines the complete set of steps that are used for end-to-end training of GraphOpt. An epoch starts with initial state representation computed using state encoder when there is

no edge between the nodes (line 4-7). For every step in the environment, the policy either creates a new edge or repeats an existing edge (line 8). It receives a reward based on current reward function and the state of the environment is updated along with replay buffer and current trajectory information (line 9-13). We train the policy network, Q-network, and state encoder after every few steps taken by the environment (line 15). If all existing edges have repeated k times, the episode ends (line 16-19), the environment is reset and new trajectory τ is initialized. The reward network is trained after end of each epoch (line 21-23). D.1 details the gradient updates. In contrast to generative adversarial approaches to (6.2) for imitation learning [171], which converge to an uninformative discriminator, and in contrast to behavioral cloning [172], which does not provide an explanatory mechanism, maximum entropy IRL satisfies the key objective of our work by recovering a useful latent reward. Figure 6.2 (b) provides an overview of our algorithm.

6.4 Experiments

In this section, we aim to answer the following questions to evaluate the efficacy of our approach:

(i) Can GraphOpt discover a useful latent objective that is operationally equivalent to some underlying mechanism of a graph-structured domain, and thereby *transferable* to an unseen graph in that domain? The success and necessity of transfer is shown by how well the objective discovered on a source graph facilitates construction when optimised by a new policy on a target graph, in contrast to directly running (i.e., without fine-tuning) the trained policy on the target.

(ii) How well does GraphOpt's construction policy, learned by optimizing the discovered objective, *generalize* to downstream inference tasks? Here, we turn to the classical problem of link prediction in graphs, which requires the construction policy to generalize to predict hidden links with high accuracy, and—more crucially—perform a link prediction task for which it was not explicitly trained. We further stress test GraphOpt's generalization

capacity by deploying a trained policy on unseen target graph environments of different size and characteristics, without fine-tuning.

(iii) Can GraphOpt serve as an effective graph constructor to synthesize new graphs that exhibit structural patterns similar to those in an observed graph? We assess the performance of the learned construction policy by deploying it on the full set of training nodes and generating new graphs, analogous to standard practice in reinforcement learning [173]. GraphOpt's stochastic policy avoids copying the observed graph while preserving the statistical properties.

To the best of our knowledge, no single baseline can do all three tasks. Hence we compare GraphOpt with task-specific baselines for (ii) and (iii) and follow standard procedure in the IRL literature to report performance for (i).

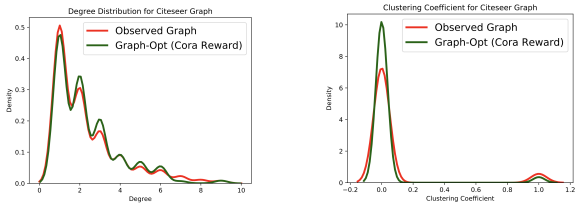
Training. All experiments begin by using the observed graph to learn the construction policy and latent reward function via 5. A key advantage of using SAC as the base RL algorithm is that it largely eliminates the need for per-task hyperparameter tuning. To encourage creation of new edges during training, we terminate an episode when the number of repeated creations of *each* existing edge reaches a threshold k, which signifies that the policy has lost the ability to explore further. We provide more details on other training configurations in D.3.4.

6.4.1 Discovering Transferable Latent Objective

For this experiment, we use two citation graphs: Cora-ML as a source environment and Citeseer as a target environment. We train on Cora-ML to discover a latent reward function, which is then transfered to train a new policy network from scratch on CiteSeer. While training on Citeseer, the reward function remains fixed and is not further trained.

Table 6.1: Transfer Performance Comparison

	Triangle Count	**Clustering Coeff.**	**Max Degree**
Cora-ML (train)	4890	0.241	168
CiteSeer (observed)	3501	0.1414	99
CiteSeer (reward)	2847.66 ± 57.13	0.098 ± 0.0010	80.66 ± 1.527
CiteSeer (direct)	2234 ± 58.96	0.084 ± 0.004	70.66 ± 2.081



(a) Degree distribution (b) Clus Coeff distribution

Figure 6.3: Degree and Clus. Coeff. distribution of graph constructed using the policy learned on CiteSeer, while optimizing the objective transferred from training on Cora-ML dataset.

After training, we input CiteSeer's node set with an empty edge set to the model and run the evaluation policy to construct edges. We collect 3 graphs and report mean and standard deviation for graph based statistics representing network patterns of the generated graphs. Table 6.1 and Figure 6.3 demonstrates that optimising the transferred objective, GraphOpt learns an effective policy to construct edge topologies that results in similar network patterns as observed in Citeseer. In contrast, the poor performance of the Cora-trained policy when directly deployed on CiteSeer without training with the transferred objective ("CiteSeer (direct)" in 6.1 and 6.4b) shows that the discovered objective is important for transfer. This experiment demonstrates GraphOpt's effectiveness in discovering a useful latent objective that serves as an explanation for the formation of observed network patterns across citation graphs, thereby lending itself to transfer across graphs within this domain.

6.4.2 Policy Generalization to Prediction Task

We first discuss the task of link prediction, which demonstrates GraphOpt's ability to learn a construction policy that generalizes to unseen task for which it was not explicitly trained. We then show the performance of the policy to generalize to new graphs of different sizes and characteristics.

Link Prediction

Setup. We conduct link prediction experiments on a variety of graphs from both non-relational and relational domains[2]. We compare our performance with both explicit baselines that employ a dedicated hand-designed link prediction objective and implicit models that generalize to the link prediction task without using explicit link prediction objectives during training; GraphOpt falls under the latter category. For all experiments, we follow the protocol in [25] by randomly removing 10% of edges to form a held-out test set and randomly sampling the same number of nonexistent links to form negative test samples. Training is then performed on the remaining graph. For relational graphs, we include edge type as an extra feature in the message passing scheme in the state encoder.

After training, we provide the observed graph as initial state and run the policy to assess how well it can predict hidden edges. For non-relational baselines, we label each edge from the test set as 1 if the policy created it and 0 otherwise, and compare with true labels to report AUC (Area under curve) and AP (average precision). For relational baselines, for a test triple ((e_s, r, e_o)), we collect all the edges that are created for tuple (e_s, r) and rank them in order of creation to report MRR and HITS@10—this signifies policy's preference to create one test edge over another. We also perform link prediction experiments to assess the usefulness of node representations learned by our state encoder.

Performance. Tables 6.2 and 6.3 show that GraphOpt's link prediction performance surpasses implicit baselines on most datasets. It is highly competitive with NetGAN (non-relational),which uses a GAN based objective, and shows significant improvement over RL based Minerva (relational), which uses LSTM encoder for state representation and a fixed reward value of +1/-1 for each step. Superior prediction performance demonstrates that GraphOpt learns a model with strong generalization capacity. GraphOpt's success in this aspect can be attributed to the combination of a stochastic policy that encourages exploration during training and the ability of the GNN to encode state representations that gen-

[2]Table D.2 and D.3 in D.3.1 provide dataset details

Table 6.2: Link Prediction on non-relational data: (*) is used to signify better performer amongst GraphOpt and method with implicit objective. Bold numbers are best two performers overall.

	Cora-ML		**Political Blogs**		**E. Coli**	
	AUC	**AP**	**AUC**	**AP**	**AUC**	**AP**
VGAE	94.70	96.10	92.60	93.44	93.22	93.10
Node2Vec	91.12	91.78	87.22	85.51	79.99	74.32
NetGAN	94.20*	95.22*	**95.51***	90.00	93.17	94.50
SEAL	**97.21**	**97.99**	95.32	**96.10**	**97.12**	**97.50**
GraphOpt-Policy	93.50	94.87	92.21	92.33*	94.43*	95*
GraphOpt-Embed	**96.21**	**96.66**	**95.50**	**95.32**	**97.20**	**97.44**

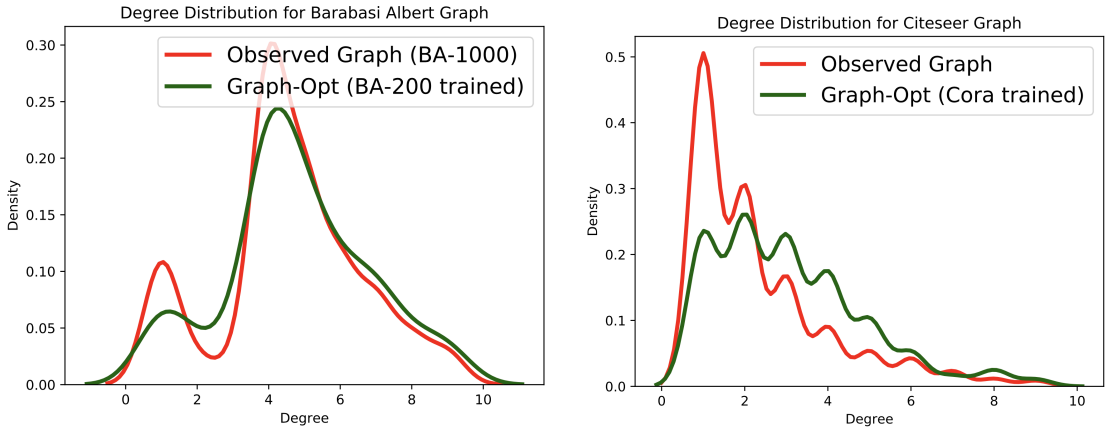
Table 6.3: Link Prediction performance on relational data: (*) is used to signify better performer amongst GraphOpt and RL method with +1/-1 reward. Bold numbers indicate best two performers overall.

	Kinship		**FB15K-237**		**WN18RR**	
	MRR	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**
ConvE	**87.1**	**98.1**	**43.5**	**62.2**	**44.9**	**54**
NeuralLP	61.9	91.2	22.7	34.8	46.3	65.7
Reward Shaping	**87.8**	**98.2**	**40.7**	56.4	47.2	54.2
Minerva	72.0	92.4*	29.3	45.6	44.8*	51.3
GraphOpt-Policy	82.2*	92.33	33.12*	53.22*	44.2	53.6*
GraphOpt-Embed	84	96.12	39.66	**58.51**	**47.3**	**58.43**

Table 6.4: Generalization Performance Comparison

	Triangle Count	**Clustering Coeff.**	**Max Degree**
BA-200 (train)	780	0.12	43
BA-1000 (observed)	1632	0.0407	115
BA-1000 (eval)	1470.66 ± 25.71	0.036 ± 0.0044	119.33 ± 2.081
Cora (train)	4890	0.241	168
CiteSeer (observed)	3501	0.1414	99
Citeseer (eval)	2234 ± 58.96	0.084 ± 0.004	70.66 ± 2.081

eralize to inference time. As a tradeoff for its greater generality, GraphOpt does not have the luxury of domain-specific architectures or objectives; hence its competitive but often slightly worse performance against state-of-the-art dedicated link prediction baselines is not surprising. However, GraphOpt's comparable performance demonstrates its greater potential for domains where an objective function is not known *a priori* and hand-designing an objective is difficult—this opens up exciting avenues of research for improvement. Surprisingly, the embedding based prediction shows strong performance, often surpassing baselines on various datasets (last row in 6.2 and 6.3). This demonstrates that GraphOpt learns a representation network that can be independently leveraged to perform various downstream tasks. As the encoder is trained in the same computational graph as the policy, through optimising the discovered latent objective, this further supports the usefulness of the discovered objective.



(a) BA-200 trained policy

(b) Cora trained policy

Figure 6.4: Policy Transfer across different size (Barabasi-Albert graph) and different graph (Cora-ML→Citeseer).

Evaluating Policy performance on Unseen Environments

We now focus on the performance of the construction policy when trained on a source graph and deployed on an unseen target graph without further training[3]. Specifically, we investigate two aspects of this direct policy transfer:

(i) Transfer from small to large graph from same distribution. We train GraphOpt on a source BA graph of 200 nodes. Then we input the node set of a target BA graph with 1000 nodes to the learned policy and run it without further training to generate 3 graphs. 6.4a and 6.4 (top 3 rows) demonstrate that generated graphs exhibit similar properties to the BA graph of 1000 nodes. This suggests that the learned construction policy can be used to generate synthetic graphs of larger size than the training graph, while preserving the underlying structural properties. *(ii) Direct transfer from source to target graph in the same domain.* We return to the earlier experiment on citation graphs, but this time transferring the trained policy from Cora-ML to run on the CiteSeer node set, without using the learned objective. As expected, 6.4 (bottom 3 rows) and 6.4b show that the policy mostly fails to construct similar patterns as observed in the original Citeseer, but it does approximate some patterns (e.g. max degree) well. This supports the necessity of using GraphOpt's

[3]For these experiments, we are only interested in evaluating the learned policy on a target environment; hence the learned reward is not used on the target graph.

discovered reward for transfer and suggests room for further investigation.

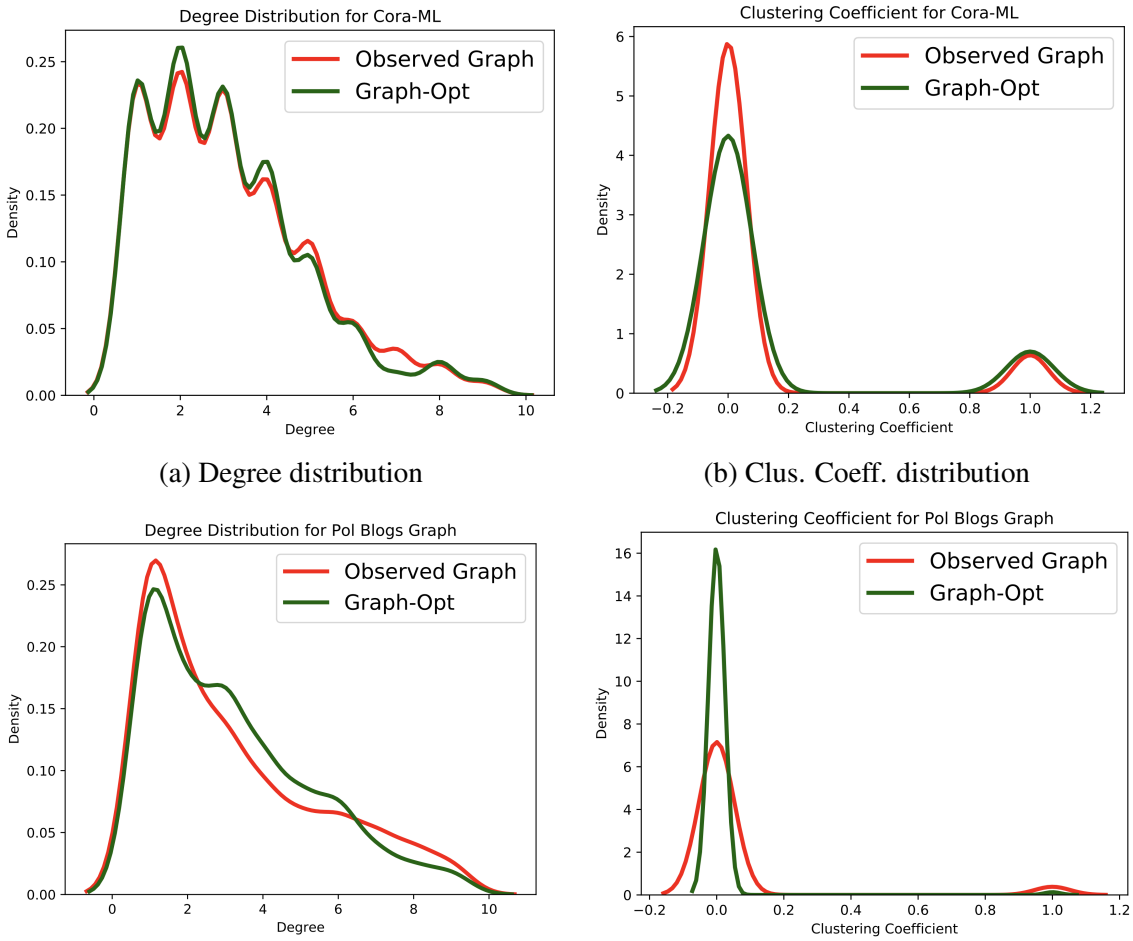
6.4.3 Synthesizing Graphs via Learned Generative Mechanism

Table 6.5: Percent deviation of graph statistics for generated graph from observed one (lower is better). First row displays the actual statistics of the observed graph. Results for more graphs and more metrics for generated graphs are available in D.4.1.

	Barabasi Albert			Political Blogs			CORA-ML		
Model	**Triangle Cnt.**	**Clust. Coeff.**	**Max Degree**	**Triangle Cnt.**	**Clust. Coeff.**	**Max Degree**	**Traingle Cnt.**	**Clust. Coeff.**	**Max Degree**
Observed Graph	504	0.1471	33	303129	0.319	351	4890	0.2406	168
DC-SBM	46.56 ± 6.58	59.44 ± 7.11	28.29 ± 7.63	52.78 ± 9.15	91.73 ± 1.18	40.86 ± 1.89	71.17 ± 1.53	68.25 ± 20.16	6.94 ± 5.40
BTER	48.02 ± 9.11	33.20 ± 1.28	33.33 ± 0	45.47 ± 7.25	54.17 ± 13.57	43.87 ± 0.75	40.06 ± 1.17	81.66 ± 1.74	16.47 ± 14.49
VGAE	70.89 ± 8.95	94.40 ± 0.81	8.08 ± 1.75	98.56 ± 0.44	99.32 ± 0.55	44.06 ± 0.92	99.56 ± 0.24	93.10 ± 2.11	94.44 ± 1.82
NetGAN	31.68 ± 6.28	40.69 ± 4.27	$\mathbf{4.04 \pm 1.74}$	44.28 ± 8.27	37.55 ± 7.2	$\mathbf{38.75 \pm 3.70}$	64.19 ± 2.15	41.12 ± 18.82	4.17 ± 2.38
GraphOpt	$\mathbf{6.28 \pm 4.05}$	$\mathbf{25.52 \pm 8.25}$	$\mathbf{5.05 \pm 4.63}$	$\mathbf{34.73 \pm 3.79}$	$\mathbf{20.34 \pm 9.1}$	$\mathbf{36.85 \pm 2.71}$	$\mathbf{19.46 \pm 1.01}$	$\mathbf{14.63 \pm 5.78}$	$\mathbf{2.58 \pm 1.24}$

Setup. We evaluate GraphOpt's ability to synthesize new graphs after learning to construct from an observed graph. We use both synthetic and real world graphs that span different domains, characteristics and sizes (Table D.1 in D.3.1). All graphs are undirected. For training, we use the complete observed graph. For evaluation, we provide the node set of the input graph and empty edge set and run the trained policy to construct edges. GraphOpt learns from a single large graph and hence we compare with strong baselines with similar setting (details in D.3.2). For all methods, we generate 3 graphs for evaluation and report mean and standard deviation of percentage error of graph based statistics between observed and generated graphs (Table 6.5). For GraphOpt, we run the evaluation policy up to either the original termination condition or a multiple of actual number of edges in observed graph, whichever is earlier. We follow reported stopping criteria for baselines.

Performance. Table 6.5 demonstrates that GraphOpt learns a construction policy that effectively captures structural patterns in the observed network and constructs graphs with similar properties to the observed graph. GraphOpt registers consistent and significantly superior performance across all datasets and against all baselines, which can perform well on some but not all metrics as they model specific statistics (except NetGAN). BTER recovers clustering coefficient statistics well but struggles on others; DC-SBM recovers Max



(a) Degree distribution

(b) Clus. Coeff. distribution

(c) Degree distribution

(d) Clus. Coeff. distribution

Figure 6.5: **(a-b)** Original vs. GraphOpt: Cora-ML (**c-d**) Original vs. GraphOpt: Pol.Blogs

Degree better than others. Further, 6.5 demonstrates the ability of Graphopt to capture intrinsic properties of graph structure. Our stochastic policy ensures that generated graphs are not merely copies of the observed graph, which is further substantiated by link prediction experiments in Section 5.2. These performance characteristics are also visible for NetGAN, which performs well in general across all metrics and datasets. However, our superior performance can be attributed to the following differences: (i) Our construction policy optimises a useful latent objective, whereas NetGAN's generator imitates the given graph by optimizing against an eventually uninformative discriminator; (ii) Our use of GNN captures better structural properties to provide rich state information to the policy network, in contrast to LSTM based path processing in NetGAN; (iii) GraphOpt allows construction of

disconnected components, often found in most real-world graphs such as in these datasets. Given these properties, we envision the use of GraphOpt as a graph constructor that ingests real-world graphs and generates synthetic versions to enrich graph repositories [174] with large-scale benchmark test sets.

We provide more details on baselines/metrics used for experiments in D.3.2 and D.3.5 respectively.

6.5 Summary

In this work, we investigate a novel setting for learning over graphs that is motivated by the optimization perspective of graph formation in network science. Our novel optimization-based learning framework, GraphOpt, models graph formation as a sequential decision-making process, learns a forward model of graph construction, and discovers a latent objective that is operationally equivalent to some underlying mechanism that could explain the formation of edges in observed graph. GraphOpt employs a novel combination of structured policy network, continuous latent action space and inverse reinforcement learning. Empirically, GraphOpt discovers a latent objective and a robust stochastic policy that transfer across graphs with different characteristics, exhibit competitive generalization for link prediction task and enable construction of graphs with similar properties as that of the observed graph. We believe that our investigation of the optimization-based perspective on network formation stemming from the wider debate in network science literature and its implications for building sophisticated models to learn effectively from graph-structured observations, coupled with the versatility of our proposed approach, would benefit the graph learning community and open exciting avenues for future research.

CHAPTER 7

LEARNING STRATEGIC NETWORK EMERGENCE GAMES

[1]

Real-world networks, especially the ones that emerge due to actions of non-inanimate agents (e.g. humans, animals), are the result of underlying strategic mechanisms aimed at maximizing individual or collective benefits. Network learning approaches built to capture these strategic insights would gain interpretability and flexibility benefits that are required to generalize beyond observations. To this end, we consider a game-theoretic formalism of network emergence that accounts for the underlying strategic mechanisms and take it to data to discover an explanation for the observed real-world networks. We propose MINE (**M**ulti-agent **I**nverse models of **N**etwork **E**mergence mechanism), a new learning framework that solves Markov-Perfect network emergence games using multi-agent inverse reinforcement learning. MINE jointly discovers agents' strategy profiles in the form of network emergence policy and the latent payoff mechanism in the form of learned reward function. In the experiments, we demonstrate that MINE learns a versatile and robust payoff mechanisms that: highly correlates with the ground truth; can be used to explain the observed network structure; and enable effective transfer to unseen environments. Further, we show that the network emergence game as a learned model supports meaningful strategic predictions, thereby signifying its applicability to a variety of challenging network analysis tasks.

7.1 Introduction

Machine Learning methods for networks [175, 18, 176, 177] typically operate on the stochastic assumption about the nature of underlying mechanisms that govern the emer-

[1] This chapter is under review at a conference. Please do not distribute.

gence of observed networks. Several networks, in spite of their different origins, indicate large commonalities among their structural properties (e.g. diameter, clustering coefficient, etc.). Solution approaches to learn from network data, therefore, rely on optimizing the likelihood of observing specific structural properties and often use surrogate objectives parameterized by modules that capture these properties, achieving notable success across different areas [51, 178, 179, 180, 181, 182]. On the other hand, many real-world networks, that emerge due to actions of non-inanimate agents, are the result of strategic behavior of individuals rather than based on probabilities. For instance, economic partnerships between financial organizations, social collaborations at work and trade between countries, all pertain to strategic actions adopted by individual entities. The network emerging out of such relationships may not correspond to any common structural parameters. Hence, it would be beneficial to model the learning problem as an equilibrium, an area that is unexplored within the existing network and relational learning literature.

Network Emergence (Formation/Creation) Games [17] provide a formal interpretable framework to characterize and analyze the decentralized process among many interacting agents, whose outcome is the observed network. This process assumes that agents follow strategic behavior in forming links with other agents and the interacting agents are considered to be inherently selfish with some capacity allowing for emergent local coordination between neighboring agents.While serving as an elegant theoretical tool to explain network emergence process, the practical applicability of Network Emergence Games beyond building stylized simulators is often hindered due to several limitations:

Existing game theoretic approaches to analyze networks [17, 53, 55, 56, 58, 59] do not directly learn from the observed network data, instead their model is hand-designed with specifications based on assumptions and intuitions about the real observations. Such specifications manifest in the form of an agent-specific *payoff (utility)*, wherein an agent is assumed to optimize this utility so as to maximize their individual benefits based on their position in the network. Simple form of payoff functions often capture only a subset of

properties related to the strategic behavior of agents, thereby diverging far from emergence mechanisms of complex real-world networks while modeling complex properties relying on hand-designed utility functions is often prone to misspecification. Further, most network games assume access to the entire network (complete information) whereas in real world networks, agents often deal with incomplete information mainly limited to its neighbors. Additionally, the emergence procedure in such games is not considered to be sequential (i.e. all agents announce all their links in one shot and the resulting network is analyzed for equilibrium after which the game is restarted). While the problem of incomplete information in network games is still under-explored even in game theory literature, recent works [58] have proposed Markov version of network games that consider network emergence as a sequential process and we discuss them in detail later.

Our Approach. In this work, we combine the interpretability merits of game theoretic modeling with practical usefulness of data-driven learning and propose an algorithmic framework, MINE, to learn strategic network emergence games from the observed network data. Concretely, our data consists of a graph structure between n-players (represented by the nodes), where the observed structure is the result of strategic interactions between these n players under some unknown payoff. A key novelty of MINE is to explicitly incorporate the network game dynamics into the learning framework tasked to jointly discover both the agents' strategies and unknown payoff mechanism that led to the emergence of the observed network. Learning the payoff directly from the observed data (inverse problem) allows to model complex mechanisms – often missed by the hand-designed specifications or difficult to express in an explicit form in the first place. We consider this network emergence game being played in a Markov setting where the agents interact with each other in a sequential manner to achieve Markov Perfect equilibrium networks and the strategies of agents at any step only depends on the current state of the network. To develop a practical learning framework, we leverage on Markov Quantal Response equilibrium (MQRE) as a solution concept and design our algorithm building on recently proposed inverse rein-

forcement learning technique [183] for solving Markov games, where the learned reward is interpreted as the payoff mechanism. Solving for equilibrium in our setting can then be viewed as searching for agents' strategy profiles (policies) that leads to equilibrium network (under the learned reward) from the set of all feasible networks which is a combinatorial search task. For efficient learning in graph structured environment, we use graph neural network (GNN) [167] based state representation, continuous action space mapped from agents' features and Soft-Actor-Critic (SAC) [150] algorithm to update policies in the inner loop of reward learning.

A key outcome of our data-driven learning of network emergence games is its ability to facilitate *interpretability* (the discovered reward function can be analyzed to characterize the observed network properties) and *generalization* (the learned modules can be transferred for use beyond observed network) – both of which are hallmarks of human intelligence and highly desirable of any learning framework. Additionally, as MINE focuses on strategic mechanisms of network emergence, it can be used for performing strategic prediction tasks, signifying its practical usefulness for network analysis of real-world agent-based networks. In the experiments, we focus on the above properties of MINE and address the following questions: (i) Can MINE effectively discover the payoff mechanisms useful to explain the characteristics of observed network?; (ii) How well does the learned payoff facilitate transfer across different set or number of players? and (iii) Can the learned network emergence game serve as an effective predictive model to perform strategic prediction tasks? We comprehensively answer these questions in the positive by analyzing reward to explain a strategic network, performing in-domain transfer for trade and movie networks and evaluating strategic link prediction performance across different networks.

7.2 Preliminaries

7.2.1 Markov Network Emergence Game

We consider an n-player Markov Network Formation Game, where agents form links with each other to maximize their individual payoffs. The game is played in a sequential manner, where at each step, the agents announce the links they want to form and their current strategies are dependent only on the current state of the network - the Markov property. A general n-player Markov game [184] is defined as $(\mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \{r_i\}_{i=1}^n, \mathcal{P}_T, \nu, \gamma)$ where $\mathcal{S}$ is the state space and $\mathcal{A}_i$ is the action space for agent i. The function $\mathcal{P}_T : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \rightarrow \mathcal{P}_T(\mathcal{S})$ specifies the transition process between states, where $\mathcal{P}_T(\mathcal{S})$ denotes probability distribution over set $\mathcal{S}$. $\nu \in \mathcal{P}_T$ specifies an initial state distribution and γ is the discount factor. A state s_t of the game at time t transitions to state s_{t+1} with probability $\mathcal{P}_T(s_{t+1}|s_t, a_1, ..., a_n)$ due to agents' actions $\{a_1, ..., a_n\}$. Each agent i receives a reward given by the function $r_i : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n$. We use bars to indicate joint quantities over all agents – $\bar{\pi}$ denote the joint policy, $\bar{r}$ denote rewards of all agents and $\bar{a}$ denote actions of all agents. Further, subscript $-i$ denotes all agents other than i – the tuple $(a_i, \bar{a}_{-i})$ denote actions of all agents. Each agent i aims to maximize its individual payoff $\mathbf{u}_i$, instantiated by the expected sum of discounted rewards, $\mathbf{u}_i = \mathbb{E}_\pi \left[\sum_{t=1}^T \gamma^t r_{i,t}\right]$, where $r_{i,t}$ is the reward received t steps into the future by agent i. Each agent selects actions according to its stochastic policy $\pi_i : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A}_i)$, where $\mathcal{P}(\mathcal{A}_i)$ is the distribution over agent i's actions space. Further, for each agent i, the expected return for a state-action pair is defined as: $\mathbf{u}_i^{\pi_i, \bar{\pi}_{-i}}(s_t, \bar{a}_t) = \mathbb{E}_{s_{t+1:T}, \bar{a}_{t+1:T}} \left[\sum_{l \geq t} \gamma^{l-1} r_i(s_l, \bar{a}_l) | s_t, \bar{a}_t, \bar{\pi}\right]$. **Network Emergence Game.** We consider a partially observable Markov Game, where each agent has access only to its local observations. For a set of n interacting agents, let $\mathcal{G} \subset \{0, 1\}^{n \times (n-1)}$ denote the set of all feasible networks. Let $G = (\mathcal{V}, \mathbf{A}, \mathcal{X}) \in \mathcal{G}$ specify one such feasible network, where $|\mathcal{V}| = n$ is the set of agents (vertices) in the game, $\mathbf{A}$ is the adjacency matrix specifying the link structure between agents and and $\mathcal{X}$ denote the feature set for agents. Below we out-

line the decision process that incorporates the standard game-theoretic network emergence dynamics:

State $\mathcal{S}$: The state of the game s_t at any time t is the graph structure $G_t = (\mathcal{V}, \mathbf{A}_t, \mathcal{X})$, where $\mathbf{A}_t$ contains information of the graph structure at time t. $G_0 = (\mathcal{V}, \mathbf{A}_0 = \mathbf{0}, \mathcal{X})$ defines the initial state of the game s_0. Any agent i can only access its local observation $o_{i,t} = (\eta_{i,t}, \mathcal{X}_{\eta_{i,t}}) \in \mathcal{O}$ from the game's overall state $s_t \in \mathcal{S}$, where $\eta_{i,t} = \{j | \mathbf{A}_t^{ij} = 1\}$ is the neighborhood of agent i at time t. **Action** $\mathcal{A}$: A step t in the game involves each agent announcing their intentions to form the links with other agents. We map this action to a continuous low-dimensional vector ($\mathbf{a}_{i,t} \in \mathbb{R}^d$ where $d \ll n$) that represents an agent's intention to form a link. The vectors announced by each agents are then matched externally (details discussed later) to compute the links that finally emerge out of that step. This action space is inspired from recent work on Geometric Network Creation Games [54]. Our continuous action space is independent of the number of nodes which facilitates scalability. **Transition Dynamics** $\mathcal{P}_T$: Let $\bar{e}$ denote set of joint edge operations that are derived from joint action profile $\bar{a}$ for all agents at state s_t. The transition function $\mathcal{P}_T$ is defined such that the $\bar{a}$ profile obtained for all agents at state $s_t = (\mathcal{V}, \mathbf{A}_t, \mathcal{X})$ produces the next state $s_{t+1} = (\mathcal{V}, \mathbf{A}_{t+1}, \mathcal{X})$. Here, $\mathbf{A}_{t+1} = \mathbf{A}_t \odot \bar{e}$ where $\odot$ represents all the edge operations (such as creation, maintenance or severance of an edge for the game) applied to adjacency $\mathbf{A}_t$ to modify the network structure. **Reward** r: A key objective of our work is to infer agent's individual payoff from the observed network and hence we do not impose any specific functional form on the reward function, instead learn the reward function r_i corresponding for each agent i directly from the data. The reward parameterization controls the shared properties of the utility function across agents. Further, we consider the localized utility setting where the reward $r_i(o_{i,t}, a_{i,t})$ for agent i is computed only with respect to its current neighborhood. We outline the parameterization of the reward function in next section.

7.2.2 Solution Concept for Network Emergence Games

Network emergence games focus on analyzing the construction of equilibrium networks where no agent want to locally change the network [53]. To setup our reinforcement learning (RL) procedure, the first step is to specify an appropriate equilibrium concept for characterizing the trajectories distribution induced by the reward function. We focus on Markov Perfect Equilibrium (MPE) [185, 186, 187, 188], as it directly relates to this work and discuss others in Appendix E.4. This concept has been investigated and formalized in recent works in stochastic game formulation for network emergence [58, 189]. MPE admits properties that map directly to reinforcement learning setting, as discussed below.

Markov Perfect Equilibrium. The definition of network emergence policy π_i for each agent i concretely specifies agents' Markov strategies – a Nash equilibrium (NE) in which is referred as Markov Perfect Equilibrium. While existence of an MPE has been long established for stochastic games [190, 191, 192], a straightforward application of Bellman's optimality principle [193] shows that solving for MPE directly maps to recursive procedure of learning optimal joint policy $\bar{\pi}^*$ by optimizing individual reward r_i using the RL procedure (details provided in Appendix E.1). However, solving for MPE requires solving for NE at each state, which is not amenable to learning due to discontinuous characteristics of NE with respect to payoff matrix. Further, NE assumes all agents to be perfectly rational which is often not the case for agents participating in real-world network emergence. Both these difficulties are addressed by Quantal Response Equilibrium (QRE) and its logistic version [194, 195], which is stochastic generalization of NE. Specifically, QRE accounts for bounded rationality using a parameter λ and models payoff matrices injected with noise, thereby introducing smoothness useful for gradient based approaches [196]. For stochastic games, QRE has been extended to Markov version by [197, 189] and referred as Markov Quantal Response Equilibrium (MQRE).

Logistic Markov Quantal Response Equilibrium. The most interesting version of MQRE is its logistic version (MLQRE) which arise from the noise that is i.i.d accord-

ing to Gumbel distribution with parameter $\lambda \in \mathbb{R}^+$, which also controls the rationality of agents. An MLQRE $\bar{\pi}^*$ can then be expressed in closed form as a solution to the following system of equations: For all states $s_t \in \mathcal{S}$, all agents i and all actions $a \in \mathcal{A}_i$: $\bar{\pi}_i^*(a_i|s) = \frac{e^{\lambda \cdot u_i(s,a,\bar{\pi}_{-i}^*)}}{\sum_{a' \in \mathcal{A}_i} e^{\lambda \cdot u_i(s,a',\bar{\pi}_{-i^*})}}$ and $V_i^*(s) = \sum_{a' \in \mathcal{A}_i} \bar{\pi}_i^*(a_i|s) \cdot u_i(s, a', \bar{\pi}_{-i}^*)$, where u_i is the expected payoff from playing action a for agent i in state s, given strategies of other players and is expressed as :$u_i(s, a, \bar{\pi}) = r_i(s, a, \bar{\pi}) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a_i, \bar{pi}_{-i}) \cdot V_i(s')$. In the quantal response framework, agent i is assumed to perceive noise injected payoff version of this expression as $\hat{u}_i(s, a, \bar{\pi}) = u_i(s, a, \bar{\pi}) + \varepsilon_i(s, a)$. λ can be interpreted as the precision with which the agents perceive the payoffs. When $\lambda \to 0$, the equilibrium is fully noisy and the agents with select actions uniformly at random. When $\lambda \to \infty$, the agents will choose actions in best response manner (greedily) and [189] has shown that its limit point converges to Markov perfect equilibrium. That is, $\bar{\pi}^{**} = \lim_{\lambda \to \infty} \bar{\pi}^*(\lambda)$ is a Markov perfect equilibrium for some LMQRE $\bar{\pi}^*(\lambda)$. This establishes LMQRE as an appropriate equilibrium concept to use in the RL setting for solving Markov Perfect Network emergence game which is our case. We discuss more details on MQRE, its convergence to Markov Perfect equilibrium and properties of π_i in the Appendix E.1.

7.2.3 Multi-Agent Inverse Reinforcement Learning

Most game-theoretic methods hand-design a reward function for theoretical analysis, but for real-world networks, it is often difficult to specify the explicit form of such reward mechanism. One of the key contributions of this paper is to discover this reward function from the observed networks. Specifically, our goal is to jointly learn the strategy profile for agents that can reconstruct the observed network and discover the latent payoff function. To achieve this, we leverage maximum entropy (MaxEnt) bases inverse reinforcement learning (IRL) [151] which aims to learn a reward function that rationalizes the expert behaviors with least commitment. Let $\mathcal{D}$ denote expert demonstrations provided by n experts in the n-player Markov game. $\mathcal{D}$ is realized as a set of M trajectories $\{\tau_j\}_{j=1}^M$, where

$\tau_j = \left\{(s_j^t, \bar{a}_j^t)\right\}_{t=1}^{T}$ denote an expert trajectory collected by sampling $s^1 \sim \nu(s), \bar{a}^t \sim \pi_E(\bar{a}^t|s^t), s^{t+1} \sim P(s^{t+1}|s^t, \bar{a}^t)$. $\mathcal{D}$, obtained from observed graph for network emergence game, contains all the available supervision for the learning procedure. Denoting Max-Ent IRL function as IRL(π_E), we have: IRL$(\pi_E) = arg\max_{r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \mathbb{E}_{\pi_E}[r(s,a)] - RL(r)$, where $RL(r) = \max_{\pi\in\Pi} \mathcal{H}(\pi) + \mathbb{E}_\pi[r(s,a)]$ and $\mathcal{H}(\pi) = \mathbb{E}_\pi[-\log \pi(a|s)]$ is the policy entropy.

For this work, we consider MA-AIRL [183], a recently proposed IRL algorithm for multi-agent setting. MA-AIRL uses Logistic Stochastic Best Response Equilibrium (LSBRE) as a solution concept to characterize the trajectory distributions induced by the reward functions of agents $\{r_i(s,\bar{a})\}_{i=1}^N$. Given a Markov game with horizon T, LSBRE is defined as a sequence of T stochastic policies $\{\pi^t\}_{t=1}^T$, where each joint policy $\bar{\pi} : \mathcal{S} \to \mathcal{P}(\mathcal{A}_1 \times ... \times \mathcal{A}_n)$ is given by:$\bar{\pi}^t(a_1, ..., a_n|s^t) = P\left(\bigcap_i \{z_i^{t,(\infty)}(s^t) = a_i\}\right)$. Here, z_i is a mapping from state to action expressed as:$z_i^{t,(k+1)}(s^t) \sim P_i^t(a_i^t|\bar{a}_{-i}^t = \bar{z}_{-i}^{t,(k)}(s^t), s^t) = \frac{\exp\left(\lambda Q_i^{\bar{\pi}^{t+1:T}}(s^t, a_i^t, \bar{z}_{-i}^{t,(k)}(s^t))\right)}{\sum_{a_i'}\exp\left(\lambda Q_i^{\bar{\pi}^{t+1:T}}(s^t, a'^t_i, \bar{z}_{-i}^{t,(k)}(s^t))\right)}$, where $\lambda \in \mathbb{R}^+$ is noise parameter controlling rationality of agents and $\{P_i^t\}_{i=1}^N$ specifies set of conditional distributions. We specifically note the close relation of the form of z_i in LSBRE with that of π_i in MLQRE that allows us to design our practical algorithm building on MA-AIRL [183] while using LMQRE as a solution concept for network emergence games. We discuss more details on connection between MLQRE and LSBRE in the Appendix E.1.

7.3 Proposed Model

In this section, we first describe the architecture details of the MINE model and then outline the learning procedure that jointly learns both the reward function and the network emergence policy.

Architecture. Being a network emergence game, the performance of our agents depend on effectively learning over the graph structure of the problem. To achieve this, we design a graph neural network [167] based policy network that embeds the observation into

a continuous vector space, further processed by the next layers of policy network to output the action vectors. We outline the details on our structured strategy network below along with the specifications on environment updates based on the continuous action vector. Further, we also discuss the parameterization of learned reward function and its connection to conventional game-theoretic approaches.

Structured Strategy Network. We leverage structured policy network [166] to design and implement a mapping function from state $s_t = G_t$ to the embedding matrix $\mathbf{H} \in \mathbb{R}^{n \times d}$ where each row of the matrix represent the embedding $\mathbf{h}_i$ for agent i. The mapping starts from the initial agent features $\mathbf{h}_i^{(0)}$ which are problem dependent. A P-step message passing procedure updates these features by aggregating information from P-hop neighborhood of agent i as follows: $\mathbf{h}_i^{(p+1)} = U\left(\mathbf{h}_i^{(p)}, \left\{(m_{ij}, \mathbf{h}_j^{(p)})\right\}_{j \in \mathcal{N}(i)}\right)$, where U is the update function, m_{ij} is the message on the edge between agents i and agent j and $\mathcal{N}(i)$ is the neighborhood agent set for agent i. The update to all agents' embeddings occur at the end of the environment step and the the policy function takes the agent i specific local observations (now mapped to embeddings) as input to compute the action for agent i. For computing a graph/subgraph embedding, we use an attention based aggregation of the participant agents' embeddings from last message passing iteration $\mathbf{h}_i^{(P)}$. Next, we design a stochastic policy that takes as input observation $\bar{o}_{i,t}$ and outputs a link formation action $a_{i,t}$ for agent i. Under a Gaussian policy π_ϕ, the next action is computed as follows: $[\boldsymbol{\mu}, \log(\boldsymbol{\sigma^2})] = \pi(s_t) = \alpha(g_{\phi_i}(\bar{o}_{i,t}))$ and $\mathbf{a}_{i,t} \sim \mathcal{N}(\boldsymbol{\mu}, \log(\boldsymbol{\sigma}^2))$, where g_{ϕ_i} is a two layer MLP with the policy parameters ϕ_i for agent i and α is the activation function. External to the policy network, we map the action vector $\mathbf{a}_{i,t}$ for all agents to discrete edge operation set $\bar{e}$ described in Section 2.1, making our approach fully differentiable: We interpret the value of action vector $\mathbf{a}_{i,t} \in \mathbb{R}^d$ as a direct prediction of agent j's embedding with which agent i wants to form a link. It is possible that the action vector of an agent i maps to action vectors of multiple agents j. We capture this insight by first stacking action vectors of all the agents into action matrix $\mathbf{a} \in \mathbb{R}^{n \times d}$. We then compute the prob-

abilities of forming a link between two agents as: $\mathbf{a}_{\text{prob}} = \sigma(\mathbf{a}^T\mathbf{a})$, where $\mathbf{a}_{\text{prob}} \in \mathbb{R}^{n\times n}$ is the link probability matrix and σ represents sigmoid function. Subsequently, the entire environment structure is updated by modifying the adjacency matrix $\mathbf{A}_t$ as follows: $\mathbf{A}^{ij}_{t+1} = 1$ if $\mathbf{a}^{ij}_{\text{prob}} > 0.5$ and $\mathbf{A}^{ij}_{t+1} = 0$ otherwise. Analogous to game-theoretic approaches – a link is formed/maintained in the next step between two agents only with mutual consent (i.e. both the agents i and j select actions close to each other in the latent space). If either agent do not consent, the link is never formed or severed if it existed in $\mathbf{A}_t$ previously.

Neural Payoff Mechanism. We use a novel parameterization for the reward function r_i for each agent i, inspiring from the game-theoretic insights where the payoff is designed with an assumption that the agents optimize their position in the network. We use a local 1-layer GNN to compute observation input ($\bar{o}_{i,(t+1)}$) to the reward function for agent i. As the actions effectively map to (choice of) agents, we are able to use the state only reward function. Further, the observations for all agents would have been updated based on their actions before computing the reward and hence each agent will have access to other agent's actions and its outcomes (strategy) locally.

Learning. We design an efficient training procedure for learning network emergence games by building on the recently proposed multi-agent adversarial reinforcement learning (MA-AIRL [183]) algorithm. For matching the efficiency and scalability demands of learning in the challenging graph-structured environment, we modify the the original algorithm to use multi-agent attention actor critic (MAAC [198]) to solve the inner RL loop of the MA-IRL algorithm. We further account for the graph structured environment by modifying the critic to use graph attention networks [199]. Algorithm 1 in Appendix E.2 outlines complete training procedure. An important consideration for the inverse methods is the extraction of expert demonstrations. Unlike conventional RL environments, where the expert demonstrations are readily available, for observed graphs, we only have access to final graph structure (experts' outcome). Hence, we need to extract useful and valid trajectories. Following previous multi-agent IRL work, we also extract joint trajectories of

each construction in the graph where at each step of the trajectory, we sample an edge for each agent either via random permutation order or BFS ordering on the entire graph and use it to define the action of agents. While such expert trajectories will only show growing graph, the action space of learning agents still need to consider severance to account for wrong edges that they create which they need to remove over time.

7.4 Experiments

In this section, we provide insights into the important aspects of learning network emergence games. First we demonstrate the ability of MINE to discover a paypoff mechanism that has high correlation to the ground truth game-theoretic utility and then use a toy real-world network to illustrate that MINE is able to recover the strategic links in the observed network using the learned policy. We further assess the interpretability benefits by qualitatively analyzing the learned reward behavior with respect to the observed network structure. Finally, we evaluate the transfer properties of MINE across different settings. We conclude our experiments by demonstrating MINE's capability to facilitate effective prediction of future strategies (links) of agents given a state of the network. We provide more details on experimental setup in Appendix E.3 and dataset statistics in Table 7.2(c). Code and Data will be made available upon publication.

7.4.1 Payoff Function

MINE learns the underlying payoff mechanisms from the observed networks and hence it is important to evaluate its ability to learn a meaningful payoff function useful for interpretation and transfer purposes. Below, we outline our analysis of the learned reward with respect to these properties:

Quality. To evaluate the quality of the learned reward, we perform two different experiments: For real-world networks, we do not have access to the ground truth utility that was originally optimized by the involved players. Hence, we first perform a synthetic ex-

Table 7.1: Analysis of the learned reward using: **(a)** Game-theoretic reward function **(b)** Zachary Karate club data (no ground truth reward). Correct links are fraction of original links recovered.

	Agent#1	Agent#2	Agent#3	Agent#4	Agent#5	Average
PCC	0.842	0.928	0.883	0.763	0.681	0.8194
Expert	-7.213	-12.221	-10.65	-6.441	-12.294	-
MINE	-8.331	-12.252	-10.31	-8.045	-10.797	-

(a)

	Leader#Red	Leader#Purple	Community#Red	Community#Purple
Correct Links	72	77	75	81
Original	-132.33	-83.42	-99.98	-74.61
Policy	-141.71	-85.22	-112.32	-77.04
Perturbed	-221.4	-118.93	-199.74	-101.65

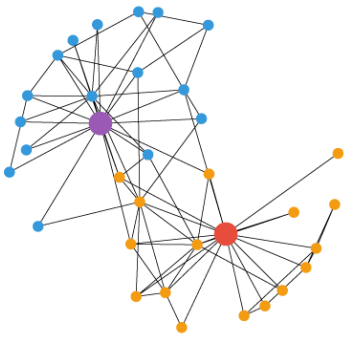
(b)

periment, where an expert is trained to optimize a specific form of a game-theoretic reward function using the inner MAAC algorithm of MINE (no reward learning). Specifically, we use the following form of the reward inspired from game-theoretical model of social network emergence [200]:

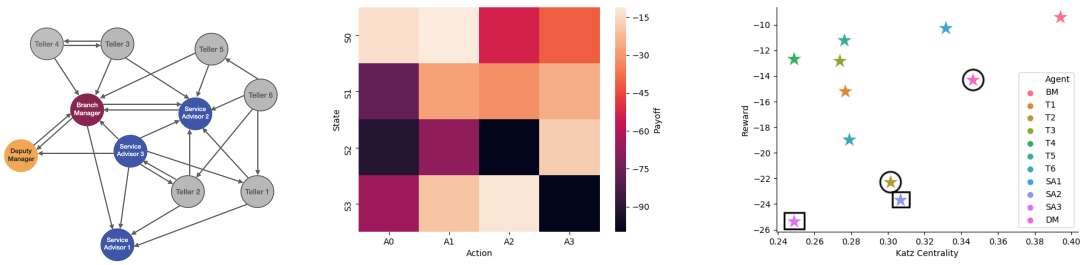
$$r_i(o_i) = \sum_{j \in \mathcal{N}(i)} \left(\sum_{k=1}^{K} b_k \max(z_{jk} - z_{ik}, 0) - \|\mathbf{c} \odot (\mathbf{z}_j - \mathbf{z}_i)\|_2 \right),$$

where, $\mathbf{b}$ and $\mathbf{c}$ are benefit and cost parameters respectively with fixed values and k is dimension of agent embeddings $\mathbf{z}$. We perform this experiment with $N = 5$ agents that play the game defined by MINE's MDP but optimize the above reward. We report the correlation (Pearson Correlation Coefficient (PCC)) between the learned and the expert reward and show comparison between expected returns of the learned and the expert policies. Results in Table 7.1(**a**) demonstrate that MINE successfully learns a reward function that has high correlation with a ground-truth game-theoretic utility. Further, the learned policies that optimizes this reward imitates the experts well.

Figure 7.1: Karate network



Next, we consider a toy real-world network of Zachary's karate club (34 agents, 78 links) that contains two clearly different communities (Figure 7.1). For learning, we extract expert trajectories as described in previous section. We evaluate the learned policy under following criteria: fraction(%) of correct links recovered (Table 7.1(**b**) top row) and policy performance in terms of expected return (Table 7.1(**b**) bottom 3 rows). Community#Red and Community#Purple results are averaged across their follower vertices respectively. The



(a) (b) (c)

Figure 7.2: Payoff interpretability in relation to the real-world Australian Bank network. **(a)** Observed Network (Darker nodes have more importance). **(b)** Marginal Payoff heatmap (lighter color signify higher utility) for state-action pairs where state is a single node of particular type and action is the link formation with a new node: (S0): Teller, (S1): Service Advisor, (S2) Deputy Manager and (S3) Branch Manager **(c)** Payoff behavior for each agent with respect to its Katz centrality in the network.

first row demonstrates that MINE recovers a significant portion of strategic interactions (links). To substantiate that this is not the result of mimicking only the structural properties, we report the utility values for 3 different network states: original graph, policy generated graph and perturbed graph, where the perturbation swaps two leaders with their followers. This preserves structural configuration of the graph but results in strategically different network. Table 7.1(**b**) bottom rows show that network emerging as a result of learned policy has closer behavior to the original graph compared to the perturbed graph under the learned payoff, thereby confirming the vital role of learned objective in recovering real-world strategies.

Interpretability. We consider Australian bank dataset (Figure 7.2(a)), a network of strategic confiding relationships between branch personnel representing hierarchy among the employees. We first study how the the learned payoff can be interpreted with respect to the strategic behavior of agents. After training, we compute marginal utility of individual agents for proposing an action (link formation choice w.r.t other agents). In the heatmap (Figure 7.2(b)), A0 to A3 are signify actions (chosen agents). In real-world setting, agents often confide in other agents at the same or next level higher up but not otherwise. The heatmap demonstrates similar properties of the learned reward (e.g. a Teller (S0) receives high payoff for proposing a link with other Tellers (A0) and Service advi-

sors (SA) (A1) but not with managers). However, this behavior changes towards the top of hierarchy where clustering behavior is not observed due to fewer agents and confiding relationships become reciprocating with agents at lower level, which is also captured by the learned reward (e.g. Branch manager (BM) (S3) gets high payoff for proposing link with both Deputy manager (DM) (A2) and SA (A1)).

We also investigate how the learned payoff relates to the strategic importance of the agents in the network in terms of Katz centrality (KC) [201], a widely used measure in game-theoretic payoff functions [202]. Each agent in the observed network (Figure 7.2(a)) has importance attribute not based on KC. After training, we modify the network such that it affects the KC of particular agents while keeping others same and then compute the state-only reward for each agent (Figure 7.2(c)). Tellers with low KC get high rewards, explained by peripheral roles of tellers. But we modified the local structure of one teller (T2, golden star in Figure 7.2(c)) to increase its KC, keeping its (low) importance attribute same. T2 gets low utility value owing to contrast to its (low) importance value. Further, in spite of having lower KC value for Branch Manager than SA in observed network, BM gets high reward than SA. This shows that MINE captures intricate properties beyond structure that may not be modeled by hand-designed specifications. For instance, KC considers the entire network however, in real-world, an agent often only have access to its local observations. Finally, we modified the observed network to decrease the KC for SA2 such that its incoming links are removed. As expected, in spite having high importance values, SA2 gets a low reward value for not being involved in confiding with tellers.

Transfer. In this section, we evaluate the ability of learned payoff to facilitate effective transfer across games. For all the experiments, we first train our policy and reward functions on source network (that provides expert demonstrations). We then transfer the learned reward function to the target graph, where the policy is re-trained to optimize the transferred reward. For comparison, we train the full model on the target network (no reward transfer). Additionally, we perform an experiment to evaluate the generalization ca-

Table 7.2: **(a)**Transfer Performance (Top row: transfer across #agents, Bottom 2 rows: transfer across set of agents). **(b)** Strategic Link Prediction Performance: Number are AUC. **(c)** Dataset Statistics.

Dataset	**Correct Links**	**PCC**	**Training Episodes**
Andorra			
Target Trained Reward (No Transfer)	76.5	-	100000
Source Trained Reward (Transfer)	70.2	0.681	72000
Policy Transfer	68.88	0.59	-
Trade			
Target Trained Reward (No Transfer)	87.2	-	25000
Source Trained Reward (Transfer)	81	0.848	18000
Policy Transfer	62.45	0.648	-
Movie			
Target Trained Reward (No Transfer)	62.54	0.712	60000
Source Trained Reward (Transfer)	53.09	0.631	45000
Policy Transfer	51.1	0.598	-

(a)

Methods	**Trade**	**Company**	**Arxiv GR-QC**
GT_core	0.834	0.762	0.943
Social Game Embed	0.968	0.987	0.857
svII	0.821	0.774	0.658
Seal	0.971	0.933	0.912
Graphite	0.942	0.889	0.823
MINE	0.91	0.819	0.855

(b)

Datasets	**Nodes**	**edges**
Andorra	32,829	513,931
Trade	100	703
Company	1984	12,751
Movie	2788	10,399
Arxiv GR-QC	5242	14496

(c)

pacity of the learned policy. Specifically, we train a full model on source network and then directly evaluate the learned policy on target network without re-training (zero-shot generalization). We report our results on three criteria: fraction of correct links recovered on target network, correlation between the policy performance between target trained model and transferred models (both retrained one and policy transfer) and number of training episodes for convergence.

Transfer across different network sizes– To this end, we consider Andorra phone call network with attributes such as phone type (apple, Samsung and others), location and internet usage. We train MINE on a sub-network of 100 agents to learn the payoff (source training). We then fix the learned payoff and re-optimize the policy over the full network (transfer). Table 7.2(a) top row demonstrates successful transfer with an on par performance compared to the model fully trained on the large network and notably requires lesser episodes, thereby providing speedup. *Transfer across different set of players*– We consider two strategic networks from different domains: a trade network between countries and bipartite movie network between directors and cast. We spilt each network into two connected components disjoint set of agents. We train MINE on one component (source training) and transfer it to the other. Table 7.2(a) showcases highly competitive transfer performance for Trade data, signifying its applicability to extract a useful strategic mecha-

nism from an observed network to train network games with different configurations. For the movie network, the performance degrades slightly and we suspect it is due to its bipartite nature that requires further constraints in the model.

Finally, the policy transfer performs worse than re-optimizing the policy on target network. Nevertheless, its moderate success (with no re-training) has applications in scenarios where quick testing may be useful first step. This usefulness of MINE for zero-shot generalization is due to the use of GNN for encoding agent representations that generalize across unseen states of the environment.

7.4.2 Strategic Prediction

In this section, we evaluate the ability of learned network emergence games as a model to support meaningful strategic predictions. Concretely, we focus on classical link prediction task that forms the basis of many further network analysis tasks. We consider three networks from different domains: a financial trade network, a company network of communication between members at different hierarchy and a co-authorship network of General relativity and Quantum Cosmology field. We split the networks into train (80%) and test(20%) edge sets and train a reward and policy over the training edges. At convergence, we rollout the evaluation policy asking agents to form links between them. We report the standard link prediction metrics Area under the curve (AUC) in Table 7.2(b). For baselines, we use 3 game theoretic approaches for link prediction as a direct comparison: GT_core [203] and Social Game Embed [200] that combines network embedding approaches with game-theoretic payoff functions and svII [204], a recently proposed similarity measure used to perform link prediction based on agent similarities. For completeness, we compare with a state-of-art discriminative model SEAL [25] and generative model Graphite [205] for link prediction. The results in Table 7.2(b) demonstrate that a learned MINE model has strong predictive capabilities that often outperforms or achieves comparable performance to stylized game-theoretic approaches. The high performance of Social Game Embed for Trade

and Company dataset is attributed to its dataset specific payoff function but its performance degrades on dataset which uses a different strategy than a social game while MINE demonstrates consistently good performance. The superior performance of learning baselines for link prediction is expected as these baselines use a task-dedicated architecture and training objective, which stands in contrast to MINE, which discovers the objective from the observed network and learns a generic network emergence strategy model. This demonstrates compelling generalization properties of MINE that is coupled with the interpretability benefits, both absent in deep learning approaches such as the dedicated baselines. Finally, none of the above approaches facilitate seamless transfer across new players which makes our approach versatile.

7.5 Summary

In this paper, we investigate the problem of learning network emergence games directly from the observed networks without any assumptions on the underlying strategic mechanisms. We propose, MINE, a data-driven learning framework that incorporates Markov-Perfect Network emergence game dynamics into its sequential decision process formulation and solves it using multi-agent reinforcement learning. MINE jointly discovers agents' strategy profiles in the form of learned policy and the latent payoff mechanism in the form of learned reward function. Our experimental evaluation of the predictive, transfer and explanatory properties of MINE demonstrates that MINE successfully combines the interpretability benefits of game-theoretic frameworks with the practical applicability of learning approaches.This opens up new avenues for research on leveraging game theoretic approaches to build interpretable and transferable learning frameworks for network analysis.

CHAPTER 8
CONCLUSION

Graphs are ubiquitous representation of information across domains such as social networks, knowledge graphs, financial systems, protein-protein networks and many more. Machine Learning over graph structured data has enjoyed successful advancements in the past decade with the increasing availability of standardized benchmarks, proliferation of conceptual models and a simultaneous upsurge in the application domains that leverage structured knowledge for decision-making. Given the versatile presence of graph structures, it is important to study the processes that leads to the emergence or governs the evolution of these graphs. In this thesis, we follow the principle of tightly coupling the recent advancements in machine learning over graphs with the classical modeling approaches. We leverage this principle to build machine learning frameworks for modeling, learning and inferring such processes over graph structured data. Specifically, we make following contributions:

8.1 Contributions

As a part of this thesis, we build both generative and discriminative modules to learn over graph structured data. In the generative case, we focus on modeling evolution process of dynamic networks and learning dynamic node representations that evolve over time. This work has been specialized to perform temporal reasoning over multi-relational dynamic knowledge graphs. Further, we focus on the process of network formation and build learning approaches that aim to discover global optimization models of graph formation and learn local strategic network emergence mechanisms. Finally, in the discriminate case, we propose a deep relational learning architecture that can jointly perform representation learning and entity linkage over multiple graph sources in an end-to-end fashion.

Dynamics. For modeling and learning dynamic graphs, we propose a novel deep learning framework grounded in rich mathematical model of temporal point process and show that temporal point process based graph evolution model supports capturing fine-grained temporal dynamics. We also propose novel two-time scale modeling of dynamics, localized embedding propagation and temporal attention all of which serve as useful inductive biases for our deep learning architecture. Finally, we demonstrate that the proposed work is useful for any event based applications and supports accurate event based predictions for variety of domains that exhibit temporal evolution.

Global Formation Mechanisms. We investigate the implications of adopting the optimization perspective for building graph learning approaches and found that modeling optimization mechanisms of graph formation is a promising approach for learning over graphs. Modeling these mechanisms allow for building approaches that function effectively in a non-probabilistic setting and lend itself to transfer within a given domain. In this work, we specifically focus on discovering global optimization mechanisms that govern the formation process of graphs and demonstrate that models learned with these properties generalize well to prediction task and further serve as an effective graph constructor.

Local Strategic Mechanisms. We further our investigation into building optimization based network learning approaches to the setting where agents are strategic and rational and they participate in network formation process with the aim of optimizing local utility or benefits. Subsequently, we focus on network emergence games studied extensively in game theory and take it to the data for supporting practical applications on real-world data. Our approach incorporates network game dynamics explicitly into the learning framework jointly tasked to learn payoff mechanisms and strategy profiles. We demonstrate that our approach learns interpretable and transferable mechanisms and the learned game as a model is useful to perform strategic link prediction.

Multi-graph Representation Learning. Many data driven organizations such as Google and Microsoft take the approach of constructing a unified super-graph by integrating data

from multiple sources. Such unification has shown to significantly help in various applications, such as search, question answering, and personal assistance. To this end, there exists a rich body of work on linking entities and relations, and conflict resolution. Still, the problem remains challenging for large scale knowledge graphs and this paper proposes a deep learning solution that can play a vital role in this construction process. To address this, we propose a novel relational learning framework that learns entity and relationship embeddings across multiple graphs. The proposed representation learning framework leverage an efficient learning and inference procedure which takes into account the duplicate entities representing the same real-world entity in a multi-graph setting. We demonstrate superior accuracies on link prediction and entity linkage tasks compared to the existing approaches that are trained only on individual graphs. In real-world setting, we envision our method to be integrated in a large scale system that would include various other components for tasks like conflict resolution, active learning and human-in-loop learning to ensure quality of constructed super-graph.

As the focus of this thesis is to build approaches that tightly couple classical modeling approaches/insights with deep learning techniques for networks, several of these works adopt a novel perspective compared to the prevalent ones in the graph learning community. This contributes towards opening up several new research directions and avenues. While our work on formation mechanisms is fairly recent and demonstrates the promise of high impact on graph learning community, both our earlier work on dynamic graphs and multi-graph representation learning have received have gained significant attention both in academia and industry. Our work on dynamic graphs have forged a line of follow up works on modeling dynamic processes over graphs and designing various temporal attention based architecture. Simultaneously, our work on multi-graph representation learning has led to a multi-year project at Amazon on using representation learning for unifying graphs from different sources and constructing Amazon Product Graph, a billion-entity scale knowledge graph of Amazon products.

8.2 Limitations and Future Work

The overall theme of this thesis is to build a synergy between classical approaches that analyze and study networks and the success in learning over complex graph structures with an aim to combine the benefits of both worlds specifically focusing on moving the needle towards generalizable and interpretable approaches to learning over graphs in heterogeneous settings. While our contributions take several initial steps in this direction, there are lot of open problems and challenges that remain to be solved as a part of future work in each of the proposed directions:

Dynamics. Our current work on modeling dynamic graphs does not support shrinkage due to following reasons: (i) It is difficult to procure data with fine grained deletion time stamps and (ii) The temporal point process model requires more sophistication to support deletion. Another interesting future direction could be to support encoding higher order dynamic structures. Further, sophisticated approaches that can account for birth and death process, multi-time scale dynamics, causal interventions and many more characteristics need to be researched and developed for supporting decision critical applications.

Optimization Mechanisms Viewpoint. Our work on building learning approaches based on optimization viewpoint is a novel approach for graph learning approaches and hence we expect a several follow up work that adopt this viewpoint and focus on different tasks. For immediate extensions, our current approach learns from single graph and hence the next step will be to support multiple input graphs when samples from distribution over graphs is available. Further, we start out with supporting general objective functions, but it would be interesting to specialize these methods to specific applications that require constrained learning. Finally, our current work only relies on the final outcome observation but it would be useful to build approaches that support learning from construction process whenever it is available atleast partially in the form of evolving graphs.

Learning over multiple graphs. Our current work in this direction mainly focuses on

aligning two graphs while jointly learning representations over them. For future work, it would be interesting to extend the current evaluation of our work from a two-graph setting to multiple graphs. A straightforward approach is to create a unified dataset out of more than two graphs by combining set of triplets, and apply learning and inference on the unified graph without any major change in the methodology. Alternatively, one can develop sophisticated approaches with iterative merging and learning over pairs of graphs until exhausting all graphs in an input collection. Further, more sophisticated learning methods based on graph matching and optimal transport can be leveraged to conduct sophisticated learning tasks over multiple graphs.

Appendices

APPENDIX A

RELATIONAL LEARNING OVER MULTI-SOURCE KNOWLEDGE

A.1 Discussion and Insights on Entity Linkage Task

Entity linkage task is novel in the space of multi-graph learning and yet has not been tackled by any existing relational learning approaches. Hence we analyze our performance on the task in more detail here. We acknowledge that baseline methods are not tailored to the task of entity linkage and hence their low performance is natural. But we observe that our model performs well even in the unsupervised scenario where essentially the linkage loss function is switched off and our model becomes a relational learning baseline. We believe that the inductive ability of our model and shared parameterization helps to capture knowledge across graphs and allows for better linkage performance. This outcome demonstrates the merit in multi-graph learning for different inference tasks. Having said that, we admit that our results are far from comparable to State-of-the-art linkage results (Das et al., 2017) and much work needs to be done to advance representation and relational learning methods to support effective entity linkage. But we note that our model works for multiple types of entities in a very heterogeneous environment with some promising results which serves as an evidence to pursue this direction for entity linkage task.

We now discuss several use-case scenarios where our model did not perform well to gain insights on what further steps can be pursued to improve over this initial model:

Han Solo with many attributes (False-negative example). Han Solo is a fictional character in Star Wars and appears in both D-IMDB and D-FB records. We have a positive label for this sample but we do not predict it correctly. Our model combines multiple compo-

nents to effectively learn across graphs. Hence we investigated all the components to check for the failures. One observation we have is the mismatch in the amount of attributes across the two datasets. Further, this is compounded by multi-value attributes. As described, we use paragraph2vec like model to learn attribute embeddings where for each attribute, we aggregate over all its values. This seems to be computing embeddings that are very noisy. As we have seen attributes are affecting the final result with high impact and hence learning very noisy attributes is not helping. Further, the mismatch in number of types is also an issue. Even after filtering the types, the difference is pretty large. Types are also included as attributes and they contribute context to relation embeddings. We believe that the skew in type difference is making the model learn bad embeddings. Specifically this happens in cases where lot of information is available like Han Solo as it lead to the scenario of abundant noisy data. With our investigation, we believe that contextual embeddings need further sophistication to handle such scenarios. Further, as we already learn relation, type and attribute embeddings in addition to entity embeddings, aligning relations, types and attributes as integral task could also be an important future direction.

Alfred Pennyworth is never the subject of matter (False-negative example). In this case, we observe a new pattern which was found in many other examples. While there are many triples available for this character in D-IMDB, very few triplets are available in D-FB. This skew in availability of data hampers the learning of deep network which ends up learning very different embeddings for two realizations. Further, we observe another patter where Alfred Pennyworth appears only as an object in all those few triplets of D-FB while it appears as both subject and object in D-IMDB. Accounting for asymmetric relationships in an explicit manner may become helpful for this scenario.

Thomas Wayne is Martha Wayne! (False-positive example). This is the case of abundance of similar contextual information as our model predicts Thomas Wayne and Martha

Wayne to be same entity. Both the characters share a lot of context and hence many triples and attributes, neighborhood etc. are similar for of them eventually learning very similar embeddings. Further as we have seen before, neighborhood has shown to be a weak context which seems to hamper the learning in this case. Finally, the key insight here is to be able to attend to the very few discriminative features for the entities in both datasets (e.g. male vs female) and hence a more sophisticated attention mechanism would help.

In addition to the above specific use cases, we would like to discuss insights on following general concepts that naturally occur when learning over multiple graphs:

- **Entity Overlap Across Graphs.** In terms of overlap, one needs to distinguish between *real* and *known* overlap between entities. For the known overlap between entities, we use that knowledge for linkage loss function L_{lab}. But our method does not need to assume either types of overlap. In case there is no real overlap, the model will learn embeddings as if they were on two separate graphs and hence will only provide marginal (if any) improvement over State-of-art embedding methods for single graphs. If there is real overlap but no known overlap (i.e., no linked entity labels), the only change is that Equation (13) will ignore the term $(1-b) \cdot L_{lab}$. Table 3 shows that in this case (corresponding to AUPRC (Unsupervised)), we are still able to learn similar embeddings for graph entities corresponding to the same real-world entity.

- **Disproportionate Evidence for entities across graphs.** While higher proportion of occurrences help to provide more evidence for training an entity embedding, the overall quality of embedding will also be affected by all other contexts and hence we expect to have varied entity-specific behavior when they occur in different proportions across two graphs

- **Ambiguity vs. Accuracy.** The effect of ambiguity on accuracy is dependent on the type of semantic differences. For example, it is observed that similar entities with

major difference in attributes across graphs hurts the accuracy while the impact is not so prominent for similar entities when only their neighborhood is different.

A.2 Implementation Details

A.2.1 Additional Dataset Details

We perform light pre-processing on the dataset to remove self-loops from triples, clean the attributes to remove garbage characters and collapse CVT (Compound Value Types) entities into single triplets. Further we observe that there is big skew in the number of types between D-IMDB and D-FB. D-FB contains many non-informative type information such as $\#base.*$. We remove all such non-informative types from both datasets which retains 41 types in D-IMDB and 324 types in D-FB. This filtering does not reduce the number of entities or triples by significant number (less than 1000 entities filtered)

For comparing at scale with baselines, we further reduce dataset using similar techniques adopted in producing widely accepted FB-15K or FB-237K. Specifically, we filter relational triples such that both entities in a triple contained in our dataset must appear in more than k triples. We use $k = 50$ for D-FB and $k = 100$ for D-IMDB as D-IMDB has orders of magnitude more triples compared to D-FB in our curated datasets. We still maintain the overall ratio of the number of triples between the two datasets.

Positive and Negative Labels. We obtain 500662 positive labels using the existing links between the two datasets. Note that any entity can have only one positive label. We also generate 20 negative labels for each entity using the following method: (i) randomly select 10 entities from the other graph such that both entities belong to the same type and there exist no positive label between entities (ii) randomly select 10 entities from the other graph such that both entities belong to different types.

A.2.2 Training Configurations

We performed hyper-parameter grid search to obtain the best performance of our method and finally used the following configuration to obtain the reported results:
– Entity Embedding Size: 256, Relation Embedding Size=64, Attribute Embedding Size = 16, Type Embedding Size = 16, Attribute Value Embedding Size = 512. We tried multiple batch sizes with very minor difference in performance and finally used size of 2000. For hidden units per layer, we use size = 64. We used $C = 50$ negative samples and $Z = 20$ negative labels. The learning rate was initialized as 0.01 and then decayed over epochs. We ran our experiments for 5 epochs after which the training starts to convert as the dataset is very large. We use loss weights b as 0.6 and margin as 1. Further, we use $K = 50$ random walks of length $l = 3$ for each entity We used a train/test split of 60%/40% for both the triples set and labels set. For baselines, we used the implementations provided by the respective authors and performed grid search for all methods according to their requirements.

A.2.3 Contextual Information Formulations

Here we describe exact formulation of each context that we used in our work.

Neighborhood Context: Given a triplet (e^s, r, e^o), the neighborhood context for an entity e^s will be all the nodes at 1-hop distance from e^s other than the node e^o. This will capture the effect of other nodes in the graph surrounding e^s that drives e^s to participate in fact (e^s, r, e^o). Concretely, we define the neighborhood context of e^s as follows:

$$\mathbf{N_c}(e^s) = \frac{1}{n_{e'}} \sum_{\substack{e' \in \mathcal{N}(e^s) \\ e' \neq e^o}} \mathbf{v}^{\mathbf{e}'} \tag{A.1}$$

where $\mathcal{N}(e^s)$ is the set of all entities in neighborhood of e^s other than e^o. We collect the neighborhood set for each entity as a pre-processing step using a random walk method. Specifically, given a node e, we run k rounds of random-walks of length l and create the neighborhood set $\mathcal{N}(e)$ by adding all unique nodes visited across these walks.

Please note that we can also use $\max$ function in (A.1) instead of sum. $\mathbf{N_c}(e^s) \in \mathbb{R}^d$ and the context can be similarly computed for object entity.

Attribute Context. For an entity e^s, the corresponding attribute context is defined as

$$\mathbf{A_c}(e^s) = \frac{1}{n_a} \sum_{i=1}^{n_a} \mathbf{a_i^{e^s}} \tag{A.2}$$

where n_a is the number of attributes. $\mathbf{a_i^{e^s}}$ is the embedding for attribute i. $\mathbf{A_c}(e^s) \in \mathbb{R}^y$.

Type Context. We use type context mainly for relationships i.e. for a given relationship r, this context aims at capturing the effect of type of entities that have participated in this relationship. For a given triplet (e^s, r, e^o), we define type context for relationship r as:

$$\mathbf{T_c}(r) = \frac{1}{n_t^r} \sum_{i=1}^{n_t^r} \mathbf{v_i^{t'}} \tag{A.3}$$

where, n_t^r is the total number of types of entities that has participated in relationship r and $\mathbf{v_i^{t'}}$ is the type embedding that corresponds to type t. $\mathbf{T_c}(r) \in \mathbb{R}^q$.

APPENDIX B

REPRESENTATION LEARNING OVER DYNAMIC GRAPHS

B.1 Pictorial Exposition of DyRep Representation Network

B.1.1 Localized Embedding Propagation

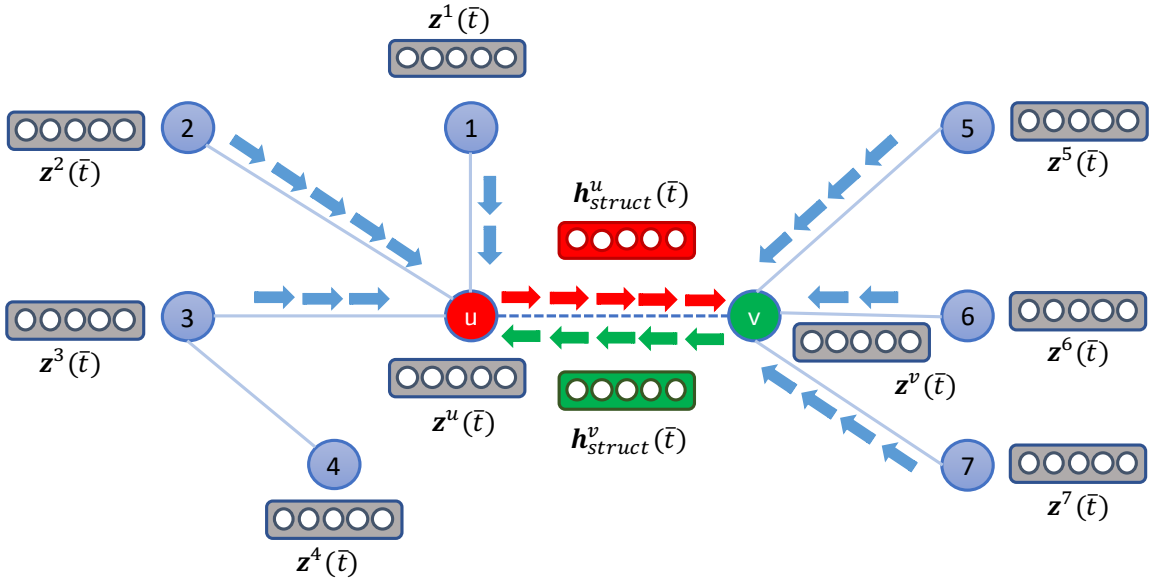


Figure B.1: **Localized Embedding Propagation**: An event is observed between nodes u and v and k can be 0 or 1 i.e. It can either be a topological event or interaction event. The first term in Eq 4. contains $\mathbf{h}_{struct}$ which is computed for updating each node involved in the event. For node u, the update will come from $\mathbf{h}^v_{struct}$ (green flow) and for node v, the update will come from $\mathbf{h}^u_{struct}$ (red flow). Please note all embeddings are dynamically evolving hence the information flow after every event is different and evolves in a complex fashion. With this mechanism, the information is passed from neighbors of node u to node v and neighbors of node v to node u. (i) Interaction events lead to temporary pathway - such events can occur between nodes which are not connected. In that case, this flow will occur only once but it will not make u and v neighbors of each other (e.g. meeting at a conference). (ii) Topological events lead to permanent pathway - in this case u and v becomes neighbor of each other and hence will contribute to structural properties moving forward (e.g. being academic friends). The difference in number of blue arrows on each side signify different importance of each node to node u and node v respectively.

Overall Embedding Update Process. As a starting point, *neighborhood* only includes

nodes connected by a structural edge. On observing an event, we update the embeddings of two nodes involved in the event using Eq 4. For a node u, the first term of Eq 4 (**Localized Embedding Propagation**) requires $\mathbf{h}_{struct}$ which is the information that is passed from neighborhood (N_v) of node v to node u via node v (one can visualize v as being the message passer from its neighborhood to u). This information is used to update the embedding of node u. However, we posit that node v does not relay equal amount of information from its neighbors to node u. Rather, node v receives its information to be relayed based on its communication and association history with its neighbors (which relates to importance of each neighbor). This requires to compute the attention coefficients on the structural edges between node v and its neighbors. For any edge, we want this coefficient to be dependent on rate of events between the two nodes (thereby emulating real world phenomenon that one gains more information from people one interacts more with). Hence, we parameterize our attention module with the temporal point process parameter $\mathcal{S}_{uv}$. Algorithm 1 outlines the process of computing the value of this parameter.

B.1.2 Computing $\mathbf{h}_{struct}$: Temporal Point Process based Attention

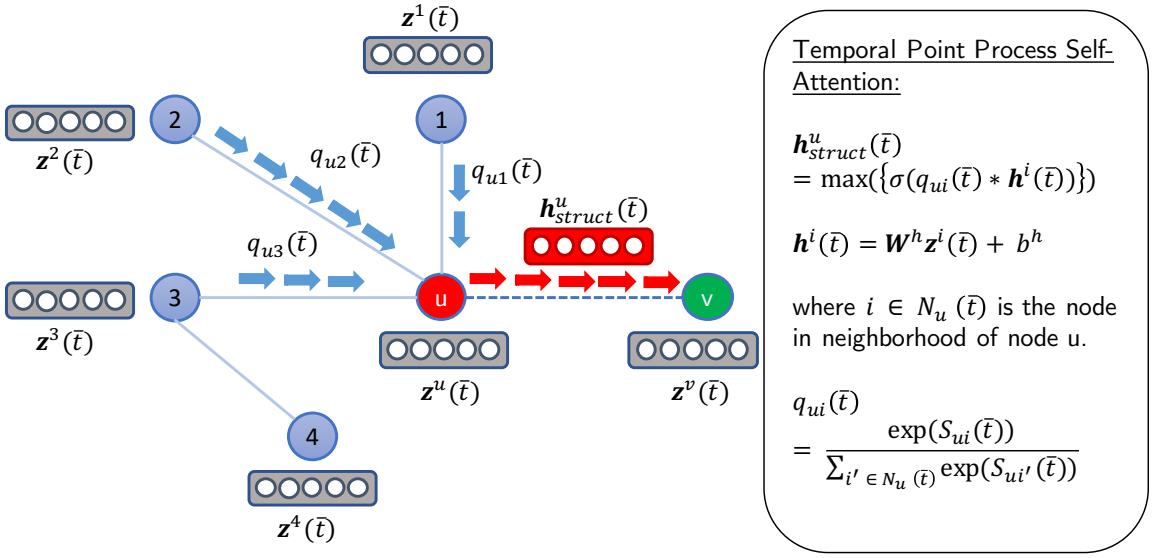


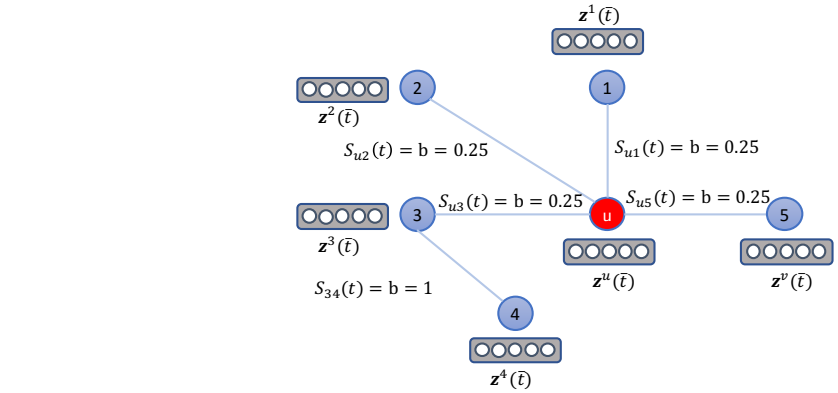
Figure B.2: Temporal Point Process based Self-Attention: This figure illustrates the computation of $\mathbf{h}^u_{struct}$ for node u to pass to node v for the same event described before between nodes u and v at time t with any k. $\mathbf{h}^u_{struct}$ is computed by aggregating information from neighbors (1,2,3) of u. However, Nodes that are closely connected or has higher interactions tend to attend more to each other compared to nodes that are not connected or nodes between which interactions is less even in presence of connection. Further, every node has a specific attention span for other node and therefore attention itself is a temporally evolving quantity. DyRep computes the temporally evolving attention based on association and communication history between connected nodes. The attention coefficient function (q's) is parameterized by $\mathcal{S}$ which is computed using the intensity of events between connected nodes. Such attention mechanism allows the evolution of importance of neighbors to a particular node (u in this case) which aligns with real-world phenomenon.

B.1.3 Computing $\mathcal{S}$: Algorithm 1

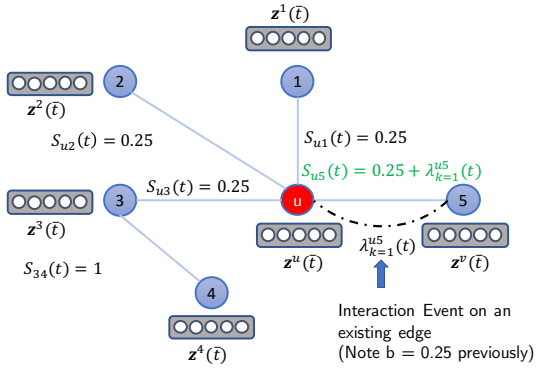
Please check Figure B.3 on next page.

B.2 Rationale Behind DyRep Framework

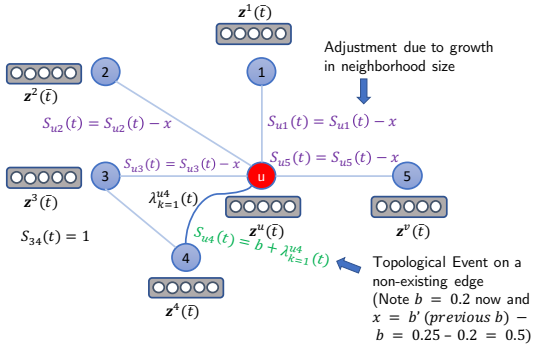
Connection to Marked Point Process. From a mathematical viewpoint, for any event e at time t, any information other than the time point can be considered a part of mark space describing the events. Hence, for DyRep, given a one-dimensional timeline, one can con-



(a) Start State



(b) Update to $\mathcal{S}$ after Interaction Event



(c) Update to $\mathcal{S}$ after Topological Event

Figure B.3: Computing $\mathcal{S}$. Illustration of the update to $\mathcal{S}$ under two circumstances for events that involve node u: (i) Interaction events between neighbors (ii) Topological Event between non-neighbors. We only illustrate one node but update will happen for both nodes in the event (e.g. for (u, v), rows of both nodes will be updated asymmetrically due to different neighborhood size. **(a)** shows the initial state where u has 4 neighbors and hence background attention is uniform $b = 0.25$. **(b)** u has an interaction event with node 5. Update only happens to $\mathcal{S}_{u5}$ and $\mathcal{S}_{5u}$ based on intensity of the event. **(c)** u has a topological event with node 4. b changes to 0.2. $b' = 0.25$ which is the previous b. Update happens to $\mathcal{S}_{u4}$ and $\mathcal{S}_{4u}$ based on intensity of event. Next attention for all other neighbors of both nodes (We only show for u here) are adjusted to reflect neighborhood size change. The matrix $\mathcal{S}$ is used for computing attention and hence does not get updated for interaction events between nodes which do not have an edge (for e.g. pair (1,2) may have an interaction event $\mathcal{S}_{12}$ won't be updated as they are not neighbors.

sider $\mathcal{O} = \{(u, v, k)_p, t_p)_{p=1}^P$ as a marked process with the triple (u, v, k) representing the mark.

However, from machine learning perspective, using a single-dimensional process with such marks does not allow to efficiently and effectively discover or model the structure in the

point process useful for learning intricate dependencies between events, participants of the events and dynamics governing those events. Hence, it is often important to extract the information out of the mark space and build an abstraction that helps to *discover the structure* in point process and make this learning *parameter efficient*. In our case, this translates to two components:

1. The nodes in the graph are considered as dimensions of the point process, thus making it a multi-dimensional point process where an event represents interaction/structure between the dimensions, thus allowing us to explicitly capture dependencies between nodes.

2. The topological evolution of networks happen at much different temporal scale than activities on a fixed topology network (e.g. rate of making friends vs liking a post on a social network). However both these processes affect each other's evolution in a complex and nonlinear fashion. Abstracting k to associate it with these different scales of evolution facilitates to model our purpose of expressing dynamic graphs at two time scales in a principled manner. It also provides an ability to explicitly capture the influential dynamics [96] of topological evolution on dynamics of network activities and vice versa (through the learned embedding – aka **evolution through mediation**.

Note that this distinction in use of mark information is also important as we learn representations for nodes (dimensions) but not for k. It is important to realize that k representing two different scales of event dynamics is not same as edge or interaction type. For instance, in case of typed persistent edge (e.g. wasbornIn, livesIn) or typed interaction (e.g. visit, fight), one would add type as another component in the mark space to represent an event while k still signifying different dynamic scales.

Comparison to [28]. In the similar vein as above, the point process specification of [28] can also be considered as a marked process that models the typed interaction dynam-

ics at a single time-scale and does not model topological evolution. In contrast to that, our method explicitly models dynamic graph process at two time scales. While both models use a point process based formulation for modeling temporal dynamics, there are several significant methodological differences between the two approaches:

Deep Point Process Model — While one can augment the event specification in (Trivedi et. al. 2017) with additional mark information, that itself is not adequate to achieve DyRep's modeling of dynamical process over graphs at multiple time scales. We employ a soft-plus function for f_k which contains a dynamic specific scale parameter ψ_k to achieve this while (Trivedi et al. 2017) uses an exponential ($\exp$) function for f with no scale parameter. Their intensity formulation attains a Rayleigh distribution which leads to a specific assumption about underlying dynamics which models fads where intensity of events drop rapidly between events after increasing. Our two-time scale model is more general and induces modularization, where each of two components allow complex, nonlinear and dependent dynamics towards a non-zero steady state intensity.

Graph Structure— As shown in [95], the key idea behind representation learning over graphs is to capture both the global position and local neighborhood structural information of node into its representations. Hence, there has been significant research efforts invested in devising methods to incorporate graph structure into the computation of node representation. Aligned with these efforts, DyRep proposes a novel and sophisticated Localized Embedding Propagation principle that dynamically incorporates graph structure from both local neighborhood and faraway nodes (as interactions are allowed between nodes that do not have an edge). Contrary to that, [28] uses single edge level information, specific to the relational setting, into their representations.

Deep Temporal Point Process Based Self-Attention— For learning over graphs, attention has been shown to be extremely valuable as importance of nodes differ significantly relative to each other. The state-of-the-art approaches have focused solely on static graphs with Graph Attention Networks [122] being the most recent one. Our attention mechanism for

dynamic graphs present a significant and principled advancement over the existing state-of-the-art Graph based Neural Self-Attention techniques which only support static graphs. As [28] do not incorporate graph structure, they do not use any kind of attention mechanism.

Support for Node Attributes and Edge Types. Node types or attributes are supported in our work. In Eq. 4, $z^v(\bar{t_p}^v)$ induces recurrence on node v's embedding, but when node v is observed for first time, $z^v(\bar{t_p}^v) = \mathbf{x}_v$ where $\mathbf{x}_v$ is randomly initialized or contains the raw node features available in data (which also includes type). One can also add an extra term in Eq. 4 to support high-dimension node attributes. Further, we also support different types of edges. If either the structural edge or an interaction has a type associated with it, our model can trivially support it in Eq. 3 and Eq. 4, first term $\mathbf{h}_{struct}$. Currently, for computing $\mathbf{h}_{struct}$, the formulation is shown to use aggregation over nodes. However, this aggregation can be augmented with edge type information as conventionally done in many representation learning frameworks [95]. Further, for more direct effect, Eq 3 can include edge type as third feature vector in the concatenation for computing g_k.

Support for new nodes. As mentioned in Section 2.3 of the main paper, the data contains a set of dyadic events ordered in time. Hence, each event involves two nodes u and v. A new node will always appear as a part of such an event. Now, as mentioned above, the initial embedding of any new node u is given by $z^u(\bar{t_p}^u)$ which can be randomly initialized or using the raw feature vector of the node u, $\mathbf{x}_u$. This allows the computation of intensity function for the event involving new node in Eq 1. Due to the inductive ability of our framework, we can then compute the embedding of the new node using Eq 4. There are two cases possible: Either one of the two nodes are new or both nodes are new. The mechanism for these two cases work as follows:

- *Only one new node in observed event* — To compute the embedding of new nodes, $\mathbf{h}_{struct}$ is computed using neighborhood of the existing (other) node, $\mathbf{z}(t_0^u)$ s the feature vector of

the node or random and drift is 0. To compute the new embedding of existing node, $\mathbf{h}_{struct}$ is the feature vector of the new node, self-propation uses the most recent embedding of the node and drift is based on previous time point.

- *Both nodes in the observed event are new* — $\mathbf{h}_{struct}$ is the feature vector of the feature vector of the other nodes, $\mathbf{z}(t_0^u)$ s the feature vector of the node or random and drift is 0.

Finally, Algorithm 1 does not require to handle new nodes any differently. As already available in the paper, both $\mathbf{A}$ and $\mathcal{S}$ are qualified by time and hence the matrices get updated every time. The starting dimension of the two matrices can be specified in two ways: (i) Construct both matrices of dimension = total possible no. of nodes in dataset and make the rows belonging to unseen nodes 0. (ii) Expand the dimensions of matrices as you start seeing new nodes. While we implement the first case, (ii) will be required in real-world streaming scenario.

B.3 Ablation Study

DyRep framework unifies several components that contribute to its effectiveness in learning rich node representation over complex and nonlinear processes in dynamic graphs. In this section, we provide insights on each component and how it is indispensable to the learning mechanism by performing an ablation study on various design choices of our model. Specifically, DyRep can be divided into three main parts: **Multi-time scale point process model**, **Representation Update Formulation** and **Conditional Intensity Based Attention Mechanism**. We focus on design choices available in each component and evaluate them on large github dataset. DyRep in the Figure B.4 is the full model.

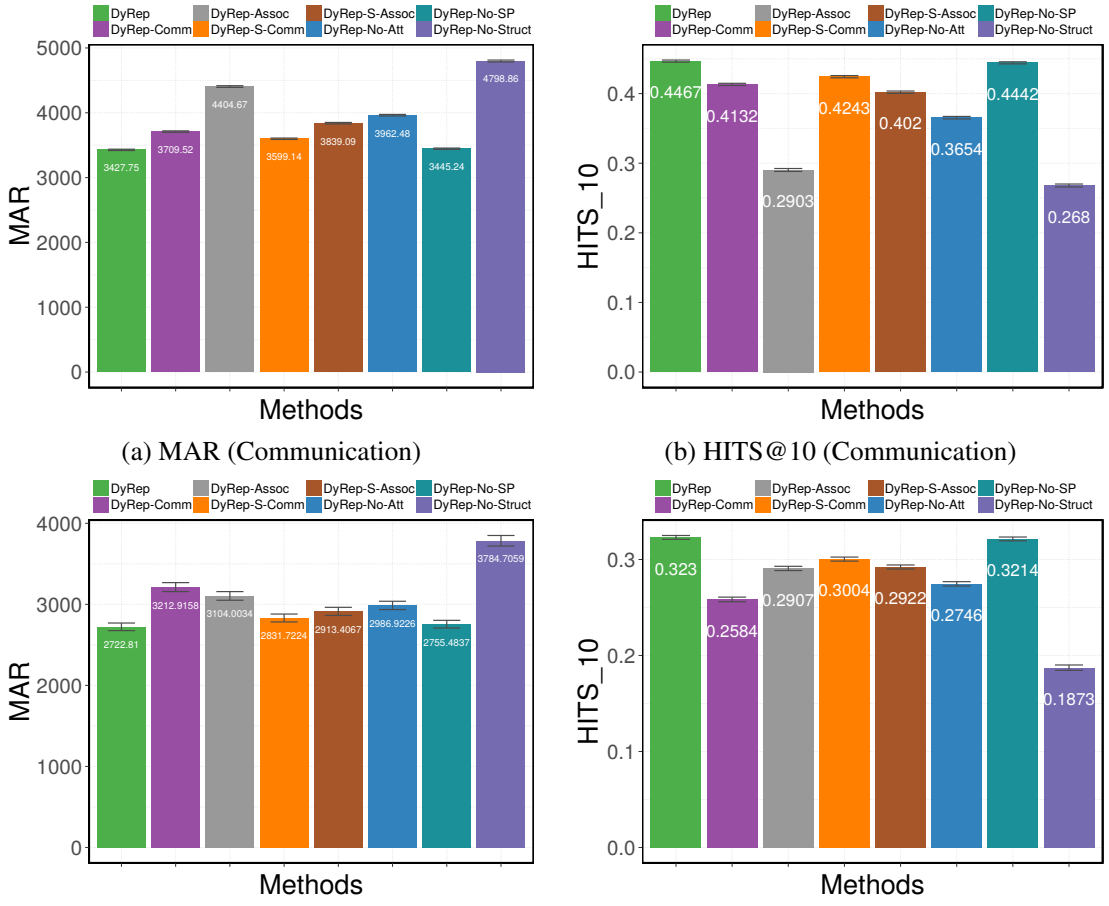
Multiple Time- Scale Processes. For this component, we perform two major tests:

- **DyRep-Comm.** In this variant, we make Eq 1., time-scale independent (i.e. remove k) and we train on only Communication Events. But we evaluate on both communication and association events. Please note that this is possible as our framework can compute representations for unseen nodes. Hence during training they will only learn representation parameters based on communication events. It is observed that compared to the full model, the performance of model degrades in prediction for both types of events. But the decline is more prominent for the Association events compared to Communication Events.

- **DyRep-Assoc.** In this variant, similar to above, we make Eq 1., time-scale independent and we train on only Association Events. But we evaluate on both communication and association events. It is observed that compared to the full model, the performance of model degrades in prediction for both types of events. But the decline is more prominent for the Communication events compared to Association Events.

The above two experiments show that considering events at a single time scale and not

distinguishing between the processes hurt the performance. Although the performance is hurt more when communication events are not considered which may be due to the more availability of communication events due to its rapid frequency. We also performed a small test by training on all events but using a single scale parameter (ψ). The performance for both the dynamics degrades which demonstrates the effectiveness of ψ_k.



(a) MAR (Communication)

(b) HITS@10 (Communication)

(c) MAR (Association)

(d) HITS@10 (Association)

Figure B.4: Ablation Study on Github Dataset

Representation Update Formulation. For this component, we focus on Eq. 4 and switch off the components to observe its effect.

- **DyRep-No-SP.** In this variant, we switch off the self-propagation component and we observe that the overall performance is not hurt significantly by not using self-propagation. In general, this term provides a very weak feature and mainly captures

the recurrent evolution of one's own latent features independent of others. It is observed that the deviation has increased for Association events which may point to the reason that there are few nodes who have links but highly varying frequency of communication and hence most of their features are either self-propagated or completely associated with others.

- **DyRep-No-Struct.** In this variant, we remove the structural part of the model and as one would expect, the performance drops drastically in both the scenarios. This provides evidence to the necessity of building sophisticated structural encoders for dynamic graphs.

Intensity Attention Mechanism. For this component, we focus on Section 3.2 which builds the novel intensity based attention mechanism. Specifically, we carry following test:

- **DyRep-No-Att.** Here we completely remove the attention from the structural component and we see a significant drop in the performance.

- **DyRep-S-Comm.** In this variant, we focus on Algorithm 1 and we only make update to the $\mathcal{S}$ matrix for Communication events but do not do it for Association events. This leads to slightly worse performance which helps to see how the $\mathcal{S}$ matrix is helping to mediate the two processes and not considering association events leads to loss of information.

- **DyRep-S-Assoc.** In this variant, we focus on Algorithm 1 and we only make update to the $\mathcal{S}$ matrix for Association events but do not do it for Communication events. This leads to a significant drop in performance again validating the need for using both processes but its prominent effect also shows that communication events (dynamics on the network) is more important while considering the influence of neighbors. Please note that this version is temporal analogous of GAT.

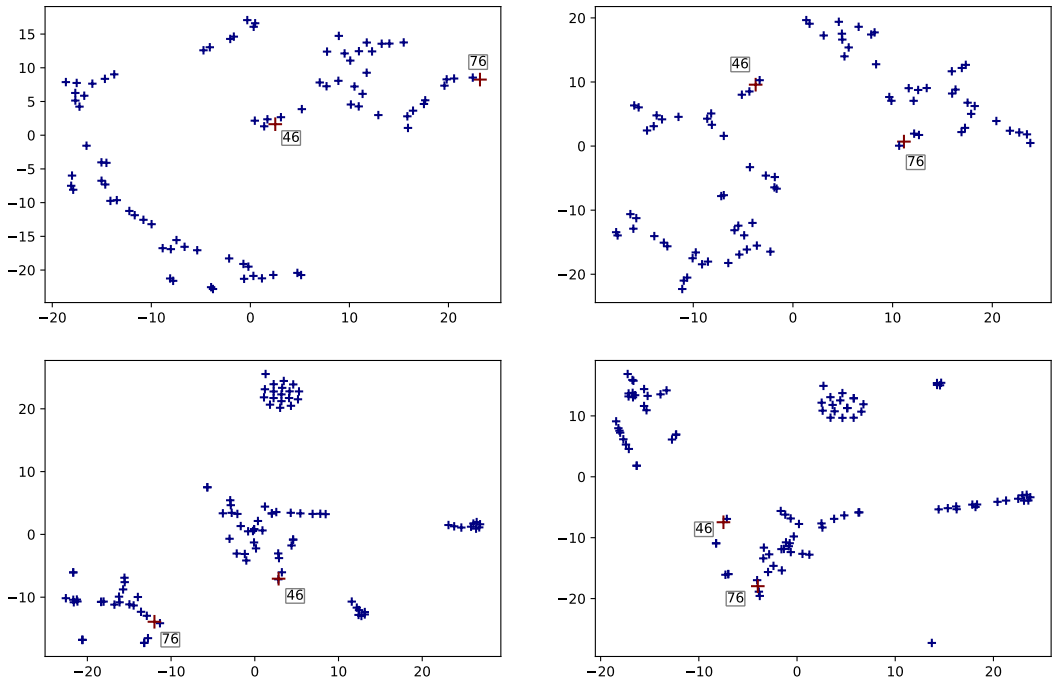
B.4 Exploratory Analysis

We assess the quality of learned embeddings and the ability of model to capture both temporal and structural information. Let t_0 be the time point when train ended. Let t_1 be the timepoint when the first test slot ends.

Effect of Association and Communication on Embeddings. We conducted this experiment on Social dataset. We consider three use cases to demonstrate how the interactions and associations between the nodes changed their representations and visualize them to realize the effect.

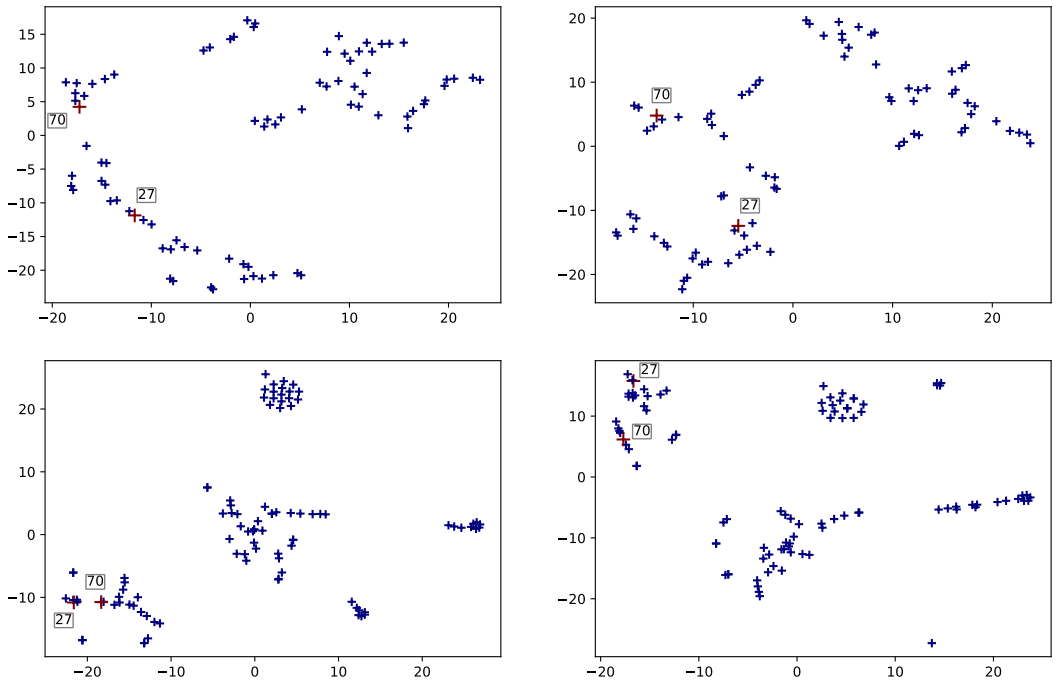
- **Nodes that did not have association before test but got linked during first test slot.** Nodes 46 and 76 got associated in test between test points 0 and 1. This reduced the cosine distance in both models but DyRep shows prominent effect of this association which should be the case. DyRep reduces the cosine distance from 1.231 to 0.005. Also, DyRep embeddings for these two points belong to different clusters initially but later converge to same cluster. In GraphSage, the cosine distance reduces from 1.011 to 0.199 and the embeddings still remain in original clusters. Figure B.5 shows the visualization of embeddings at the two time points in both the methods. This demonstrates that our embeddings can capture association events effectively.

- **Nodes that did not have association but many communication events (114000).** Nodes 27 and 70 is such a use case. DyRep embeddings consider the nodes to be in top 5 nearest neighbor of each other, in the same cluster and cosine distance of 0.005 which is aligned with the fact that nodes with large number of events tend to develop similar features over time. Graphsage on the other hand considers them 32nd nearest neighbor, puts them in different clusters with cosine distance - 0.792. Figure B.6 shows the visualization of embeddings at the two time points in both the methods. This demonstrates the ability of DyRep's embedding to capture communication events and their temporal effect on embeddings effectively.



(a) Train End Time (b) Test Slot 1 End Time.

Figure B.5: Use Case I. **Top row:** GraphSage Embeddings. **Bottom Row:** DyRep Embeddings.



(a) Train End Time (b) Test Slot 1 End Time.

Figure B.6: Use Case II. **Top row:** GraphSage Embeddings. **Bottom Row:** DyRep Embeddings.

- **Temporal evolution of DyRep embeddings.** In figure B.7 we visualize the embedding positions of the nodes (tracked in red) as they evolve through time and forms and breaks from clusters.

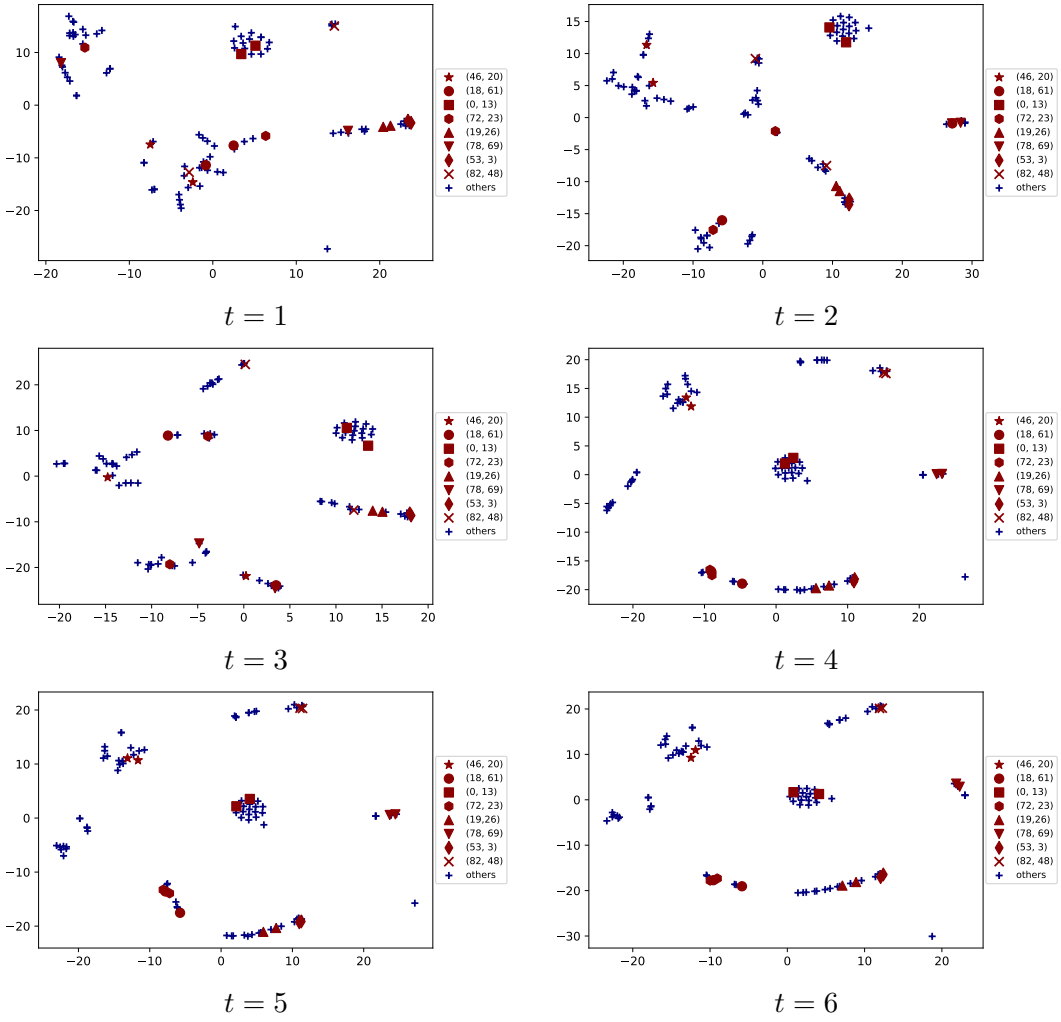


Figure B.7: Use Case IV: DyRep Embeddings over time - From left to right and top to bottom. t are the timepoints when test with that id ended. Hence, $t = 1$ means the time when test slot 1 finished.

B.5 Full Experiment Results for both Datasets

Figure B.8 provides HITS@10 results in addition to the MAR results reported for Link Prediction in Section 5 (Experiments) of the main paper.

B.6 Detailed Related Work

Static Embedding Approaches. Representation Learning approaches for static graphs can be broadly classified into two categories – Node embedding approaches aim to encode structural information pertaining to a node to produce its low-dimensional representation [88, 89, 90, 91, 92, 93, 94]. As they learn each individual node's representation, they are inherently transductive. Recently, [18] proposed GraphSage, an inductive method for learning functions to compute node representations that can be generalized to unseen nodes. Sub-graph embedding techniques learn to encode higher order graph structures into low dimensional vector representations [99, 100, 101]. Further, various approaches to use convolutional neural networks [102, 103, 104] over graphs have been proposed to capture sophisticated feature information but are generally less scalable. Most of these approaches only work with static graphs or can model evolving graphs without temporal information.

Other models for dynamic networks. There exists a rich body of literature on temporal modeling of dynamic networks [7] that focus on link prediction tasks but their goal is orthogonal to us as they build task specific methods and do not focus on representation learning. Further, there are several approaches in graph mining and temporal relational learning community [8, 9, 10, 11] that consider dynamic networks but are orthogonal to our current work. Research on learning dynamic embeddings has also progressed in linguistic community where the aim is to learn temporally evolving word embeddings [206, 207]. [105, 106] include some other approaches that propose model of learning dynamic embeddings in graph data but none of these models consider time at finer level and do not capture both topological evolution and interactions. [208] proposes subgraph pattern neural networks that focuses on evolution of subgraphs instead of single nodes and links. They build a novel neural network architecture for supervised learning where the hidden layers

represent the subgraph patterns observed in the data and output layer is used to perform prediction. [209] induces a dynamic graph from videos based on the visual correlation of object proposal that spans across the video. They further propose an LSTM based architecture to capture temporal dependencies over this induced graph and perform object detection. [210] proposes a dynamic probabilistic model in bipartite case of user-item recommendation where the goal is to learn the evolution of user and item latent features under the context of Poisson factorization, thus considering the evolution processes of users' and items' latent features as independent of each other.

Deep Temporal Point Process Models. Recently, [107] has shown that fixed parametric form of point processes lead into the model misspecification issues ultimately affecting performance on real world datasets. [107] therefore propose a data driven alternative to instead learn the conditional intensity function from observed events and thereby increase its flexibility. Following that work, there have been increased attraction in topic of learning conditional intensity function using deep learning[108] and also intensity free approach using GANS [109] for learning with deep generative temporal point process models.

B.7 Implementation Details

B.7.1 Additional Dataset Details

Table B.1: Dataset Statistics for Social Evolution and Github.

Dataset	#Nodes	#Initial Associations	#Final Associations	#Communications	Clustering Coefficient
Social Evolution	83	376	791	2016339	0.548
Github	12328	70640	166565	604649	0.087

For the social evolution dataset, we consider Proximity, Calls and SMS records between users as communication events (k=1) and all Close Friendship records as association events (k=0). For Github dataset, we consider Star/Watch records as communication events (k=1) and Follow records as association events (k=0). The Social Evolution data is collected from Jan 2008 to to June, 30 2009. We consider the association events between user from Jan

2008-Sep 10, 2008 (survey date) to form the initial network and use the rest of data for our experiments. We collected Github data from Jan 2013 - Dec 2013. For the nodes in 2013, we consider Follow link that existed between them before 2013 to form the initial network. We pre-process both datasets to remove duplicate (not recurrent in time) records and self-loops. We also process Github dataset to only contain users (and not organizations) as nodes and we select nodes that have at least 40 communication (watch) events and 10 association (follow) events.

Temporal Train/Test Split: For all the experiments, the data is divided into train and test based on time line. For Social Evolution Dataset, we train on data from Sep 11, 2008 to Apr 30, 2009 and use May 1, 2009-Jun, 30 2009 data for test which gives 10 days of time per test slot. This leads to an approximate 70/30 (train/test) split. For Github data, we train from Jan 1, 2013 to Sep 30, 2013 and test for Oct 1, 2013 - Dec, 31 2013 which gives 15 days of events per time slot. This leads to an approximate 65/35 (train/test) split.

B.7.2 Training Configurations

We performed hyper parameter search for best performance for our method and all the baselines and used the following hyper-parameters to obtain the reported results:
– For social dataset: Num nodes = 100, Num Dynamics = 2, bptt (sequence length) = 200, embed_size = 32, hidden_unit_size = 32, nsamples (for survival) = 5, gradient_clip = 100 and no dropout.
– For github dataset: Num nodes = 12328, Num Dynamics = 2, bptt (sequence length) = 300, embed_size = 256, hidden_unit_size = 256, nsamples (for survival) = 5, gradient_clip = 100.

For baselines, we used the implementations provided by their authors and we report the range of configurations used for baseline here: $max\_iter = \{1000, 5000, 10000\}, bptt = \{100, 200, 300\}, lr = \{0.0005, 0.0050.5, 0.1, 1\}, embed\_E = \{32, 64, 128, 256\}, embed\_R = \{32, 64, 128, 256\}, hidden = \{32, 64, 128, 256\}, warm = 0, t\_scale = 0.0001, w\_scale =$

$0.1, num\_epochs = \{10, 50, 100, 500, 1000\}$. As mentioned in experiment section, we always train baselines with warmstart in a sliding window training fashion.

Know-Evolve: The code provided by the authors was implemented in C++.

GraphSage: The code was implemented in Tensorflow by the authors. We use only the unsupervised train module to generate embeddings.

Node2Vec: We use the original python code with few changes in the hyper-parameters. We fix q in the node2vec as 0.8 for Social Dataset and 1 for Github dataset.

DynGEM: We experiment on the original code implemented in Keras with Theano back-end by the authors.

DynTrd: We use original code provided by the authors.

For tSNE embedding visualization in Figure 4, we used $sklearn.manifold.TSNE$ library to plot this figure with n components = 2, learning rate = 200, perplexity = 30, metric = "euclidean", min_grad_norm = 1e-9, early exaggeration = 4 and ran for 40,000 iterations.

B.8 Monte Carlo Estimation for Survival Term in $\mathcal{L}$ for Section 4

Algorithm 6 Computation of integral term in $\mathcal{L}$ for a mini-batch

Input: Minibatch $\mathcal{M} = \{m_q = (u, v, t, k)_q\}_{q=1}^{|\mathcal{M}|}$. Minibatch node list l, sample size N.

Output: Minibatch survival loss L_{surv}

$L_{surv} = 0.0$

for $q = 0\ to\ |\mathcal{M}| - 1$ **do**

$t_{curr} = m_q \rightarrow t$; $u_{curr} = m_q \rightarrow u$

$v_{curr} = m_q \rightarrow v$; $u_{surv} = 0$; $v_{surv} = 0$

for N samples **do**

select $u_{other} \in$ l uniformly randomly s.t. $u_{other} \notin \{u_{curr}, v_{curr}\}$

select $v_{other} \in$ l uniformly randomly s.t. $v_{other} \notin \{u_{curr}, v_{curr}\}$

for $k \in \{0, 1\}$ **do**

$u_{surv} += \lambda_k^{u_{curr}, v_{other}}(t_{curr})$

$v_{surv} += \lambda_k^{u_{other}, v_{curr}}(t_{curr})$

end for

end for

$L_{surv} += (u_{surv} + v_{surv})/N$

end for

return L_{surv}

Algorithm 6 is a simple variant of Monte Carlo trick to compute the survival term of log-likelihood equation. Specifically, in each mini-batch, we sample non-events instead of considering all pairs of non-events (which can be millions). Let m be the mini-batch size and N be the number of samples. The complexity of Algorithm 6 will then be $\mathcal{O}(2mkN)$ for the batch where the factor of 2 accounts for the update happening for two nodes per event which demonstrates linear scalability in number of events.

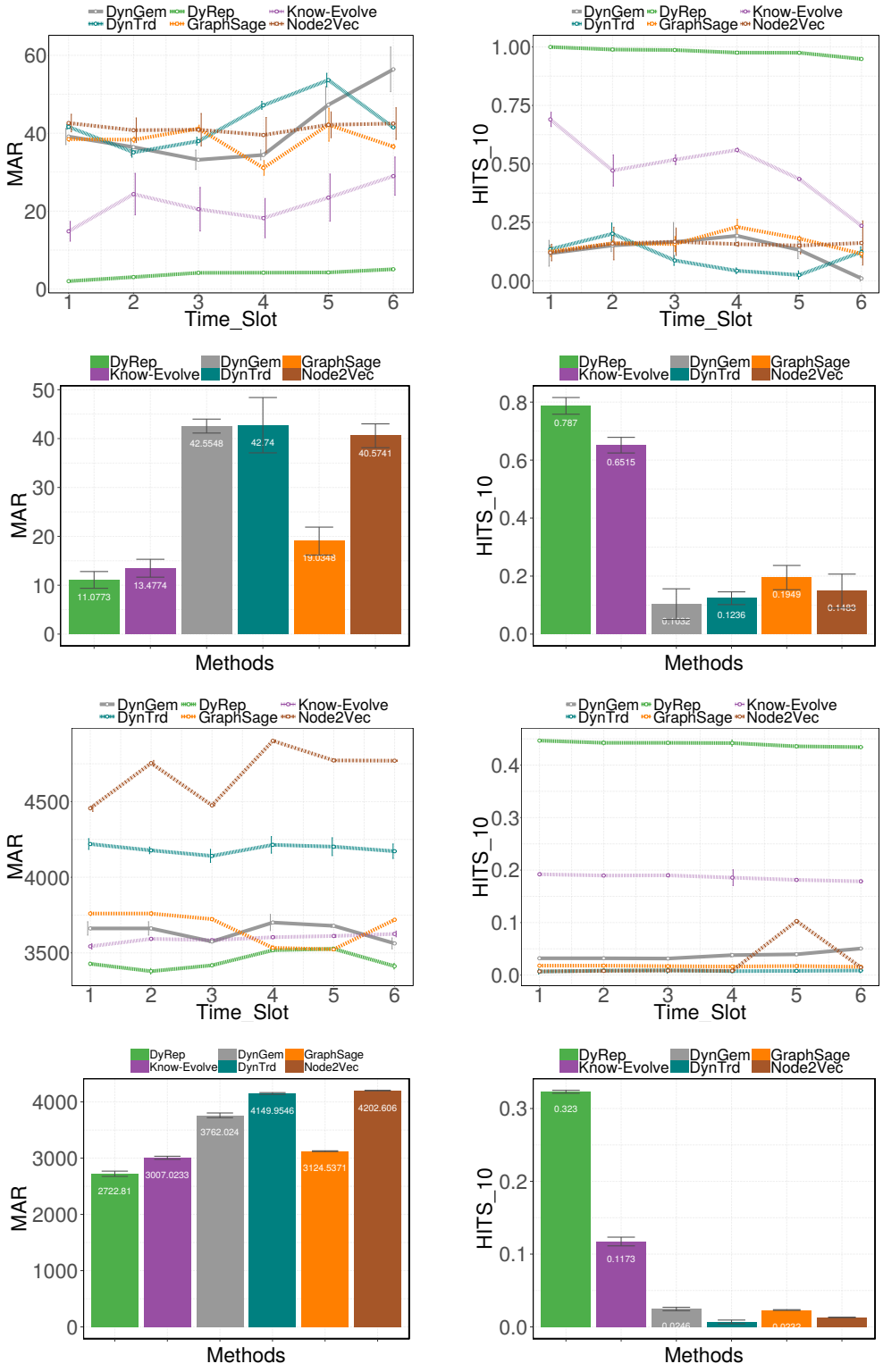


Figure B.8: Dynamic Link Prediction Performance: **Top 2 rows** show performance for Social Evolution Dataset. **Bottom 2 rows** show performance for Github Dataset. 1st and 3rd row show performance for Communication Events while 2nd and 4th row show performance for Association Events.

APPENDIX C

TEMPORAL REASONING OVER DYNAMIC KNOWLEDGE

C.1 Algorithm for Global BPTT Computation

As mentioned in Section 4 of main paper, the intricate relational and temporal dependencies between data points in our setting limits our ability to efficiently train by decomposing events into independent sequences. To address this challenge, we design an efficient Global BPTT algorithm presented below. During each step of training, we build computational graph using consecutive events in the sliding window of a fixed size. We then move sliding window further and train till the end of timeline in similar fashion which allows to capture dependencies across batches while retaining efficiency.

Algorithm 7 Global-BPTT

Input: Global Event Sequence $\mathcal{O}$, Steps s, Stopping Condition max_iter
$cur\_index = 0$, $t\_begin = 0$
for $iter = 0$ **to** max_iter **do**
  if $cur\_index > 0$ **then**
    $t\_begin = \mathcal{O}[cur\_index - 1] \rightarrow t$
  end if
  e_mini_batch = $\mathcal{O}[cur\_index : cur\_index + s]$
  Build Training Network specific to e_mini_batch
  Feed Forward inputs over network of s time steps
  Compute Total Loss $\mathcal{L}$ over s steps:
  $\mathcal{L} = -\sum_{p=1}^{s} \log\left(\lambda_r^{e^s,e^o}(t_p|\bar{t_p})\right)$ + Survival loss computed using Algorithm 6
  Backpropagate error through s time steps and update all weights
  if $cur\_index + s > \mathcal{O}.size$ **then**
    $cur\_index = 0$
  else
    $cur\_index = cur\_index + s$
  end if
end for

C.2 Data Statistics and Sparsity of Knowledge Tensor

Table C.1: Statistics for each dataset.

Dataset Name	# Entities	# Relations	# Events
GDELT-full	14018	20	31.29M
GDELT-500	500	20	3.42M
ICEWS-full	12498	260	0.67M
ICEWS-500	500	256	0.45M

Table C.2: Sparsity of Knowledge Tensor.

Dataset Name	# Possible Entries	# Available Entries	% Proportion
GDELT-full	3.93B	4.52M	0.12
GDELT-500	5M	0.76M	15.21
ICEWS-full	39.98B	0.31M	7e-3
ICEWS-500	64M	0.15M	0.24

C.3 Implementation Details

Know-Evolve. Both Algorithm 6 and Algorithm 7 demonstrate that the computational graph for each mini-batch will be significantly different due to high variations in the interactions happening in each window. To facilitate efficient training over dynamic computational graph setting, we leverage on graph embedding framework proposed in [101] that allows to learn over graph structure where the objective function may potentially have different computational graph for each batch. We use Adam Optimizer with gradient clipping for making parameter updates. Using grid search method across hyper-parameters, we set mini-batch size = 200, weight scale = 0.1 and learning rate = 0.0005 for all datasets. We used zero initialization for our entity embeddings which is reasonable choice for dynamically evolving entities.

Competitors. We implemented all the reported baselines in Tensorflow and evaluated all methods uniformly. For each method, we use grid search on hyper-parameters and embedding size and chose the ones providing best performance in respective methods. All the baseline methods are trained using contrastive max-margin objective function described in [75]. We use Adagrad optimization provided in Tensorflow for optimizing this objective function. We randomly initialize entity embeddings as typically done for these models.

C.4 Parameter Complexity Analysis

We report the dimensionality of embeddings and the resulting number of parameters of various models. Table C.3 illustrates that Know-Evolve is significantly efficient in the number of parameters compared to Neural Tensor Network while being highly expressive as demonstrated by its prediction performance in Section 5 of main paper. The overall number of parameters for different dataset configurations are comparable to the simpler relational models in order of magnitude.

Table C.3: Comparison of our method with various relational methods for memory complexity. Last two columns provide example realizations of this complexity in full versions for GDELT and ICEWS datasets. H_a and H_b correspond to hidden layers used in respective methods.H_e and H_r correspond to entity and relation embedding dimensions respectively. N_e and N_r are number of entities and relations in each dataset. For GDELT, $N_e = 14018$ and $N_r = 20$. For ICEWS, $N_e = 12498$ and $N_r = 260$. We borrow the notations from [19] for simplicity.

Method	Memory Complexity	GDELT		ICEWS	
		$H_e/H_r/H_a/H_b$	# Params	$H_e/H_r/H_a/H_b$	# Params
NTN	$N_e^2 H_b + N_r(H_b + H_a) + 2N_r N_e H_a + N_e H_e$	100/16/60/60	11.83B	60/32/60/60	9.76B
RESCAL	$N_r H_e^2 + N_e H_e$	100/-/-/-	1.60M	60/-/-/-	1.69M
TransE	$N_e H_e + N_r H_e$	100/-/-/-	1.40M	60/-/-/-	0.77M
TransR	$N_e H_e + N_r H_r + N_r H_r^2$	100/20/-/-	1.41M	60/32/-/-	1.02M
ER-MLP	$N_e H_e + N_r H_r + H_a + H_a(2H_e + H_r)$	100/20/100/-	1.42M	60/32/60/-	0.77M
Know-Evolve	$H_e(N_e + N_r H_e) + N_r H_r + H_a * (2H_e + H_r) + H_a * H_b + 2H_b$	100/20/100/100	1.63M	60/32/60/60	1.71M

C.5 Exploratory Analysis

C.5.1 Temporal Reasoning

We have shown that our model can achieve high accuracy when predicting a future event triplet or the time of event. Here, we present two case studies to demonstrate the ability of evolutionary knowledge network to perform superior reasoning across multiple relationships in the knowledge graphs.

Case Study I: Enemy's Friends is an Enemy

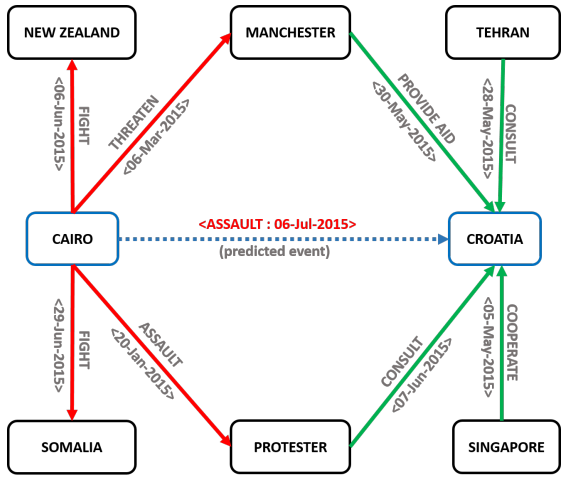


Figure C.1: Relationship graph for Cairo and Croatia. Dotted arrow shows the predicted edge. Direction of the arrow is from subject to object entity.

We concentrate on the prediction of a quadruplet (Cairo,Assault,Croatia,July 5,2015) available in test set. This event relates to the news report of an assault on a Croation prisoner in Cairo on July 6 2015. Our model gives rank-1 to the object entity Croatia while the baselines did not predict them well ($rank > 250$).

We first consider relationship characteristics for Cairo and Croatia. In the current train span, there are 142 nodes with which Cairo was involved in a relationship as a subject (total of 1369 events) and Croatia was involved in a relationship as an object (total of 1037 events). As a subject, Cairo was involved in an assault relationship only 59 times while as

an object, Croatia was involved in assault only 5 times. As mentioned earlier, there was no direct edge present between Cairo and Croatia with relationship type assault.

While the conventional reasoning methods consider static interactions of entities in a specific relationship space, they fail to account for the temporal effect on certain relationships and dynamic evolution of entity embeddings. We believe that our method is able to capture this multi-faceted knowledge that helps to reason better than the competitors for the above case.

Temporal Effect. It is observed in the dataset that many entities were involved more in negative relationships in the last month of training data as compared to earlier months of the year. Further, a lot of assault activities on foreign prisoners were being reported in Cairo starting from May 2015. Our model successfully captures this increased intensity of such events in recent past. The interesting observation is that overall, Cairo has been involved in much higher number of positive relationships as compared to negative ones and that would lead conventional baselines to use that path to reason for new entity – instead our model tries to capture effect of most recent events.

Dynamic Knowledge Evolution. It can be seen from the dataset that Cairo got associated with more and more negative events towards the mid of year 2015 as compared to start of the year where it was mostly involved in positive and cooperation relationships. While this was not very prominent in case of Croatia, it still showed some change in the type of relationships over time. There were multiple instances where Cairo was involved in a negative relationship with a node which in turn had positive relationship with Croatia. This signifies that the features of the two entities were jointly and non-linearly evolving with the features of the third entity in different relationship spaces.

Below we provide reference links for the actual event news related to the edges in Figure C.1.

Predicted Edge.

- (Cairo, Assault, Croatia, 06-Jul-2015): https://www.bloomberg.com/news/articles/2015-

08-05/islamic-state-egypt-affiliate-threatens-to-kill-croatian-citizen

Other Edges.

- (Cairo, Assault, Protester, 20-Jan-2015):http://usa.chinadaily.com.cn/world/2015-04/22/content_20501452
- (Cairo, Threaten, Manchester, 06-Mar-2015): http://www.manchestereveningnews.co.uk/news/greater-manchester-news/anthony-filz-stashed-deadly-machine-8788541
- (Protester, Consult, Croatia, 07-Jun-2015): http://globalvoicesonline.org/2015/06/07/veterans-of-croatias-war-of-independence-are-still-knocking-on-the-governments-door/
- (Manchester, Provide Aid, Croatia, 30-May-2015): http://www.offthepost.info/blog/2015/05/liverpool-meet-inter-to-discuss-mateo-kovacic-deal/

Case Study II: Common enemy forges friendship

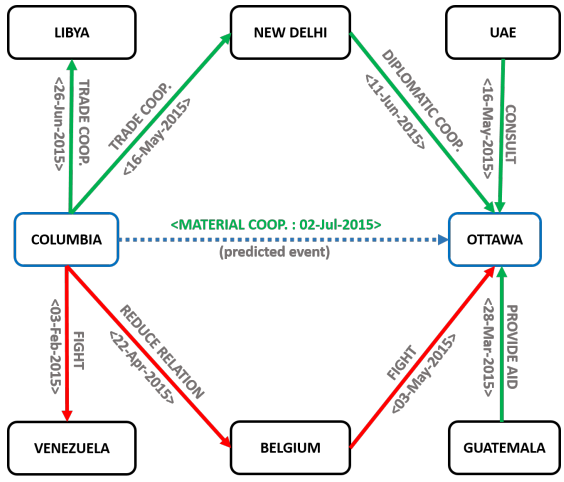


Figure C.2: Relationship graph for Columbia and Ottawa. Dotted arrow shows the predicted edge. Direction of the arrow is from subject to object entity.

We concentrate on the prediction of a quadruplet (Colombia,Engage in Material Cooperation,Ottawa,July 2 2015) available in test set. This event relates to the news report of

concerns over a military deal between Colombia and Canada on July 2 2015 and reported in Ottawa Citizen. Our model gives rank-1 to the object entity Ottawa while the other baselines do not predict well ($rank > 250$). The above test event is a new relationship and was never seen in training.

As before, we consider relationship characteristics between Colombia and Ottawa. In the current train span, there are 165 nodes for which Colombia was involved in a relationship with that node as a subject (total of 1604 events) and on the other hand, Ottawa was involved in a relationship with those nodes as an object total of 733 events). As a subject, Colombia was involved in a cooperation relationship 71 times while as an object, Ottawa was involved in cooperation 24 times.

Temporal Effect. It is observed in the dataset that Colombia has been involved in hundreds of relationships with Venezuela (which is natural as they are neighbors). These relationships range across the spectrum from being as negative as "fight" to being as positive as "engagement in material cooperation". But more recently in the training set (i.e after May 2015), the two countries have been mostly involved in positive relationships. Venezuela in turn has only been in cooperation relationship with Ottawa (Canada). Thus, it can be inferred that Colombia is affected by its more recent interaction with its neighbors while forming relationship with Canada.

Dynamic Knowledge Evolution. Overall it was observed that Colombia got involved in more positive relationships towards the end of training period as compared to the start. This can be attributed to events like economic growth, better living standards, better relations getting developed which has led to evolution of Colombia's features in positive direction. The features for Ottawa (Canada) have continued to evolve in positive direction as it has been involved very less in negative relationships.

More interesting events exemplifying mutual evolution were also observed. In these cases, the relationship between Colombia and third entity were negative but following that relationship in time, the third entity forged a positive relationship with Ottawa (Canada).

One can infer that it was in Colombia's strategic interest to forge cooperation (positive relation) with Ottawa so as to counter its relationship with third entity. Below we provide reference links for the actual event news related to the edges in Figure C.2.

Predicted Edge.

- (Columbia, Material Coop., Ottawa, 02-Jul-2015): http://ottawacitizen.com/news/politics/report-on-military-executions-casts-shadow-over-lav-deal-with-colombia

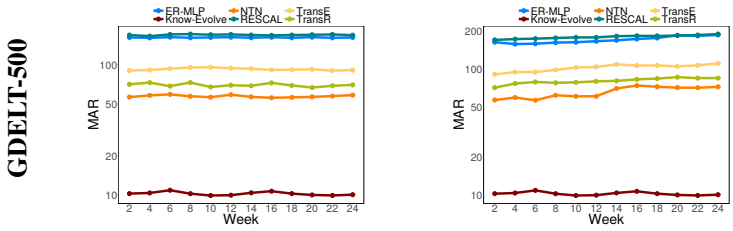
Other Edges.

- (Columbia, Trade Coop., New Delhi, 16-May-2015): http://www.newindianexpress.com/business/2015/may/16/Petroleum-Minister-Dharmendra-to-Lead-Business-Delegation-to-Mexico-Colombia-761494.html
- (Columbia, Fight, Venezuela, 03-Feb-2015):http://www.turkishpress.com/news/421947/
- (New Delhi, Diplomatic Coop., Ottawa, 28-May-2015):http://www.marketwatch.com/story/art-of-living-set-to-showcase-the-yoga-way-2015-06-11-61734555
- (Belgium, Fight, Ottawa, 05-May-2015): https://www.durhamregion.com/news-story/5597504-9-facts-about-in-flanders-fields-on-its-100th-anniversary/

C.6 Sliding Window Training Experiment

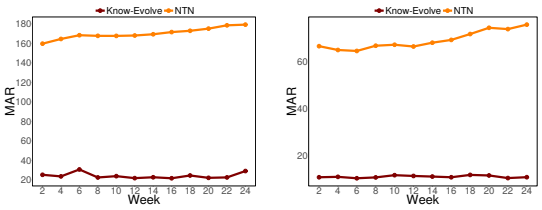
Unlike competitors, the entity embeddings in our model get updated after every event in the test, but the model parameters remain unchanged after training. To balance out the advantage that this may give to our method, we explore the use of sliding window training paradigm for baselines: We train on first six months of dataset and evaluate on the first test window. Next we throw away as many days (2 weeks) from start of train set as found in test set and incorporate the test data into training. We retrain the model using previously learned parameters as warm start. This can effectively aid the baselines to adapt to the evolving knowledge over time. Figure C.3 shows that the sliding window training contributes to

stable performance of baselines across the time window (i.e.the temporal deterioration is no longer observed significantly for baselines). But the overall performance of our method still surpasses all the competitors.



(c) Sliding Window Training (d) Non-sliding window Training

Figure C.3: Performance comparison of sliding window vs. non-sliding window training.



(a) New facts only (b) Recurrent Facts Only

Figure C.4: Comparison with NTN over recurrent and non-recurrent test version.

C.7 Recurrent Facts vs. New facts

One fundamental distinction in our multi-relational setting is the existence of recurrence relations which is not the case for traditional knowledge graphs. To that end, we compare our method with the best performing competitor - NTN on two different testing setups: 1.) Only Recurrent Facts in test set 2.) Only New facts in test set. We perform this experiment on GDELT-500 data. We call a test fact "new" if it was never seen in training. As one can expect, the proportion of new facts will increase as we move further in time. In our case, it ranges from 40%-60% of the total number of events in a specific test window. Figure C.4 demonstrates that our method performs consistently and significantly better in both cases.

APPENDIX D

LEARNING OPTIMIZATION MODELS OF GRAPH FORMATION

D.1 Gradient Updates for GraphOpt Algorithm

In this section, we provide details on the gradient update algorithm:

Algorithm 8 Parameter Update

procedure TRAIN_POLICY($B, \psi, \phi, \theta, \bar{\psi}, \omega$)
  for each gradient step **do**
    Sample mini-batch of $\mathcal{M}$ transitions
    $\{(s_t, a_t, r_t(s_t, a_t), s_{t+1})\}$ from B
    Compute $\hat{\nabla}_\psi J_V(\psi), \hat{\nabla}_\theta J_Q(\theta), \hat{\nabla}_\phi J_\pi(\phi)$ using
    Eq. D.2,D.3, D.5 and compute $\nabla_\omega J$ for repr. network
    Update the parameters based on following:
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_\theta J_Q(\theta_i) \;\; i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow l\bar{\psi} + (1-l)\bar{\psi}$
    $\omega \leftarrow \omega - \lambda_{emb} \hat{\nabla}_\omega J_{emb}(\omega)$
  end for
end procedure

For completeness, we present the gradients of each objective below, however, they are adopted from [150] and we encourage interested readers to refer to the original manuscript for more details:

1. **Soft Q-function** is trained to minimize the Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim B} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t) \right)^2 \right] \tag{D.1}$$

where,

$$\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E} s_{t+1} \sim p[V_{\bar{\psi}}(s_{t+1})] \tag{D.2}$$

where $V_{\bar{\psi}}$ is a target value network and $\bar{\psi}$ is an exponentially moving average of the

value network weights for stabilizing training. The gradients for Eq. D.1 is given by:

$$\begin{aligned}\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t, a_t)(Q_\theta(s_t, a_t) \\ - r(s_t, a_t) - \gamma V_{\bar{\psi}}(s_{t+1}))\end{aligned} \tag{D.3}$$

2. **Policy Network** is trained using the following objective function:

$$\begin{aligned}J_\pi(\phi) = \mathbb{E}_{s_t \sim B, \epsilon_t \sim \mathcal{N}}[\log \pi_\phi(f_\phi(\epsilon_t; s_t)|s_t) \\ - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))]\end{aligned} \tag{D.4}$$

where f_ϕ is the transformation applied to the policy network to induce reparameterization that helps in building a low variance estimator. The gradient for Eq. D.4 is then given by:

$$\begin{aligned}\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t|s_t) \\ + \nabla_\phi f_\phi(\epsilon_t; s_t)\left(\nabla_{a_t} \log \pi_\phi(a_t|s_t)\right. \\ \left. - \nabla_{a_t} Q(s_t, a_t)\right)\end{aligned} \tag{D.5}$$

D.2 More Related Work

Reinforcement Learning for Graphs. Recent advancements in deep learning techniques over graph structured data [177, 18] and progress in deep RL [168, 169] has stimulated increased interest in casting the general task of learning over graphs into a sequential decision process, whereby actions correspond to the discrete set of nodes to be sequentially connected to a partially constructed graph. This procedure optimizes task-specific objectives in the form of the reward function. Specifically, deep reinforcement learning has been used in three major learning paradigms: *Learning to Generate* [48] proposes a generative model for molecular structure using a graph convolutional policy network to optimize a domain specific reward that captures various properties of molecules. *Learning to Walk* [211, 164, 212] uses RL for tasks of link prediction and QA over Knowledge Graphs. The key goal in these works is to find an optimal path from a query entity to a target answer entity. *Learning to Optimize* [22, 23, 24]: builds a combination of graph neural networks and RL to learn optimization algorithms for NP-hard problems (e.g. MaxCut).

Maximum Entropy Reinforcement Learning. Deep Reinforcement Learning [213, 214, 215], specifically Actor-critic algorithms [170] has recently achieved great success in a variety of tasks that require search over combinatorial space and have inspired many new architectures that can be broadly categorized into: on-policy algorithms [216, 169] that build on standard on-policy policy gradients and off-policy algorithms [168] that use off-policy samples from replay buffer. Both categories exhibit trade-off between sample complexity and stability with on-policy algorithms being more stable while off-policy counterpart being more sample efficient. We build our framework on recently proposed Soft Actor-Critic (SAC) [150] algorithm that has been shown to be both sample efficient and stable. SAC falls in the category of Maximum entropy based algorithms [151, 217, 218, 219, 220] that are based on maximum entropy learning [221] and have been shown to be robust in the face of model and estimation errors while improving exploration [222]. **Inverse Reinforcement**

Learning (IRL) — Methods in IRL [163] and imitation learning [223] seek to recover a reward function given measurements of near-optimal expert trajectories. The maximum entropy IRL framework [151] leads to a sample-based method that learns a neural network approximation of the reward, without requiring knowledge of the MDP transition function [152].

Deep Generative Models of Graph Generation. Recently, there have been significant research efforts in building deep generative models of graph generation as they allow to effectively capture complex structural properties observed in a graph and use that information to output realistic graphs. Most of these works can be broadly categorized into two classes: (i) Methods that learn from collection of graphs (e.g. DeepGMG [47], GraphRNN [20], GCPN [48]) and (ii) Methods that learn from a single graph (e.g VGAE [49], GraphGan [50], MolGAN [51], NetGan [52]). As discussed in the main paper, our current approach falls into the second category. DeepGMG builds probabilistic model where the partially generated graph is encoded by the graph neural network (GNN) and the representation is used to make decision of constructing next node or edge. However, it suffers from scalability issues. GraphRNN solves this problem by proposing an auto-regressive model of graph generation, wherein the focus is on generating sequence of adjacency vectors that be mapped to graph structure. It employs hierarchical recurrent architecture to encode the historical path information. GCPN combines GCN with RL and learns a deep generative model using an objective specific to domain of chemistry. In the second category, methods like GraphGan and GVAE are implicit models but their main focus is to learn graph representations and hence perform weakly on generation tasks and have limited scalability. NetGan is a recently proposed implict graph generator model exhibiting generalization properties. However, unlike GraphOpt, NetGAN optimizes a GAN-based objective which converge to an uninformative discriminator, thereby not useful for transfer, which in contrast is a key objective of our approach.

D.3 Additional Details on Experiments

D.3.1 Datasets

Table D.1 provide statistics and reference to the dataset used for Graph Construction experiments. Table D.2 provides dataset statistics for non-relational datasets and Table D.3 provides dataset statistics for relational datasets, both used for link prediction.

Table D.1: Dataset Statistics for Construction Experiments

Graph	Nodes	Edges	Density	Avg. Degree	Source
Barabasi-Albert (BA)	100	384	0.0384	7.68	Synthetic
Erdos-Renyi	500	6152	0.02461	24.608	Synthetic
Political Blogs	1224	16718	0.0127	27.316	[224]
CORA-ML	2995	8158	0.001	3.898	[225]
PubMed	19717	44327	0.00011	4.496	[226]
CiteSeer	3327	4676	0.00042	2.811	[226]

Table D.2: Social, Metabolic and Citation Graphs for Link Prediction

Graph	Nodes	Edges	Density	Average Deg.	Source
Cora-ML	2995	8158	0.001	3.898	[225]
Political Blogs	1224	16718	0.0112	27.316	[224]
E. Coli	1805	14660	0.0045	12.55	[25]

Table D.3: Knowledge Graphs for Link Prediction

Graph	Nodes	Egdes	Relations	Density	Average Deg.	Source
Kinship	104	10686	25	0.9879	82.15	[227]
FB15k-237	14541	310116	237	0.0013	19.74	[228]
WN18RR	40943	93003	11	0.00005	2.19	[229]

D.3.2 Baselines

As we learn from single graph as an observation, we compare with the state-of-art baselines that can operate in this setting for a fair comparison. Below we provide references and some details on each baseline that we compare with:

- For graph construction experiments, we compare against two traditional generators: Degree Corrected Stochastic Block Model (DC-SBM) [230] that extends classical stochastic block models to account for heterogeneous degree of the vertices and block two level Erdos-Reniy (BTER) random graph model [231] that generates graph

with dense subgraphs, each being an ER graph in itself and has a degree preserving property. Next, we use Variational Graph Autoencoder (VGAE) [49] that uses an autoencoding mechanism to learn node embeddings. Being an autorecnoder, it has the capacity to generate graphs and hence serves as a representative node embedding approach for graph generation. Finally, NetGAN [52] is the state-of-art implicit model of graph generation that learns to mimic network patterns observed in a graph by building a probabilistic models of random walks over the graph. NetGAN employs a GAN based objective where the discriminator attempts to distinguish the generated random walks from the observations.

- For link prediction experiments in non-relational domains, we compare again with NetGan and GVAE. NetGan is an implicit model of graph generation that generalizes to the task of link prediction which is also the property of our model and hence provides a strong baseline for our work. Further, GVAE being a node embedding approach, is naturally suitable for link prediction. Next, we compare Node2Vec [232], a classical node embedding approach based on random walks. Finally, we compare with a strong state-of-art link prediction baseline, SEAL [25], that uses a graph neural network based architecture to perform link prediction. It is important to note that SEAL employs a task specific architecture and task-specific objective and hence provide a strong baseline to compare with.

- For relational link prediction on knowledge graphs, we cover a spectrum of approaches through their representative baselines. ConvE [229] is the state-of-art embedding based relational learning approach that has been shown to outperform most of earlier baselines in relational learning over knowledge literature [233]. We consider two Rl based method, Minerva [164] and Reward Shaping [165]. Minerva employs an LSTM encoder to encode the path information into the state representation and then solve a Markov Decision Process using policy gradients more specifically

by REINFORCE algorithm [234]. Reward Shaping advances the methods in MINERVA to build a strong state-of-art baseline. However, it is again important to note both ConvE and Reward Shaping are dedicated baselines with task specific architecture and objective analogous to SEAL above (Reward shaping specifically shapes the reward using embedding model which essentially makes it very close to link prediction objective). On the other hand, MINVERVA can be considered somewhat closer in spirit to our work as it uses more general form of reward (+1/-1) which is also sparse while still using task-specific architecture. Finally, we also compare with NeuralLP [235] that introduces a differential rule learning system using symbolic operaters to learn logic rules to perform reasoning.

D.3.3 Evaluation Protocol for Link Prediction using Learned Embeddings

Our state encoder learns to represent the nodes into low dimensional representation. This experiment evaluates how well the encoder network was learned by testing the ability of embeddings learned by the encoder itself to perform link prediction. Once the model is trained, we pass the training graph through the encoder to get the final trained embedding of nodes. For non-relational case, we use dot product based embedding similarity criterion and label edges as before. For relational case, we use score function similar to ComplEx [236] and rank the entities as before.

D.3.4 GraphOpt Implementation

We closely follow SAC's architecture in using two Q-functions for efficient learning and parameterizing them with standard multi-layered perceptron. Similarly, the reward function $\mathcal{R}$ is parameterized by standard multi-layer perceptron. As described in Section 3, we use GNN based policy network. We use the rlkit library[1] available online and adapt it for our framework. In general, SAC does not have any aggressive hyper-parameters that needs

[1] https://github.com/vitchyr/rlkit

Table D.4: Hyper Parameter Configuration Table

GraphOpt Algorithm			
HyperParameters	**Default Values**	**HyperParameters**	**Default Values**
num_epochs	10000	reward_lr	0.01
num_steps_per_epoch	500	soft_target_tau	0.001
num_steps_per_eval	1000	policy_lr	1.00E-04
num_steps_before_training_online	25	qf_lr	1.00E-03
replay_buffer_size	100000	optimizer	Adam
batch_size	128	train_policy_with_reprarameterization	TRUE
max_path_length	1000	eval_deterministic	FALSE
discount	0.99	use_automatic_entropy_tuning	TRUE
reward_iter	30	gen_from_policy	10
irl_episode_per_train	10	term_threshold	100
meas_samples	5	l1_coeff	0.1
gen_samples	10	n_embed_size	32
prop_rounds	2	net_size	256

to be tuned – Table E.1 provides complete list of hyper-parameters with their reasonable default values that were used for maximum number of experiments and here we describe the ones that were mostly tuned using validation performance.

For evaluation, SAC provides two choices to use either deterministic or stochastic and we use stochastic for our evaluation as stochastic action is an important factor in our work. We tune the number of epochs ranging from 10000-50000 epochs depending on the graph environment. For inverse learning, we vary reward iterations from 30-300. Most of other hyper-parameters remain fairly constant across the environments. For state encoder, we tune prop-round between 2-4 and found 2 to be best choice. term_threshold ranges from 100-500 based on dataset and node emebeding size is tuned in the range $\{32, 64, 128, 256, 512\}$. In the general parameters, net_size represents the size of the hidden units of MLP that were used for policy, Q, representation and reward networks. We tune this parameter again in the range $\{32, 64, 128, 256, 512\}$ based on the environment.

All the experiments were conducted on Intel Xeon CPU V4 2.20 GHZ with 64 GB memory and Nvidia GeForce 1080 GPU.

D.3.5 Metrics

We provide several results on our constructed graphs and link predictions and below we discuss some information about the metrics reported in the tables and figures:

1. Graph Construction Experiments We chose the following statistics that cover various aspects of graph structure:

   - Triangle Count: Number of triangles in the graph
   - Clustering Coefficient: measure of the degree to which nodes in a graph tend to cluster together.
   - Longest Connected Component: Size of the largest connected component.
   - Assortativity: Pearson correlation of degrees of connected nodes, $A = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$ where the (x_i, y_i) pairs are the degrees of connected nodes.
   - Max Degree: Maximum degree of all nodes in the graph.

2. Non-relational Link Prediction Experiments

   - AUC (Area under ROC Curve): Measures how well a parameter / model distinguish between correct and wrong connections.
   - Average Precision: Classical measure used to evaluate link prediction based on the precision recall performance of the model

3. Relational Link Prediction Experiments

   - MRR (Mean Reciprocal Rank): This is a more robust statistic for evaluating link prediction compared to mean average rank and is widely used in relational learning tasks.
   - HITS@K: It measures how many correct predictions made by the policy lie in the top K predictions.

Finally, we provide percent Deviation values with error in main paper in 6.5. This metric represent the percentage difference in graph bases statistics between the original graph and the generated graphs. As we generate multiple graphs to measure variation, we take the percent deviation with each of the generated graphs individually and then report the mean and error over these readings.

D.4 Additional Experiment Results

D.4.1 Synthesizing Graphs Via Learned Generative Mechanism

In this section, we provide more comprehensive results on the construction capabilities of GraphOpt with results on two more datasets: Erdos-Renyi (Table D.6) and Pubmed (Table D.9). We also present results for the datasets we showed in the main paper along with these two as it contains two more statistics (Largest connected component and Assortativity) that we evaluated which we could not show in main paper due to space constraint.

Table D.5: Percent deviation of graph statistics for generated graphs from observed BA graph (lower is better)

Model	Triangle Count	Clustering Coeff.	Largest Connected Component	Assortativity	Max Degree
Observed Graph	504	0.147	100	-0.096	33
DC-SBM	46.56 ± 6.58	59.44 ± 7.11	27.33 ± 2.52	91.61 ± 0.64	28.29 ± 7.63
BTER	48.02 ± 9.11	33.20 ± 1.28	35.33 ± 0.58	86.06 ± 1.85	33.33 ± 0.00
VGAE	70.89 ± 8.95	94.40 ± 0.81	9.00 ± 1.00	92.40 ± 1.77	8.08 ± 1.75
NetGAN	31.68 ± 6.28	40.69 ± 4.27	$\mathbf{4.00 \pm 1.73}$	62.12 ± 35.49	$\mathbf{4.04 \pm 1.74}$
GraphOpt	$\mathbf{6.28 \pm 4.05}$	$\mathbf{25.52 \pm 8.25}$	$\mathbf{6.00 \pm 2.65}$	$\mathbf{8.24 \pm 0.78}$	$\mathbf{5.05 \pm 4.63}$

Erdos-Renyi Graph

Table D.6: Percent deviation of graph statistics for generated graphs from the observed Erdos-Renyi graph (lower is better)

Model	Triangle Count	Clustering Coeff.	Largest Connected Component	Assortativity	Max Degree
Observed Graph	7335	4.84E-02	500	-0.019	37
DC-SBM	54.67 ± 1.07	94.91 ± 0.74	35.53 ± 2.10	70.17 ± 9.36	30.63 ± 5.63
BTER	60.11 ± 0.85	59.81 ± 0.86	38.87 ± 2.48	91.22 ± 2.59	8.11 ± 4.68
VGAE	85.62 ± 0.44	83.12 ± 1.27	$\mathbf{2.33 \pm 0.31}$	82.81 ± 11.05	6.31 ± 1.56
NetGAN	51.54 ± 2.78	49.47 ± 5.89	3.73 ± 3.36	$\mathbf{31.58 \pm 22.94}$	$\mathbf{2.73 \pm 2.70}$
GraphOpt	$\mathbf{26.29 \pm 2.89}$	$\mathbf{23.96 \pm 3.68}$	6.07 ± 1.22	$\mathbf{33.33 \pm 21.27}$	$\mathbf{5.41 \pm 2.70}$

Political Blogs Graph

Table D.7: Percent deviation of graph statistics for generated graphs from the observed Political Blogs Graph (lower is better)

Model	Triangle Count	Clustering Coeff.	Largest Connected Component	Assortativity	Max Degree
Observed Graph	303129	0.320	1222	-0.221	351
DC-SBM	52.78 ± 9.15	91.73 ± 1.18	65.25 ± 10.79	88.49 ± 4.93	40.86 ± 1.89
BTER	45.47 ± 7.25	54.17 ± 13.57	55.46 ± 3.73	90.75 ± 4.83	43.87 ± 0.75
VGAE	98.56 ± 0.44	99.32 ± 0.55	42.01 ± 5.78	97.86 ± 2.03	44.06 ± 0.92
NetGAN	44.28 ± 8.27	37.55 ± 7.20	29.59 ± 5.00	$\mathbf{24.95 \pm 13.79}$	$\mathbf{38.75 \pm 3.70}$
GraphOpt	$\mathbf{34.73 \pm 3.79}$	$\mathbf{20.34 \pm 9.12}$	$\mathbf{24.30 \pm 2.05}$	$\mathbf{21.79 \pm 4.78}$	$\mathbf{36.85 \pm 2.71}$

CORA-ML Graph

Table D.8: Percent deviation of graph statistics for generated graphs from the observed Cora-ML graph

Model	Triangle Count	Clustering Coeff.	Largest Connected Component	Assortativity	Max Degree
Observed Graph	4890	0.241	2485	-0.066	168
DC-SBM	71.17 ± 1.53	68.25 ± 20.16	1.76 ± 2.21	26.04 ± 7.49	6.94 ± 5.40
BTER	40.06 ± 1.17	81.66 ± 1.74	5.15 ± 2.85	89.96 ± 6.37	16.47 ± 14.49
VGAE	99.56 ± 0.24	93.10 ± 2.11	$\mathbf{0.05 \pm 0.02}$	96.86 ± 1.52	94.44 ± 1.82
NetGAN	64.19 ± 2.15	41.12 ± 18.82	0.52 ± 0.11	$\mathbf{2.84 \pm 3.16}$	4.17 ± 2.38
GraphOpt	$\mathbf{19.46 \pm 1.01}$	$\mathbf{14.63 \pm 5.78}$	2.09 ± 0.34	$\mathbf{4.10 \pm 3.53}$	$\mathbf{2.58 \pm 1.24}$

PubMed Graph

Table D.9: Percent deviation of graph statistics for generated graphs from the observed Pubmed graph

Model	Triangle Count	Clustering Coeff.	Largest Connected Component	Assortativity	Max Degree
Observed Graph	37560	6.02E-02	19717	-0.0436	171
DC-SBM	40.56 ± 5.33	85.33 ± 8.31	38.22 ± 3.77	83.87 ± 18.17	39.57 ± 3.43
BTER	35.17 ± 4.69	65.17 ± 2.00	41.74 ± 2.34	69.57 ± 7.62	37.43 ± 5.064
VGAE	N/A	N/A	N/A	N/A	N/A
NetGAN	23.39 ± 3.79	53.32 ± 7.13	29.99 ± 3.16	57.57 ± 12.76	40.94 ± 2.03
GraphOpt	$\mathbf{19.95 \pm 2.30}$	$\mathbf{42.47 \pm 2.33}$	$\mathbf{13.63 \pm 1.39}$	$\mathbf{18.19 \pm 11.87}$	$\mathbf{23.19 \pm 6.00}$

APPENDIX E

LEARNING STRATEGIC NETWORK EMERGENCE GAMES

E.1 Network Emergence Games and Multi-Agent Inverse Reinforcement Learning

In this section, we first provide details on Markov Perfect equilibrium (MPE) as a solution concept for Markov Network Emergence games and outline how the procedure for solving it maps to the reinforcement learning setting. Next, we discuss the connection of MPE with more general solution concept of Markov Quantal Response Equilibrium (MQRE) that aids in building a gradient based learning approach. Finally, we discuss how the logistic version of MQRE is an appropriate solution concept for solving network emergence games with multi-agent reinforcement learning. We note that several materials discussed in this section can be found scattered across different literature resources cited in the main paper, but we discuss some of these details here in a consolidated manner using canonical notations for ease of exposition and establishing direct correspondence to our approach.

The description of network emergence policy (in Section 2.1) $\pi_i : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A}_i)$, where $\mathcal{P}(\mathcal{A}_i)$ is the distribution over agent i's actions space, concretely specifies the Markov strategy for an agent i.

Definition 1 (Markov Perfect Equilibrium)**.** *A Markov perfect equilibrium* $(\pi_i^*)_{i \in N}$ *is a Nash equilibrium in Markov strategies.*

While existence of a Markov Perfect Equilibrium (MPE) in stochastic games has been long established for stochastic games [190, 191, 192], an application of Bellman's optimality principle [193] shows that solving for MPE directly maps to recursive procedure of learning optimal joint policy $\bar{\pi}^*$ by optimizing individual reward r_i as done in a reinforcement learning procedure.

Theorem 2 (Recursive representation of MPE)**.** *A joint Markov strategy profile $\bar{\pi}^*$ constitutes a Markov perfect equilibrium if and only if:*

1. *For all agents i, there exist state value function $V_i^* : \mathcal{S} \to \mathbb{R}$ such that:*

$$V_i^*(s) = \max_{a_i \in \mathcal{A}_i} r_i(s, a_i, \bar{\pi}_{-i}^*) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a_i, \bar{\pi}_{-i}^*) \cdot V_i^*(s') \tag{E.1}$$

   holds for all states $s_t \in \mathcal{S}$, where $\bar{\pi}_{-i}^*$ represent optimal strategy of all agents other than agent i.

2. *For all states $s_t \in \mathcal{S}$, the joint strategy profile $\bar{\pi}^*$ constitutes a Nash equilibrium of normal-form game (Nash for current state) with action space $\mathcal{A}$ and agent-specific payoff value:*

$$\hat{r}_i(s, \bar{a}) = r_i(s, \bar{a}) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(\bar{a}) \cdot V_i^*(s') \tag{E.2}$$

   for $\bar{a} \in \mathcal{A}$ and all agents i.

Proof. See [237] Page 374 for a proof sketch. □

In stochastic games such as network emergence, the subsequent course of play which depends on strategies of all players, affects the final outcome in addition to the payoff i.e. decisions are based on:

$$\hat{r}_i(s, \bar{\pi}) = r_i(s, \bar{\pi}) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(\bar{\pi}) \cdot V_i(s') \tag{E.3}$$

As the payoff and strategies are interdependent, MPE can be found by simultaneously solving for state values $\bar{V}^*$ and strategies $\bar{\pi}^*$ using the following maximization operations for all $s_t \in \mathcal{S}$ and all agents i:

$$\begin{aligned} V_i^* &= \max_{\pi_i} \hat{r}_i(s, \pi_i, \bar{\pi}) \\ \pi^* &= \arg\max_{\pi_i} \hat{r}_i(s, \pi_i, \bar{\pi}) \end{aligned} \tag{E.4}$$

Multi-agent Reinforcement learning algorithms are capable of solving the above system of equations in (E.4) efficiently, where optimizing individual reward functions for learning joint optimal policy ($\bar{\pi}$) maps to solving for Markov Perfect Equilibrium. However, solving directly for MPE requires solving for Nash equilibrium at each state, which is not amenable to learning due to discontinuous characteristics of Nash Equilibrium with respect to payoff matrix. Further, Nash equilibrium assumes all agents to be perfectly rational which is often not the case for agents participating in the network emergence games. Fortunately, both these difficulties are addressed by another solution concept, Quantal Response Equilibrium (QRE) and its logistic version [194, 195], which is stochastic generalization of Nash equilibrium. Specifically, QRE accounts for bounded rationality using a parameter λ and models the situations where payoff matrices are injected with some noise, thereby introducing smoothness useful for gradient based learning approaches [196]. In the context of stochastic games, QRE has been extended to Markov version by [197, 189] and referred as Markov Quantal Response Equilibrium (MQRE).

Let the expected payoff from playing action a for agent i in state s, given strategies of other players, is denoted by:

$$\hat{r}_i(s, a, \bar{\pi}) = r_i(s, a, \bar{\pi}) + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}(a_i, \bar{\pi}_{-i}) \cdot V_i(s') \tag{E.5}$$

In the quantal response framework, agent i is assumed to perceive noise injected payoff version of Eq. (E.5) as:

$$\tilde{r}_i(s, a, \bar{\pi}) = \hat{r}_i(s, a, \bar{\pi}) + \varepsilon_i(s, a) \tag{E.6}$$

The noise vector for all actions $\boldsymbol{\varepsilon}_i(s) = (\varepsilon_i(s, a))_{a \in \mathcal{A}_i}$ follows a joint distribution with zero mean and density function $f_i(\boldsymbol{\varepsilon}_i(s))$. Let agent i's response set of action a in state s

denoted by:

$$R_i(s,a) = \left\{ \boldsymbol{\varepsilon}_i(s) \in \mathbb{R}^{|\mathcal{A}_i|} : \tilde{r}_i(s,a,\bar{\pi}) > \tilde{r}_i(s,a',\bar{\pi}) \ \ \forall a' \in \mathcal{A}_i \right\} \tag{E.7}$$

$R_i(s,a)$ specifies the realization of $\boldsymbol{\varepsilon}_i(s)$ such that agent i in state s perceives action a as the one with the highest payoff.

Definition 3 (Markov quantal response equilibrium)**.** *A Markov quantal response equilibrium is a strategy profile $\bar{\pi}^*$ such that*

$$\bar{\pi}^* = \int_{R_i(s,a)} f_i(\boldsymbol{\varepsilon}_i(s))\, d\boldsymbol{\varepsilon} \tag{E.8}$$

where the probability mass function of the response set of agent i specifies the probability that agent i in state s takes action a. An interesting version of MQRE is its logistic version (MLQRE) which arise from the noise that is i.i.d according to Gumbel distribution with parameter $\lambda \in \mathbb{R}_0^+$, which also controls the rationality of agents. An MLQRE $\bar{\pi}^*$ can then be expressed in closed form as a solution to the following system of equations: For all states $s_t \in \mathcal{S}$, all agents i and all actions $a \in \mathcal{A}_i$:

$$\begin{aligned} \bar{\pi}_i^*(a_i|s) &= \frac{e^{\lambda \cdot \hat{r}_i(s,a,\bar{\pi}^*_{-i})}}{\sum_{a' \in \mathcal{A}_i} e^{\lambda \cdot \hat{r}_i(s,a',\bar{\pi}^*_{-i})}} \\ V_i^*(s) &= \sum_{a' \in \mathcal{A}_i} \bar{\pi}_i^*(a_i|s) \cdot \hat{r}_i(s,a',\bar{\pi}^*_{-i}) \end{aligned} \tag{E.9}$$

When $\lambda \to 0$, the equilibrium is fully noisy and the agents will select actions uniformly at random. When $\lambda \to \infty$, the agents will choose actions in best response manner (greedily). Recently, [189] has shown that if the logit MQRE converges as $\lambda \to \infty$, the limit point is Markov perfect equilibrium.

Theorem 4. *Let $\bar{\pi}^{lim}$ be the limit point of some logit MQRE $\bar{\pi}^*(\lambda)$, i.e.*

$$\bar{\pi}^{lim} = \lim_{\lambda \to \infty} \bar{\pi}^*(\lambda) \tag{E.10}$$

Then, $\bar{\pi}^{lim}$ is a Markov perfect equilibrium.

Proof. See [189] Page 13, Theorem 4 for a proof by contradiction. □

[189] further provides proof on the existence of the limit point for stochastic games which establishes logit MQRE an appropriate equilibrium concept for characterizing trajectory distributions induced by the reward functions specified in the decision process of Markov Perfect network emergence games. However, the system of equations in Eq. (E.9) provide a solution to a set of constraints which do not explicitly specify concrete joint policy profiles that can be used to maximize the likelihood of the observed data as a function of rewards, a key objective of this work. To address this exact challenge, [183] introduces a new solution concept, referred as Logistic Stochastic Best Response Equilibrium (LSBRE), that corresponds to the outcome of repeated application of a stochastic process where each agent attempts to optimize its actions while keeping other agents' actions fixed. Please refer to Section 2.3 for the complete definition of LSBRE for a Markov game with horizon T - a key component of which is the form of z_i which is given as:

$$z_i^{t,(k+1)}(s^t) \sim P_i^t(a_i^t | \bar{a}_{-i}^t = \bar{z}_{-i}^{t,(k)}(s^t), s^t) = \frac{\exp\left(\lambda Q_i^{\bar{\pi}^{t+1:T}}(s^t, a_i^t, \bar{z}_{-i}^{t,(k)}(s^t))\right)}{\sum_{a'_i} \exp\left(\lambda Q_i^{\bar{\pi}^{t+1:T}}(s^t, a'^t_i, \bar{z}_{-i}^{t,(k)}(s^t))\right)} \tag{E.11}$$

where, $Q_i^{\bar{\pi}^{t+1:T}}$ is recursively defined as:

$$\begin{aligned} Q_i^{\bar{\pi}^{t+1:T}}(s^t, a_i^t, \bar{\pi}_{-i}^t) = \hat{r}_i(s^t, a_i^t, \bar{\pi}_{-i}^t) + \mathbb{E}_{s^{t+1} \sim P(\cdot|s^t, \bar{a})} \Big[\mathcal{H}(\pi_i^{t+1}(\cdot|s^{t+1})) \\ + \mathbb{E}_{a^{t+1} \sim \bar{\pi}^{t+1}(\cdot|s^{t+1})} [Q_i^{\bar{\pi}^{t+2:T}}(s^{t+1}, \bar{a}^{t+1})] \Big] \end{aligned} \tag{E.12}$$

The reward term $\hat{r}_i(s^t, a_i^t, \bar{\pi}_{-i}^t)$ in Eq. E.12 corresponds to the payoff term $\hat{r}_i(s, a, \bar{\pi}_{-i}^*)$ in Eq. E.9. Further, [183] establishes that the trajectory induced by LSBRE policies can be characterized with energy-based formulation such that the probability of a trajectory increases exponentially as the sum of rewards increases. This allows them to build a practical inverse reinforcement learning algorithm for multi-agent setting where the expert policies are assumed to form a unique LSBRE under some unknown reward. We build our approach based on this algorithm and extend it to network emergence games which we discuss in the next section.

E.2 MINE Algorithm

In this section, we outline the multi-agent inverse reinforcement learning procedure to learn network emergence games.

Algorithm 9 builds on the recently proposed multi-agent adversarial reinforcement learning [183] technique by extending it to support graph structured environment and modifying it to use multi-agent attention actor-critic (MAAC [198]) algorithm in the inner RL loop for efficient off-policy learning. At the start of each epoch, the game is reset such that there exists no links between the agents (line 7, $\mathbf{A}_t$ is the adjacency matrix at step t). At this stage, agents' low-dimensional embeddings are initialized from their features using one round of the the GNN based state encoder (due to lack of edges there will be no message passing initially). Next, we rollout several forward steps to collect experience in replay buffer (line 13) and build agents' trajectories (line 14). During each step in the game, we sample actions for all agents using our Gaussian policy (line 10, corresponds to each agents' announcements of forming links with other agents as discussed in Section 3). These continuous actions are mapped (externally to the network) to links between agents and used to update $\mathbf{A}_t$. Based on the new structure, the state of the environment is updated using SE (line 12). After several rollouts, few iterations of gradient updates are used to update the parameters of all the networks. For each update, we obtain set of trajectory samples from both ex-

Algorithm 9 MINE Algorithm

1: **procedure** MINE
2: **Input:** Expert Demonstrations $D_E = \{\tau_j^E\}$, Empty list of Agent's trajectories D_π,
3: Replay Buffer $\mathcal{B}$, Agent feature matrix $\mathcal{X}$
4: Initialize the parameters for policies $\bar{pi}$, attentive critic $\bar{Q}$, reward estimators $\bar{g}$, potential
5: functions $\bar{h}$ and state encoder (SE) with $\bar{\phi}$, $\bar{\psi}$, $\bar{\omega}$, $\bar{\theta}$ and φ respectively
6: **repeat**
7: Reset the environment and set adjacency matrix $\mathbf{A}_0 = \mathbf{0}$ (no links)
8: Initialize $\tau^\pi = \{\}$
9: **for** each step do **do**
10: Sample $\bar{a} \sim \bar{\pi}_{\bar{\phi}}(\bar{a}_t|\bar{o}_t)$
11: Update $\mathbf{A}_t$ based on $\bar{a}$ (c.f. Section 3)
12: Update $s_{t+1} = \mathsf{SE}_\varphi(s_t, \mathbf{A}_{t+1})$
13: Add $(\bar{o}, \bar{a}, \bar{o'})$ to $\mathcal{B}$
14: Update $\tau^\pi \leftarrow \mathsf{concat}(\tau^\pi, (\bar{o}, \bar{a}, \bar{o'}))$
15: **end for**
16: Add τ^π to D_π and reset $\tau^\pi \leftarrow \{\}$
17: **for** each training iteration do **do**
18: Sample $(\bar{o}, \bar{a}, \bar{o'})_\pi$ triples T_π from D_π and $(\bar{o}, \bar{a}, \bar{o'})_E$ triples T_E from D_E
19: Update $\bar{\omega}$, $\bar{\theta}$ using:
20: $\mathbb{E}_{T_E}[\log D((\bar{o}, \bar{a}, \bar{o'})_E)] + \mathbb{E}_{T_\pi}[\log(1 - D((\bar{o}, \bar{a}, \bar{o'})_\pi))]$
21: Update reward estimates $\bar{r}(\bar{o}, \bar{a}, \bar{o'})$ with $\bar{g}_{\bar{\omega}}(\bar{o}, \bar{a})$
22:
23: Update $\bar{\phi}$, φ w.r.t. $\bar{r}(\bar{o}, \bar{a}, \bar{o'})$ using MAAC policy gradients:
24: $\nabla_{\phi_i} J(\bar{\pi}_\phi) = \mathbb{E}_{\bar{o}\sim\mathcal{B}, \bar{a}\sim\bar{\pi}}[\nabla_{\phi_i} \log(\pi_{\phi_i}(a_i|o_i))(-\alpha \log(\pi_{\phi_i}(a_i|o_i))$
25: $+ Q_{\psi_i}(\bar{o}, \bar{a}) - b(\bar{o}, a_{-i}))]$
26:
27: Update critic parameters $\bar{\psi}$ by minimizing the TD-error:
28: $\mathbb{E}_{(\bar{o},\bar{a}\sim B}[\sum_{i=1}^N (Q_{\psi_i}(\bar{o}, \bar{a}) - y_i)^2]$ where
29: $y_i = r_i(\bar{o}, \bar{a}) + \gamma \mathbb{E}_{a'\sim\bar{\pi}_{\bar{\phi}}(\bar{o'})}[Q_{\psi_i}(\bar{o'}, \bar{a'}) - \alpha \log(\pi_{\phi_i}(a_i'|o_i'))]$
30: **end for**
31: **until** Convergence
32: **Output:** Learned policies $\bar{\pi}_\phi$, reward functions $\bar{g}_\omega$ and state encoder SE_φ
33: **end procedure**

pert and agents' trajectories (line 18). For the loss function in line 20, we follow [183] and use the following structure for the discriminators: $D_i(o_i, a_i, o_i') = \frac{\exp(f_{\omega_i}(o_i,a_i,o_i')}{\exp(f_{\omega_i}(o_i,a_i,o_i'))+\pi_{\phi_i}(a_i|o_i)}$, where $f_{\omega_i}(o_i, a_i, o_i') = g_{\omega_i}(o_i, a_i) + \gamma h_{\theta_i}(o_i') - h_{\theta_i}(o_i)$. g_ω is the reward estimator and h_θ is the potential function. We use this discriminator definition to update parameters $\bar{\omega}$ and $\bar{\theta}$ (line 19-20). The update reward estimator g_ω serves as an updated the reward function

$\bar{r}$ (line 21). The parameters of the structured strategy network (which includes both the policy network and the state encoder network) are updated with respect the newly estimated reward function $\bar{r}$ using the soft-policy gradients (line 24). Our updates are based on off-policy gradients as we use MAAC [198] for policy learning that extends Soft-Actor Critic [150] algorithm to multi-agent setting with attention based critics. In line 24-25, α is the temperature parameter that balances the trade off between policy entropy and reward maximization. Further, b is the baseline advantage function used to solve the credit assignment problem in multi-agent setting and we follow [198] to compute it. Finally, the attentive critics are updated by minimizing the TD-error as shown in line (28-29). We note that there is no explicit objective function for directly optimizing the state encoder parameters φ, however, the policy gradients backpropagate through the state encoder network in an end-to-end fashion, thereby updating the encoder network. At convergence, MINE returns the learned policy network $\bar{\pi}$, reward function $\bar{g}$ and state encoder SE, which are then used for performing evaluation tasks as described in next section.

E.3 Further Experiment Details

E.3.1 Datasets

In this section, we provide more details on the properties of the datasets used in our work.

Andorra[1]. The Andorra dataset contains call detail records between 32,829 citizens where a link between two citizens if they were involved in atleast one call interaction during the period from July 2015 to June 2016. The dataset contains three attributes for each agent – Phone type (takes values Apple, Samsung, others), frequent city (takes values between 0 and 6) and cellular usage (real value). These attributes are strongly correlated with important unobserved individual characteristics such as phone type may be related to income, frequent city to the place of dwelling and cellular usage to the daily online activities.

Trade[1]. This is the 2014 international trade data between countries provided by the

[1] Dataset can be downloaded from repository for [200] at https://github.com/yuany94/endowment/tree/master/data

United Nations Statistical Division (UN Comtrade Database: https://comtrade.un.org/). The network contains 100 countries, a link among which indicates that the trade value between two countries is greater than 1 billion dollars in both directions. the dataset contains three attributes – Continent (Africa, America, Asia/Pacific and Europe), Economic Complexity Index [] and GDP (real value). While continent affects the location based strategic trade decision, ECI captures the diversity and sophistication of country's export and GDP directly impacts a country's ability to perform trade partnerships. In this dataset, an entire country represents an agent in the game and hence this is an example where the network emerges due to indirect impact of human beings.

Movie[1]. The movie dataset is specific type of social network where edges signify collaborations between directors and actor/actresses (cast). The links in this network are formed strategically based on the benefits to both parties. The overall network structure is close to bipartite network (some nodes are both directors and cast members) with 160 directors, 2628 cast members and 10,399 links between them. The dataset was collected for 3493 movies in the period of 2000-2016 and contains two attributes for each agent – Occupation (director, cast) and Gender (male, female).

Company[1]. This data consists of network between employees in a company where an edge between them signify a call or text communication. Each employee is either a manger or a subordinate (which is also the only attribute for agents in this network) and there are 420 managers and 1564 subordinates with 12,751 edges between them. In this network, managers are mostly connected to other managers and similar for subordinates with occasional links between subordinates with their specific manager. This is an example of a hierarchical network that the properties with the Australian bank toy network that we used to analyze the reward interpretability.

Arxiv GR-QC[2]. This is a collaboration network between authors in the field of General Relativity and Quantum Cosmology where an edge between them indicate they co-authored

[2] Data is available at: https://snap.stanford.edu/data/ca-GrQc.html

atleast one paper between them. The network consists of 5242 nodes with 13396 edges between. This network does not contain any attributes hence we only use one-hot identity map as initial feature vector for agents. In this network, the strategy of link formation between two authors would depend on their common neighbors and hence the network emerges based on development of its own structure.

E.3.2 Baselines

In this section, we provide details on baselines used for comparison in strategic prediction task.

GT_core [203]. This method incorporates game-theoretic models into node representation learning methods for learning latent node features that are used for downstream link prediction. Specifically, this method focuses on two node representation learning methods – node2vec and DeepWalk and enhances them by proposing a novel form of biased random walk. In this biased walk, the next node is not chosen randomly, instead it is chosen based on link probability between the current node and set of possible next nodes. The link probability is computed as a product of two weight values corresponding to that edge. These two weight values are computed using game-theoretic model based utility function and k-core decomposition respectively. The game-theoretic weights are further computed using two different utility models - a co-authorship model [238] and influence games [239]. Once the biased random walk is performed to select the neighbors, standard gradient based training is done and link prediction is performed based on learned node representations.

Social Game Embed [200]. This work takes an economic view of link formation between heterogeneous agents, where the link formation is considered to be driven by the tradeoff between exchange benefits and coordination costs between interacting agents. Based on this view, a social network formation model is proposed with utility function designed to capture this relation between benefits and cost. The agents in the network are characterized by vectors, called endowment vectors and agents are assumed to maximize

their utility by comparing their endowment vectors with those of others. This vectors are learned from the observed network by solving an optimization problem. Following is the form of the utility function of agent i with respect to agent j: $\sum_{k=1}^{K} b_k \max(z_{jk} - z_{ik}, 0) - \|\mathbf{c} \odot (\mathbf{z}_j - \mathbf{z}_i)\|_2$, where the parameters $\mathbf{b}, \mathbf{c}, \mathbf{W}$ are learned by minimizing the loss function $\mathcal{L}(\mathbf{b}, \mathbf{c}, \mathbf{W}|D)$. Note that the hand-designed reward function used for the synthetic experiments in our work is inspired from the above function. Once the parameters are learned, link prediction is performed by using the utility of an agent i with respect to other agents j as a predictor.

svII [204]. svII is a game-theory based interaction index that captures the notion of similarity between agents. The interaction index is build upon two well-known solution concepts from game theory - the Shapely value and the Bazhaf index. The payoff for an agent in the network is a function of its sphere of influence parameterized by k, where k is the degree of influence. Given this influence game, the above index measures similarity between the spheres of influence of any two nodes. Link prediction is performed by computing the similarity between any pair of nodes not having an edge in current graph and connect the most similar pair.

SEAL [25]. This is the state-of-art discriminative model for link prediction that uses graph neural network to learn general graph structure features from local enclosing subgraphs, embeddings and attributes, thereby capturing higher order properties. Link prediction is performed using standard technique used in learning based approaches.

Graphite [205]. Graphite is a state-of-art latent variable generative model for graphs based on variational auto-encoding. Graphite learns a directed model expressing a joint distribution over the entries of adjacency matrix of graphs and latent feature vectors for every node. Graph neural networks are used in straightforward manner for inference (encoding), while the decoding of these latent features to reconstruct the original graph (generation) is done using a multi-layer iterative procedure. Link prediction is performed by first training the model on a subgraph, adding the test edges back to the graph and evaluating the

probabilities assigned to the test edges.

E.3.3 Evaluation Protocol

In this section, we elaborate and clarify the evaluation protocol used in the modified (perturbed) network setting (quality and interpretability tasks in Section 4) and strategic prediction setting. As described in Algorithm 1, at the start of every epoch, the environment is reset to an empty graph state (only agents, no links) and then the agents learn to form links by observing the real network. However, for evaluation purposes, it is not required to start from an empty graph state. and at convergence, one can input a graph with edges as initial input to MINE and perform evaluation tasks thereof. Note that both the policy network and reward network employ a GNN based architecture and hence any graph structure provide as input will be encoded into continuous representation by these GNN before being used as input to either policy or reward network. WE outline the evaluation steps for each task below:

Quality. For the experiments on Karate network (Table 7.1(b)), the values for three different graphs are the agent specific utilities for two leaders and average of agent-specific utilities across all agents of each community. To obtain these numbers, we first train MINE using the observed network data. Once the model is trained, we input the three graph configurations to the learned reward model and compute the agent-specific utilities directly. The perturbation is done on the original network and the policy generated graph is obtained by running the evaluation policy starting from an empty network.

Interpretability. For the experiments on Australian bank dataset with respect to Katz centrality, we first train MINE using the original network. We then modify the original network to alter centrality of some nodes. Next, we provide this new network as input to the learned reward function without running the evaluation policy. Note that reward estimator is a learned function that takes agent-specific local observation structure (encoded via GNN) as input and hence it directly processes the input network observations for each

agent and return the reward value that is used to report the analysis.

Strategic Prediction. For this task, after the training ends, we provide the graph with 80% training edges as an initial input to MINE. Next we run the evaluation policy and collect the new links formed by the agents. We use these links as predictions and compare them with the test edges to report the results.

E.3.4 Training Configurations

We design our approach based on recently proposed MA-AIRL algorithm but replace its inner loop on-policy RL algorithm with off-policy MAAC algorithm. Hence, we closely follow the architecture and training configurations of MA-AIRL for the descriminators while we adopt the configurations of MAAC for policy learning. MAAC uses Soft-Actor critic to update the policies and in general, SAC does not have any aggressive hyper-parameters that needs to be tuned. We use a target critic function $Q_{\hat{\psi}}$ following [198] as it helps to stabilize the use of experience replay for off-policy reinforcement learning with neural network function approximators. For GNN, we tune propagation round P between 2-4 but found 2 to be best choice. We tune the node embedding dimension in the range of $\{32, 64, 128, 256\}$ based on the graph. We reset each environment after every 100 steps. After 100 steps, we perform 4 updates for the attention critic and for all policies. We perform gradient descent on the Q-function loss objective, as well as the policy objective, using Adam optimizer. Further, following [198], we use 4 heads for our attention critics and dimension of 128 for all hidden units across networks. Table E.1 provides list of hyper-parameters that were used across all the experiments:

Table E.1: Hyper Parameter Configuration Table

HyperParameters	**Values**	**HyperParameters**	**Values**
discount factor	0.99	replay buffer size	1e6
batch size	1024	policy and critic learning rate	0.001
policy entropy temperature	0.01	target policy and critic update rate	0.005
discriminator entropy temperature	0.01	discriminator learning rate	0.0005

All the experiments were conducted on Intel Xeon CPU V4 2.20 GHZ with 64 GB memory and Nvidia GeForce 1080 GPU.

E.4 More Related Work

In this section, we discuss more literature on inverse reinforcement learning.

Inverse Reinforcement Learning. As stated before, the MaxEnt IRL framework [151] aims to recover a reward function that rationalizes the expert behaviors with least commitment and denoted as IRL(π_E): IRL(π_E) $= arg\max_{r\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \mathbb{E}_{\pi_E}[r(s,a)] - RL(r)$ where $RL(r) = \max_{\pi\in\Pi} \mathcal{H}(\pi) + \mathbb{E}_{\pi}[r(s,a)]$. Here, $\mathcal{H}(\pi) = \mathbb{E}_{\pi}[-\log\pi(a|s)]$ is the policy entropy. The forward RL problem in the inner loop makes the above procedure less efficient and hence various improvements have been proposed in the literature. [240, 241] propose one such framework, Adversarial IRL, a sampling based approximation to MaxEnt IRL that uses adversarial training framework where the discriminator takes following specific form: $D(s,a) = \frac{\exp(f(s,a))}{exp(f(s,a)+q_(a|s)}$, where $f(s,a,s') = g(s,a) + \gamma h(s') - h(s)$. The policy is trained to maximize $[\log D - \log(1-D)]$ and the specific form of f is used to alleviate the reward shaping ambiguity where many reward function can explain an optimal policy. It has been shown that at optimality, either f or g will approximate the advantage function of the expert policy and thereby recover the reward upto some approximation while q will approximate the expert policy. Our approach is built on [183] that extends the above setup to multi-agent case and we have discussed all the relevant details in the earlier sections. The above method is general in a sense that it supports cooperative, competitive and mixed environments hence useful for our case. There are other inverse learning methods specific to various multi-agent settings and tasks that include cooperative inverse reinforcement learning [242], non-cooperative inverse reinforcement learning [243] and competitive multi-agent inverse reinforcement learning with suboptimal demonstrations [244]. [245] is a very early work in multi-agent inverse reinforcement learning that form the basis of recent advancements in multi-agent inverse reinforcement learning.

REFERENCES

[1] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Comput. Surv.*, 2006.

[2] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 601–610.

[3] H. Liu and Y. Yang, "Cross-graph learning of multi-relational associations," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[4] M. Pershina, M. Yakout, and K. Chakrabarti, "Holistic entity matching across knowledge graphs," in *2015 IEEE International Conference on Big Data (Big Data)*, 2015.

[5] D. Koutra, H. Tong, and D. Lubensky, "Big-align: Fast bipartite graph alignment," in *2013 IEEE 13th International Conference on Data Mining*, 2013.

[6] P. Buneman and S. Staworko, "Rdf graph alignment with bisimulation," *Proc. VLDB Endow.*, 2016.

[7] B. Kim, K. Lee, L. Xue, and X. Niu, "A review of dynamic network models with latent variables," *arXiv:1711.10421*, 2017.

[8] C. Loglisci and D. Malerba, "Leveraging temporal autocorrelation of historical data for improving accuracy in network regression," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 10, no. 1, pp. 40–53, 2017.

[9] C. Loglisci, M. Ceci, and D. Malerba, "Relational mining for discovering changes in evolving networks," *Neurocomputing*, vol. 150, Part A, pp. 265–288, 2015.

[10] C. Esteban, V. Tresp, Y. Yang, S. Baier, and D. Krompa, "Predicting the co-evolution of event and knowledge graphs," in *FUSION*, 2016.

[11] T. Jiang, T. Liu, T. Ge, L. Sha, S. Li, B. Chang, and Z. Sui, "Encoding temporal information for time-aware link prediction," in *ACL*, 2016.

[12] M. Farajtabar, Y. Wang, M. Gomez-Rodriguez, S. Li, H. Zha, and L. Song, "Coevolve: A joint point process model for information diffusion and network coevolution," in *NIPS*, 2015, pp. 1954–1962.

[13] F. Papadopoulos, M. Kitsak, M. Á. Serrano, M. Boguná, and D. Krioukov, "Popularity versus similarity in growing networks," *Nature*, 2012.

[14] R. M. D'souza, C. Borgs, J. T. Chayes, N. Berger, and R. D. Kleinberg, "Emergence of tempered preferential attachment from optimization," *Proceedings of the National Academy of Sciences*, 2007.

[15] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou, "Heuristically optimized trade-offs: A new paradigm for power laws in the internet," in *International Colloquium on Automata, Languages, and Programming*, Springer, 2002, pp. 110–122.

[16] A.-L. Barabási, "Network science: Luck or reason," *Nature*, 2012.

[17] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker, "On a network creation game," in *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*, 2003.

[18] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[19] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, 2016.

[20] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "Graphrnn: Generative realistic graphs with deep auto-regressive mdoels," *arXiv preprint arXiv:1802.08773*, 2018.

[21] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," *arxiv:2002.09405*, 2020.

[22] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *NIPS*, 2017.

[23] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *NeurIPS*, 2018.

[24] S. V. Bojja, M. Alizadeh, and P. Viswanath, "Graph2seq: Scalable learning dynamics for graphs," *arXiv:1802.04948v3*, 2018.

[25] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *arxiv:1802.09691*, 2018.

[26] R. Trivedi, B. Sisman, J. Ma, C. Faloutsos, H. Zha, and X. L. Dong, "Linknbed: Multi-graph representation learning with entity linkage," in *ACL*, 2018.

[27] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *ICLR*, 2019.

[28] R. Trivedi, H. Dai, Y. Wang, and L. Song, "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs," in *ICML*, 2017.

[29] R. Trivedi, J. Yang, and H. Zha, "Graphopt: Learning optimization models of graph formation," in *ICML*, 2020.

[30] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin, "Entity disambiguation for knowledge base population," in *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

[31] Z. He, S. Liu, M. Li, M. Zhou, L. Zhang, and H. Wang, "Learning entity representation for entity disambiguation," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[32] H. Huang, L. Heck, and H. Ji, "Leveraging deep neural networks and knowledge graphs for entity disambiguation," *arXiv:1504.07678v1*, 2015.

[33] W. M. Campbell, L. Li, C. Dagli, J. Acevedo-Aviles, K. Geyer, J. P. Campbell, and C. Priebe, "Cross-domain entity resolution in social media," *arXiv:1608.01386v1*, 2016.

[34] W. Fang, J. Zhang, D. Wang, Z. Chen, and M. Li, "Entity disambiguation by knowledge and text jointly embedding," in *CoNLL*, 2016.

[35] A. Globerson, N. Lazic, S. Chakrabarti, A. Subramanya, M. Ringaard, and F. Pereira, "Collective entity resolution with multi-focal attention," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[36] J. Pujara and L. Getoor, "Generic statistical relational entity resolution in knowledge graphs," in *Sixth International Workshop on Statistical Relational AI*, 2016.

[37] M. Chen, Y. Tian, M. Yang, and C. Zaniolo, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," 2017.

[38] L. Zhu, d. Guo, J. Yin, G. V. Steeg, and A. Galstyan, "Scalable temporal latent space inference for link prediction in dynamic social networks," *TKDE*, 2016.

[39] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *IJCAI International Workshop on Representation Learning for Graphs*, 2017.

[40] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *AAAI*, 2018.

[41] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," *arXiv:1612.07659*, 2016.

[42] N. Du, Y. Wang, N. He, and L. Song, "Time sensitive recommendation from recurrent user activities," in *NIPS*, 2015, pp. 3492–3500.

[43] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anamoly detection in dynamic networks," in *KDD*, 2018.

[44] J. Li, H. Dani, X. Hu, J. Tang, Y. Change, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *CIKM*, 2017.

[45] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *KDD*, 2018.

[46] G. H. Ngyuyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *WWW*, 2018.

[47] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," *arXiv preprint arXiv:1803.03324*, 2018.

[48] J. You, B. Liu, R. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal directed molecule generation," in *NIPS*, 2018.

[49] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[50] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," *1711.08267*, 2017.

[51] N. D. Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," *arXiv:1805.11973*, 2018.

[52] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[53] C. H. Papadimitriou, "Algorithms, games, and the internet," in *ACM Symposium on Theory of Computing*, 2001.

[54] D. Bilò, T. Friedrich, P. Lenzner, and A. Melnichenko, "Geometric network creation games," in *ACM Symposium on Parallelism in Algorithms and Architectures*, 2019.

[55] M. Jackson and A. Wolinsky, "A strategic model of social and economic networks," *Journal of Economic Theory*, 1996.

[56] R. Myerson, "Game theory: Analysis of conflict," *Harvard University Press*, 1991.

[57] F. Bloch and M. O. Jackson, "The formation of networks with transfers among players," *Journal of Economic Theory*, 2007.

[58] R. Lee and K. Fong, "Markov-perfect network formation: An applied framework for bilateral oligopoly and bargaining in buyer-seller networks," 2013.

[59] T. R. Johnson, "Dynamic network formation: Theory and estimation," 2017.

[60] G. Ridder and S. Sheng, "Estimation of large network formation games," *arXiv:2001.03838*, 2020.

[61] J. Dalton, L. Dietz, and J. Allan, "Entity query feature expansion using knowledge base links," in *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.

[62] E. Gabrilovich and S. Markovitch, "Wikipedia-based semantic interpretation for natural language processing," *J. Artif. Int. Res.*, 2009.

[63] R. Catherine and W. Cohen, "Personalized recommendations using knowledge graphs: A probabilistic logic programming approach," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[64] W. Cui, Y. Xiao, H. Wang, Y. Song, S.-w. Hwang, and W. Wang, "Kbqa: Learning question answering over qa corpora and knowledge bases," *Proc. VLDB Endow.*, 2017.

[65] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *SIGMOD Conference*, 2008.

[66] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a web of open data," in *The Semantic Web*, 2007.

[67] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[68] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.

[69] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 809–816.

[70] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[71] 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, pp. 2787–2795.

[72] H. Xiao, M. Huang, and X. Zhu, "Transg: A generative model for knowledge graph embedding," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[73] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv:1412.6575*, 2015.

[74] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep feature learning for graphs," *arXiv:1704.08829*, 2017.

[75] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems*, 2013, pp. 926–934.

[76] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *arXiv preprint arXiv:1405.4053*, 2014.

[77] R. Kadlec, O. Bajgar, and J. Kleindienst, "Knowledge base completion: Baselines strike back," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017.

[78] J. Feng, M. Huang, Y. Yang, and X. Zhu, "Gake: Graph aware knowledge embedding," in *COLING*, 2016.

[79] Q. Liu, H. Jiang, A. Evdokimov, Z.-H. Ling, X. Zhu, S. Wei, and Y. Hu, "Probabilistic reasoning via deep learning: Neural association models," *arXiv:1603.07704v2*, 2016.

[80] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," 2016.

[81] K. Toutanova, X. V. Lin, W.-t. Yih, H. Poon, and C. Quirk, "Compositional learning of embeddings for relation paths in knowledge bases and text," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2016, pp. 1434–1444.

[82] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relatinal embeddings," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[83] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, 2014.

[84] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *AAAI*, 2011.

[85] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," 2014.

[86] Y. Lin, Z. Liu, M. Sun, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," 2015.

[87] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," 2015.

[88] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015.

[89] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *KDD*, 2016.

[90] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *arXiv preprint arXiv:1403.6652*, 2014.

[91] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[92] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016.

[93] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *AAAI*, 2017.

[94] L. Xu, X. Wei, J. Cao, and P. Y. Yu, "Embedding identity and interest for social networks," in *WWW*, 2017.

[95] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv:1709.05584*, 2017.

[96] B. Chazelle, "Natural algorithms and influence systems," *Communications of the ACM*, 2012.

[97] D. Farine, "The dynamics of transmission and the dynamics of networks," *Journal of Animal Ecology*, vol. 86, no. 3, pp. 415–418, 2017.

[98] O. Artime, J. J. Ramasco, and M. S. Miguel, "Dynamics on networks: Competition of temporal and topological correlations," *arXiv:1604.04155*, 2017.

[99] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *Neural Networks, IEEE Transactions on*, vol. 20, no. 1, pp. 61–80, 2009.

[100] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[101] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," *CoRR*, vol. abs/1603.05629, 2016.

[102] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[103] ——, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[104] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[105] C. Yang, M. Liu, Z. Wang, L. Liu, and J. Han, "Graph clustering with dynamic embedding," *arXiv:1712.08249*, 2017.

[106] P. Sarkar, S. Siddiqi, and G. Gordon, "A latent space approach to dynamic embedding of co-occurence data," in *AISTATS*, 2007.

[107] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *KDD*, 2016.

[108] H. Mei and J. M. Eisner, "The neural hawkes process: A neurally self-modulating multivariate point process," in *NIPS*, 2017.

[109] S. Xiao, M. Farajtabar, X. Ye, J. Yan, L. Song, and H. Zha, "Wasserstein learning of deep generative point process models," in *NIPS*, 2017.

[110] D. Daley and D Vere-Jones, *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. 2007.

[111] O. Aalen, O. Borgan, and H. Gjessing, *Survival and event history analysis: a process point of view*. Springer, 2008.

[112] M. Farajtabar, N. Du, M. G. Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *NIPS*, 2014.

[113] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[114] Y. Wang, B. Xie, N. Du, and L. Song, "Isotonic hawkes processes," in *ICML*, 2016, pp. 2226–2234.

[115] B. Tabibian, I. Valera, M. Farajtabar, L. Song, B. Schölkopf, and M. Gomez-Rodriguez, "Distilling information reliability and source trustworthiness from digital traces," in *WWW*, 2017.

[116] V. Isham and M. Westcott, "A self-correcting pint process," *Advances in Applied Probability*, vol. 37, pp. 629–646, 1979.

[117] M. Farajtabar, X. Ye, S. Harati, L. Song, and H. Zha, "Multistage campaigning in social networks," in *NIPS*, 2016.

[118] A. Zarezade, A. Khodadadi, M. Farajtabar, H. R. Rabiee, and H. Zha, "Correlated cascades: Compete or cooperate.," in *AAAI*, 2017.

[119] M. Farajtabar, J. Yang, X. Ye, H. Xu, R. Trivedi, E. Khalil, S. Li, L. Song, and H. Zha, "Fake news mitigation via point process based intervention," in *ICML*, 2017.

[120] L. Tran, M. Farajtabar, L. Song, and H. Zha, "Netcodec: Community detection from individual activities," in *SDM*, 2015.

[121] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[122] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[123] J. Zhang, X. Shi, J. Xie, h. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated attention networks for learning on large spatiotemporal graphs," in *UAI*, 2018.

[124] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in temporal networks," in *WSDM*, 2017.

[125] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone," *ISA Annual Convention*, 2013.

[126] E. Boschee, J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, and M. Ward, "Icews coded event data," 2017.

[127] D. Cox and P. Lewis, "Multivariate point processes," *Selected Statistical Papers of Sir David Cox: Volume 1, Design of Investigations, Statistical Methods and Applications*, vol. 1, p. 159, 2006.

[128] D. Daley and D. Vere-Jones, *An introduction to the theory of point processes: volume II: general theory and structure*. Springer, 2007, vol. 2.

[129] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.

[130] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 307–361, 2012.

[131] A. Schein, M. Zhou, D. Blei, and H. Wallach, "Bayesian poisson tucker decomposition for learning the structure of international relations," *arXiv:1606.01855*, 2016.

[132] S.-H. Yang and H. Zha, "Mixture of mutually exciting processes for viral diffusion," in *ICML*, 2013, pp. 1–9.

[133] M. Farajtabar, N. Du, M. Gomez-Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *NIPS*, 2014, pp. 2474–2482.

[134] Y. Wang, N. Du, R. Trivedi, and L. Song, "Coevolutionary latent feature processes for continuous-time user-item interactions," in *NIPS*, 2016, pp. 4547–4555.

[135] Y. Wang, E. Theodorou, A. Verma, and L. Song, "A stochastic differential equation framework for guiding online user activities in closed loop," *arXiv preprint arXiv:1603.09021*, 2016.

[136] Y. Wang, G. Williams, E. Theodorou, and L. Song, "Variational policy for guiding point processes," in *ICML*, 2017.

[137] Y. Wang, X. Ye, H. Zha, and L. Song, "Predicting user activity level in point processes with mass transport equation," in *NIPS*, 2017.

[138] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Deep coevolutionary network: Embedding user and item features for recommendation," *arXiv preprint arXiv:1609.03675*, 2016.

[139] U. Sharan and J. Neville, "Temporal-relational classifiers for prediction in evolving domains," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 540–549.

[140] C. Esteban, V. Tresp, Y. Yang, S. Baier, and D. Krompaß, "Predicting the co-evolution of event and knowledge graphs," in *2016 19th International Conference on Information Fusion (FUSION)*, 2016, pp. 98–105.

[141] T. Jiang, T. Liu, T. Ge, S. Lei, S. Li, B. Chang, and Z. Sui, "Encoding temporal information for time-aware link prediction," 2016.

[142] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," *arXiv:1707.07435*, 2017.

[143] V. Singh and P. Lio, "Towards probabilistic generative models harnessing graph neural networks for disease-gene prediction," *arxiv:1907.05628*, 2019.

[144] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[145] A.-L. Barabási and M. Pósfai, *Network science*. Cambridge University Press, 2016.

[146] M. Newman, *Networks: An Introduction*. OUP Oxford, 2010.

[147] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, 2002.

[148] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins, and E. Upfal, "The web as a graph," in *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2000, pp. 1–10.

[149] A. Vázquez, A. Flammini, A. Maritan, and A. Vespignani, "Modeling of protein interaction networks," *Complexus*, vol. 1, no. 1, pp. 38–44, 2003.

[150] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *arxiv:1812.05905*, 2018.

[151] B. D. Ziebart, A. Maas, J. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008.

[152] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[153] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An introduction to exponential random graph (p*) models for social networks," *Social Networks*, vol. 29, no. 2, pp. 173–191, 2007.

[154] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, 2007.

[155] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," in *Advances in Neural Information Processing Systems 21*, 2009.

[156] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, 2010.

[157] P. Erdös and A. Rényi, "On random graphs," *Publ. Math. Debrecen*, vol. 6, pp. 290–291, 1959.

[158] A. D. Broido and A. Clauset, "Scale-free networks are rare," in *Nature Communications*, 2018.

[159] Y. Dong, R. A. Johnson, J. Xu, and N. V. Chawla, "Structural diversity and homophily: A study across more than one hundred big networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[160] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," *arxiv:1802.03480*, 2018.

[161] A. D. Paula, S. Richards-Shubik, and E. Tamer, "Identifying preferences in networks with bounded degree," *Econometrica*, 2018.

[162] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, 1999.

[163] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *in Proc. 17th International Conf. on Machine Learning*, Morgan Kaufmann, 2000, pp. 663–670.

[164] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," 2018.

[165] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," *arxiv:1808.10568*, 2018.

[166] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy via graph neural networks," in *ICLR*, 2018.

[167] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, 2009.

[168] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[169] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[170] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[171] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[172] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 4950–4957.

[173] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[174] B. Liu, W. Hu, M. Zitnik, and J. Leskovec, "Open graph benchmark," *To appear*, 2020.

[175] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *arXiv:1806.01261*, 2018.

[176] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[177] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *arXiv:1812.04202*, 2018.

[178] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, "Convolutional embedding of attributed molecular graphs for physical property prediction," *Journal of Chemical Information and Modeling*, 2017.

[179] F. Dutil, J. P. Cohen, M. Weiss, G. Derevyanko, and Y. Bengio, "Towards gene expression convolutions using gene interaction graphs," *arxiv:1806.06975*, 2018.

[180] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert, "Distance metric learning using graph convolutional networks: Application to functional brain networks," *arxiv:1703.02161*, 2017.

[181] J. Ma, C. Zhou, P. Cui, H. Yang, and W. Zhu, "Learning disentangled representations for recommendation," *arxiv:1910.14238*, 2019.

[182] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018.

[183] L. Yu, J. Song, and S. Ermon, "Multi-agent adversarial inverse reinforcement learning," in *International Conference on Machine Learning*, 2019.

[184] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *International Conference on Machine Learning*, 1994.

[185] L. S. Shapley, "Stochastic games," in *Proceedings of the National Academy of Sciences*, 1953.

[186] E. Maskin and T. J., "A theory of dynamic oligopoly i: Overview and quantity competition with large fixed costs," *Econometrica*, 1988.

[187] ——, "A theory of dynamic oligopoly ii: Price competition, kinked demand curves, and edgeworth cycles," *Econometrica*, 1988.

[188] ——, "Markov perfect equilibrium: I. observable actions," *Journal of Economic Theory*, 2001.

[189] S. Eibelshäuser and D. Poensgen, "Markov quantal response equilibrium and a homotopy method for computing and selecting markov perfect equilibria of dynamic stochastic games," 2019.

[190] A. M. Fink, "Equilibrium in a stochastic n-person game," *Journal of Science of the Hiroshima University*, 1964.

[191] M. Takahashi, "Equilibrium points of stochastic, noncooperative n-person games," *Journal of Science of the Hiroshima University*, 1964.

[192] M. J. Sobel, "Non-cooperative stochastic games," *Annals of Mathematical Statistics*, 1971.

[193] R. E. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, 1954.

[194] R. D. McKelvey and T. R. Palfrey, "Quantal response equilibria for normal form games," *Games and Economic Behavior*, 1995.

[195] ——, "Quantal response equilibria for extensive form games," *Experimental Economics*, 1998.

[196] C. K. Ling, F. Fang, and J. Z. Kolter, "What game are we playing? end-to-end learning in normal and extensive form games," in *International Joint Conference on Artificial Intelligence*, 2018.

[197] Y. Breitmoser, J. H. Tan, and D. J. Zizzo, "Understanding perpetual r and d races," *Economic Theory*, 2010.

[198] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2019.

[199] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[200] Y. Yuan, A. Alabdulkareem, and A. Pentland, "An interpretable approach for social network formation among heterogeneous agents," in *Nature Communications*, 2018.

[201] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, 1953.

[202] N. Pagan and F. Dörfler, "Game theoretical inference of human behavior in social networks," *Nature Communications*, 2019.

[203] E. Nasiri, A. Bouyer, and E. Nourani, "A node representation learning approach for link prediction in social networks using game theory and k-core decomposition," *The European Physical Journal B*, 2019.

[204] P. L. Szczepanski, A. Barcz, T. P. Michalak, and T. Rahwan, "The game-theoretic interaction index on social networks with applications to link prediction and community detection," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015.

[205] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[206] R. Bamler and S. Mandt, "Dynamic word embedding," in *ICML*, 2017.

[207] M. Rudolph and D. Blei, "Dynamic embeddings for language evolution," in *WWW*, 2018.

[208] C. Meng, C. S. Mouli, B. Ribeiro, and J. Neville, "Subgraph pattern neural networks for high-order graph evolution prediction," in *AAAI*, 2018.

[209] Y. Yuan, X. Liang, X. Wang, D.-Y. Yeung, and A. Gupta, "Temporal dynamic graph lstm for action-driven video object detection," in *ICCV*, 2017.

[210] G. Jerfel, M. E. Basbug, and B. E. Engelhardt, "Dynamic collaborative filtering with compund poisson factorization," in *AISTATS*, 2017.

[211] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," *arXiv preprint arXiv:1707.06690*, 2017.

[212] Y. Shen, J. Chen, P.-S. Huang, Y. Guo, and J. Gao, "M-walk: Learning to walk over graphs using monte carlo tree search," in *NeurIPS*, 2018.

[213] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[214] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, 2017.

[215] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[216] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[217] M. Toussaint, "Robot trajectory optimization using approximate inference," in *ICML*, 2009.

[218] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference (extended abstract)," in *IJCAI*, 2013.

[219] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *UAI*, 2016.

[220] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arxiv:1805.00909*, 2018.

[221] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," PhD thesis, Machine Learning Department, Carnegie Mellon University, 2010.

[222] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *ICML*, 2017.

[223] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[224] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 u.s. election: Divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery*, 2005.

[225] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, 2000.

[226] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and Eliassi-Rad, "Collective classification in network data," *Ai Magazine*, 2008.

[227] S. Kok and P. Domingos, "Statistical predicate invention," in *ICML*, 2007.

[228] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *EMNLP*, 2015.

[229] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, 2018.

[230] B. Karrer and M. Newman, "Stochastic blockmodels and community structure in networks," *Arxiv preprint arXiv:1008.3926*, 2010.

[231] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of erdos-renyi graphs," *Physical Review E*, 2012.

[232] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *KDD*, 2016.

[233] G. A. Geses, R. Biswas, M. Alam, and H. Sack, "A survey on knowledge graph embeddings with literals: Which model links better literal-ly?" *arXiv:1910.12507*, 2019.

[234] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, 1992.

[235] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base completion," *arxiv:1702.08367*, 2017.

[236] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, 2016.

[237] U. Doraszelski and J. F. Escobar, "A theory of regular markov perfect equilibria in dynamic stochastic games: Genericity, stability, and purification," *Theoretical Economics*, 2010.

[238] M. O. Jackson, "A survey of models of network formation: Stability and efficiency," 2003.

[239] T. P. Michalak, K. V. Aadithya, P. L. Szczepanski, B. Ravindran, and N. R. Jennings, "Efficient computation of the shapley value for game-theoretic network centrality," *Journal of Artificial Intelligence Research*, 2013.

[240] C. Finn, P. Christiano, P. Abbeel, and S Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv:1611.03852*, 2016.

[241] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations*, 2018.

[242] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances in Neural Information Processing Systems 29*, 2016.

[243] X. Zhang, K. Zhang, E. Miehling, and T. Basar, "Non-cooperative inverse reinforcement learning," in *Advances in Neural Information Processing Systems 32*, 2019.

[244] X. Wang and D. Klabjan, "Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations," in *International Conference on Machine Learning*, 2018.

[245] S. Natarajan, G. Kunapuli, K. Judahy, P. Tadepalliy, K. Kerstingzand, and J. Shavlik, "Multi-agent inverse reinforcement learning," in *International Conference on Machine Learning and Applications*, 2010.