An Improved Link Forecasting Framework for Temporal Knowledge Graphs

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering *by Research*

by

ABINASH MAHARANA
2018111033
abinash.maharana@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
DECEMBER 2023

International Institute of Information Technology

Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "An Improved Link Forecasting Framework for Temporal Knowledge Graphs" by Abinash Maharana, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Advisor: Prof. P. Krishna Reddy

To *Baba* and *Mama*

Acknowledgments

No amount of acknowledgment can sufficiently profess the impact that other people have had in my life. Whatever I have achieved, I owe it to every single one of my teachers, friends, and family; and I apologise if I miss anyone's name here. Know that I am grateful for having had the pleasure of your company.

Thank you, Mom and Dad, for giving me the strength to pursue my dreams. Your love and support mean the world to me, and I can never truly comprehend the sacrifices that you have made to give me a better future. This thesis is for you. I also thank my uncles, who have been as invested in my life as anybody else. I am a better person because of their guidance.

The world is a better place because of teachers. I feel fortunate to have had Prof. P. Krishna Reddy as my advisor. He has played an invaluable role in shaping my perspective on research. His guidance related to life and academics has helped me become a better person. Secondly, I would like to thank Prof. Rage Uday Kiran, who has been the constant driving force behind my work. He has taught me the fundamentals of doing good research work and the art of projecting ideas into text. He has been an excellent guide and a friend. Finally, I would like to thank Mrs. Madhusmita Nath, my high school Computer Science teacher, who sowed the seeds of problem-solving in me from an early age.

I want to thank all my lab members, Vipul, Sriharshita, Shiridi, Bhoomendra, Haranadh, Saideep sir, Narendra sir, Srinivas sir, and Mamatha ma'am. Doing research work at DSAC has been a pleasure. Also, I am very fortunate to have met some fantastic people here at IIIT Hyderabad. Their friendship over the years has been an integral part of my life, and they have also helped me through some tough times. I want to thank, in no particular order: Shivam, Guru, Sanskar, Nikhil, Dhruv, Prabhav, Abhyudit, Istasis, Anshul, Avlok, Akshat, Shivansh, Priyanshu, Tanmay, Shahbaz, Saransh, and Sumanth.

Abstract

Representing knowledge in a diagrammatic form has been a long-standing goal of humanity. Early efforts in the field of knowledge representation, such as symbolic logic and semantic net, have led to the development of The Semantic Web, which provided a strong foundation for the development of Knowledge Graphs (KGs). KGs were first introduced in 1986 and were popularized recently in 2012 with the introduction of the Google knowledge graph. Most KGs suffer from the problem of incompleteness, which has led to many efforts in the field of KG completion. In this thesis, we address a specific subproblem from this field called link forecasting. Link forecasting is the problem of predicting future links in a given temporal knowledge graph (TKG) using the existing data. Although several link forecasting frameworks exist in the literature, most previous studies suffer from reduced performance as they cannot efficiently capture the time dynamics of a TKG.

We present a novel rule-based link forecasting framework by introducing two new concepts: *relaxed temporal random walks* and *link-star rules*. The former concept involves generating rules by performing random walks on a TKG, considering the real-world phenomenon that the order of any two events may be ignored if their occurrence time gap is within a threshold value. The latter concept defines a class of acyclic rules generated based on the natural phenomenon that history repeats after a particular time. Our framework also accounts for the problem of combinatorial rule explosion, making our framework practicable. Experimental results demonstrate that our framework outperforms the state-of-the-art by a substantial margin.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The human pursuit of making sense out of data has led to the development of the field of *knowledge representation*. Early efforts in this field date back to the development of symbolic logic in the General Problem Solver by Newell et al. [39] and the concept of semantic net proposed by R.H. Richens [46]. The Semantic Web [4] was the first large-scale project aimed in this direction, which is aimed at increasing the machine-understanding of the data available on the world wide web. The Semantic Web was built on two foundational pillars, the Resource Description Framework (RDF) [43] to capture the *semantics* of the web, and the Web Ontology Language (OWL) [32] to capture the *ontology* of the same. These efforts, along with other developments in the field of logic and reasoning systems led to the development of 'Knowledge Graphs (KGs)', a term first coined by Stokman and Vries [51] in 1986, when they formalized the process of building a knowledge graph.

Knowledge graphs rose to prominence with the announcement of the Google Knowledge Graph [50] in 2012. Since then, large conglomerates such as Microsoft, Facebook, and Uber have developed enterprise KGs [42]. Table 1.1 reflects the scale of some of these KGs. They have also seen extensive usage [1] in the domains such as medicine and healthcare, education, cybersecurity, telecom, finance, tourism, transportation, and natural sciences. This versatility in application outlines the importance of knowledge graphs.

In this thesis, we propose an improved approach for link forecasting in knowledge graphs. We develop a rule-based link forecasting framework to address the drawbacks of the previous methods. We also demonstrate the benefits of our approach with extensive experimentation and analysis.

The remainder of this chapter is organized as follows: We first explain the concept of knowledge graphs. Next, we discuss temporal knowledge graphs and explain the related research issues. Subsequently, we explain the problem of link forecasting and summarize the proposed framework. At the end of this chapter, we list our contributions and provide the organization of the thesis.

Organization	Entities	Facts
Microsoft	2 Bn.	55 Bn.
Google	1 Bn.	70 Bn.
Facebook	50 Mn.	500 Mn.
IBM	100 Mn.	100 Mn.

Table 1.1: The scale of various real-world knowledge graphs. Source: Noy et al. [42] (data from 2019)

1.1 Knowledge Graphs

Although the definition of a KG varies slightly across literature, a recent survey work by Hogan et al. [20] defines a Knowledge Graph as "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent the entities of interest and whose edges represent the relations between these entities".

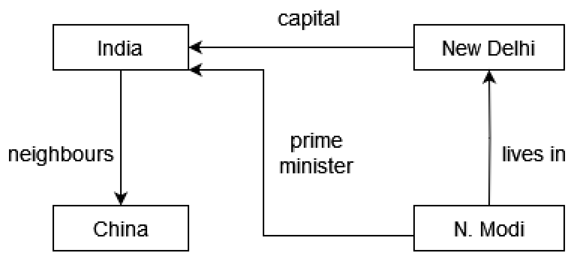

Figure 1.1: A static knowledge graph

Figure 1.1 shows a hypothetical KG, where the vertices are real-world actors such as people and countries while the edges represent some relation that is true in their context. For example, the directed edge with a label 'capital' indicates that the entity 'New Delhi' has a relation of 'capital' with the entity 'India'. This information represented in the triple format as (New Delhi, capital, India) is also called as a *link*[1].

We now discuss a few KGs that are being used in the real-world, to highlight their importance and versatility. Broadly, such graphs can be divided into two categories: (i) Open Knowledge Graphs, and (ii) Enterprise Knowledge Graphs.

[1]A link is sometimes also referred to as an *atom*. To avoid confusion, we will use the term 'link' throughout this thesis

1.1.1 Open Knowledge Graphs

The Open Data Philosophy[2] defines openness as follows: "Open data and content can be freely used, modified, and shared by anyone for any purpose". The Knowledge Graphs published under this philosophy are considered as 'open knowledge graphs'. Some notable examples of such KGs include DBPedia, YAGO, WikiData, and FreeBase. DBpedia [3] is a project that extracts structured data from Wikipedia articles and enriches it with external resources. YAGO [45] also extracts data from Wikipedia, integrating it with the hierarchical structure of WordNet [36] to create a lightweight ontology. Freebase [5], on the other hand, solicits contributions directly from human editors and aimed to address information integration challenges. Wikidata [53] serves as a central knowledge graph for Wikipedia, providing structured data that can be used to update articles across different languages automatically. It allows collaborative editing of both data and schema and has been widely adopted by various applications, famously including Apple's Siri.

1.1.2 Enterprise Knowledge Graphs

The use of knowledge graphs in industry has gained significant attention, with many companies developing proprietary 'enterprise knowledge graphs' to achieve various goals. These goals include improving search capabilities, providing user recommendations, implementing conversational agents, enhancing targeted advertising, empowering business analytics, connecting users, extending multilingual support, facilitating research and discovery, assessing and mitigating risk, tracking news events, and increasing transport automation, among others.

Enterprise knowledge graphs have been deployed in various industries [42, 20, 1]. Commerce companies such as Amazon, eBay, Airbnb, and Uber have developed KGs to enhance product search, recommendations, and semantic features for their platforms. Social networking services like Facebook, LinkedIn, and Pinterest have employed KGs to connect people, provide recommendations, and improve content discovery. The financial sector has leveraged KGs for financial data analytics, sentiment analysis, risk assessment, and investment research. Other industries, including healthcare, transportation, oil and gas, have also explored the use of KGs for applications like drug discovery, genomics research, driving automation, and risk mitigation.

1.2 Temporal Knowledge Graphs

Notably, static KGs cannot capture temporal information and so it is very difficult to assert the validity of each link. For example, as shown in Figure 1.1, India being neighbours of China will always

[2]http://opendefinition.org/

remain true, but N. Modi being the prime minister of India might be valid only for a certain period. Since static knowledge graphs could not capture time-related information, Temporal Knowledge Graphs (TKGs) were developed. A TKG is a knowledge graph that additionally captures time-related information through timestamps.

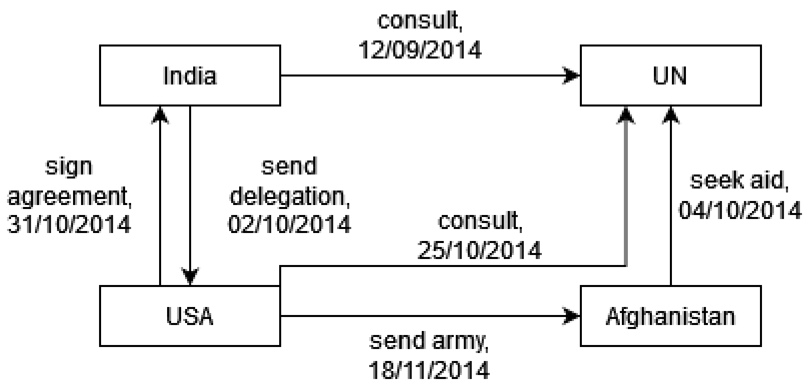

Figure 1.2: A temporal knowledge graph

A sample TKG is shown in Fig. 1.2. Notice how in comparison to Fig. 1.1, there is a timestamp associated with each edge, thus recording temporal information. For example, the directed edge 'consult, 12/09/2014' indicates that the entity 'India' has a relation of 'consult' with the entity 'UN', that is valid on the date '12/09/2014'. This can be interpreted in simple terms as follows: India consulted the UN on 12/09/2014.

TKGs extend the traditional knowledge graph model to capture temporal dynamics in the relationships between the entities. A TKG enables the represention of not only what entities are related but also how and when these relationships change over time. For example, a TKG could represent the evolution of a social network over time, capturing how individuals join or leave the network and how their relationships with other individuals change over time.

Temporal knowledge graphs have a wide range of applications that are made possible by the inclusion of temporal information. For example, TKGs can be used to analyze social networks [16], financial assessment [21], and generate personalized recommendations [35]. In each of these applications, including temporal information is critical to the accuracy and usefulness of the TKG. In this manner, a TKG enables us to make predictions, detect anomalies, and optimize performance in various domains by capturing how relationships and behaviors change over time.

1.3 Research on Knowledge Graphs

The research on knowledge graphs can be classified into three domains:

1. **Knowledge Representation Learning** deals with the study of embedding-spaces in the context of machine learning. This aids in capturing the semantic information of entities and relations in a KG. Specifically, this includes the study of the representation space [15], scoring functions [37], and auxiliary information [54].

2. **Knowledge-aware Applications** refers to the application-level research of KGs, which includes the development of Question Answering systems [8], Recommender systems [17], and Semantic Search [60]. Domain-specific applications of KGs [1] is also an important research area in this category.

3. **Knowledge Acquisition** deals with the study of extracting knowledge from text sources, semi-structured, and structured data in order to create knowledge graphs. It can be classified into three categories: entity discovery [56], relation extraction [40], and knowledge graph completion (KGC) [7]. An essential problem in the context of KGC is *link forecasting*, which is the problem of predicting future connections in a given TKG. This thesis aims to propose an improved approach to tackle the link forecasting problem.

1.4 About link forecasting

Link forecasting is an important subproblem in the area of knowledge graph completion that aims to predict a future event based on a given set of past events drawn from a TKG. Formally, the problem is to find a suitable candidate for a missing entity in a given link with a future unseen timestamp, as illustrated in the following example.

Example 1 *Given a query link* $(Afghanistan, consult, ?, 29/11/2014)$*, the problem of link forecasting is to traverse the TKG shown in Fig. 1.2, which contains all the past events and use them to identify an appropriate candidate entity to replace* $?$*. Notice that, the timestamp in the query '*$29/11/2014$*' is a future unseen timestamp because the newest existing timestamp in the TKG is '*$18/11/2014$*'.*

In the real-world, link forecasting can be used for event prediction [30], medical applications for predicting drug patterns and behaviour [31], as well as applications in social network graphs to predict possible future links [11]. Apart from that, the broader problem of knowledge graph completion will always remain very important as large-scale data is often incomplete in nature.

In the past, many embedding-based frameworks [14, 15, 18, 25, 26, 27, 59] have exploited temporal information in a graph to perform link forecasting. However, these frameworks lack explainability, which is crucial for developing transparent and interpretable applications. Recently, Liu et al. [30] exploited the concept of 'temporal logical rules' and presented an explainable state-of-the-art framework known as TLogic. However, even the TLogic framework misses many useful rules, which in turn, negatively affect its forecasting efficiency. In this thesis, we study its shortcomings in detail, and discuss the reasons behind the framework missing useful rules for link forecasting. We then propose a new framework which takes these gaps into account to make an improved link forecasting framework.

1.5 Overview of the proposed approach

We propose an improved framework called Temporally Relaxed Knowledge Graph Miner (TRKG-Miner). Our framework improves over the previous approaches via the means of two novel ideas, in the form of adding temporal relaxation, and the introduction of acyclic rules for link forecasting.

We have observed that, the current time constraints imposed on the graph random walks required for generating logical rules are too strict. Due to strictness, many useful walks are being missed out and leads to missing many useful rules for link forecasting. On the other hand, eliminating time constraints from this process leads to the generation of numerous spurious rules. To overcome this problem and reach an ideal middle-ground, the proposed approach employs a new time-relaxation parameter called as *maximum time gap* (δ). This parameter is used to capture the irregular occurrence order of temporal events by allowing for a small time-gap between the consecutive events. This aids in finding useful rules, which were previously being missed out.

Existing rule-based methods for link forecasting mine only cyclic rules from temporal knowledge graphs. We have observed that, cyclic rules alone are insufficient for link forecasting as they are not able to capture all the patterns that exist in a TKG. In an effort to get closer to the goal of extracting as much useful information as possible from a TKG, we propose a framework to generate and use acyclic rules for link forecasting, while also ensuring that there is no rule-explosion.

1.6 Contributions

The major contributions of this thesis are as follows:

- To improve link forecasting, we propose two new concepts: temporal relaxation, and link-star rules.
- We develop a novel explainable rule-based link forecasting framework called TRKG-Miner for temporal knowledge graphs using the above concepts.
- We perform extensive experimentation to verify the effectiveness of our proposed framework, and present a thorough analysis of the results.

1.7 Thesis organization

The rest of the thesis is organized as follows.

- In Chapter 2, we discuss the related work.
- In Chapter 3, we discuss the background associated with rule-based link forecasting.
- In Chapter 4, we present the proposed framework and the experimental results.
- Finally, in Chapter 5, we conclude the thesis with future research directions.

Chapter 2

Related Work

In this chapter, we discuss the related work. First, we discuss the work related to the development of knowledge graphs over time. Next, we discuss the various areas of research that are ongoing for KGs. We discuss the related works on the problem of link forecasting. Finally, we discuss various approaches to solving it using embedding-based and random-walk-based methods and how the proposed approach differs from the preceding approaches in the literature.

2.1 Knowledge Graphs

The concept of knowledge graphs can be traced back to the origins of diagrammatic forms of knowledge representation, which began with Aristotle's notion of visual reasoning through forms such as Euler circles and Venn diagrams. Representation of knowledge with computers dates back to the development of symbolic logic in the General Problem Solver by Newell et al. [39] and the concept of semantic net proposed by R.H. Richens [46]. The next most significant step in this timeline was the introduction of The Semantic Web [4], which was proposed to enhance the search and machine-understanding of the World Wide Web.

In the semantic web, information is given well-defined meaning. The semantic web is built upon two central concepts: Resource Description Framework (RDF) [43], and Web Ontology Language (OWL) [32]. RDF is a foundation for processing *metadata* of the web, which is aimed at developing machine-understanding of the same. OWL is intended to provide a language for establishing and creating web ontologies. This additional layer of interpretation captures the *semantics* of the data. This also resulted in many open knowledge bases/ontologies being published, such as YAGO, DBPedia, WordNet, and Freebase.

In 1986, Stokman and Vries [51] proposed the idea of structuring knowledge in a graph, hence giving rise to the term 'Knowledge Graphs'. However, it was much later in 2012, when knowledge graphs gained popularity with the introduction of the Google Knowledge Graph, which was aimed at enhancing the search capabilities of their search engine. Notably, Google also introduced the Knowledge Vault [12], which was a new way to build large-scale knowledge graphs. These series of events accelerated the research on knowledge graphs, which can be classified into three domains:

1. **Knowledge Representation Learning:** deals with the study of embedding-spaces in the context of machine learning. This aids in capturing the semantic information of entities and relations in a KG. Specifically, this includes the study of the following: (i) Representation spaces are used to learn low-dimensional embeddings of entities and relations. Goel et al. [15] provide a comprehensive survey of many such methods. (ii) Scoring functions are used to measure the plausibility of facts. The survey by Ji et al. [24] covers a wide variety of scoring functions used in knowledge representation learning. (iii) Auxiliary information is often used in conjunction with knowledge graphs to enhance the information present within them. Ji et al. [24] also cover a range of methods that use auxiliary information for knowledge representation learning.

2. **Knowledge-aware Applications:** refers to the application-level research of KGs, which includes the development of the following: (i) Question Answering Systems, which leverage KGs to find answers to natural language questions input by human users. The survey by Chakraborty et al. [8] covers many such neural network-based question answering systems. (ii) Recommendation Systems, which are used to model user preferences. Guo et al. [17] covers many such systems which leverage the power of knowledge graphs to create recommender systems. (iii) Semantic Search is important to find entities that share a similar meaning. For this problem, Zhu and Iglesias [60] created a framework that provides similarity tools and datasets that allow users to perform semantic search in a KG. (iv) KGs have also seen extensive usage in various domains in the real-world. The work by Abu et al. [1] covers a wide range of such domain-specific applications of KGs.

3. **Knowledge Acquisition:** deals with the study of extracting knowledge from text sources, semi-structured, and structured data in order to create knowledge graphs. It can be classified into three categories: (i) Entity discovery is the problem of discovering new entities from various sources such as news streams, web pages, and social feeds. The work by Wu et al. [56] contains a study of several embedding-based models that are being used for entity discovery. (ii) Relation extraction is the problem of extracting semantic relations between pairs of entities in text. This problem has primarily been approached using several language processing and deep learning methods as shown in the work by Nguyen et al. [40]. (iii) Knowledge graph completion [49, 7] is the problem of filling the missing information in the KG using various machine learning and graph analytical techniques. We discuss this problem in further detail in the next section.

2.2 Knowledge Graph Completion

Since large-scale data is often incomplete, researchers faced a crucial problem in the form of filling the missing information in knowledge graphs [20]. This led to the growth of the field of knowledge graph completion (KGC). KGC is the problem of predicting and completing the missing parts of a link. The survey on KG completion by Shen et al. [48] provides a comprehensive study of several KGC techniques. The survey by Cai et al. [7] presents a study of several TKG completion techniques. KGC frameworks are generally tested on one or more of the following subtasks:

1. **Link prediction:** The problem of link prediction is to find the best candidate for a missing entity in a given link. We discuss this problem in further detail in the next section.

2. **Triple classification:** Triple classification is the problem of determining whether to add a given link to a KG by estimating whether it is true. Xie et al. [57] suggested a KGC model that takes advantage of the hierarchical structure present in the entities within the knowledge graph to perform KGC. They test their model on the subtasks of link prediction and triple classification. Jaradeh et al. [23] suggested leveraging pre-trained language models [38] to perform KGC on a scholarly knowledge graph. They benchmarked their model using triple classification.

3. **Relation prediction:** The problem of relation prediction is to compute the probability of establishing a relation between two given entities. Cui et al. [10] suggested a framework that augments existing embedding-based KGC models with type information from the knowledge graph. They test their framework on the subtask of relation prediction. Similarly, Li et al. [28] leverage multi-hop paths in KGs to capture the global semantics of the KG for relation prediction.

Out of the above tasks, the one that interests us the most is link prediction, specifically, a variant of it known as 'link forecasting.' While link prediction is a problem that applies to all KGs, link forecasting is a problem specific to TKGs. The problem of link forecasting is to find the best candidate for a missing entity in a given link *with a future timestamp*. That is, we are essentially forecasting a future event instead of filling in missing information. The rest of this chapter discusses literature specific to our problem, beginning from the first approaches for link prediction on static KGs.

2.2.1 Link prediction on static knowledge graphs

Trouillon et al. [52] propose an embedding-based link prediction model called 'ComplEx' which uses a three-way tensor factorization approach that embeds entities and relationships into a continuous vector space. The model is able to learn latent representations of both entities and relationships, allowing it to predict missing or unknown relationships in the data. It presents a new approach for analyzing complex datasets with multiple relationships, but there are also some potential drawbacks and limitations to consider. One major drawback of the ComplEx model is its computational complexity. The model uses a three-way tensor factorization approach that embeds entities and relationships into a continuous vector space. This requires a large number of parameters, which can make training and inference computationally expensive. As the size of the dataset and the number of relationships increase, the model's computational requirements can become prohibitively high.

Bordes et al. [6] present a new model for learning from multi-relational data called TransE. The model represents entities and relations as continuous vectors in a low-dimensional space and uses translation operations to capture the relationships between entities. One major limitation of the TransE model is that it assumes that all relationships between entities are one-to-one, meaning that each entity is connected to only one other entity by a single relation. In reality, many relationships between entities may be more complex, involving multiple entities and/or multiple relations.

DistMult is a model proposed by Yang et al. [58] for learning from multi-relational data that represents entities and relations as low-dimensional vectors in a latent space. Unlike some other models, DistMult models relations as diagonal matrices, which allows it to capture correlations between entities and relations more effectively. The model uses a bilinear form to calculate the plausibility of a relationship between two entities based on the embeddings of the entities and the relation connecting them.

Nickel et al. [41] proposes a new model, called RESCAL (REscaling for SCAlable Learning of Multi-relational data), for learning from multi-relational data. The model represents entities as low-dimensional vectors and relations as matrices that act as linear transformations between entity vectors. One limitation of RESCAL is that it assumes that all relations between entities are equally important, and it does not provide a way to weight the importance of different relations. This can be problematic in some domains where some relations may be more significant than others. RESCAL also requires all relations to be represented as matrices, which can be computationally expensive for large datasets with many relations.

2.2.2 Related work on temporal knowledge graphs

TTransE by Leblay et al. [27] adds a temporal dimension to TransE by incorporating time as an additional entity in the model. In TTransE, each entity and relation is associated with a continuous embedding vector in the same space, and the goal is to learn these embeddings such that the model can accurately predict missing relationships in the graph. TTransE achieves this by introducing a time dimension, where each triple in the graph is associated with a timestamp indicating the time when the relationship occurred. TTransE models temporal dynamics by treating time as a special entity, which is added to the head and tail embeddings of each triple. These temporal embeddings capture the change in the relationships over time, allowing the model to distinguish between different versions of the same relationship.

García-Durán et al. [14] proposed a model called Temporal Attention-based DistMult (TA-DistMult) . TA-DistMult is a tensor factorization model that uses attention mechanisms to handle temporal dynamics and complex relations. The model is based on DistMult, which is a bilinear model that projects entities and relations into a shared embedding space. In TA-DistMult, attention mechanisms are used to capture the temporal dynamics of the knowledge graph. The model uses an attention mechanism to weight the embeddings of entities and relations based on the time they occur. This allows the model to capture how the relationships in the graph change over time.

TNTComplEx by Lacroix et al. [26] is a knowledge graph embedding model that extends the ComplEx model to handle temporal dynamics in the KG. This model combines a tensor factorization approach with attention mechanisms to capture both the structure and the temporal dynamics of the KG. In TNTComplEx, each entity and relation is represented by a complex-valued embedding vector. The model uses a tensor factorization approach to capture the structure of the KG, where the embeddings of entities and relations are combined using tensor products. To capture temporal dynamics, TNTComplEx uses an attention mechanism that weights the embeddings of entities and relations based on their temporal proximity to the query time. This allows the model to capture the evolution of the relationships in the KG over time.

CyGNet is a deep learning-based approach for link prediction in TKGs, proposed by Zhu et al. [59]. CyGNet is designed to address the challenges of link prediction in TKGs, including the sparsity and noise in the data, and the complex temporal dependencies between the entities and their relationships. It leverages the concept that facts/links often show repetition to describe an embedding-based model which doesn't only predict future events, but also identifies facts with recurrence.

RE-Net [25] is a neural network-based model proposed by Jin et al. for the task of link prediction in KGs. The main idea behind RE-Net is to use a neural network architecture that combines relational information with entity information to make predictions. The model consists of three components: and entity encoder, a relation encoder, and a relation network. The key innovation in RE-Net is the use of relation networks to capture the interactions between entities and relationships. The relation network allows the model to capture complex patterns in the data that may not be easily captured by traditional methods such as rule-based or statistical approaches.

xERTE (eXtended Entity-Relation-Typing with Entity descriptions) is a method proposed by Han et al. [18] for link prediction in knowledge graphs. The approach of xERTE is based on a combination of entity descriptions and relation types to enhance link prediction. The xERTE model consists of three main components: an entity encoder, a relation encoder, and a scoring function. The entity encoder learns representations for entities using their descriptions, related entities, and their types. The relation encoder learns representations for relation types using the entities that participate in those relations. The scoring function computes the likelihood of a relation between two entities based on their representations and the relation type.

2.2.3 Rule-based link forecasting frameworks

Rule-based methods and embedding-based methods are two popular approaches for link forecasting in knowledge graphs. While both methods have their own strengths and weaknesses, rule-based methods offer several advantages over embedding-based methods in terms of explainability and interpretability. While embeddings are hidden and obscure in nature, association rules are explainable because they have a logical human-interpretation.

AMIE by Galárraga et al. [13] is one of the earliest known frameworks for mining association rules from KGs. AMIE focuses on leveraging ontological KGs and limited factual information to discover meaningful patterns using association rule mining. It does so by applying a set of novel algorithms that take into account the inherent uncertainty and incompleteness of knowledge bases.

AnyBURL (Anytime Bottom-Up Relational Learning) is a method proposed by Meilicke et al. [34] for learning expressive rules from KGs. The aim of AnyBURL is to identify patterns or regularities in the relationships between entities in a KG and use these patterns to make predictions about unknown relationships. The approach of AnyBURL is based on a bottom-up learning strategy that starts with simple rules and iteratively refines them to produce more complex and expressive rules. AnyBURL can learn a wide range of rules, including Horn rules, non-Horn rules, and rules with existential quantification. Overall, while AnyBURL is a powerful method for learning expressive rules from knowledge graphs, it may not be well-suited for handling temporal knowledge graphs without appropriate modifications or extensions. Thus, frameworks such as TLogic have been proposed.

TLogic is a framework proposed by Liu et al. [30] for link forecasting in TKGs. Unlike traditional approaches that use statistical or machine learning models, TLogic leverages logical rules extracted via temporal random walks to make predictions about the likelihood of future links in a network. In addition to increasing explainability, this method also outperforms most embedding-based methods.

2.3 Differences with the existing approaches

The proposed approach differs from the preceding approaches in the following ways: Static KG link prediction approaches overlook the crucial temporal information present in TKGs. Similarly, it was observed that frameworks extended from static KG link prediction approaches do not capture the temporal information of TKGs properly and thus still suffer from performance issues. This is evident in their inferior performance as compared to the other methods during experimentation. While embedding based link prediction methods have become popular in recent years due to their high predictive accuracy in various applications, these methods lack explainability and transparency in their predictions. Our proposed approach is a rule-based framework that makes use of random walks for link forecasting, which helps with explainability while also performing significantly better than the previous approaches.

2.4 Summary

In this chapter, we have discussed several approaches for link forecasting on knowledge graphs. We have observed that even state-of-the-art approaches for link forecasting in TKGs such as TLogic are not able to mine enough rules, which makes them inefficient for link forecasting. In the following chapter, we build up the background necessary to understand rule-based link forecasting.

Chapter 3

Preliminaries

The integration of time information into Knowledge Graphs has led to the development of Temporal Knowledge Graphs (TKGs). In TKGs, nodes and edges are associated with temporal intervals, allowing the representation of time-varying relations between entities. In this chapter, we formalize the concept of TKGs, introduce random walks, Horn rules, and discuss the problem of link forecasting, which aims to predict the existence or absence of a link between entities in a TKG at a future time. We conclude by defining the central problem statement addressed by this thesis.

3.1 Temporal Knowledge Graphs

Formally, a TKG can be described as follows: Let $\mathcal{E} = \{e_1, e_2, \cdots, e_m\},\ m \geq 1$, be a set of entities. Let $\mathcal{R} = \{r_1, r_2, \cdots, r_n\},\ n \geq 1$, be a set of relations. Let $\mathcal{T} = \{t_1, t_2, \cdots, t_o\},\ o \geq 1$, be an ordered set of timestamps. A link (or an edge), denoted as $lk_i,\ i > 0$, is a quadruple (e_{sub}, r, e_{obj}, t), where $e_{sub} \in \mathcal{E}$ corresponds to a subject entity, $r \in \mathcal{R}$ represents a relation, $e_{obj} \in \mathcal{E}$ is an object entity, and $t \in \mathcal{T}$ represents a timestamp. For each link $lk_i = (e_{sub}, r, e_{obj}, t)$, there exists an inverse link, denoted as $l\hat{k}_i = (e_{obj}, r^{-1}, e_{sub}, t)$, that allows us to perform graph walks along this link in both directions. The relation r^{-1} is called the *inverse relation* of r. A temporal Knowledge Graph, denoted as TKG, is a set of links, i.e., $TKG = \cup_{i=1}^{p} lk_i,\ p \geq 1$.

Example 2 *Let $\mathcal{E} = \{Fred, Robin, France, Belgium, Spain\}$ be the set of entities.*

Let $\mathcal{R} = \{talk, visit\}$ be the set of relations. Let $\mathcal{T} = \{19/05, \cdots, 05/06\}$ be the set of timestamps. A link, say $lk_1 = (Robin, visit, France, 05/06)$, where 'Robin' is a subject entity, 'visit' is a relation, 'France' is an object entity, and '05/06' is a timestamp. For this link, there exists an inverse link $l\hat{k}_1 = (France, visit^{-1}, Robin, 05/06)$. A hypothetical TKG containing all such links is shown in Fig. 3.1. We will use this TKG as the running example to explain our framework.

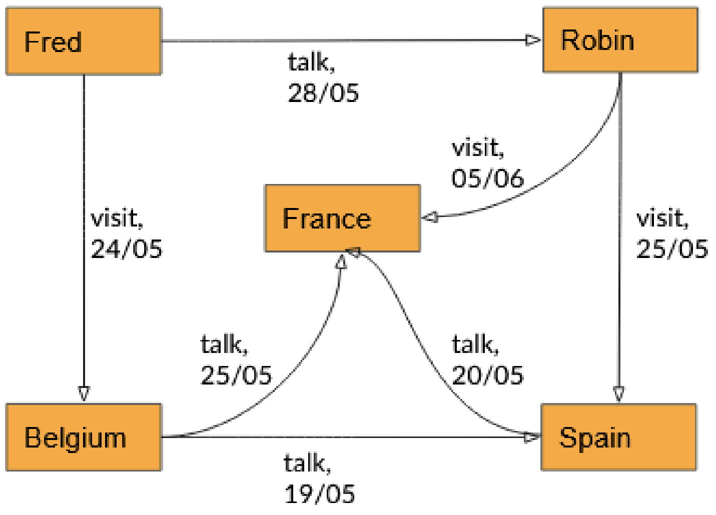

Figure 3.1: A hypothetical temporal knowledge graph. We will use this TKG to explain the proposed approach.

3.2 Random Walks

Random walks on TKGs refer to a stochastic process in which a walker moves from one entity to another through the relationships of the graph, taking into account the timestamps associated with those relationships. Random walks on TKGs have several applications, such as in recommender systems, link forecasting, and anomaly detection. For example, a recommender system aims to recommend items to users based on their past interactions with the system. Random walks can be used to model the user's behavior over time and to identify related items that the user may be interested in.

One approach to random walks on TKGs is to use a modified version of the standard random walk algorithm, called the time-aware random walk. In this algorithm, the walker moves from one entity to another according to the probabilities associated with the relationships of the graph, taking into account the timestamps associated with those relationships. Specifically, the probability of moving from entity i to entity j is proportional to the weight of the edge between i and j, multiplied by a time-dependent factor that reflects the recency of the relationship. Based on this concept, we present the formal definition of a random walk on a Temporal Knowledge Graph.

Definition 1 ***Random Walk:*** *A Random Walk of length $n \in \mathbb{N}$ on a Temporal Knowledge Graph is an ordered set of connected links. That is, $W = \langle (e_1, r_1, e_2, t_1), (e_2, r_2, e_3, t_2) \dots (e_n, r_n, e_{n+1}, t_n) \rangle$*

Example 3 *The following sequence of links is a random walk on the TKG in Fig. 3.1: $\langle (Belgium, visit^{-1}, Fred, 24/05), (Fred, talk, Robin, 28/05), (Robin, visit, Spain, 25/05) \rangle$.*

3.3 Horn Clauses and Temporal Rules

Horn clauses [22] are a type of logical formula widely used in logic programming. A Horn clause is a special logical formula with at most one positive literal (i.e., an atomic proposition that is true) and any number of negative literals (i.e., atomic propositions that are false). Horn clauses are essential in logic programming because they can be efficiently solved by an inference engine, making it possible to derive solutions to problems quickly.

There are two main types of Horn clauses:

1. Fact: A fact is a Horn clause with no negative literals, and it simply states that a particular predicate is true. For example, the Horn clause $\texttt{p}$ is a fact that simply states that $\texttt{p}$ is true.

2. Rule: A rule is a Horn clause that has one positive literal and one or more negative literals. A rule is used to derive a conclusion from a set of conditions. For example, the Horn clause $\texttt{p} \leftarrow \texttt{q} \wedge \texttt{r}$ is a rule that states that if $\texttt{q}$ and $\texttt{r}$ are true, then $\texttt{p}$ must also be true.

Additionally, there are two ways to represent a Horn clause: Disjunction and Implication.

1. Disjunction form: A Horn clause is represented as a disjunction of atomic propositions, where at most one of the propositions is true. In disjunction form, the Horn clause can be thought of as a set of constraints, where each proposition is a constraint that must be satisfied.

2. Implication form: In implication form, a Horn clause is represented as an implication of atomic propositions, where the conclusion is the positive literal and the conditions are the negative literals. In implication form, the Horn clause can be thought of as a rule where the conditions must be true in order for the conclusion to be true.

For example, the Horn clause in the implication form $\texttt{p} \leftarrow \texttt{q} \wedge \texttt{r}$ can be represented in disjunction form as $\texttt{p} \vee\neg\, \texttt{q} \vee\neg\, \texttt{r}$. Here, the $\neg$ symbol denotes negation, the $\vee$ symbol denotes disjunction, the $\wedge$ symbol denotes conjunction, and the $\leftarrow$ symbol denotes implication. Both disjunction form and implication form are equivalent representations of Horn clauses and can be used interchangeably depending on the needs of the problem.

Horn rules can be used for graph analysis [33, 9, 47], where the goal is to predict the presence or absence of links between pairs of nodes in a graph. Horn rules allow us to incorporate prior knowledge and domain expertise into the link forecasting process. We can use existing knowledge about the structure and properties of the graph to formulate Horn rules that capture the underlying patterns and relationships. For example, we can use Horn rules to encode the intuition that nodes with similar attributes are likely to be connected or that nodes with common neighbors are more likely to be connected.

Horn rules are well-suited for efficient computation, which is critical for large-scale link forecasting problems. Horn rules can be solved using a forward-chaining algorithm [19], which only considers the rules that are relevant to the problem at hand. This makes it possible to quickly derive solutions to link forecasting problems even for very large graphs. Further, Horn rules are highly flexible and expressive, which means they can capture complex patterns and relationships in the graph. Horn rules can incorporate both positive and negative information about the graph, which allows them to capture subtle patterns that may be missed by other methods.

In summary, Horn rules offer several advantages for link forecasting, including the ability to incorporate domain knowledge, efficient computation, flexibility and expressivity. Keeping these advantages in mind, we define Temporal Rules as follows:

Definition 2 ***Temporal Rule:*** *Let E_i and T_i represent entity and timestamp variables, respectively. A Temporal Rule of length $n \in \mathbb{N}$ is a Horn Rule composed in the following form: $((E_s, r_h, E_o, T_h) \leftarrow \wedge_{lk_i=1}(E_i, r_i, E_{i+1}, T_i))$*

In a Temporal Rule, the left-hand side of the arrow is called the *rule head*, while the right-hand side is called the *rule body*, which is represented by an ordered conjunction of body links (E_i, r_i, E_{i+1}, T_i). All the entities and timestamps are variables, whereas all the relations are fixed. r_h is called as the *head relation*. A rule of this form implies that if the rule body holds along with the given time constraints, then the rule head is true.

Example 4 *The following is an example of a Temporal Rule:*

$$(E_1, host, E_3, T_3) \leftarrow (E_1, call, E_2, T_1) \wedge (E_2, invite, E_3, T_2) \tag{3.1}$$

This rule indicates that if E_1 calls E_2 and E_2 invites E_3, then E_1 will host E_3.

Having established all the required background, we move on to define the problem statement of this thesis.

3.4 Link forecasting problem

The problem of link forecasting is to predict a future event from a given set of past events present in the form of a temporal knowledge graph. Formally, we present the problem statement as follows:

Given a query containing a future timestamp, say $(e_{sub}, r_x, ?, t_{future})$ or $(?, r_x, e_{obj}, t_{future})$, the goal of link forecasting is to predict a missing (subject or object) entity. The term t_{future} represents a future timestamp that does not exist in the TKG. That is, $t_{future} > t_o$ and $t_{future} \notin \mathcal{T}$.

Example 5 *The TKG shown in Fig.3.1 records the events that happened from '19/05' to '05/06' in a year. Given a query with an unseen future timestamp, say (Fred, visit, ?, 13/06), the goal of link forecasting is to predict an appropriate candidate entity to replace ?.*

3.5 Summary

In this chapter, we introduced the necessary background required to understand the problem and the proposed solution. We began with a formal definition for TKGs and explained random walks and temporal rules, concluding with defining the link forecasting problem. In the next chapter we will introduce TRKG-Miner, our proposed link forecasting framework.

Chapter 4

Proposed Framework

In this chapter, we present the proposed approach to improve link forecasting in TKGs, called Temporally Relaxed Knowledge Graph Miner (TRKG-Miner). In the previous chapters we have introduced knowledge graphs and discussed work relevant to the problem of link forecasting. We have also built up the necessary background required to understand the concepts that will be applied in this chapter. Now, we explain the motivation behind our approach as well as the ideas and algorithms that constitute our approach. We conduct extensive experimentation and a discussion of the results, which show a significant improvement of the proposed approach over the previous best approach.

4.1 Introduction

We have defined the problem of link forecasting in temporal knowledge graphs in the preceding chapter. Briefly, link forecasting is the problem of predicting a missing entity in a link with a future timestamp for a given knowledge graph. In the related work, we have discussed several approaches aimed at tackling this problem. Among those approaches, TLogic is a rule-based link forecasting framework that performs better than all the other discussed approaches while also being explainable. However, we have observed several drawbacks in the TLogic framework that affect its performance negatively. We address some of these problems by using two concepts, namely, temporal relaxation and link-star rules. We use these concepts to build TRKG-Miner. We follow it up by performing extensive experimentation to test the effectiveness of our approach's effectiveness and the usefulness of all the algorithm parameters. We can observe that our approach outperforms the previous approaches by a substantial margin.

The rest of the chapter is organized as follows. In the next section, we briefly explain the TLogic framework and discuss the drawbacks found by us. Then, we discuss the basic ideas behind our approach and explain how they are integrated into our framework. This is followed by a section that describes our proposed approach, along with the algorithms involved in our framework. After that, we discuss the experimental setup. We then present and discuss the results of these experiments.

4.2 TLogic and its drawbacks

TLogic is a framework proposed by Liu et al. [30] for link forecasting in TKGs. Unlike traditional approaches that use statistical or machine learning models, TLogic leverages logical rules to predict the likelihood of future links in a network. This makes it one of the best frameworks for link forecasting on TKGs. At a high level, the TLogic framework involves two key steps:

- Rule Generation: The process of generating logical rules from a given TKG. This step can be divided into two parts:
  - Rule Extraction: The first step involves extracting logical rules from the observed network data. These rules capture the relationships and patterns in the network and can be represented using first-order logic (FOL).
  - Rule Abstraction: In this step, the extracted rules are abstracted to make them more generalizable and applicable to new situations. This is done by replacing specific entities and predicates in the rules with variables, which allows the rules to be applied to different contexts.
- Rule Evaluation: Finally, the abstracted rules are evaluated to make predictions about the likelihood of future links in the network. This is done by applying the rules to the current state of the network and then using logical inference to determine the probability of a new link forming between two nodes.

Upon close inspection of the TLogic framework, the following drawbacks can be observed:

1. Rule abstraction trade-offs: While rule abstraction allows the extracted rules to be more generalizable, there are trade-offs to consider. For example, too much abstraction can lead to oversimplification and loss of important details, while too little abstraction can result in rules that are too specific and not applicable to new contexts. In comparison, AnyBURL uses semi-abstract rules as well.

2. Absence of entity-aware random walks: The probabilities for jumping from one node to another are decided based on a similarity metric that considers only the gap between timestamps. In this context, entity knowledge can also be used for decision making during the walks. For instance, it is more likely that a relation ‘talk’ is valid between two people rather than a building and a vehicle.

3. Brute-force cycle finding method: The process for obtaining cyclic walks is brute-force, and thus there is a lot of scope for improvement there by using cycle-finding techniques such as simple

depth/breadth first search, or union-find. Additionally, in the context of logical rules only small cycles of lengths 2-4 are required, and this constraint can be exploited to develop specific cycle-finding algorithms.

4. Not using acyclic rules: The TLogic framework implicitly assumes that cyclic rules alone are sufficient for link forecasting, and there is no provision for generating or using acyclic rules. However, cyclic rules alone can be insufficient to capture all the diverse patterns that can arise in a KG. Thus, there is a need to generate acyclic rules as well.

5. Presence of strict temporal constraints: The temporal constraints imposed while performing random walks and generating rules is too strict. However, in the real world, events happening at around the same time may or may not happen in a specific temporal order. For example, multiple events can happen simultaneously, or there can be cyclical relationships between two events.

Out of the aforementioned drawbacks, we address the last two drawbacks i.e. not using acyclic rules and the presence of strict temporal constraints. These two problems collectively amount to the problem of *the inability to mine many useful rules*. We discuss both of these drawbacks one-by-one in detail and explain the plan to tackle them in the next section.

4.3 Basic Idea: Temporal relaxation and Link-star rules

In this section we introduce the basic ideas behind our framework: (i) Temporal relaxation and (ii) Link-star rules. We first introduce the motivation behind these ideas with a discussion of the problems that they address. We then introduce the methods and parameters that are used to integrate these ideas into our proposed approach.

4.3.1 Temporal relaxation

Since the real world is complex and driven by convenience, events may or may not happen in an exact temporal order. In general, the temporal order of events is an important aspect of causality, as it helps us understand the cause-and-effect relationships between events. For example, if event A happens before event B, we may infer that event A caused event B. However, there are situations where events happening close to each other temporally may not follow any exact order, making it difficult to establish a clear cause-and-effect relationship between them.

One such situation is when multiple events happen simultaneously. In such cases, it is impossible to determine which event caused the other, as they are happening at the same time. For example, in a car accident involving multiple vehicles, it may be challenging to determine which vehicle caused the

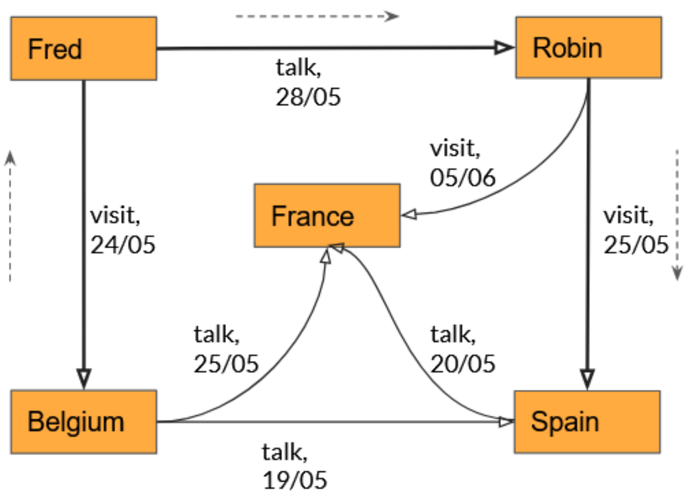

Figure 4.1: A relaxed temporal random walk on a TKG. The dotted lines indicate the direction of the walk.

accident, as multiple vehicles may have collided simultaneously. Additionally, some events may have a non-linear or cyclical relationship with one another, making it challenging to establish a clear temporal order. For example, the waves that occurred during the peak of the COVID-19 pandemic.

In summary, while the temporal order of events is an important aspect of causality, there are situations where events happening close to each other temporally may not follow any exact order. Multiple events happening simultaneously, events happening in a short span of time, and events with non-linear or cyclical relationships are some examples of such situations.

The temporal constraints enforced in the TLogic framework [30] state that there must be a strict temporal order between consecutive events/links which encompasses both the steps of rule generation and rule application. The following problems arise because of the aforementioned time constraints:

1. Not mining enough rules: During Rule Generation, in the step for performing a random walk, several significant links are being missed, which subsequently leads to the inability to mine a large number of useful logical rules.

2. Inability to find candidates during rule application: During rule application, in the step for the search for candidate entries, several candidates are being missed by TLogic.

4.3.1.1 Maximum Time Gap (δ)

Our proposed solution for both of these problems is to introduce temporal relaxation in the form of a parameter called ‘Maximum Time Gap’, denoted by δ. The idea behind this parameter is to allow for a small interval of time between two events which occur temporally close to each other to allow for the previously mentioned relaxation in the case of multiple events happening simultaneously, events happening in a short span of time, and events with non-linear or cyclical relationships. Equipped with this parameter, we give the formal definition for a temporally relaxed random walk.

Definition 3 ***Relaxed Temporal Random Walk:*** *A Relaxed Temporal Random Walk, denoted as W, is an ordered set of links such that the time gap between any two consecutive links is no more than the user-specified* maximum time gap *(δ). That is,* $W = \langle(e_1, r_1, e_2, t_1), (e_2, r_2, e_3, t_2) \ldots (e_l, r_l, e_{l+1}, t_{l+1})\rangle$, $l > 0$, $t_i \geq (t_{i-1} - \delta)$, *and* $i \in [1, l]$. *The length of the walk* W, *denoted as* $len(W)$, *represents the total number of links in* W. *That is,* $len(W) = |W|$, *where* $|W|$ *represents the number of links in* W.

Example 6 *Consider the following sequence of links in Fig. 4.1:* $\langle(Belgium, visit^{-1}, Fred, 24/05)$, $(Fred, talk, Robin, 28/05), (Robin, visit, Spain, 25/05)\rangle$. *The existing frameworks do not consider the above sequence of links as a random walk because the link* $(Fred, talk, Robin, 28/05)$ *violates the strict temporal constraint that it has to occur before the link* $(Robin, visit, Spain, 25/05)$. *However, if we relax the temporal constraint that the actual order of events can be ignored if their occurrence is within a particular time gap, say 4 days, i.e.,* $\delta = 4$, *we can consider the above sequence of links a relaxed temporal random walk. The length of this walk is 3 as it contains only three links.*

After having introduced temporal relaxation to random walks to fix the problem occurring during rule generation, we move to present a similar time-relaxed version of temporal rules which can solve the problems that arise during rule application.

Definition 4 ***Relaxed Temporal Rule:*** *Let* E_i *and* T_i *represent entity and timestamp variables, respectively. A Relaxed Temporal Rule of length* $n \in \mathbb{N}$ *is defined as* $((E_s, r_h, E_o, T_h) \leftarrow \wedge_i(E_i, r_i, E_{i+1}, T_i))$, *with the constraint* $T_i \geq T_{i-1} - \delta; i \in [1, n]$.

Example 7 *The following is an example of a Relaxed Temporal Rule extracted from the TKG in Fig. 4.1:* $(E_1, visit, E_3, T_3) \leftarrow (E_1, visit, E_2, T_1) \wedge (E_2, talk, E_3, T_2)$.

Having established Relaxed Temporal Random Walks and Relaxed Temporal Rules, we divide the walks and rules into two types, cyclic and acyclic, where each type has its own process of performing random walks and generating rules from the said walks.

Definition 5 ***Cyclic Relaxed Temporal Random Walk:*** *A relaxed temporal random walk,* W, *is said to be a Cyclic Relaxed Temporal Random Walk, denoted as* CW, *if the walk starts and end at the same entity (or node). That is,* $CW = \langle(e_1, r_1, e_2, t_1), (e_2, r_2, e_3, t_2) \ldots (e_n, r_n, e_1, t_n)\rangle$

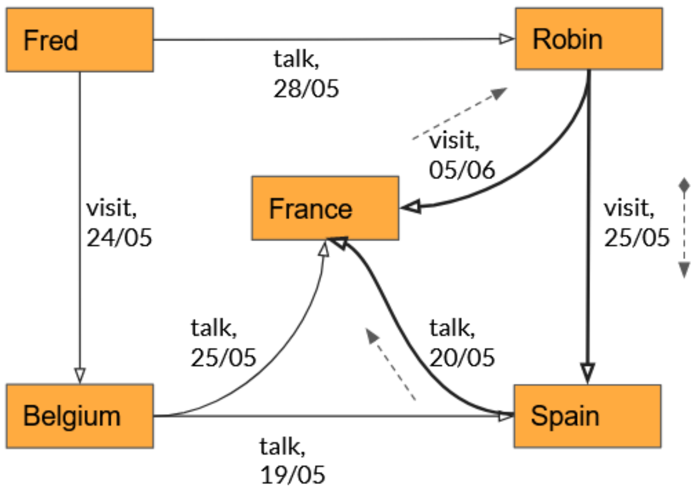

Figure 4.2: A cyclic relaxed temporal random walk on a TKG. The dotted lines indicate the direction of the walk.

Example 8 *Consider the following walk on the TKG in Fig. 4.2:*

$$\langle (Robin, visit, Spain, 25/05), (Spain, talk, France, 20/05), (France, visit^{-1}, Robin, 05/06)\rangle \tag{4.1}$$

This is a cyclic relaxed temporal random walk because it starts and ends with the same entity, i.e., Robin.

Definition 6 ***Relaxed Temporal Cyclic Rule:*** *A Relaxed Temporal Cyclic Rule is a Relaxed Temporal Rule generated from a cyclic walk by generalizing the entities and timestamps with variables. While the inverse of the last link becomes the rule head $(E_1, r_h^{-1}, E_n, T_n)$, the other links are mapped to body atoms, where each link (e_i, r_i, e_{i+1}, t_i) is converted to the body atom (E_i, r_i, E_{i+1}, T_i). That is, the final rule is of the form $(E_n, r_h^{-1}, E_1, T_n) \leftarrow \wedge_{i=1}^{l}(E_i, r_i, E_{i+1}, T_i)$*

The following example illustrates the process of converting a cyclic walk to a cyclic rule.

Example 9 *Let us consider the cyclic walk from Example 8:*

$$\langle (Robin, visit, Spain, 25/05), (Spain, talk, France, 20/05), (France, visit^{-1}, Robin, 05/06)\rangle \tag{4.2}$$

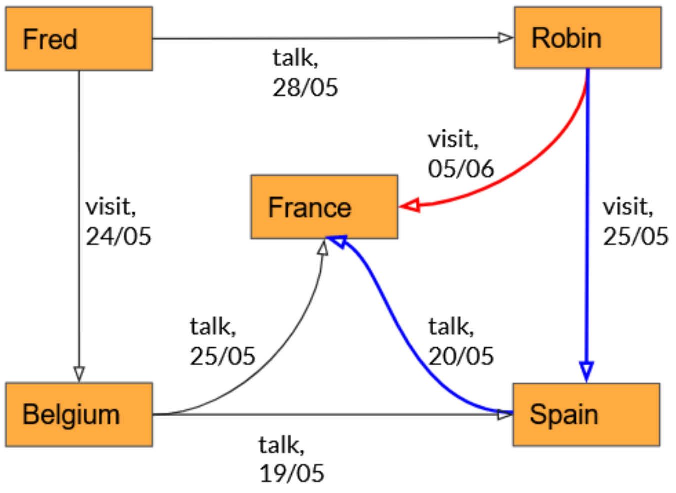

Figure 4.3: A cyclic relaxed temporal random walk on a TKG that can be converted to a rule. The red and blue links correspond to the head link and the body links, respectively.

Upon replacing the entities and timestamps with variables, we get the following:

$$\langle (E_1, visit, E_2, T_1), (E_2, talk, E_3, T_2), (E_3, visit^{-1}, E_1, T_3) \rangle \tag{4.3}$$

where E_1, E_2, E_3, T_1, T_2 and T_3 are generalized from 'Robin,' 'Spain,' 'France,' '25/05,' '20/05,' and '05/06' respectively. Then, as shown in Fig. 4.3 we consider the inverse of the last link as the rule head and present the final rule:

$$(E_1, visit, E_3, T_3) \leftarrow (E_1, visit, E_2, T_1) \wedge (E_2, talk, E_3, T_2) \tag{4.4}$$

4.3.2 Link-star rules

Acyclic rules are a common appearance in knowledge-based systems owing to their versatile nature. There are several real-world patterns that cannot be captured by cyclic rules alone. Cyclic rules can be understood as providing two alternate paths between any two entities, where one path is the rule head and the other path is the rule body. But this is just one of the many types of relationships that can exist between entities in a TKG. For instance, AnyBURL [33] is a rule-learning framework that takes advantage of acyclic rules to extract useful information from KGs. Their work suggests that acyclic rules are a vital and relevant part of link prediction on KGs.

However, with the onset of acyclic rules, comes the overhead of processing them, and often the most glaring problem in this regard is combinatorial rule explosion [44], which involves producing too many

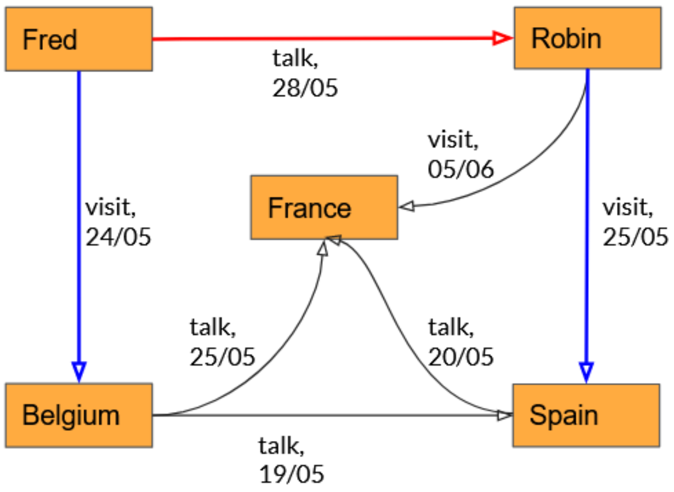

Figure 4.4: An acyclic relaxed temporal random walk on a TKG that can be converted to a rule. The red and blue links correspond to the head link and the body links, respectively.

rules. There is a computational and runtime cost associated with each performed walk, and it becomes intractable to control this cost when we begin mining acyclic rules. This also takes us back in time to the development of pruning, and data mining in general. As revolutionized by the seminal work of Agrawal and Srikant [2], pruning techniques are vital in case of combinatorial bottom-up rule building techniques to control combinatorial rule explosion. However, we can work our way around this problem by ignoring rules of length 2, while at the same time not ignoring them. We achieve this solution with the help of link-star (L-star) rules.

Before formalizing a definition for L-star rules, we define Acyclic walks, which are used to form the said rules.

Definition 7 ***Acyclic Relaxed Temporal Random Walk:*** *An Acyclic Relaxed Temporal Random Walk is a temporal random walk which does not contain any cycles (repeated entities) in the walk.*

Example 10 *As shown in Fig. 4.1, the following sequence of links constitutes an acyclic relaxed temporal random walk.*

$$\langle (Belgium, visit^{-1}, Fred, 24/05), (Fred, talk, Robin, 28/05), (Robin, visit, Spain, 25/05) \rangle$$

Definition 8 ***Link-star (L-star) Rule:*** *A Link-star (or L-star) Rule is a Relaxed Temporal Rule generated from an acyclic walk of length 3 by generalizing the entities and timestamps with variables. While*

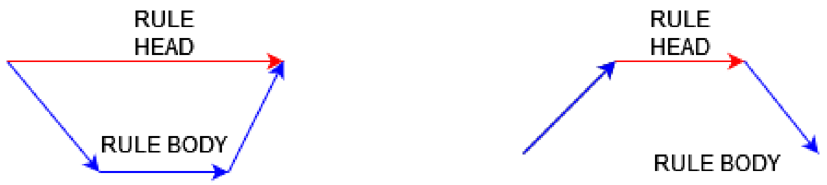

Figure 4.5: The direction of the rule head (RED) and the rule body (BLUE) in a cyclic rule vs. a link-star rule. The two different types of rules capture different types of patterns. It is important to note that sometimes in a rule, the links can be inverted.

the second link in the walk becomes the rule head (E_2, r_h, E_3, T_2)*, the other two links are mapped to body atoms, where each link* (e_i, r_i, e_{i+1}, t_i) *is converted to the body atom* (E_i, r_i, E_{i+1}, T_i)*. The final rule is of the form* $(E_1, r_h, E_2, T_1) \leftarrow (E_0, r_1, E_1, T_0) \wedge (E_2, r_3, E_3, T_2)$

Example 11 *Consider the following sequence of links in Fig. 4.1:*

$$\langle (Belgium, visit^{-1}, Fred, 24/05), (Fred, talk, Robin, 28/05), (Robin, visit, Spain, 25/05) \rangle$$

Upon replacing the entities and timestamps with variables, we get the following:

$$\langle (E_0, visit^{-1}, E_1, T_0), (E_1, talk, E_2, T_1), (E_2, visit, E_3, T_2) \rangle \tag{4.5}$$

Then, as shown in Fig. 4.4 we consider the middle link as the rule head and the rest of the links as the rule body and present the final rule as follows:

$$(E_1, talk, E_2, T_1) \leftarrow (E_0, visit^{-1}, E_1, T_0) \wedge (E_2, visit, E_3, T_2) \tag{4.6}$$

In graph theory, a star [29] represents a set of vertices where a vertex (called as core) is adjacent to all other vertices (called as leaves). Star patterns occur when a central entity has multiple edges connecting it to other entities, but those entities do not have any edges connecting them to each other. Thus the core shares some information with all the leaves. Analogous to that, in a link-star rule *we consider the rule head as a core vertex while its adjacent links form the rule body*. A Link-star rule is formed only from acyclic (relaxed temporal random) walks of length 3. Link-star rules are a simple version of star patterns found in graphs, where neighbourhood relations are considered to be sharing some information with the core. Fig. 4.5 illustrates this idea of the core being at the center of a link-star rule in a simplified manner.

Rule explosion occurs when a large number of rules are generated from the dataset. This is especially common when the dataset is large or when the minimum support and confidence thresholds used to filter the rules are low. The problem of rule explosion makes it difficult to identify the most interesting and useful patterns from the dataset, as there are simply too many rules to examine manually. This issue is amplified further in the case of link forecasting in TKGs because unlike cyclic rules, that have a constraint of being generated from a cyclic walk, acyclic rules are not bound by any constraint, and thus almost every search leads to the formation of a rule (sparing those rules which do not satisfy the *minsup* threshold.

Rule explosion can lead to a number of issues. First, it can make it difficult to identify the most important patterns from the dataset. With so many rules to examine, it can be challenging to identify the patterns that are most relevant to the user's needs. Second, rule explosion can lead to a high number of false positives. When there are too many rules to examine, it becomes more likely that some of the rules are not actually interesting or useful. Also, rule explosion can make it challenging to scale pattern mining algorithms to larger datasets, as the time and computational resources required to generate and filter the rules can become prohibitively large.

To tackle this challenge, a single link-star rule encompasses multiple acyclic rules. Consider a crude generalization of a link-star rule as $B \leftarrow A \wedge C$ where the alphabets represent distinct links. This rule inherently contains several rules. $B \leftarrow A$, $B \leftarrow C$, and $B \leftarrow A \wedge C$ can all be considered as subsets of this rule. In this way, we are able to prevent repetition of acyclic rules. Admittedly, we are losing some accuracy by not considering explicit rules of length 2, but the tradeoff gained in terms of saving time and computation far outweighs this loss.

We define a few more concepts that are necessary for specifying our framework as follows.

4.3.2.1 ACR and the number of searches

We define a hyperparameter which can be controlled by the user to allow for flexibility in the number of walks performed during rule generation. **Acyclic to Cyclic Ratio** or ACR gives the ratio of acyclic walks to cyclic walks that are to be performed during rule generation. The **number of searches** (denoted by 's') is the number of cyclic walks sampled inorder to create rules. This value is multiplied by ACR to compute the number of acyclic walks to be sampled during rule generation.

For example, if $s = 100$ and $ACR = 0.5$, then 50 acyclic walks and 100 cyclic walks will be performed during rule generation. Our experimentation indicates that increasing ACR results in some improved forecasting accuracy but it also increase the runtime by a noticeable amount. We discuss this in further detail in the experiments section.

4.3.2.2 Rule Confidence

Since all the generated rules are probabilistic, we need to specify a confidence measure to establish the likelihood of a rule being applicable to a given query. To evaluate the quality of the generated rules, we consider the conventional confidence measure. The confidence (denoted by *conf*) of a cyclic rule is the ratio of its *rule groundings* to *body groundings* in the TKG.

Definition 9 ***Body Grounding:*** *A body grounding for a rule is a tuple of entities and timestamps* $(e_1, t_1, e_2, t_2, \cdots, t_n, e_{n+1})$*, such that each body link* (e_i, r_i, e_{i+1}, t_i) *exists in the TKG.*

Definition 10 ***Rule Grounding:*** *For a cyclic rule, a body grounding is also a rule grounding if there are two entities in the grounding such that the link* $(e_1, r_h, e_{n+1}, t_{n+1}); t_n \leq t_{n+1} + \delta$*, also exists in the TKG.*

Example 12 *Let us compute the confidence of the rule given by equation 4.4. One possible body grounding for this rule is as follows:*

$$(Fred, 24/05, Belgium, 25/05, France) \tag{4.7}$$

However, there is no link of the form $(Fred, visit, France, t_x); t_x > 25/05$*. Thus it is not a rule grounding. Another valid body grounding for that rule is*

$$(Robin, 25/05, Spain, 20/05, France) \tag{4.8}$$

Further, this TKG contains the link $(Robin, visit, France, 05/06)$*, and* $05/06 > 25/05$*. Thus, this body grounding is also a rule grounding.*

Total rule groundings = 1

Total body groundings = 2

Hence, the confidence of this rule $= 1/2 = 0.5$

We do not use the same measure for L-star rules. Instead, we implicitly mine acyclic rules with high confidence. We achieve this by ensuring the presence of the head relation while performing acyclic walks. That is, while sampling acyclic walks, we begin with the head relation and then find the body links afterwards. This approach is different from cyclic walks where we perform walks for the rule body and match the rule head later. This saves a significant amount of time and computation resources, and also results in better forecasting accuracy.

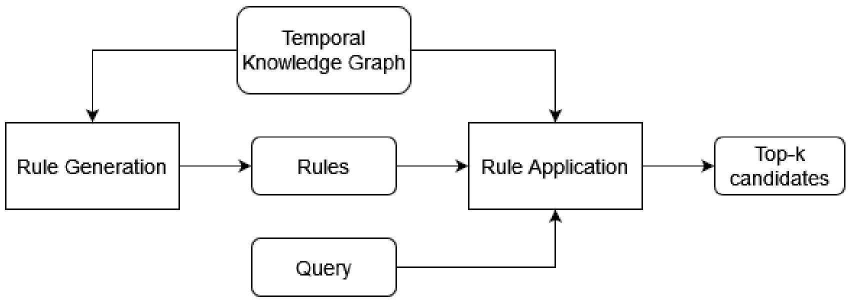

Figure 4.6: The basic flow of TRKG-Miner. The inward and outward arrows indicate Inputs and Outputs, respectively.

4.4 Proposed approach

In this section we present the algorithms which are central to our framework, having established the problem and the required definitions and notations. As discussed previously, Temporally Relaxed Knowledge Graph Miner (TRKG-Miner) consists of two broad steps, and each of these steps corresponds to one algorithm. The overall flow of our approach has been explained in Fig. 4.6. The first algorithm is for rule generation, which consists of performing random walks on the TKG as well as generating rules from them. This algorithm takes the TKG as an input and returns a set of human-understandable rules which are stored in the secondary memory. The second algorithm is for rule application, which takes a query and the generated rules as input and returns the top-k suitable candidates for the given query.

4.4.1 Rule Generation

The rule generation process is shown in Algorithm 1. First, we iterate over all the relations and perform both cyclic and acyclic random walks.

For cyclic walks, we perform the walk in a reverse temporal order, that is, starting from the last link and going backwards in time. That is essentially the same thing as performing a normal cyclic walk, but in this case we can ensure that we go through all the relations.

Example 13 *Consider the following walk on the TKG in Fig. 4.2:*

$$\langle (Robin, visit, Spain, 25/05), (Spain, talk, France, 20/05), (France, visit^{-1}, Robin, 05/06) \rangle \tag{4.9}$$

In TRKG-Miner, we sample this walk in the following order:

Algorithm 1 Rule Generation

```
Input: A temporal knowledge graph
Parameters: Number of searches $s \in \mathbb{N}$, $ACR$, $\delta$
Output: Set of Relaxed Temporal rules $RT_R$
1: for links with relation $r \in \mathcal{R}$ do
2:     for i ∈ $[s]$ do                    ▷ Repeat with [$ACR * s$] for L-star rules
3:         Sample a walk $w$
4:         Create rule $rt$ from $w$
5:         Compute rule confidence $conf(rt)$
6:         $RT_{\mathcal{R}} \leftarrow \cup(rt_r, conf(rt))$
7:     end for
8: end for
```

$$\langle (Robin, visit, France, 05/06), (France, talk^{-1}, Spain, 20/05), (Spain, visit^{-1}, Robin, 25/05) \rangle \tag{4.10}$$

For acyclic walks, we take a similar approach (of beginning with the relation in the rule head) and then find the rest of the body links. Taking this specific path has an added advantage in the case of L-star rules. Since the rule head occurs between the two body links, by beginning with it, we are ensuring that the rules we mine are of high confidence by themselves. In a way, this is analogous to both frequent patterns and association rules, where we get the best of both worlds. The naïve way to perform an acyclic walk would entail starting with the first link, and then finding the rule head, in which case we cannot ensure that all the relations are considered for the rule head. Thus, that approach is discarded in favour of our approach which implicitly mines rules with high confidence.

Example 14 *Assume that we wish to find a rule where the head relation is 'talk'. Consider the following sequence of links in Fig. 4.1:*

$$\langle (Belgium, visit^{-1}, Fred, 24/05), (Fred, talk, Robin, 28/05), (Robin, visit, Spain, 25/05) \rangle \tag{4.11}$$

In this case, we begin by sampling the link (Fred, talk, Robin, 28/05) first, and then sample the other two links.

During walk sampling, there can be multiple possible next links. In this case, the priority is given to the links which have a timestamp closer to the head link. This distribution is a weighted normalized probability distribution given by Equation 4.12. (t_u is the current timestamp and $T_c = \{t_{c_1}, ...t_{c_i}, ..., t_{c_s}\}$ is a set of timestamps of the candidates)

$$\mathbb{P}(t_c; T_c, t_u) = \frac{exp(-|t_u - t_c|)}{\sum_{T_c} exp(-|t_u - t_{c_i}|)} \tag{4.12}$$

As explained in the previous section, for each walk we perform a generalization (opposite of grounding) and replace the entities and timestamps with variables to obtain rules. The final step is to compute rule confidence (Subsection 4.3.2.2) following which all the rules are stored for later use.

4.4.2 Rule Application

Given a query of the form $(e_{query}, r_{query}, ?, t_{query})$, and a set of rules generated from the previous algorithm, the goal is to find a ranked list of top-k candidates to replace '?'. The basic intuition behind this algorithm is that past events and patterns tend to repeat themselves, and they do so quite frequently. This can be leveraged to get an accurate forecast for a future event.

Algorithm 2 Rule Application

```
Input: $RT_{\mathcal{R}}$, Query $(e_{query}, r_{query}, ?, t_{query})$
Parameters: No. of candidates required ($k$), minsup
Output: Answer candidates $A$
1: $RT_R \leftarrow \{rt | support(rt) > minsup\}$
2: for each $rt$ with relation $r_{query}$ do
3:     Find all body groundings for $rt$
4:     for each body grounding do
5:         Get candidate $a$ and candidate timestamp $t_a$
6:         $score(a) \leftarrow \alpha(exp(-\lambda(t_{query} - t_a))) + (1 - \alpha)conf(rt)$
7:     end for
8: end for
9: Return top-$k$ candidates ordered by score
```

First, all the rules are pruned according to a chosen minimum support (*minsup*) and ranked in decreasing order of rule confidence. Next, the rules are filtered to use only those whose head relation is r_{query}.

Next, body groundings are determined for each rule. For cyclic rules, body groundings are found for each rule by performing a relaxed temporal walk starting from e_{query}. Then, from each grounding of the form $(e_{query}, t_0, ..., t_i, e_i, ..., t_n, e_n)$, the last entity e_n and timestamp t_n are selected as a candidate and its candidate timestamp.

In the case of L-star rules, this body grounding is determined by starting from e_{query} and finding a pre-link (link connected to the subject entity in the rule head). After this, a post-link (link connected to the object entity in the rule head) is determined by matching link which are 1 link apart from e_{query}. In this way, multiple candidates are found. From each grounding of the form $(e_0, t_0, e_{query}, t_1, e_2, t_2, e_3)$, e_2 represents a candidate and t_1 represents its candidate timestamp.

To rank the different candidates that are obtained, a score is calculated for each of them. Given a candidate a with a candidate timestamp t_a extracted from a rule rt, the candidate score, denoted by $score(a)$ is calculated as follows:

$$score(a) = \alpha(exp(-\lambda(t_{query} - t_a))) + (1 - \alpha)conf(rt) \tag{4.13}$$

λ and α are experimentally-determined parameters. The reason for considering this scoring function are as follows:

1. Recent events are more relevant to the query than older events
2. The candidates extracted using rules with a better rule quality metric should receive a higher score

Cyclic rules get lesser priority over acyclic rules because the search criteria for generating cyclic walks is based on loops *of a short length like 2-3* which does not work well in sparse segments of the TKG. This is not the case with L-star rules, since they do not have any such criteria and they are acyclic by nature. Finally, the top-k candidates ordered by score are returned.

Having explained our basic idea and its implementation in the form of a framework, we move on to define the datasets and the experimental setup, and then we present our findings.

4.5 Experiments

In this section we discuss the experimental setup, error metrics, and the datasets used for the experiments.

4.5.1 Datasets

The three ICEWS [55] datasets - ICEWS14, ICEWS18, and ICEWS0515 - are collections of world event data that contain event information from the years 2014, 2018, and 2005-15, respectively. The events in these datasets include various types of political activity such as protests, negotiations, violent incidents, and state repression, among others. This data is present in the datasets in the form of several thousand tuples of entity-relation-entity-timestamp (see Table 4.3, that can be directly used as links to build a temporal knowledge graph. Using the ICEWS datasets for link forecasting creates an important real-world tool to identify potential significant events that may occur in the future.

The ICEWS datasets are created by a team of researchers who use automated methods to collect and preprocess data from various sources such as news wires, online news outlets, local newspapers, social media. Here's a brief discussion of data collection and preprocessing for the ICEWS datasets:

4.5.2 Data Collection

The ICEWS data collection process starts with the collection of raw news articles from a wide range of sources, including news wires, local newspapers, and online news outlets. The articles are collected in real-time and stored in a centralized database. A team of trained analysts then reads and codes the articles using the CAMEO coding scheme, which categorizes each event according to its type, actors involved, and other relevant information. The coded data is then stored in a structured format that can be easily analyzed and processed.

We split the data into train, test, and validation sets, where the test set contains timestamps which are relatively in the future from the training and validation sets. The distribution can be found in Table 4.2. Notably, we can see that the test and valid sets contain a good fraction of new unseen entities from the train sets. More importantly, all the timestamps are new and unseen from the train sets. For consistency and reproducibility, we use the same train-test-validation split as xERTE [18].

Table 4.1: Statistics of the used datasets

–	ICEWS14	ICEWS18	ICEWS0515
Links	90730	468558	461329
Entities	7128	23033	10488
Relations	222	256	251
Timestamps	262	304	4017

4.5.3 Experimental setup

We run our experiments on a multicore server with 128 cores, with each core having a maximum clock speed of 2.9 GHz. Since the number of processes can be passed as a parameter while running the framework, more available cores means easy parallelization and subsequently a faster run-time.

Table 4.2: train-valid-test distribution for the experiments. '(N)' indicates new.

	ICEWS14			ICEWS18			ICEWS0515		
–	train	valid	test	train	valid	test	train	valid	test
Links	63685	13823	13222	373018	45995	49545	322958	69224	69147
Links (N)	-	964	865	-	1411	1955	-	3241	8559
Ents.	6180	2968	2845	21085	7730	8243	8853	4851	4751
Ents. (N)	-	530	496	-	910	1140	-	939	1089
Rels.	222	164	171	251	204	213	248	222	217
Rels. (N)	-	5	4	-	3	3	-	2	2
TS	262	52	51	240	30	34	2775	679	563
TS (N)	-	52	51	-	30	34	-	679	563

Table 4.3: Sample Links from ICEWS14

Arsen Avakov, *make statement*, Military (Ukraine), 2014-06-17
Muhammadu Buhari, *make an appeal or request*, Citizen (Nigeria), 2015-01-23
Thailand, *host a visit*, Japan, 2015-01-23
Malaysia, *sign formal agreement*, Japan, 2015-10-07

Experiments on ICEWS14 can be performed on a regular personal computer as well, but the other two datasets need significantly more computing power and time. Especially, since almost every sampled acyclic walk is converted to a rule (as compared to cyclic walks, which depend on the walk being a cycle), the time needed for rule generation is greatly influenced by ACR.

We use hits@k (k={1,3,10}) and Mean Reciprocal Rank as performance metrics. The hits@k metrics measures the fraction of times that the correct entity is present among the top-k returned candidates. For example, $hits@1$ measures the fraction of times when the top ranked candidate is the correct entity for a given query. MRR is defined as the average of all reciprocal ranks of the correct query answers across all queries, where reciprocal rank is $1/x$ for a rank x. MRR can also be considered as the weighted average of hits@1 (h@1), hits@3 (h@3), and hits@10 (h@10).

The primary motivation behind choosing these experiments is to establish that the proposed framework is efficient and works as described throughout the manuscript. Secondly, we our model is indeed superior to other approaches, and thus can be regarded as a positive contribution to the field. Thridly, we do an ablation study/hyperparameter analysis to show that our introduced variables/concepts are indeed impactful and not just random additions to the framework.

During experimentation, we are mainly concerned with the dynamics of the following: ACR, minsup, and δ. Thus, we test the influence of each parameter while keeping the other two constant. We also observe the number of generated rules and analyze them across the experiments. In the first and central experiment, we compare the performance of the proposed framework, TRKG-Miner, against eleven existing frameworks (DistMult [58], ComplEx [52], AnyBURL [33], TTransE [27], TA-DistMult [14], DE-SimplE [15], TNTComplEx [26], CyGNet [59], RE-Net [25], xERTE [18], and TLogic [30]) on the three datasets. After that, we study the performance of the framework on varying the parameters ACR, δ, and $minsup$. We are limited by the fact that some of these experiments take several hours, especially in the case of ICEWS18 and ICEWS0515, where this can transition into days. Thus, we had to modify a few parameters to obtain results in a reasonable time and complete our experimentation. We present an analysis of the observed error metrics as well as the runtime for each experiment in the next section.

4.6 Results and Discussion

In this section we present the results of our experiments accompanied by a discussion of the same.

4.6.1 Performance comparison with other frameworks

Table 4.4 presents the performance results of TRKG-Miner against the existing link forecasting frameworks. It can be observed that TRKG-Miner outperforms all of its competitors on all of the evaluated datasets. Notably, the results of TRKG-Miner for *hits@1* metrics improve by approximately 45% on average, which means the accuracy of finding the correct result from the first ranked candidate increases. This helps in making accurate predictions in one shot without having to rely upon multiple candidate predictions. There is also a significant improvement in *hits@3*. Consequently, the *MRR* is improved by about 23%. This momentous improvement in forecasting accuracy can be attributed to the introduction of temporal relaxation and Link-star rules.

Comparing *hits@1* across the three datasets shows that the performance on ICEWS18 achieves the most significant improvement over previous approaches. More importantly, we can observe that the improvement in the results is not affected by scaling the knowledge graph, since both ICEWS18 and ICEWS0515 achieve comparable or even a higher percentage of improvement over ICEWS14, even though ICEWS18 has about three times the number of entities and ICEWS0515 has ten times the number of timestamps over ICEWS14. Having more training data certainly helps, since more and more new rules can be generated and used for link forecasting.

Dataset	**ICEWS14**				**ICEWS18**				**ICEWS0515**			
Model	MRR	h@1	h@3	h@10	MRR	h@1	h@3	h@10	MRR	h@1	h@3	h@10
DistMult	0.2767	0.1816	0.3115	0.4696	0.1017	0.0452	0.1033	0.2125	0.2873	0.1933	0.3219	0.4754
ComplEx	0.3084	0.2151	0.3448	0.4958	0.2101	0.1187	0.2347	0.3987	0.3169	0.2144	0.3574	0.5204
AnyBURL	0.2967	0.2126	0.3333	0.4673	0.2277	0.1510	0.2544	0.3891	0.3205	0.2372	0.3545	0.5046
TTransE	0.1343	0.0311	0.1732	0.3455	0.0831	0.0192	0.0856	0.2189	0.1571	0.0500	0.1972	0.3802
TA-DistMult	0.2647	0.1709	0.3022	0.4541	0.1675	0.0861	0.1841	0.3359	0.2431	0.1458	0.2792	0.4421
DE-SimplE	0.3267	0.2443	0.3569	0.4911	0.1930	0.1153	0.2186	0.3480	0.3502	0.2591	0.3899	0.5275
TNTComplEx	0.3212	0.2335	0.3603	0.4913	0.2123	0.1328	0.2402	0.3691	0.2754	0.1952	0.3080	0.4286
CyGNet	0.3273	0.2369	0.3631	0.5067	0.2493	0.1590	0.2828	0.4261	0.3497	0.2567	0.3909	0.5294
RE-Net	0.3828	0.2868	0.4134	0.5452	0.2881	0.1905	0.3244	0.4751	0.4297	0.3126	0.4685	0.6347
xERTE	0.4079	0.3270	0.4567	0.5730	0.2931	*0.2103*	0.3351	0.4648	0.4662	*0.3784*	0.5231	0.6392
TLogic	*0.4304*	*0.3356*	*0.4827*	*0.6123*	*0.2982*	0.2054	*0.3395*	*0.4853*	*0.4697*	0.3621	*0.5313*	*0.6743*
TRKG-Miner	**0.5028**	**0.4514**	**0.5189**	**0.6130**	**0.3796**	**0.3262**	**0.3905**	**0.4952**	**0.5856**	**0.5360**	**0.6028**	**0.6912**

Table 4.4: Results of link forecasting using various approaches. The best results in each category are highlighted in bold while the second best ones are italicised. Parameters: $\delta = 1$, $ACR = 3$ for ICEWS14, $ACR = 1$ for ICEWS18 and ICEWS0515

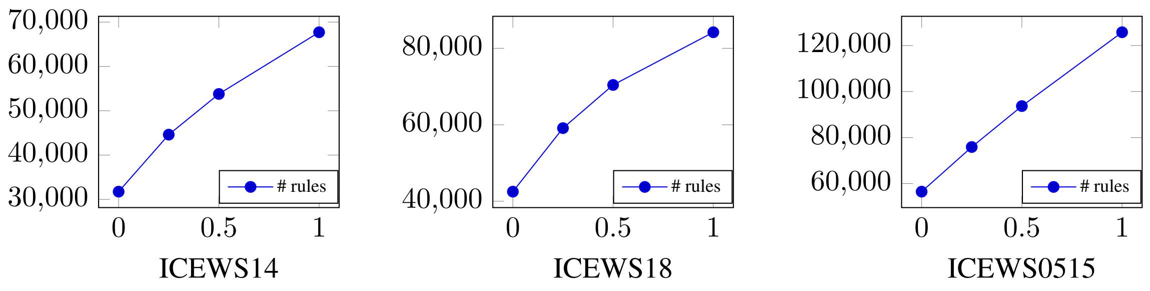

Figure 4.7: A plot of the number of rules generated with a variation in ACR

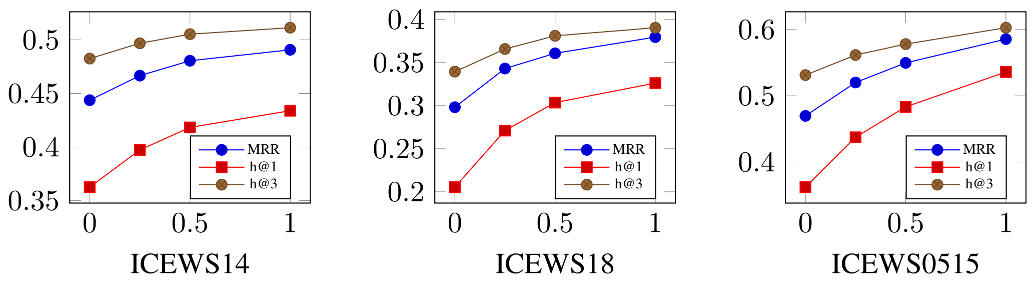

Figure 4.8: A plot of the error metrics with a variation in ACR

4.6.2 Varying ACR

Fig. 4.7 records the variation in the number of rules used for rule application on varying ACR during rule generation, and Fig. 4.8 records the change in the error metrics when varying the same. We find that increasing the fraction of L-star rules leads to an increase in prediction accuracy. The steepest increase happens initially when acyclic rules are introduced, and the slope gradually decreases. It is important to note that meanwhile the number of rules is also increasing, and with it comes added time for both generating and applying each rule. Another important thing to keep in mind is that during the search, almost every acyclic walk is converted to an L-star rule, and thus this increase in run time is guaranteed, unlike cyclic rules where the formation of a rule depends on the sampled walk being a cycle. Thus, we cannot leave this parameter unchecked, because for a minor gain in accuracy we might have to tradeoff a significant amount of run time. Thus, this parameter should be carefully decided based on the time and computing power available.

Another important observation is that when $ACR = 0$, only cyclic rules are mined. This demonstrates how our approach generates a significant number of new useful rules as compared to previous approaches. This goes against the conventional idea that often rule mining algorithms mine a lot of unnecessary rules as a result of which they need pruning. We believe that that is the case only when rule searches are brute force, such as combinatorial apriori search. Integrating contextual and structural knowledge while mining patterns can result in finding a much better set of patterns from the get go. We sincerely hope that the reader takes note of this point, and we believe that this is the core essence of this entire work.

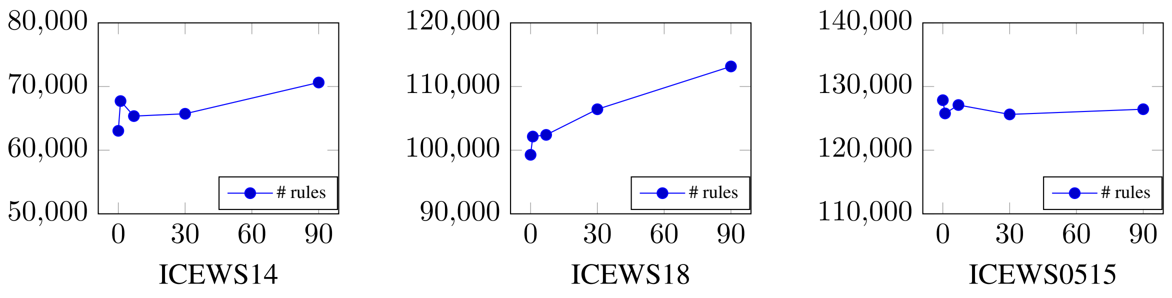

Figure 4.9: A plot of the number of rules generated with a variation in maximum time gap

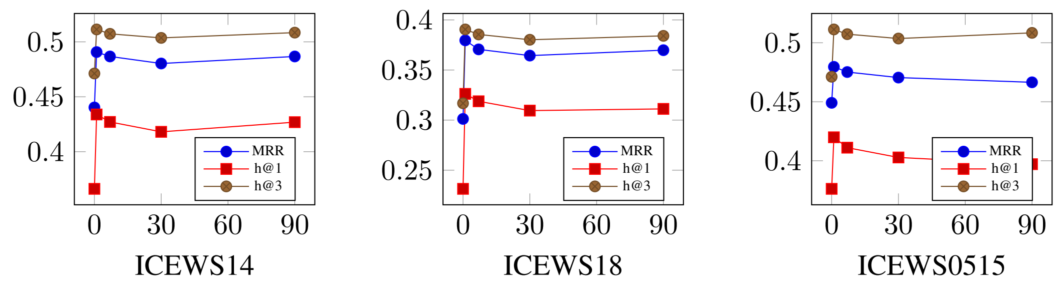

Figure 4.10: A plot of the error metrics with a variation in maximum time gap

4.6.3 Varying Maximum Time Gap

Fig. 4.9 records the variation in the number of rules used for rule application on varying *maximum time gap* (δ), and Fig. 4.10 records the change in the error metrics when varying the same. With an increase in δ, the number of generated rules increases. There is also a significant increase in forecasting efficiency when changing δ from 0 to 1, which indicates the usefulness of introducing temporal relaxation. Interestingly, the forecasting accuracy seems to deteriorate afterwards. This experiment validates our hypothesis, that mutually exclusive events happening close to each other need not follow any strict order. However, we also find that this matters only when the events remain within an acceptable time gap. Once we go beyond a certain limit, the number of patterns increases while MRR decreases.

Unlike the last experiment, we we can observe that the increase in the number of rules as compared to increase in δ is nominal. This is consistent with the idea that the dataset has an even distribution of facts over the temporal dimension. Another important observation is that initially, an increase in δ is not followed by an increase in the number of rules. This makes sense since this parameter primarily affects the walks, which means that walks are found which range over a larger span of time, and thus they do not find the minimum support to qualify for being converted to a rules.

The change in the error metrics initially follows the number of rules, but as explained in the previous paragraph, that behaviour in anomalous, and in general an increasing δ is associated with a decrease in forecasting accuracy. In fact, δ can be considered as a handy tool which lets us to arrive at a midpoint between the two approaches of either being too strict temporally (e.g. TLogic), or completely disregarding temporal dynamics (e.g. static KG completion approaches). This, in our opinion, is the second most important takeaway of this thesis, which is (also consistent with the real-world idea) that often a compromise between two extreme views helps explain a concept better than either of them.

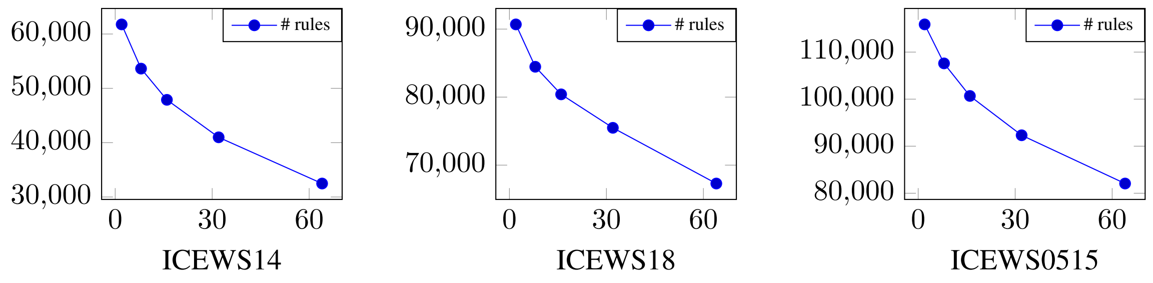

Figure 4.11: A plot of the number of rules used for rule application with a variation in minimum support

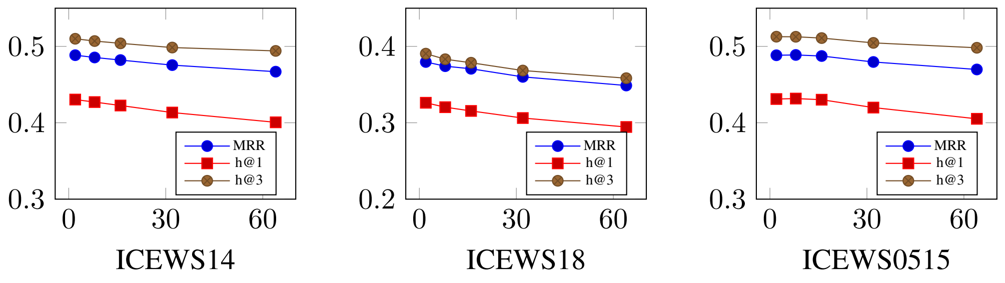

Figure 4.12: A plot of the error metrics with a variation in the minimum support

4.6.4 Varying *minsup*

Fig. 4.11 records the variation in the number of rules used for rule application on varying the minimum support parameter, and Fig. 4.12 records the change in the error metrics when varying the minimum support. Contrary to conventional beliefs, we observe that an increase in *minsup* almost always leads to a decrease in the forecasting accuracy.

In conventional data mining tasks, having a minsup as low as 2 can lead to an explosion in the number of spurious rules. However, in the case of link forecasting using our approach, this does not seem to be the case. Even a minor increase in *minsup* leads to a decrease in model performance.

Although we acknowledge that the runtime is quite large, it is acceptable for experimentation because some parameters have been tweaked inorder to complete all the experiments in a reasonable amount of time. Notably, this behavior is consistent with TLogic, where an increase in *minsup* led to a decrease in the forecasting accuracy [30]. Subsequently, with increasing minimum support the performance gets worse. Our final experiment reinforces an important overall takeaway from this work, which is that conventional ideas need not always apply to every problem, and often seemingly unintuitive paths lead to surprising discoveries.

4.6.5 Analysis of the generated rules

Table 4.5 shows a few of the rules generated by using TRKG-Miner on ICEWS14. The first three rules are acyclic while the other three are cyclic rules. We can observe the existence of loops in the cyclic rules (apart from the main loop formed by the rule head and the rule body). We can also observe that the rule length varies among the cyclic rules only, and all the acyclic rules are of length 3. We can also extract some inferences from the mined rules, which indicate the closeness between different entities, which is quite consistent with the events happening in the real-world.

Table 4.5: Sample Rules generated from ICEWS14

Cooperate militarily $(X_1, X_2, T_1) \leftarrow$ Use conventional military force $(X_0, X_1, T_0) \wedge$ Provide aid(X_2, X_3, T_2)
Cooperate economically $(X_1, X_2, T_1) \leftarrow$ Impose embargo, boycott, or sanctions $(X_0, X_1, T_0) \wedge$ Make a visit (X_2, X_3, T_2)
Defy norms, law $(X_1, X_2, T_1) \leftarrow$ Demand $(X_0, X_1, T_0) \wedge$ Accuse (X_2, X_3, T_2)
Accede to demands for change in policy $(X_0, X_1, T_3) \leftarrow$ Retreat or surrender militarily $(X_0, X_1, T_0) \wedge$ Make statement $(X_1, X_2, T_1) \wedge$ Accuse (X_2, X_1, T_2)
Accuse $(X_0, X_2, T_3) \leftarrow$ Criticize or denounce $(X_0, X_1, T_0) \wedge$ Criticize or denounce $(X_1, X_0, T_1) \wedge$ Accuse (X_0, X_2, T_2)
Mediate $(X_0, X_1, T_1) \leftarrow$ Host a visit (X_0, X_1, T_0)

4.7 Summary

In this chapter we introduce TRKG-Miner, an improved rule-based link forecasting framework. TRKG-Miner improves the forecasting efficiency by finding additional useful rules over previous approaches. This is achieved because of two main improvements: (i) To better capture the irregular occurrences of events within a time gap, we added time relaxation in the form of *maximum time gap* (ii) Based on the principle that past events repeat themselves, we introduced and developed an algorithm to generate Link-star rules, a special type of acyclic rules. Our experimentation indicates that we achieve significantly better results than the past approaches. The next chapter concludes our thesis by summarising our key contributions. We then acknowledge any potential limitations of the experiments and suggests future research directions.

Chapter 5

Conclusion

Temporal Knowledge Graphs find a wide range of applications in various fields. One of their most significant applications is predicting future events. By analyzing the patterns and trends in temporal data, temporal knowledge graphs can provide insights into what may happen in the future. TKGs are used for real-world applications such as event-forecasting, medical data analysis, traffic congestion analysis, and social network analysis. Formally known as Link forecasting, the problem of predicting future events can be formulated as finding missing entities in a given query. This problem of link forecasting has led to joint efforts from various domains, such as machine learning and graph theory.

In this thesis, we have carried out a discussion on approaching the problem of link forecasting in temporal knowledge graphs. Notably, we identify and tackle the following research gaps: First, we observe that the number of rules generated by previous frameworks is too less because of strictly imposed temporal constraints on both rule generation and rule application. To mitigate this we propose temporal relaxation in the form of a parameter called the *maximum time gap*. Secondly, there is no provision for finding acyclic rules, which often play a major role alongside cyclic rules in many applications. For this we introduce Link-star rules, a special class of acyclic rules and use them for link forecasting. The experimental results indicate that our framework is a substantial improvement over the past approaches.

We conclude that rule-based approaches are one of the best paradigms for link forecasting in TKGs. From our discussion throughout this thesis, we can conclude that temporal relaxation and the use of cyclic and acyclic rules positively impact link forecasting frameworks, and there is much scope for real-world applications of rule-based systems in the future.

5.1 Limitations

While our study provides valuable insights into the topic at hand, several factors should be considered when interpreting our findings. It is crucial to address these limitations to ensure our conclusions are not overgeneralized and to provide direction for future research. The following are two notable limitations of our approach.

- Although our framework can handle large amounts of data, this comes at the cost of using a large amount of resources in the form of high memory usage and high computation time.
- We are not using any entity-related contextual information, for example, the fact that USA is a country, and typically it will not have links shared with, say, a farmer in Vietnam. These ideas are captured only in an implicit but inefficient format via the means of temporal rules.

Although beyond the scope of this thesis, we present a few future directions for possible improvements and extensions in the next section.

5.2 Future Work

Future extensions of our work can include the following:

- Frameworks that can perform link forecasting on datasets with heterogeneous timestamps. These datasets contain timestamps where time gaps are a mix of seconds, minutes, days, or even months, unlike the current work where the time gap is consistent, that is, one day.
- A framework could also be developed for handling time intervals (and not just timestamps), which would mean dealing with a different type of data and developing rule generation and rule application algorithms for it.
- There is scope for adding more complex variations of cyclic and acyclic rules to make a better link forecasting framework.

Related Publications

1. Rage, U.K.*, **Maharana, A.***[1], Polepalli, K.R. (2023). A Novel Explainable Link Forecasting Framework for Temporal Knowledge Graphs Using Time-Relaxed Cyclic and Acyclic Rules. In: Kashima, H., Ide, T., Peng, WC. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2023. Lecture Notes in Computer Science, Vol. 13935. Springer, Cham.

[1]Both authors contributed equally to the work

Bibliography

[1] B. Abu-Salih. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications*, 185:103076, 2021. 1, 3, 5, 9

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th International Conference on Very Large Databases, VLDB*, volume 1215, pages 487–499. Santiago, Chile, 1994. 27

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pages 722–735. Springer, 2007. 3

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001. 1, 8

[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008. 3

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, 26, 2013. 11

[7] B. Cai, Y. Xiang, L. Gao, H. Zhang, Y. Li, and J. Li. Temporal knowledge graph completion: A survey. *arXiv preprint arXiv:2201.08236*, 2022. 5, 9, 10

[8] N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer. Introduction to neural network based approaches for question answering over knowledge graphs. *arXiv preprint arXiv:1907.09361*, 2019. 5, 9

[9] J. Chen, H. He, F. Wu, and J. Wang. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6271–6278, 2021. 17

[10] Z. Cui, P. Kapanipathi, K. Talamadupula, T. Gao, and Q. Ji. Type-augmented relation prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7151–7159, 2021. 10

[11] N. N. Daud, S. H. Ab Hamid, M. Saadoon, F. Sahran, and N. B. Anuar. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications*, 166:102716, 2020. 5

[12] 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, pages 601–610, 2014. 9

[13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422, 2013. 13

[14] A. García-Durán, S. Dumančić, and M. Niepert. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202*, 2018. 5, 12, 37

[15] R. Goel, S. M. Kazemi, M. Brubaker, and P. Poupart. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3988–3995, 2020. 5, 9, 37

[16] T. Gracious, S. Gupta, A. Kanthali, R. M. Castro, and A. Dukkipati. Neural latent space model for dynamic networks and temporal knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4054–4062, 2021. 4

[17] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568, 2020. 5, 9

[18] Z. Han, P. Chen, Y. Ma, and V. Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*, 2020. 5, 12, 35, 37

[19] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1983. 18

[20] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. d. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4):1–37, 2021. 2, 3, 10

[21] C. Hong, M. Tan, S. Wang, J. Wang, M. Li, and J. Qiu. Small and medium-sized enterprises credit risk assessment based on temporal knowledge graphs. In *2021 IEEE 20th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 166–172. IEEE, 2021. 4

[22] A. Horn. On sentences which are true of direct unions of algebras1. *The Journal of Symbolic Logic*, 16(1):14–21, 1951. 17

[23] M. Y. Jaradeh, K. Singh, M. Stocker, and S. Auer. Triple classification for scholarly knowledge graph completion. In *Proceedings of the 11th on Knowledge Capture Conference*, pages 225–232, 2021. 10

[24] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021. 9

[25] W. Jin, M. Qu, X. Jin, and X. Ren. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. *arXiv preprint arXiv:1904.05530*, 2019. 5, 12, 37

[26] T. Lacroix, N. Usunier, and G. Obozinski. Canonical tensor decomposition for knowledge base completion. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2863–2872. PMLR, 10–15 Jul 2018. 5, 12, 37

[27] J. Leblay and M. W. Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776, 2018. 5, 11, 37

[28] Q. Li, D. Wang, S. Feng, C. Niu, and Y. Zhang. Global graph attention embedding network for relation prediction in knowledge graphs. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6712–6725, 2021. 10

[29] J. Liu, M. Zhou, P. Fournier-Viger, M. Yang, L. Pan, and M. Nouioua. Discovering representative attribute-stars via minimum description length. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 68–80. IEEE, 2022. 28

[30] Y. Liu, Y. Ma, M. Hildebrandt, M. Joblin, and V. Tresp. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 5, 13, 21, 23, 37, 42

[31] F. MacLean. Knowledge graphs and their applications in drug discovery. *Expert opinion on drug discovery*, 16(9):1057–1069, 2021. 5

[32] D. L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004. 1, 8

[33] C. Meilicke, M. W. Chekol, M. Fink, and H. Stuckenschmidt. Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv preprint arXiv:2004.04412*, 2020. 17, 26, 37

[34] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, and H. Stuckenschmidt. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International Semantic Web Conference*, pages 3–20. Springer, 2018. 13

[35] H. Mezni. Temporal knowledge graph embedding for effective service recommendation. *IEEE Transactions on Services Computing*, 15(5):3077–3088, 2021. 4

[36] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 3

[37] S. K. Mohamed, V. Novácek, P.-Y. Vandenbussche, and E. Muñoz. Loss functions in knowledge graph embedding models. *DL4KG@ ESWC*, 2377:1–10, 2019. 5

[38] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–35, 2021. 10

[39] A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem solving program. In *IFIP Congress*, volume 256, page 64. Pittsburgh, PA, 1959. 1, 8

[40] T. H. Nguyen and R. Grishman. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, 2015. 5, 9

[41] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011. 11

[42] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor. Industry-scale knowledge graphs: Lessons and challenges: Five diverse technology companies show how it's done. *Queue*, 17(2):48–75, 2019. x, 1, 2, 3

[43] L. Ora. Resource description framework (rdf) model and syntax specification. *http://www. w3. org/TR/REC-rdf-syntax/*, 1999. 1, 8

[44] L. I. Perlovsky. Conundrum of combinatorial complexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):666–670, 1998. 26

[45] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II 15*, pages 177–185. Springer, 2016. 3

[46] R. H. Richens. General program for mechanical translation between any two languages via an algebraic interlingua. *Report on research: Cambridge Language Research Unit, Mechanical Translation*, 3, 1956. 1, 8

[47] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32, 2019. 17

[48] T. Shen, F. Zhang, and J. Cheng. A comprehensive overview of knowledge graph completion. *Knowledge-Based Systems*, page 109597, 2022. 10

[49] Y. Shen, K. Yuan, J. Dai, B. Tang, M. Yang, and K. Lei. Kgdds: a system for drug-drug similarity measure in therapeutic substitution based on knowledge graph curation. *Journal of Medical Systems*, 43:1–9, 2019. 9

[50] A. Singhal et al. Introducing the knowledge graph: things, not strings. *Official Google blog*, 5(16):3, 2012. 1

[51] F. N. Stokman and P. H. de Vries. *Structuring knowledge in a graph*. Rijksuniversiteit Groningen, 1986. 1, 8

[52] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080. PMLR, 2016. 10, 37

[53] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. 3

[54] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, 2014. 5

[55] M. D. Ward, A. Beger, J. Cutler, M. Dickenson, C. Dorff, and B. Radford. Comparing gdelt and icews event data. *Analysis*, 21(1):267–297, 2013. 34

[56] Z. Wu, Y. Song, and C. Giles. Exploring multiple feature spaces for novel entity discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. 5, 9

[57] R. Xie, Z. Liu, M. Sun, et al. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, volume 2016, pages 2965–2971, 2016. 10

[58] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2015. 11, 37

[59] C. Zhu, M. Chen, C. Fan, G. Cheng, and Y. Zhang. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4732–4740, 2021. 5, 12, 37

[60] G. Zhu and C. A. Iglesias. Sematch: Semantic similarity framework for knowledge graphs. *Knowledge-Based Systems*, 130:30–32, 2017. 5, 9