Machine Learning with Knowledge Graphs for Explainable Artificial Intelligence

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universität München

Yushan Liu

München, 2023

Machine Learning with Knowledge Graphs for Explainable Artificial Intelligence

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik der Ludwig-Maximilians-Universität München

vorgelegt von

Yushan Liu

aus Beijing, China

München, den 18.12.2023

Erstgutachter:	Prof. Dr. Volker Tresp
Zweitgutachter:	Prof. Dr. Florian Büttner
Drittgutachter:	Prof. Dr. Michael Färber
Tag der Disputation:	09.04.2024

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8 Abs. 2 Pkt. 5.)

Hiermit erkläre ich, Yushan Liu, an Eides statt, dass die vorliegende Dissertation von mir selbstständig und ohne unerlaubte Beihilfe angefertigt worden ist.

München, den 18.12.2023 _____

Yushan Liu

Contents

Abstract **vii**

Zusammenfassung **ix**

Acknowledgments **xi**

List of Publications and Declaration of Authorship **xiii**

1 Introduction **1**

1.1 Motivation 1

1.2 Summary of Contributions 3

2 Graph Machine Learning **5**

2.1 Fundamentals of Graphs 5

2.2 Machine Learning Tasks 7

2.2.1 Tasks 7

2.2.2 Evaluation 8

2.3 Subsymbolic Approaches 9

2.3.1 Knowledge Graph Embeddings 9

2.3.2 Graph Neural Networks 13

2.4 Symbolic Approaches 15

2.5 Path-Based Approaches 18

3 Explainable Artificial Intelligence **21**

3.1 Explainability in Graph Machine Learning 22

3.2 Calibration in Graph Machine Learning 23

4 Neural Multi-Hop Reasoning with Logical Rules on Biomedical Knowledge Graphs 25

5 TLogic: Temporal Logical Rules for Explainable Link Forecasting on Temporal Knowledge Graphs 49

6 A Knowledge Graph Perspective on Supply Chain Resilience 63

7 On Calibration of Graph Neural Networks for Node Classification 75

8 Conclusion 93

Bibliography 97

Abstract

In many real-world datasets, relations exist between the involved entities, such as in drug-disease interactions, supplier-customer partnerships, and citation networks. To visualize and model such data, knowledge graphs have been established, in which entities (e.g., diseases, suppliers, or publications) are represented as nodes and the relations between them (e.g., treat, supply, or cite) as edges. As knowledge is often not static but has a dynamic nature, like in event data, the edges in the graph can also be equipped with temporal information to indicate the validity of a fact at a specific timestamp or during a certain time range, leading to the concept of a temporal knowledge graph.

Even though knowledge graphs already contain a large amount of curated information, they tend to be incomplete, meaning that there is likely information regarding nodes and edges that is not included in the graph. Therefore, two common tasks concerning knowledge graph completion are node classification and link prediction in order to find missing node labels and edges, respectively. For temporal knowledge graphs, a task of interest is link forecasting, which predicts links involving future timestamps.

Due to the high domain complexity and expenditure of time required for manual effort, machine learning algorithms have been applied to knowledge graph completion tasks to great success. However, a drawback of many graph machine learning algorithms is their back-box property, i.e., it is often not transparent why exactly the algorithm has arrived at a particular result and how the output values can be interpreted, which impacts their usability in real-world applications. In classification tasks, the output is usually associated with a probability value, which reflects the confidence of the model regarding the prediction. Uncalibrated probabilities could lead to dangerous situations, e.g., accidents with autonomous vehicles, if the user's decisions depend strongly on the output values. Explainable artificial intelligence aims at increasing the interpretability, transparency, and trustworthiness of machine learning models, thus promoting user acceptance and facilitating a wider adoption of artificial intelligence.

Abstract

In this thesis, we consider the tasks of link prediction, link forecasting, and node classification with respect to different use cases, for which we introduce novel graph machine learning approaches that achieve state-of-the-art performance while providing explainability for the results. First, we study drug repurposing and predict links that connect drugs with new treatment targets in a biomedical knowledge graph. We retrieve policy-guided walks based on reinforcement learning and integrate metapaths as background information for prognosticating drug efficacy. The traversed paths in the graph can serve as explanations for the predictions. Next, we address the extrapolation task of link forecasting in the context of future event prediction, based on temporal knowledge graphs. We learn and apply temporal logical rules, which are extracted via temporal random walks. The logical rules act as explanations and can also be transferred to related datasets with a common vocabulary. Furthermore, we explore the industrial use case of identifying important suppliers to assist supply chain managers in the automatic detection of criticalities in the supply network. After the application of knowledge graph completion methods to the supply chain knowledge graph, we deploy graph analytics to rank the suppliers according to their importance in the graph. Finally, we look into the calibration of graph neural networks for the node classification task. We investigate the calibration of several graph neural network models on citation networks and propose a topology-aware method that yields improved calibration compared to baselines.

Zusammenfassung

In vielen realen Datensätzen existieren Relationen zwischen den involvierten Entitäten, z. B. bei Interaktionen zwischen Medikamenten und Krankheiten, Kunden-Lieferanten-Beziehungen und in Zitationsnetzwerken. Für die Visualisierung und Modellierung solcher Daten wurden Wissensgraphen eingeführt, die Entitäten (z. B. Krankheiten, Lieferanten oder Publikationen) als Knoten und deren Relationen (z. B. behandeln, beliefern oder zitieren) als Kanten in einem Graphen repräsentieren. Da Wissen aber oft nicht statisch, sondern dynamisch ist, wie z. B. bei Eventdaten, können die Kanten in einem Graphen auch mit zeitlichen Informationen ausgestattet werden, um die Gültigkeit einer Aussage zu einem spezifischen Zeitpunkt oder während eines bestimmten Zeitraums zu signalisieren. Ein Wissensgraph mit zeitlichen Angaben wird temporaler Wissensgraph genannt.

Auch wenn Wissensgraphen bereits große Mengen an kuratierten Informationen umfassen, sind sie meist unvollständig, d. h., es gibt vermutlich Informationen zu Knoten und Kanten, die nicht im Graphen enthalten sind. Daher sind zwei übliche Aufgaben zur Vervollständigung von Wissensgraphen die Knotenklassifizierung und Kantenvorhersage, um jeweils fehlende Knotenkategorien und Relationen zu finden. Bei temporalen Wissensgraphen ist unter anderem die Prognose von Kanten mit zukünftigen Zeitstempeln von Interesse.

Bedingt durch die hohe Domänenkomplexität und den zeitintensiven Aufwand bei manuellen Bemühungen wurde maschinelles Lernen äußerst erfolgreich zur Vervollständigung von Wissensgraphen angewandt. Jedoch sind viele maschinelle Lernalgorithmen Black Boxes, bei denen oft die Transparenz fehlt, wieso der Algorithmus zu einem bestimmten Ergebnis gekommen ist und wie die ausgegebenen Werte interpretiert werden können, was ihre Nutzbarkeit in realen Anwendungen beeinträchtigt. Die Vorhersage bei Klassifizierungen ist in der Regel mit einem Wahrscheinlichkeitswert verbunden, der das Vertrauen des Modells in das Ergebnis widerspiegelt. Unkalibrierte Wahrscheinlichkeitswerte können zu gefährlichen Situationen führen, z. B. Unfällen mit autonomen Fahrzeugen, falls

die Entscheidungen des Nutzers stark von diesen Werten abhängen. Erklärbare künstliche Intelligenz hat das Ziel, die Interpretierbarkeit, Transparenz und Vertrauenswürdigkeit von maschinellen Lernmodellen zu erhöhen, damit die Benutzerakzeptanz zu fördern und eine breitere Einführung von Methoden der künstlichen Intelligenz zu ermöglichen.

In der vorliegenden Arbeit betrachten wir die Aufgaben Kantenvorhersage, Kantenprognose und Knotenklassifizierung für verschiedene Anwendungsfälle und stellen neuartige Methoden vor, die sowohl state-of-the-art Performance als auch erklärbare Ergebnisse liefern. Zuerst prognostizieren wir Kanten für den Anwendungsfall Drug Repurposing, die existierende Medikamente mit neuen Behandlungszielen in einem biomedizinischen Wissensgraphen verbinden. Wir durchlaufen Pfade im Graphen auf Basis von bestärkendem Lernen und integrieren Metapfade als Hintergrundinformation über die Wirksamkeit der Medikamente. Diese Pfade können dann als Erklärungen für die Vorhersagen dienen. Als Nächstes befassen wir uns mit der Kantenprognose im Kontext von Eventdaten auf temporalen Wissensgraphen. Wir lernen und verwenden temporale logische Regeln, die über temporale zufällige Irrfahrten extrahiert werden. Die logischen Regeln verbessern die Verständlichkeit der Ergebnisse und können außerdem auf verwandte Datensätze mit einem gemeinsamen Vokabular übertragen werden. Darüber hinaus betrachten wir den industriellen Anwendungsfall der Identifizierung wichtiger Lieferanten, um Supply Chain Managern bei der automatischen Erkennung von Kritikalitäten im Liefernetzwerk zu unterstützen. Nach der Anwendung von Methoden zur Vervollständigung des Graphen führen wir Graphanalysen durch, um die Lieferanten nach ihrer Relevanz im Netzwerk zu sortieren. Zuletzt untersuchen wir die Kalibrierung von neuronalen Netzen auf Graphen für Knotenklassifizierung. Wir analysieren die Kalibrierung von mehreren Modellen auf Zitationsnetzwerken und schlagen eine Methode vor, die die Topologie berücksichtigt und eine bessere Kalibrierung als die Vergleichsmethoden erreicht.

Acknowledgments

This dissertation is the product of the last four years that I spent with Siemens and the Ludwig-Maximilians-Universität of Munich. During this time, I have been supported by many amazing people without whom the completion of the thesis would not have been possible.

First of all, I would like to express my sincere gratitude to my supervisor Prof. Dr. Volker Tresp for guidance throughout my entire PhD journey. Volker gave me the freedom to explore interesting research directions and define my own research questions. He was always available for discussions, giving constructive advice and invaluable feedback. Volker is an inspirational role model of how to become a great researcher and inventor. In addition, I am very honored that Prof. Dr. Florian Büttner and Prof. Dr. Michael Färber have agreed to be the external examiners of my thesis.

I would like to acknowledge Siemens and BMWK for funding my research within the projects RAKI and CoyPu. Special thanks go to Dr. Steffen Lamparter who has offered me a PhD position in his research group SMR and has provided the necessary resources and a great environment for conducting my research.

I am deeply grateful to all my co-authors and collaborators in my projects, who have contributed to the publications in significant ways, especially Dr. Marcel Hildebrandt, Dr. Mitchell Joblin, Dr. Yunpu Ma, and Tong Liu for the development of many innovative ideas, insightful discussions, and continuous motivation. I would like to extend my appreciation to all colleagues in SMR and all fellow PhD students in Volker's PhD group, in particular Dr. Zhiliang Wu, Dr. Martin Ringsquandl, Bailan He, Hang Li, Dr. Zhen Han, Dr. Gerd Völksen, Anna Himmelhuber, and Dagmar Beyer. Thank you for your support and commitment, thought-provoking exchanges, and fruitful collaborations.

Last but not least, I would like to thank my parents for their unconditional love and encouragement in any situation. I am also thankful to my partner Luca who has always believed in me. I am very fortunate to have you by my side.

List of Publications and Declaration of Authorship

- **Yushan Liu**, Marcel Hildebrandt, Mitchell Joblin, Martin Ringsquandl, Rime Raissouni, and Volker Tresp. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In *The 18th Extended Semantic Web Conference*, volume 12731, pages 375-391, 2021. DOI: 10.1007/978-3-030-77385-4_22

  > *Marcel Hildebrandt and I conceived of the original research contributions. I performed the main implementations, experiments, and evaluations. Marcel Hildebrandt and Mitchell Joblin supported me in conducting experiments on two baselines. I wrote the initial draft of the manuscript with assistance from Marcel Hildebrandt, Mitchell Joblin, and Martin Ringsquandl. All co-authors discussed this work regularly and contributed to improving the manuscript.*

  This publication serves as Chapter 4 of this thesis.

- **Yushan Liu**, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. TLogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *The 36th AAAI Conference on Artificial Intelligence*, volume 36(4), pages 4120-4127, 2022. DOI: 10.1609/aaai.v36i4.20330

  > *I conceived of the original research contributions and performed all implementations, experiments, and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. All co-authors discussed this work regularly and contributed to improving the manuscript.*

  This publication serves as Chapter 5 of this thesis.

- **Yushan Liu**, Bailan He, Marcel Hildebrandt, Maximilian Buchner, Daniela Inzko, Roger Wernert, Emanuel Weigel, Dagmar Beyer, Martin Berbalk, and Volker Tresp. A knowledge graph perspective on supply chain resilience. In *The 2nd International Workshop on Linked Data-Driven Resilience Research, Extended Semantic Web Conference*, volume urn:nbn:de:0074-3401-3, pages 1-11, 2023. URL: `https://ceur-ws.org/Vol-3401/paper3.pdf`

  > *I conceived of the original research contributions and performed all experiments and evaluations. Bailan He supported me with the implementation of the knowledge graph completion methods. I wrote the initial draft of the manuscript and did most of the subsequent corrections. All co-authors contributed to the discussion of the use case and results.*

  This publication serves as Chapter 6 of this thesis.

- Tong Liu*, **Yushan Liu***, Marcel Hildebrandt, Mitchell Joblin, Hang Li, and Volker Tresp. On calibration of graph neural networks for node classification. In *The 2022 International Joint Conference on Neural Networks*, 2022. *Equal contribution. DOI: 10.1109/IJCNN55064.2022.9892866

  > *I conceived of the original research contributions. Tong Liu performed most of the implementations. Tong Liu and I designed the experimental protocol, conducted the experiments, and evaluated the results. Tong Liu and I wrote the initial draft of the manuscript and did most of the subsequent corrections. All co-authors discussed this work regularly and contributed to improving the manuscript.*

  This publication serves as Chapter 7 of this thesis.

Other Publications

- **Yushan Liu**, Markus M. Geipel, Christoph Tietz, and Florian Buettner. TIMELY: Improving labeling consistency in medical imaging for cell type classification. In *The 24th European Conference on Artificial Intelligence*, volume 325, pages 1858-1865, 2020. DOI: 10.3233/FAIA200302

- **Yushan Liu***, Marcel Hildebrandt*, Mitchell Joblin, Martin Ringsquandl, and Volker Tresp. Integrating logical rules into neural multi-hop reasoning for drug repurposing. In *Workshop on Graph Representation Learning and Beyond, International Conference on Machine Learning*, 2020. *Equal contribution. URL: `https://grlplus.github.io/papers/60.pdf`

- Zhiliang Wu, Yinchong Yang, Yunpu Ma, **Yushan Liu**, Rui Zhao, Michael Moor, and Volker Tresp. Learning individualized treatment rules with estimated translated inverse propensity score. In *The 2020 IEEE International Conference on Healthcare Informatics*, 2020. Best Paper Award. DOI: 10.1109/ICHI48887.2020.9 374397

- Agnés Masip Gómez*, Martin Heß*, Philipp Ulrich*, Albert Eckert*, **Yushan Liu***, Theodor Isinger*, and Michael Martin*. Challenges for achieving supply chain resilience and transparency within CoyPu. In *The 1st International Workshop on Linked Data-Driven Resilience Research, Data Week Leipzig*, 2022. *Equal contribution. URL: `https://ceur-ws.org/Vol-3376/paper09.pdf`

- Caglar Demir, Anna Himmelhuber, **Yushan Liu**, Alexander Bigerl, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. Rapid explainability for skill description learning. In *The ISWC 2022 Posters, Demos and Industry Tracks, International Semantic Web Conference*, 2022. URL: `https://ceur-ws.org/Vol-3254/paper403.pdf`

Chapter 1

Introduction

1.1 Motivation

The volume and availability of data have been steadily increasing over the last decade, spanning a diverse range of domains and applications [1]. Machine learning (ML) algorithms are able to exploit massive amounts of information and make predictions based on patterns in the data, reaching or even surpassing human-level performance on specific tasks [17]. Applications of ML can be found in a variety of areas, such as recommender systems [44], social networks [47], image recognition [105], sentiment analysis [61], drug repurposing [109], and cybersecurity [85].

Data are often collected from different sources, where the involved entities are not independent but related to each other via numerous relationships, e.g., in drug-disease interactions [15], supplier-customer partnerships [5], and citation networks [27]. A potential way to model this kind of relational data is to represent the entities as nodes and their interconnections as edges in a graph, which is also termed a knowledge graph (KG) [87]. Well-known knowledge bases, which also serve as benchmarks for measuring the performance regarding graph learning and reasoning tasks, include Wikidata [103], Freebase [14], DBpedia [4], WordNet [65], Hetionet [36], and the Unified Medical Language System (UMLS) [12], just to name a few examples. Conventional KGs are static and do not contain temporal information, which indicates the validity of a fact at a certain point in time. However, knowledge usually evolves over time, e.g., the president of a country changes regularly, or a person might move to a new place of residence. To capture time-related developments, temporal KGs (tKGs) have been introduced, in which the facts can be equipped with timestamp or time range information. Many event databases have been converted to tKGs, like the

Integrated Crisis Early Warning System (ICEWS) [72], the Global Database of Events, Language, and Tone (GDELT) [53], and Wikidata [103].

Even though KGs already contain a considerable amount of curated content, they tend to suffer from incompleteness, i.e., there is information regarding nodes and edges that is missing in the graph or not up-to-date. Therefore, two classical tasks concerning knowledge graph completion are node classification and link prediction, which deal with predicting missing node labels and edges, respectively. On tKGs, a further topic of interest is link forecasting, the prediction of facts containing future timestamps. ML algorithms have been successfully applied for graph analysis as well as graph completion tasks [34, 90, 116]. Symbolic approaches perform reasoning based on symbolic representations and logical rules, which are transferable and interpretable but often not scalable to large data volumes and unable to handle noisy data. Another paradigm involves subsymbolic approaches, which make use of neural networks and vector representations for entities and relations. These approaches are more scalable and can handle noisy data but are black boxes, so it is not directly apparent why the model has arrived at a certain conclusion or how exactly the results can be interpreted. The combination of symbolic and subsymbolic approaches, called neuro-symbolic methods, tries to alleviate the respective limitations and incorporate the best of both worlds [115].

For many real-world applications, the explainability of the model outputs is important to ensure the usability of the method, especially in safety-critical situations. The field of explainable artificial intelligence (XAI) aims to provide explanations for AI models to increase the interpretability, transparency, and trustworthiness of these models [3]. Possible explanation techniques include textual descriptions, visualizations, feature relevance scores, illustrative examples, and logical rules [13]. In node classification tasks, the model output is usually associated with a probability value that reflects the confidence of the model regarding the prediction. The calibration of these probabilities could increase the user's trust in the results and support the user in making more informed decisions. XAI tackles a crucial issue in AI and is an essential step towards greater user acceptance and wider adoption of AI in many domains.

In this thesis, we consider the tasks of link prediction, link forecasting, and node classification with applications in drug repurposing, future event prediction, identification of criticalities in supply chains, and calibration on citation networks. We introduce novel graph ML approaches that yield state-of-the-art performance while providing improved explanability for the results.

1.2 Summary of Contributions

This section summarizes the main contributions of the thesis and provides an overview of the included publications.

- Chapter 2 provides an introduction to the graph structures and ML tasks addressed in the subsequent publications. We give an overview of different approaches to solving graph-related ML tasks, where we differentiate between subsymbolic, symbolic, and path-based approaches. The focus lies on the description of methods and related literature that are necessary to follow our work.

- Chapter 3 gives a brief overview of the field of explainable artificial intelligence (XAI), with an emphasis on explainability in graph ML and calibration of graph learning approaches.

- In Chapter 4, we study the use case of drug repurposing and predict links that connect existing drugs with new treatment targets in a biomedical KG [57]. We propose the neuro-symbolic approach Policy-Guided Walks with Logical Rules (PoLo), which combines representation learning and logic. Given a compound, we predict a treatable disease by retrieving policy-guided walks based on reinforcement learning and integrating logical rules in the form of metapaths as background information for prognosticating drug efficacy. The metapaths are part of a novel reward function, which guides the reinforcement learning agent according to domain principles. We apply our method to the biomedical KG Hetionet [36] and show that the integration of metapaths leads to better performance, while the extracted paths from the graph can serve as explanations for the predictions.

- In Chapter 5, we address the extrapolation task of link forecasting in the context of future event prediction based on tKGs [58]. Future event prediction has many application areas, such as supply chain management [69], catastrophe modeling [91], and clinical decision support [118]. We focus on the tKGs based on ICEWS [72], which model political events as facts equipped with timestamps. We introduce the symbolic approach TLogic, which learns and applies temporal logical rules, extracted from the graph via temporal random walks. Experiments on three ICEWS benchmark datasets show better overall performance compared to state-of-the-art baselines. The logical rules act as explanations for the forecasts and can also be transferred to related datasets with a common vocabulary without retraining.

- In Chapter 6, we explore the industrial use case of identifying important suppliers to assist supply chain managers in the automatic detection of criticalities in supply networks [56]. Due to the complexity and intransparency of supply chains [18], it is difficult for supply chain managers to obtain precise predictions about prospective risks manually. We connect and model supply chain-related entities as a KG and apply KG completion methods to find missing relationships. Then, we deploy graph analytics, including the calculation of centrality measures, to rank the suppliers according to their importance in the graph. The interpretation of the computed graph measures and resulting scores are discussed with domain experts, making the process and findings more comprehensible.

- In Chapter 7, we look into calibration of graph neural networks for the node classification task [55]. The calibration of deep neural networks has already been analyzed in many works [33, 96, 97], but the calibration of graph neural networks has not been sufficiently explored yet. We investigate the calibration of several graph neural network models on three citation networks under varying model capacity, graph density, and loss functions. We further propose the topology-aware calibration method Ratio-Binned Scaling (RBS), which takes the predicted labels of neighboring nodes into account and yields improved calibration compared to other post-processing baselines. In many cases, calibrated probabilities can lead to a deeper understanding of the results and increased trustworthiness in the ML algorithms.

Chapter 2

Graph Machine Learning

2.1 Fundamentals of Graphs

Definition 2.1.1. A *graph* is defined as a tuple $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes a set of nodes and $\mathcal{E}$ a set of edges. An *edge* $e \in \mathcal{E}$ is described by a pair of nodes $(v_i, v_j) \in \mathcal{V} \times \mathcal{V}$, with $i, j \in \{1, 2, \ldots, |\mathcal{V}|\}$. A graph is *undirected* if the existence of an edge $(v_i, v_j) \in \mathcal{E}$ implies $(v_j, v_i) \in \mathcal{E}$. Otherwise, a graph is called *directed*.

Nodes in a graph represent entities or concepts, while the edges show the interactions or relations between the nodes.

Definition 2.1.2. A node $v \in \mathcal{V}$ can be associated with a *feature vector* $\mathbf{x}_v \in \mathbb{R}^f$. The *feature matrix* containing the feature vectors of all nodes is denoted by $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times f}$.

Feature vectors specify more detailed information about the nodes, e.g., a collection of words for nodes representing documents.

Definition 2.1.3. A graph can be represented by an *adjacency matrix* $\mathbf{A} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$. With $\mathbf{A}[i, j]$ denoting the entry in the i-th row and j-th column, $\mathbf{A}[i, j] = 1$ if $(v_i, v_j) \in \mathcal{E}$, and $\mathbf{A}[i, j] = 0$ otherwise.

Definition 2.1.4. A graph $G = (\mathcal{V}, \mathcal{E})$ is *homogeneous* if G only contains one node type and one edge type. A graph is *heterogeneous* if it contains multiple node types or multiple edge types.

An example of a homogeneous graph is a social network in which the nodes represent people and the edges the friendship relation (see Figure 2.1). An example of a heterogeneous graph is a supply chain network that includes entities such as suppliers, countries,

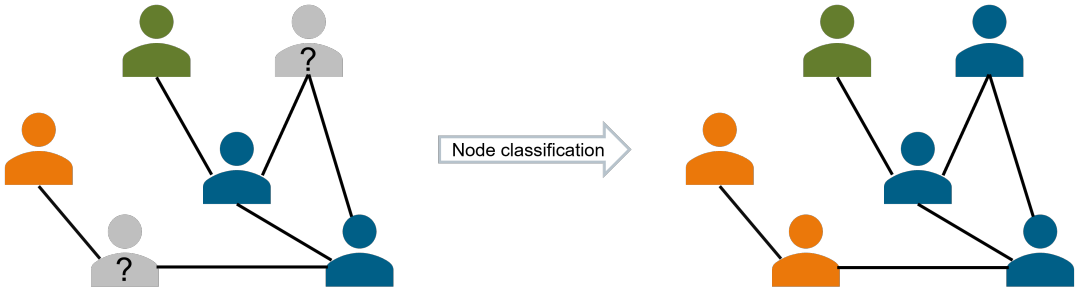


Figure 2.1: A social network graph. The nodes represent people, and the edges display their friendship relations. The people have different labels indicated by distinct colors, which represent, e.g., favorite sports, political orientations, or locations. The node classification task aims to predict missing node labels in the graph.

and industries, which are connected via different edge types (see Figure 2.2). Heterogeneous graphs are often referred to as knowledge graphs.

Definition 2.1.5. A *knowledge graph* (KG) is defined as a set of triples $\mathcal{KG} \subset \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, where $\mathcal{V}$ denotes a set of nodes and $\mathcal{R}$ a set of binary relations. A triple in the form *(subject, predicate, object)* $\in \mathcal{KG}$ indicates a directed edge (or link) between the *subject* node and the *object* node via the *predicate* relation. For a relation $r \in \mathcal{R}$, the *inverse relation* is represented by r^{-1}, so the *inverse triple* is *(object, predicate*$^{-1}$*, subject)*.

Conventional KGs encode static facts, e.g., *(Leonhard Euler, born in, Basel)* or *(Budesonide, treats, Asthma)*. However, real-world knowledge is usually dynamic and evolves continuously. Events happen at a specific timestamp, e.g., Leonhard Euler was born in Basel on April 15, 1707, or properties of a person, such as the place of residence, can change during his lifetime. Temporal information can be captured by temporal knowledge graphs, which extend triples to quadruples by adding corresponding timestamps.

Definition 2.1.6. A *temporal knowledge graph* (tKG) is defined as a set of quadruples $\mathcal{TKG} \subset \mathcal{V} \times \mathcal{R} \times \mathcal{V} \times \mathcal{T}$, where $\mathcal{V}$ denotes a set of nodes, $\mathcal{R}$ a set of binary relations, and $\mathcal{T}$ a set of timestamps. A quadruple in the form *(subject, predicate, object, timestamp)* $\in \mathcal{TKG}$ indicates the occurrence or validity of the fact *(subject, predicate, object)* at the given *timestamp*. The *inverse quadruple* is represented by *(object, predicate*$^{-1}$*, subject, timestamp)*.

Figure 2.3 shows an exemplary tKG based on the event dataset ICEWS14 [30], which contains information about political events from the year 2014. Each event triple is additionally equipped with a timestamp that signifies the date of occurrence.

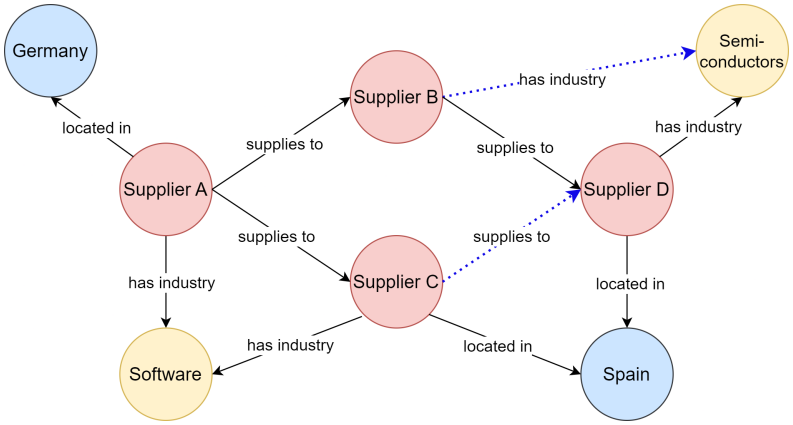


Figure 2.2: A supply chain knowledge graph including supplier, country, and industry entities with their connections. Link prediction methods propose new edges (dotted blue lines) to connect entities in the graph.

2.2 Machine Learning Tasks

2.2.1 Tasks

Many KGs contain a lot of content, but they are usually still incomplete, so common graph ML tasks involve KG completion, such as node classification on graphs, link prediction on KGs, and link forecasting on tKGs.

Node Classification In many graphs, the nodes possess labels that reflect certain properties of the nodes. As shown in Figure 2.1, some nodes might lack the label information though. The goal of the *node classification* task is to assign each node $v \in \mathcal{V}$ a unique label $\hat{y}_v \in \mathcal{C} := \{1, 2, \ldots, C\}$, where C stands for the total number of labels. The node classification task on citation networks is considered in Chapter 7.

Link Prediction Not only node labels are likely to be missing in graphs but also edges between the nodes, where the corresponding ML task for finding missing edges is called *link prediction* (see Figure 2.2). Link prediction on a KG can be phrased as an object prediction task. Given a query in the form *(subject, predicate, ?)*, the goal is to predict the correct object entity or a list of possible object candidates that are most likely to complete the query. Subject prediction can be realized by reformulating the query as *(object, predicate^{-1}, ?)*. Chapter 4 and Chapter 6 focus on the link prediction task on a biomedical KG and a supply network, respectively.

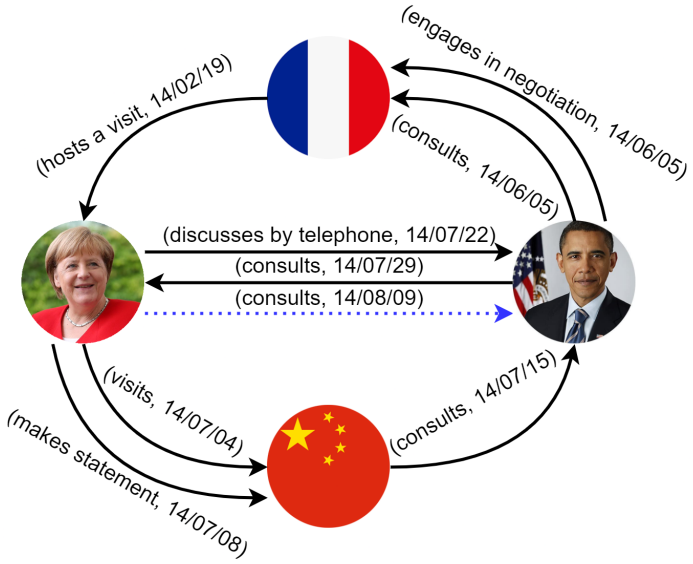


Figure 2.3: A subgraph from the event dataset ICEWS14 containing the entities *Angela Merkel*, *Barack Obama*, *France*, and *China*. The timestamps are represented as days in the format yy/mm/dd. The dotted blue line connects to the correct answer of the link forecasting task with the query *(Angela Merkel, consults, ?, 2014/08/09)*.

Link Forecasting For a tKG, let t_{min} be the earliest observed timestamp and t_{max} the latest observed timestamp in the graph so that all edges are associated with a timestamp from the interval $[t_{\text{min}}, t_{\text{max}}]$. Similarly to link prediction, we can formulate the *link forecasting* task as an object prediction problem with a prespecified timestamp. Given a query in the form *(subject, predicate, ?, timestamp)*, where the *timestamp* occurs later than t_{max}, we want to find the correct object entity or a list of possible object candidates (see Figure 2.3). Subject prediction can be performed by using the inverse relation and defining the query as *(object, predicate^{-1}, ?, timestamp)*. We study the link forecasting task on event tKGs in Chapter 5.

2.2.2 Evaluation

Node Classification For evaluating a model's performance regarding node classification, we use the metric *accuracy*, which is typically applied for evaluating ML classification tasks. Let $y_v \in \mathcal{C}$ be the ground-truth label and $\hat{y}_v \in \mathcal{C}$ the predicted label of a node $v \in \mathcal{V}$. The accuracy is the proportion of correctly classified instances, i.e., the percentage of nodes with $\hat{y}_v = y_v$.

Link Prediction and Forecasting As output of the link prediction and forecasting task, we obtain a ranked list of possible object candidates, where a higher rank (i.e., a rank with a smaller number) indicates a higher probability that the corresponding object is the correct answer. To measure the quality of the predicted list, we calculate the standard metrics *mean reciprocal rank* (MRR) and *hits@k* for $k \in \mathbb{N}$. For a rank $x \in \mathbb{N}$, i.e., the position in the ranked list of object candidates, the reciprocal rank is defined as $\frac{1}{x}$, and the MRR is the average of all reciprocal ranks of the correct query answers over all queries. The metric hits@k denotes the proportion of queries for which the correct answer appears under the top k candidates. Generally, a query might have multiple valid answers but only a unique one that is correct for a specific test query. Therefore, we filter the candidate list according to Bordes et al. [16] and remove the other valid objects from the list so that these objects do not worsen the performance of the model if they are ranked higher than the correct answer. For link forecasting, we implement time-aware filtering [35] and filter out all valid objects at the given query timestamp except for the true query object.

2.3 Subsymbolic Approaches

Subsymbolic approaches aim to automatically learn connections between input and output. The models try to imitate biological behavior by relying on distributed representations without explicitly defining logical rules. These approaches can learn from large-scale datasets and handle noisy data but are often black boxes where the reasons for particular predictions are not transparent. Subsymbolic methods include, e.g., statistical learning [39], Bayesian learning [68], genetic algorithms [41], artificial neural networks [110], and deep learning [52]. In the context of graph ML, we will take a closer look at KG embeddings and graph neural networks. For a more in-depth investigation of graph representation learning on static and temporal KGs, comprehensive overviews can be found in the following surveys: [20], [34], [42], [45], [54], and [104].

2.3.1 Knowledge Graph Embeddings

The idea of representation learning on KGs is to learn low-dimensional vector representations, called embeddings, of entities and relations that preserve both local and global information from the graph. Intuitively, we expect that two nodes with similar semantic meanings also have similar embeddings. The learned embeddings can then be used for

downstream tasks like node classification and link prediction.

The basic approach learns shallow embeddings where the entities and relations are mapped to vectors stored in a table. Each entity or relation has a unique ID for identification purposes. The optimization of the embeddings is usually based on a reconstruction loss function. The loss function measures the discrepancy between the predicted output given the embeddings and the ground-truth output with respect to the task of interest. The trained embeddings can then be retrieved by looking up the corresponding IDs in the table. First, we describe link prediction methods for static KGs. These methods learn entity and relation embeddings for a scoring function f that assigns a value $f(v_\mathrm{s}, r, v_\mathrm{o}) \in \mathbb{R}$ to a triple $(v_\mathrm{s}, r, v_\mathrm{o}) \in \mathcal{V} \times \mathcal{R} \times \mathcal{V}$, which indicates the likelihood that the triple exists in the KG. Let $\mathbf{v}_\mathrm{s}$, $\mathbf{r}$, and $\mathbf{v}_\mathrm{o}$ be the embeddings of the subject entity v_s, relation r, and object entity v_o, respectively.

- RESCAL [70] was the first approach to be published for learning KG embeddings. It models triples in the KG as a three-way tensor, applies tensor factorization, and learns a full-rank matrix $\mathbf{R}_r$ for each relation r. The scoring function is given by

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = \mathbf{v}_\mathrm{s}^T \mathbf{R}_r \mathbf{v}_\mathrm{o}, \quad \mathbf{v}_\mathrm{s},\, \mathbf{v}_\mathrm{o} \in \mathbb{R}^d, \quad \mathbf{R}_r \in \mathbb{R}^{d \times d}.$$

- DistMult [108] has the same scoring function as RESCAL but restricts the relation matrix to be diagonal to reduce the number of parameters. Thus, the scoring function can be formulated as the component-wise multiplication of three vectors and the subsequent summarization of all components, represented by the function $\langle \cdot, \cdot, \cdot \rangle$:

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = \langle \mathbf{v}_\mathrm{s}, \mathbf{r}, \mathbf{v}_\mathrm{o} \rangle, \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{v}_\mathrm{o} \in \mathbb{R}^d.$$

- ComplEx [98] is an extension of DistMult that operates on the complex space. The object embedding of an entity v is defined as the complex conjugate of the subject embedding $\mathbf{v} \in \mathbb{C}^d$. The score is represented by the real part of the resulting vector multiplication, i.e.,

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = \mathrm{Re}(\langle \mathbf{v}_\mathrm{s}, \mathbf{r}, \bar{\mathbf{v}}_\mathrm{o} \rangle), \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{v}_\mathrm{o} \in \mathbb{C}^d,$$

  where $\bar{\cdot}$ denotes the complex conjugate of a vector.

- TuckER [8] is a tensor factorization model based on the Tucker decomposition [99]. The models RESCAL, DistMult, and ComplEx can be seen as special cases of

TuckER. The scoring function is given by

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = \mathbf{W} \times_1 \mathbf{v}_\mathrm{s}^T \times_2 \mathbf{r}^T \times_3 \mathbf{v}_\mathrm{o}^T, \quad \mathbf{v}_\mathrm{s},\, \mathbf{v}_\mathrm{o} \in \mathbb{R}^{d_\mathrm{v}}, \quad \mathbf{r} \in \mathbb{R}^{d_\mathrm{r}}, \quad \mathbf{W} \in \mathbb{R}^{d_\mathrm{v} \times d_\mathrm{r} \times d_\mathrm{v}},$$

where $\mathbf{W}$ is the trainable core tensor of the Tucker decomposition and $\times_i$ the tensor product along the i-th mode.

- TransE [16] models relationships between two entities as translations between the corresponding entity embeddings, with

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = -||\mathbf{v}_\mathrm{s} + \mathbf{r} - \mathbf{v}_\mathrm{o}||, \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{v}_\mathrm{o} \in \mathbb{R}^d.$$

- RotatE [92] models relationships between two entities as rotations in the complex space. With $\odot$ denoting the component-wise multiplication,

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = -||\mathbf{v}_\mathrm{s} \odot \mathbf{r} - \mathbf{v}_\mathrm{o}||, \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{v}_\mathrm{o} \in \mathbb{C}^d,$$

  under the constraint that the absolute value of each component in $\mathbf{r}$ should be 1.

- SimplE [43] learns two embeddings for an entity v, where $\mathbf{v}(\mathrm{s})$ stands for the subject embedding and $\mathbf{v}(\mathrm{o})$ for the object embedding. The score of a triple is computed as the average of two components:

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = \frac{1}{2}\Big(\langle \mathbf{v}_\mathrm{s}(\mathrm{s}), \mathbf{r}, \mathbf{v}_\mathrm{o}(\mathrm{o})\rangle + \langle \mathbf{v}_\mathrm{o}(\mathrm{s}), \mathbf{r}^{-1}, \mathbf{v}_\mathrm{s}(\mathrm{o})\rangle\Big),$$

  where $\mathbf{v}_\mathrm{s}(\mathrm{s})$, $\mathbf{v}_\mathrm{s}(\mathrm{o})$, $\mathbf{r}$, $\mathbf{r}^{-1}$, $\mathbf{v}_\mathrm{o}(\mathrm{s})$, $\mathbf{v}_\mathrm{o}(\mathrm{o}) \in \mathbb{R}^d$. The vector $\mathbf{r}^{-1}$ denotes the embedding of the inverse relation of r.

- ConvE [25] models the interactions in a KG by convolutional and fully connected layers. The subject and relation embeddings are reshaped, concatenated, and used as input for a 2D-convolutional layer with filters $\mathbf{U}$. The result is flattened and transformed by a trainable matrix $\mathbf{W} \in \mathbb{R}^{nm_1m_2 \times d}$, where n represents the number of filters and m_1 and m_2 are the dimensions of the feature maps. The complete scoring function is given by

$$f(v_\mathrm{s}, r, v_\mathrm{o}) = \sigma\Big(\mathrm{flatten}\Big(\sigma([\tilde{\mathbf{v}}_\mathrm{s}; \tilde{\mathbf{r}}] * \mathbf{U})\Big)\mathbf{W}\Big)\mathbf{v}_\mathrm{o}, \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{v}_\mathrm{o} \in \mathbb{R}^d,$$

  where $\tilde{\mathbf{v}}_\mathrm{s}$, $\tilde{\mathbf{r}} \in \mathbb{R}^{d_1 \times d_2}$ with $d = d_1 \cdot d_2$. The function σ denotes a non-linear activation function.

In tKGs, triples are additionally associated with time intervals or timestamps, and we are interested in the validity of a quadruple $(v_\mathrm{s}, r, v_\mathrm{o}, t) \in \mathcal{V} \times \mathcal{R} \times \mathcal{V} \times \mathcal{T}$. Many methods for static KG completion have been extended to take the temporal dimension into account for calculating the scoring function.

- Temporal TransE (TTransE) [51] extends TransE and adds the timestamp as another vector to the scoring function

$$f(v_\mathrm{s}, r, v_\mathrm{o}, t) = -||\mathbf{v}_\mathrm{s} + \mathbf{r} + \mathbf{t} - \mathbf{v}_\mathrm{o}||, \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{v}_\mathrm{o},\, \mathbf{t} \in \mathbb{R}^d.$$

- Temporal-Aware DistMult (TA-DistMult) [30] uses the same scoring function as the method DistMult, now including temporal information in the relation embedding. The relation type, temporal modifiers (e.g., *since*, *until*), and temporal tokens that represent a point in time are concatenated and processed by a Long Short-Term Memory (LSTM) model [37] to obtain the relation embedding $\mathbf{r}_\mathrm{temp}$. The resulting scoring function is

$$f(v_\mathrm{s}, r, v_\mathrm{o}, t) = \langle \mathbf{v}_\mathrm{s}, \mathbf{r}_\mathrm{temp}, \mathbf{v}_\mathrm{o} \rangle, \quad \mathbf{v}_\mathrm{s},\, \mathbf{r}_\mathrm{temp},\, \mathbf{v}_\mathrm{o} \in \mathbb{R}^d.$$

- TNTComplEx [48] is an extension of ComplEx and can handle both static and temporal facts by introducing two relation embeddings. The relation embedding for temporal facts is additionally multiplied with a time embedding:

$$f(v_\mathrm{s}, r, v_\mathrm{o}, t) = \mathrm{Re}(\langle \mathbf{v}_\mathrm{s}, \mathbf{r} + \mathbf{r}_\mathrm{temp} \odot \mathbf{t}, \bar{\mathbf{v}}_\mathrm{o} \rangle), \quad \mathbf{v}_\mathrm{s},\, \mathbf{r},\, \mathbf{r}_\mathrm{temp},\, \mathbf{v}_\mathrm{o},\, \mathbf{t} \in \mathbb{C}^d.$$

- DE-SimplE [32] replaces the entity embeddings in SimplE with diachronic entity embeddings. The i-th component of the diachronic embedding of an entity v at time t is defined as follows:

$$\mathbf{v}^{(t)}[i] = \begin{cases} \mathbf{a}_v[i] \cdot \sigma(\mathbf{w}_v[i]t + \mathbf{b}_v[i]) & \text{for } 1 \leq i \leq \lambda d, \\ \mathbf{a}_v[i] & \text{for } \lambda d < i \leq d, \end{cases}$$

  where $\mathbf{a}_v \in \mathbb{R}^d$ and $\mathbf{b}_v$, $\mathbf{w}_v \in \mathbb{R}^{\lambda d}$ are trainable vectors and σ is a non-linear activation function. The hyperparameter $\lambda \in [0, 1]$ determines the proportion of time-dependent entries.

- The Temporal Copy-Generation Network (CyGNet) [117] learns patterns from historical facts, based on the observation that many events occur periodically. Given an object prediction query $(v_s, r, ?, t)$, CyGNet employs a copy-generation mechanism. The copy mode infers the probabilities for objects that have appeared together with the query subject and query relation in the previous history, while the generation mode infers probabilities based on all entities in the vocabulary. The final scores are obtained by combining the probability distributions from both modes.

This section focused on shallow embeddings, where we directly optimize a vector for each entity, relation, and timestamp. However, shallow embeddings are inefficient since no parameters are shared between embeddings, incapable of using node features, and inapplicable in inductive settings, i.e., cases where previously unseen nodes, relations, or timestamps are involved. In the next section, we will look at graph neural networks, which take the graph structure as well as node features into account for learning embeddings.

2.3.2 Graph Neural Networks

Graph neural networks (GNNs) operate on graph data and learn embeddings that depend on the graph structure and existing node features. The basic underlying principle is neural message passing, where adjacent nodes exchange information in the form of vectors. For learning the embedding of a node v, information from all neighbors of v is collected, aggregated, and normalized during each message passing iteration. The aggregated message is used to update the embedding of v, and we obtain the final node embedding after a specific number of iterations. The update is often achieved by adding self-loops to the graph. After N iterations, the node embeddings contain information from their respective N-hop neighborhoods.

We first describe GNNs for a simple graph $G = (\mathcal{V}, \mathcal{E})$, with $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ denoting the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times f}$ the feature matrix. The node embeddings at iteration $n \in \mathbb{N}$, also called hidden embeddings, are saved in the matrix $\mathbf{H}^{(n)}$, where the i-th row $\mathbf{H}^{(n)}[i]$ represents the hidden embedding of the node v_i. The final embedding of the node v_i is given by the column vector $\mathbf{v}_i := \mathbf{H}^{(N)}[i]^T \in \mathbb{R}^d$.

- The Graph Convolutional Network (GCN) [46] normalizes the aggregated messages from neighboring nodes and adds self-loops to the adjacency matrix. The hidden layer at iteration $n+1$ is computed as follows:

$$\mathbf{H}^{(n+1)} = \sigma\Big(\tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(n)}\mathbf{W}^{(n)}\Big),$$

where $\mathbf{I}$ is the $|\mathcal{V}|$-dimensional identity matrix and $\tilde{\mathbf{D}}$ the degree matrix of $\mathbf{A}+\mathbf{I}$, i.e., $\tilde{\mathbf{D}}[i,i] = \sum_{j=1}^{|\mathcal{V}|}(\mathbf{A}+\mathbf{I})[i,j]$. The matrix $\mathbf{W}^{(n)}$ is a trainable weight matrix σ a non-linear activation function. Further, we set $\mathbf{H}^{(0)} := \mathbf{X}$.

- The Graph Attention Network (GAT) [101] introduces an attention mechanism to express the importance of a node during message aggregation. The attention of the node v_i towards its neighbor v_j is formulated as

$$\alpha_{ij} = \frac{\exp\Big(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{H}^{(n)}[i]^T; \mathbf{W}\mathbf{H}^{(n)}[j]^T])\Big)}{\sum_{v_{j'}\in\mathcal{N}(v_i)}\exp\Big(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{H}^{(n)}[i]^T; \mathbf{W}\mathbf{H}^{(n)}[j']^T])\Big)},$$

  where $\mathbf{W}$ and $\mathbf{a}$ are trainable parameters and $\mathcal{N}(v_i)$ denotes the neighbors of v_i. LeakyReLU [59] is applied as activation function.

- The authors of Simple Graph Convolution (SGC) [106] hypothesize that GCNs mostly benefit from the aggregation of local neighborhood information and not from the application of non-linearities. Therefore, they propose to remove the non-linear activation functions so that the resulting embeddings are given by

$$\mathbf{H}^{(N)} = \tilde{\mathbf{A}}^N\mathbf{X}\mathbf{W},$$

  where $\tilde{\mathbf{A}}^N$ is a normalized adjacency matrix to the power of N and $\mathbf{W} \in \mathbb{R}^{f\times d}$ a trainable weight matrix.

In KGs, there is typically not only one edge type, but multiple relations exist between the nodes. Conventional GNNs have been extended to multi-relational models to accommodate this situation.

- The Relational Graph Convolutional Network (R-GCN) [86] is an extension of GCN, which introduces relation-dependent transformations $\mathbf{W}_r$. The hidden embedding of the node v_i is computed by

$$\mathbf{H}^{(n+1)}[i]^T = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{v_j\in\mathcal{N}_r(v_i)}\frac{1}{c_{i,r}}\mathbf{W}_r^{(n)}\mathbf{H}^{(n)}[j]^T\right),$$

  where $\mathbf{W}_r^{(n)}$ is a trainable weight matrix and $c_{i,r} \in \mathbb{R}$ a normalization constant. The set $\mathcal{N}_r(v_i)$ denotes the neighbors of the node v_i that are connected to v_i via the relation r.

- CompGCN [100] is a composition-based multi-relational GCN, which learns joint embeddings for nodes and relations via composition operations ϕ. The embeddings of the node v_i and relation r are then obtained through the following equations:

$$\mathbf{H}^{(n+1)}[i]^T = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{v_j\in\mathcal{N}_r(v_i)}\mathbf{W}^{(n)}_{\mathrm{dir}(r)}\phi\Big(\mathbf{H}^{(n)}[j]^T,\mathbf{r}^{(n)}\Big)\right), \quad \mathbf{r}^{(n+1)} = \mathbf{W}^{(n)}_{\mathrm{rel}}\mathbf{r}^{(n)},$$

  where $\mathbf{W}^{(n)}_{\mathrm{rel}}$ is a learnable weight matrix for all relations and $\mathbf{W}^{(n)}_{\mathrm{dir}(r)}$ a learnable weight matrix that depends on the direction of the relation r.

GNNs have also been adapted to tKGs to incorporate temporal knowledge. Unlike shallow KG embeddings, GNNs can generate embeddings of previously unseen entities, relations, and timestamps, which is especially helpful regarding the link forecasting task.

- The Recurrent Event Network (RE-Net) [40] consists of an autoregressive architecture, which treats link forecasting as a multi-step inference task. The tKG is represented as a sequence of graph snapshots, where a graph snapshot contains all events that occurred at the same timestamp. RE-Net first applies a neighborhood aggregator such as R-GCN to encode the graph structure and local information within each graph snapshot. Then, a recurrent neural network [21] learns the probability distribution of events at a certain point in time while taking previous events into account. Finally, the probability of the current graph is obtained by using a feedforward artificial neural network [110] as decoder.

- xERTE [35] is an explainable reasoning framework for link forecasting on tKGs that employs a temporal relational graph attention mechanism to propagate messages and guide the extraction of relevant information from the graph. The attention mechanism introduces temporal constraints on message passing while the learned entity embeddings contain a time-dependent component. xERTE samples a query-dependent subgraph around the query subject that includes important entities and relations for answering the query. The final object is predicted as one of the nodes in the extracted subgraph, which serves as a graphical explanation for the answer.

2.4 Symbolic Approaches

Symbolic approaches have a long history in knowledge modeling and reasoning. Prominent reasoning approaches rely on logic-based systems, such as description logic [6], which consists of knowledge representation languages like the Web Ontology Language (OWL) [2],

logic programming [31], which are implemented in methods like Prolog [22] and the First-Order Inductive Learner (FOIL) [79], and fuzzy logic [112], which associates a real number as truth value with logical statements. Symbolic approaches are easier to interpret than subsymbolic algorithms and yield good performance for well-defined problems but can be computationally expensive with increasing domain complexity and have difficulties handling noisy or ambiguous data.

One basic building block of logic-based reasoning is the application of logical rules that are formulated as *Horn clauses* [38] in the form

$$h \leftarrow b_1 \wedge b_2 \wedge \cdots \wedge b_K, \quad K \in \mathbb{N},$$

where h represents the head of the rule and the conjunction of atomic formulas b_k, also called atoms, the body of the rule. The arrow denotes logical implication, i.e., if the rule body evaluates to true, then the statement in the rule head is also true. In a KG context, the rule head and body atoms can be realized as triples, e.g.,

(X:Person, born in, Y:Country) $\leftarrow$

(X:Person, born in, Z:City) $\wedge$ *(Z:City, located in, Y:Country)*,

where X, Y, and Z are variables that represent entities of the types *Person*, *Country*, and *City*, respectively. For the application of a logical rule, we can ground the logical expression by replacing the variables by instance values, i.e., for the example above, a possible grounding could be

(Isa, born in, Germany) $\leftarrow$ *(Isa, born in, Munich)* $\wedge$ *(Munich, located in, Germany).*

Since the implications in logical rules are not necessarily always true, probabilistic logic [71] introduces uncertainty by attaching a numerical value to the logical rule to reflect the probability that, given a grounded rule body, the rule head is true. For example, Markov Logic Networks (MLNs) [80] combine first-order logic [88] with probabilistic graphical networks to model logical statements with weights, where the nodes are atoms and the edges logical operators.

Logical rules can either be manually crafted by domain experts or automatically mined from the KG. The method AMIE [28] retrieves Horn rules from the KG by following the steps of rule mining and rule pruning. In the rule mining step, atoms are added iteratively to an initially empty rule to generate rule candidates that are closed, i.e., rules where each variable appears at least twice. In the second step, the candidate rules are pruned based

on their confidence values and head coverage, a normalized proportion of instantiations in the graph that are supported by the rule. The follow-up work AMIE+ [29] improves the efficiency of AMIE by refining the rule mining process and approximating the metrics for pruning. A more recent version, AMIE 3 [49], introduces even more efficient pruning strategies, faster evaluation of exact confidence values, and parallelization of specific tasks, making it possible to apply AMIE to large-scale knowledge bases.

Anytime Bottom-Up Rule Learning (AnyBURL) [62, 63] is a symbolic approach based on random path sampling, which is guided by reinforcement learning [93]. AnyBURL learns three types of rules: binary rules, unary rules with an ending dangling atom, and unary rules with an ending constant entity. The rule head of binary rules contains two variables that represent the subject and object nodes, while the rule head of unary rules contains one variable and one constant entity. The rules are constructed from the sampled paths, which are generalized to Horn rules by introducing variables and dropping redundant atoms. Due to a multi-threaded implementation in the newest version of AnyBURL [62], the method can be efficiently scaled to large knowledge bases. The framework Scalable and Fast Non-Redundant Rule Application (SAFRAN) [74] is another approach based on AnyBURL, which employs a novel rule clustering and aggregation mechanism to improve the performance.

In the case of tKGs, logical rules can be adapted to incorporate temporal information. *Temporal logical rules* can be defined by adding a timestamp to the rule head and body atoms, such as

$$(v_1, r, v_{K+1}, t) \leftarrow (v_1, r_1, v_2, t_1) \wedge (v_2, r_2, v_3, t_2) \wedge \cdots \wedge (v_K, r_K, v_{K+1}, t_K).$$

To preserve time consistency, we can add a temporal constraint $t_1 \leq t_2 \leq \cdots \leq t_K < t$, with the interpretation that the rule body expresses a sequence of events that lead to the event in the rule head at a future timestamp.

StreamLearner [73] learns temporal logical rules by first extracting static rules from a subgraph and then extending them with timestamp information. The final temporal rules are selected based on the quality metrics dynamic confidence and dynamic head coverage, the generalizations of the metrics confidence and head coverage to the dynamic case. StreamLearner retrieves rules in the form

$$(v_1, r, v_{K+1}, t + k) \leftarrow (v_1, r_1, v_2, t) \wedge (v_2, r_2, v_3, t) \wedge \cdots \wedge (v_K, r_K, v_{K+1}, t),$$

where all facts in the rule body are concurrent and occur at the timestamp t, while the rule head is an event with the future timestamp $t + k$.

Generally, the respective groundings of the logical rules in the graph can serve as an explanation of the predictions and lead to more interpretable results.

2.5 Path-Based Approaches

A further kind of approach is based on graph traversal to extract reasoning paths, which is often a combination of subsymbolic and symbolic methods [115]. Starting from the query subject of a link prediction task, the traversed path is extended sequentially, with the aim of finding the correct query object at the end of the traversal. The extracted paths provide multi-hop reasoning chains and increase the transparency of the predictions, which can be especially helpful for embedding-based methods. We define a *path* of length L in a KG as a sequence of triples

$$(v_1, r_1, v_2), (v_2, r_2, v_3), \cdots, (v_L, r_L, v_{L+1}),$$

where the triples are distinct, but the nodes can be recurrent. If the nodes v_1 and v_{L+1} are the same, we call the path closed. Some literature refers to such sequences as walks and defines only sequences with distinct nodes and edges as paths.

A possible way to retrieve paths is via random walks [76], which are stochastic processes that determine the steps and directions of a walk from a specific starting point. The Path Ranking Algorithm (PRA) [50] uses random walks to connect query subjects with potential object candidates for link prediction. PRA learns a weighting for the connecting paths and defines the paths as input to a linear model to rank the object entities.

Besides sampling random walks in a discrete space, it is also possible to guide the walk by exploiting embeddings and reinforcement learning (RL) to make it easier to evaluate similar nodes and entities based on their embeddings. DeepPath [107] is an RL-approach that extracts policy-guided random walks as part of a Markov Decision Process (MDP) [9]. The current state encodes the RL agent's location in the graph based on entity embeddings, which is then mapped to a stochastic policy to determine the next action. The reward of the agent depends on the correctness of the final entity, the length of the path, and the diversity of paths. Similar to PRA, the extracted paths are used in a linear model to predict an object ranking.

MINERVA [23], which stands for Meandering in Networks of Entities to Reach Verisimilar Answers, is also an RL-based reasoning method, which defines a Partially Observable Markov Decision Process (POMDP) [119] where the correct query answer is unknown dur-

ing graph traversal. The RL agent starts from the query subject and navigates to a node that might be a possible option for the query object. A positive reward is given to the agent if the correct answer has been reached in the final step.

The extracted paths remind of the formulation of logical rules in the previous section. One way to obtain logical rules is to generalize the paths from the graph to Horn clauses, which are then applied for inference. AnyBURL [62, 63] was at first based on sampling random walks [63] to generate rules and then added RL to achieve better performance [62]. Closed paths can be transformed into logical rules, where the relation in the rule head connects the variables from the body atoms b_1 and b_K, so the rule body describes the reason why an edge is likely to exist between the corresponding two entities.

Random walks can also be extended to tKGs, and we define a *temporal random walk* of length L as a sequence of quadruples

$$(v_1, r_1, v_2, t_1),\ (v_2, r_2, v_3, t_2),\ \cdots,\ (v_L, r_L, v_{L+1}, t_L),$$

where $t_1 \leq t_2 \leq \cdots \leq t_L$ or $t_1 \geq t_2 \geq \cdots \geq t_L$, i.e., it is only feasible to traverse the graph following one direction of time. Nguyen et al. create continuous-time dynamic network embeddings (CTDNEs) [60] by extracting temporal random walks and learning time-preserving node embeddings with a generalized Skip-Gram architecture [64]. The dynamic node embeddings can then be used in downstream tasks with suitable scoring functions. Temporal random walks produce time-consistent explanations, which are aligned with the intuition that previous events might trigger future developments.

Chapter 3

Explainable Artificial Intelligence

The emergence of high-performing ML algorithms and generative AI led to widespread adoption of AI approaches in various domains, but explainability and validation are essential to ensure reliable and secure usage. Explainable artificial intelligence (XAI) aim to provide human-understandable explanations for black-box AI models to reduce bias, make the results more interpretable, and increase the trustworthiness of the models. Explanations can have diverse forms depending on the data modality, such as rule-based explanations, saliency maps, feature importance scores, textual descriptions, prototypes, and counterfactuals [13]. Different taxonomies and categorizations exist for key concepts in XAI [3, 13, 19, 66, 84], but there is no commonly agreed definition what explainability actually means. We use the following terminology, which is mostly aligned with the definitions proposed by Arrieta et al. [3].

- *Explainability* is the ability of an AI model to provide explanations through an interface to a human, where the explanation serves as a justification of the model output.
- *Interpretability* is the ability of an AI model to provide explanations in a way that is understandable by humans.
- *Transparency* is a property of an AI model that is by itself understandable, i.e., a human can understand how the model works by looking at the model structure and algorithm without further explanations.
- *Trustworthiness* is one of the goals of XAI and can be seen as the confidence in whether the model acts in an expected way.

Another challenge is that there exist no standard metrics for evaluating the explanations. Plausible criteria and metrics have been proposed [26, 67, 89], but the general usefulness and quality of an explanation are highly dependent on the background knowledge of the target group and the use case.

3.1 Explainability in Graph Machine Learning

Knowledge graphs model concepts as nodes and their connections as edges, which are inherently understandable by humans. Therefore, paths and subgraphs that are influential for the results are often highlighted to make the reasoning process more transparent. For example, the method GNNexplainer [111] is a model-agnostic approach that can generate explanations for any GNN by identifying relevant node features and subgraphs for the prediction. The selection of critical information from the graph is based on a mean-field variational approximation [11], which helps to maximize the mutual information between predictions and explanatory subgraphs. Further, a feature mask is learned that masks out irrelevant node features.

In an effort to make KG approaches not only explainable but also scalable and robust to noise, many neuro-symbolic methods have been introduced that combine logical reasoning with neural aspects to improve the interpretability [10].

The probabilistic Logic Neural Network (pLogicNet) [78] combines KG embeddings with MLNs and optimizes the architecture through a variational expectation-maximization (EM) algorithm [24]. In the E-step, a KG embedding model performs link prediction to infer the probabilities of missing edges in the graph. Then, in the M-step, the weights of the logical rules that are modeled by an MLN are updated based on all triples.

Badreddine et al. propose Logic Tensor Networks (LTNs) [7] and Real Logic, a fully differentiable fuzzy logical language that supports LTNs regarding learning and reasoning on KGs. Entities, relations, atomic formulas, and functions are represented by tensors or tensor operations. During the optimization of LTNs, logical formulas encoded with Real Logic are inserted into the loss function to integrate constraints and background knowledge.

Neural Theorem Provers (NTP) [81] are similar to LTNs and represent entities and relations as real-valued vectors. The symbolic unification operation is defined as a differentiable function that takes these vector representations as input. Link prediction queries correspond to basic theorems, which can then be proven by following a backward chaining algorithm.

3.2 Calibration in Graph Machine Learning

Besides offering explicit explanations for the predicted results, the outputs of many graph ML models are associated with a score, such as a probability value that a node has a specific label. This score is often interpreted as the model's confidence in its prediction. If these scores are not calibrated and thus do not reflect the ground-truth probability of the results, the user could be misled, and dangerous situations could arise if the user puts too much trust in these values.

Let the output of an ML model consist of the set of labels $\mathcal{C}$ where each label is associated with a score in the range $[0, 1]$ and all scores sum up to 1. For a node v, we call the label with the highest confidence score the predicted label $\hat{y}_v$, and the corresponding confidence score is denoted by $\hat{p}_v \in [0, 1]$. Let y_v be the ground-truth label of v. Then, *perfect calibration* for a node classification task is defined as follows [33]:

$$\text{Probability}(\hat{y}_v = y_v \mid \hat{p}_v = p) = p, \quad p \in [0, 1].$$

It means that the confidence score should reflect the true probability of the prediction. For example, if 100 nodes have a confidence score of 0.7 with respect to their predicted node labels, then exactly 70 nodes are expected to be correctly classified.

A commonly used metric to judge the calibration performance of a model is the *expected calibration error* (ECE). First, the node instances are divided into bins based on their confidence scores. Then, the difference between the average confidence score and the accuracy is calculated for each bin. The ECE is the weighted sum

$$\sum_{m=1}^{M} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|,$$

where $B_1, B_2, \cdots, B_M$ are the bins and N stands for the total number of instances.

Many calibration methods are post-processing methods, which modify the output scores after inference. Classical methods, which can be used to calibrate classification approaches and are not restricted to ML on graph-structured data, include, e.g., Platt scaling [77], temperature scaling [33], histogram binning [113], and isotonic regression [114]. In some cases, these approaches can also be applied effectively to GNNs [102], but depending of the dataset, there might be other methods specifically developed for GNNs that can achieve better performance [95].

Similarly to node classification, the scores associated with link prediction results are also often unreliable. A particular challenge is the open world assumption, i.e., there is no

ground truth for the correctness of a triple since KGs are often incomplete and a triple that does not exist in the graph is not necessarily false. Tabacof and Costabello [94] generate synthetic negatives by corrupting existing triples and combine them with Platt scaling or isotonic regression for the calibration of KG embeddings models. The experiments show that the combination with isotonic regression leads to best calibration. Safavi et al. [83] conduct a user study in which they provide annotators with calibrated probability values for the link prediction task. They find that users with calibrated scores give faster and more accurate answers than users who are only provided with ranked lists of object candidates.

Chapter 4

Neural Multi-Hop Reasoning with Logical Rules on Biomedical Knowledge Graphs

This chapter contains the publication

> **Yushan Liu**, Marcel Hildebrandt, Mitchell Joblin, Martin Ringsquandl, Rime Raissouni, and Volker Tresp. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In Ruben Verborgh et al. *The Semantic Web, ESWC 2021, Lecture Notes in Computer Science*, volume 12731, pages 375-391, Springer, Cham, 2021. DOI: 10.1007/978-3-030-77385-4_22

Neural Multi-hop Reasoning with Logical Rules on Biomedical Knowledge Graphs

Yushan Liu[1,3( )], Marcel Hildebrandt[1,3], Mitchell Joblin[1], Martin Ringsquandl[1], Rime Raissouni[2,3], and Volker Tresp[1,3]

[1] Siemens, Otto-Hahn-Ring 6, 81739 Munich, Germany
{yushan.liu,marcel.hildebrandt,mitchell.joblin,martin.ringsquandl, volker.tresp}@siemens.com
[2] Siemens Healthineers, Hartmanntraße 16, 91052 Erlangen, Germany
rime.raissouni@siemens-healthineers.com
[3] Ludwig Maximilian University of Munich, Geschwister-Scholl-Platz 1, 80539 Munich, Germany

Abstract. Biomedical knowledge graphs permit an integrative computational approach to reasoning about biological systems. The nature of biological data leads to a graph structure that differs from those typically encountered in benchmarking datasets. To understand the implications this may have on the performance of reasoning algorithms, we conduct an empirical study based on the real-world task of drug repurposing. We formulate this task as a link prediction problem where both compounds and diseases correspond to entities in a knowledge graph. To overcome apparent weaknesses of existing algorithms, we propose a new method, PoLo, that combines policy-guided walks based on reinforcement learning with logical rules. These rules are integrated into the algorithm by using a novel reward function. We apply our method to Hetionet, which integrates biomedical information from 29 prominent bioinformatics databases. Our experiments show that our approach outperforms several state-of-the-art methods for link prediction while providing interpretability.

Keywords: Neural multi-hop reasoning · Reinforcement learning · Logical rules · Biomedical knowledge graphs

1 Introduction

Advancements in low-cost high-throughput sequencing, data acquisition technologies, and compute paradigms have given rise to a massive proliferation of data describing biological systems. This new landscape of available data spans a multitude of dimensions, which provide complementary views on the structure of biological systems. Historically, by considering single dimensions (i. e., single types of data), researchers have made progress in understanding many important phenomena. More recently, there has been a movement to develop statistical and

R. Verborgh et al. (Eds.): ESWC 2021, LNCS 12731, pp. 375–391, 2021.
https://doi.org/10.1007/978-3-030-77385-4_22

computational methods that leverage more holistic views by simultaneously considering multiple types of data [40]. To achieve this goal, graph-based knowledge representation has emerged as a promising direction since the inherent flexibility of graphs makes them particularly well-suited for this problem setting.

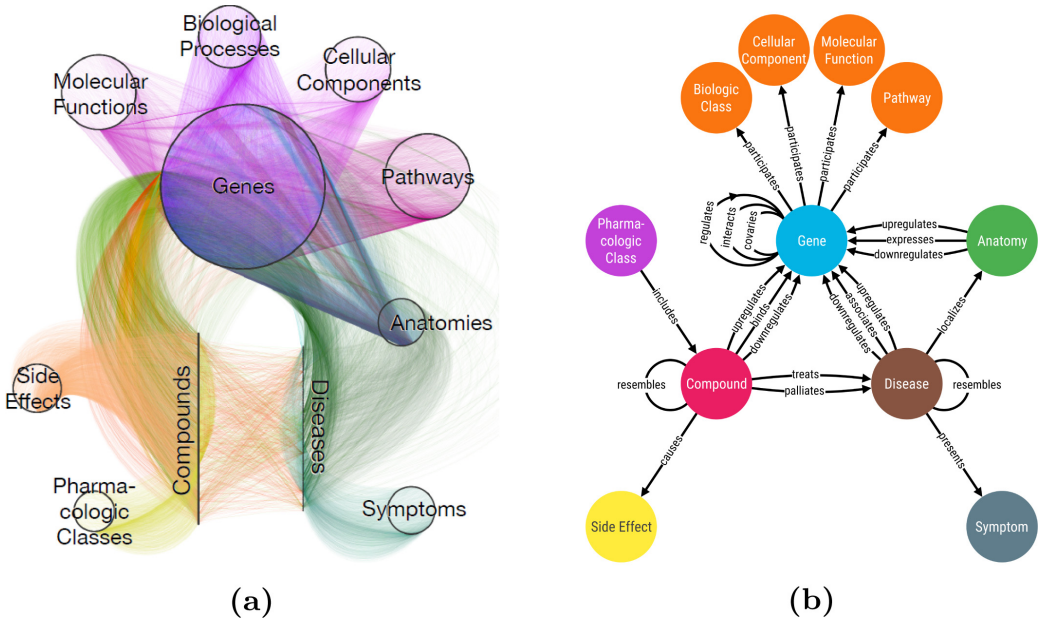


Fig. 1. (a) Visualization of the heterogeneous biomedical KG Hetionet [13] (reprint under the use of the CC BY 4.0 license). (b) Schema of Hetionet: Hetionet has 11 different entity types and 24 possible relations between them. Source: https://het.io/about/.

Biomedical knowledge graphs (KGs) are becoming increasingly popular for tasks such as personalized medicine, predictive diagnosis, and drug discovery [9]. Drug discovery, for example, requires a multitude of biomedical data types combined with knowledge across diverse domains (including gene-protein bindings, chemical compounds, and biological pathways). These individual types of data are typically scattered across disparate data sources, published for domain-specific research problems without considering mappings to open standards. To this end, KGs and Semantic Web technologies are being applied to model ontologies that combine knowledge and integrate data contained in biomedical data sources, most notably Bio2RDF [2], for classical query-based question answering.

From a machine learning perspective, reasoning on biomedical KGs presents new challenges for existing approaches due to the unique structural characteristics of the KGs. One challenge arises from the highly coupled nature of entities in biological systems that leads to many high-degree entities that are themselves densely linked. For example, as illustrated in Fig. 1a, genes interact abundantly among themselves. They are involved in a diverse set of biological pathways and molecular functions and have numerous associations with diseases.

A second challenge is that reasoning about the relationship between two entities often requires information beyond second-order neighborhoods [13].

Methods that rely on shallow node embeddings (e. g., TransE [4], DistMult [38]) typically do not perform well in this situation. Approaches that take the entire multi-hop neighborhoods into account (e. g., graph convolutional networks, R-GCN [30]) often have diminishing performance beyond two-hop neighborhoods (i. e., more than two convolutional layers), and the high-degree entities can cause the aggregation operations to smooth out the signal [16]. Symbolic approaches (e. g., AMIE+ [10], RuleN [21]) learn logical rules and employ them during inference. These methods might be able to take long-range dependencies into account, but due to the massive scale and diverse topologies of many real-world KGs, combinatorial complexity often prevents the usage of symbolic approaches [14]. Also, logical inference has difficulties handling noise in the data [24].

Under these structural conditions, path-based methods present a seemingly ideal balance for combining information over multi-hop neighborhoods. The key challenge is to find meaningful paths, which can be computationally difficult if the search is not guided by domain principles. Our goal is to explore how a path-based approach performs in comparison with alternative state-of-the art methods and to identify a way of overcoming weaknesses present in current approaches.

We consider the drug repurposing problem, which is characterized by finding novel treatment targets for existing drugs. Available knowledge about drug-disease-interactions can be exploited to reduce costs and time for developing new drugs significantly. A recent example is the repositioning of the drug remdesivir for the novel disease COVID-19. We formulate this task as a link prediction problem where both compounds and diseases correspond to entities in a KG.

We propose a neuro-symbolic reasoning approach, PoLo (**Po**licy-guided walks with **Lo**gical rules), that leverages both representation learning and logic. Inspired by existing methods [5,12,18], our approach uses reinforcement learning to train an agent to conduct policy-guided random walks on a KG. As a modification to approaches based on policy-guided walks, we introduce a novel reward function that allows the agent to use background knowledge formalized as logical rules, which guide the agent during training. The extracted paths by the agent act as explanations for the predictions. Our results demonstrate that existing methods are inadequately designed to perform ideally in the unique structural characteristics of biomedical data. We can overcome some of the weaknesses of existing methods and show the potential of neuro-symbolic methods for the biomedical domain, where interpretability and transparency of the results are highly relevant to facilitate the accessibility for domain experts. In summary, we make the following contributions:

- We propose the neuro-symbolic KG reasoning method PoLo that combines policy-guided walks based on reinforcement learning with logical rules.
- We conduct an empirical study using a large biomedical KG where we compare our approach with several state-of-the-art algorithms.
- The results show that our proposed approach outperforms state-of-the-art alternatives on a highly relevant biomedical prediction task (drug repurposing) with respect to the metrics hits@k for $k \in \{1, 3, 10\}$ and the mean reciprocal rank.

We briefly introduce the notation and review the related literature in Sect. 2. In Sect. 3, we describe our proposed method[1]. Section 4 details an experimental study, and we conclude in Sect. 5.

2 Background

2.1 Knowledge Graphs

Let $\mathcal{E}$ denote the set of entities in a KG and $\mathcal{R}$ the set of binary relations. Elements in $\mathcal{E}$ correspond to biomedical entities including, e. g., chemical compounds, diseases, and genes. We assume that every entity belongs to a unique type in $\mathcal{T}$, defined by the mapping $\tau : \mathcal{E} \to \mathcal{T}$. For example, $\tau(AURKC) = Gene$ indicates that the entity $AURKC$ has type $Gene$. Relations in $\mathcal{R}$ specify how entities are connected. We define a KG as a collection of triples $\mathcal{KG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ in the form (h, r, t), which consists of a head entity, a relation, and a tail entity. Head and tail of a triple are also called source and target, respectively. From a graphical point of view, head and tail entities correspond to nodes in the graph while the relation indicates the type of edge between them. For any relation $r \in \mathcal{R}$, we denote the corresponding inverse relation with r^{-1} (i. e., (h, r, t) is equivalent to (t, r^{-1}, h)). Triples in $\mathcal{KG}$ are interpreted as true known facts. For example, the triple $(Sorafenib, treats, Liver\ Cancer) \in \mathcal{KG}$ in Fig. 2 corresponds to the known fact that the kinase inhibitor drug sorafenib is approved for the treatment of primary liver cancer. The *treats* relation is of particular importance for this work since we frame the task of drug repurposing as a link prediction problem with respect to edges of the type *treats*. The domain of *treats* consists of chemical compounds, and the range is given by the set of all diseases.

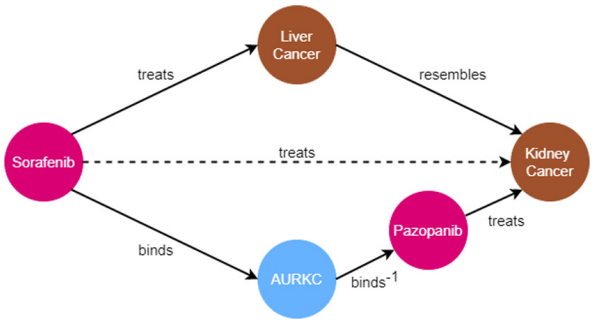


Fig. 2. Subgraph of Hetionet illustrating the drug repurposing use case. The two paths that connect the chemical compound sorafenib and the disease kidney cancer can be used to predict a direct edge between the two entities.

We further distinguish between two types of paths: instance paths and metapaths. An instance path of length $L \in \mathbb{N}$ on $\mathcal{KG}$ is given by a sequence

$$(e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \dots \xrightarrow{r_L} e_{L+1}) ,$$

where $(e_i, r_i, e_{i+1}) \in \mathcal{KG}$. We call the corresponding sequence of entity types

$$(\tau(e_1) \xrightarrow{r_1} \tau(e_2) \xrightarrow{r_2} \dots \xrightarrow{r_L} \tau(e_{L+1}))$$

[1] The source code is available at https://github.com/liu-yushan/PoLo.

a metapath. For example,

$$(Sorafenib \xrightarrow{treats} Liver\ Cancer \xrightarrow{resembles} Kidney\ Cancer)$$

constitutes an instance path of length 2, where

$$(Compound \xrightarrow{treats} Disease \xrightarrow{resembles} Disease)$$

is the corresponding metapath.

2.2 Logical Rules

Logical rules that are typically employed for KG reasoning can be written in the form *head* ← *body*. We consider cyclic rules of the form

$$(\tau_1, r_{L+1}, \tau_{L+1}) \leftarrow \bigwedge_{i=1}^{L} (\tau_i, r_i, \tau_{i+1}) \ ,$$

where $\tau_i \in \mathcal{T}$. The rule is called cyclic since the rule head (not to be confused with the head entity in a triple) connects the source τ_1 and the target τ_{L+1} of the metapath $(\tau_1 \xrightarrow{r_1} \tau_2 \xrightarrow{r_2} \ldots \xrightarrow{r_L} \tau_{L+1})$, which is described by the rule body. The goal is to find instance paths where the corresponding metapaths match the rule body in order to predict a new relation between the source and the target entity of the instance path. For the drug repurposing task, we only consider rules where the rule head is a triple with respect to the *treats* relation.
Define $CtD := (Compound, treats, Disease)$. Then, a generic rule has the form

$$CtD \leftarrow \left(Compound \xrightarrow{r_1} \tau_2 \xrightarrow{r_2} \tau_3 \xrightarrow{r_3} \ldots \xrightarrow{r_L} Disease\right) \ .$$

In particular, the rule body corresponds to a metapath starting at a compound and terminating at a disease. For example (see Fig. 2), consider the rule

$$CtD \leftarrow (Compound \xrightarrow{binds} Gene \xrightarrow{binds^{-1}} Compound \xrightarrow{treats} Disease) \ .$$

The metapath of the instance path

$$(Sorafenib \xrightarrow{binds} AURKC \xrightarrow{binds^{-1}} Pazopanib \xrightarrow{treats} Kidney\ Cancer)$$

matches the rule body, suggesting that sorafenib can also treat kidney cancer.

2.3 Related Work

Even though real-world KGs contain a massive number of triples, they are still expected to suffer from incompleteness. Therefore, link prediction (also known as KG completion) is a common reasoning task on KGs. Many classical artificial intelligence tasks such as recommendation problems or question answering can be rephrased in terms of link prediction.

Symbolic approaches have a far-reaching tradition in the context of knowledge acquisition and reasoning. Reasoning with logical rules has been addressed in areas such as Markov logic networks (MLNs) [28] or inductive logic programming [25]. However, such techniques typically do not scale well to modern, large-scale KGs. Recently, novel methods such as RuleN [21] and its successor AnyBURL [19,20] have been proposed that achieve state-of-the-art performance on popular benchmark datasets such as FB15k-237 [31] and WN18RR [7].

Subsymbolic approaches map nodes and edges in KGs to low-dimensional vector representations known as embeddings. Then, the likelihood of missing triples is approximated by a classifier that operates on the embedding space. Popular embedding-based methods include translational methods like TransE [4], more generalized approaches such as DistMult[38] and ComplEx [32], multi-layer models like ConvE [7], and tensor factorization methods like RESCAL [26]. Moreover, R-GCN [30] and CompGCN [34] have been proposed, which extend graph convolutional networks [16] to multi-relational graphs. Despite achieving good results on the link prediction task, a fundamental problem is their non-transparent nature since it remains hidden to the user what contributed to the predictions. Moreover, most embedding-based methods have difficulties in capturing long-range dependencies since they only minimize the reconstruction error in the immediate first-order neighborhoods. Especially the expressiveness of long-tail entities might be low due to the small number of neighbors [11].

Neuro-symbolic methods combine the advantages of robust learning and scalability in subsymbolic approaches with the reasoning properties and interpretability of symbolic representation. For example, Neural LP [39] and Neural Theorem Provers (NTPs) [29] integrate logical rules in a differentiable way into a neural network architecture. The method pLogicNet [27] combines MLNs with embedding-based models and learns a joint distribution over triples, while the Logic Tensor Network [8] inserts background knowledge into neural networks in the form of logical constraints. However, many neuro-symbolic approaches suffer from limited transferability and computational inefficiency. Minervini et al. have presented two more scalable extensions of NTPs, namely the Greedy NTP (GNTP) [22], which considers the top-k rules that are most likely to prove the goal instead of using a fixed set of rules, and the Conditional Theorem Prover (CTP) [23], which learns an adaptive strategy for selecting the rules.

Multi-hop reasoning or path-based approaches infer missing knowledge based on using extracted paths from the KG as features for various inference tasks. Along with a prediction, multi-hop reasoning methods provide the user with an explicit reasoning chain that may serve as a justification for the prediction. For example, the Path Ranking Algorithm (PRA) [17] frames the link prediction task as a maximum likelihood classification based on paths sampled from nearest neighbor random walks on the KG. Xiong et al. extend this idea and formulate the task of path extraction as a reinforcement learning problem (DeepPath [37]). Our proposed method is an extension of the path-based approach MINERVA [5], which trains a reinforcement learning agent to perform a policy-guided random walk until the answer entity to an input query is reached.

One of the drawbacks of existing policy-guided walk methods is that the agent might receive noisy reward signals based on spurious triples that lead to the correct answers during training but lower the generalization capabilities. Moreover, biomedical KGs often exhibit both long-range dependencies and high-degree nodes (see Sect. 4.1). These two properties and the fact that MINERVA's agent only receives a reward if the answer entity is correct make it difficult for the agent to navigate over biomedical KGs and extend a path in the most promising way. As a remedy, we propose the incorporation of known, effective logical rules via a novel reward function. This can help to denoise the reward signal and guide the agent on long paths with high-degree nodes.

3 Our Method

We pose the task of drug repurposing as a link prediction problem based on graph traversal. The general Markov decision process definition that we use has initially been proposed in the algorithm MINERVA [5], with our primary contribution coming from the incorporation of logical rules into the training process. The following notation and definitions are adapted to the use case. Starting at a query entity (a compound to be repurposed), an agent performs a walk on the graph by sequentially transitioning to a neighboring node. The decision of which transition to make is determined by a stochastic policy. Each subsequent transition is added to the current path and extends the reasoning chain. The stochastic walk process iterates until a finite number of transitions has been made. Formally, the learning task is modeled via the fixed-horizon Markov decision process outlined below.

Environment. The state space $\mathcal{S}$ is given by $\mathcal{E}^3$. Intuitively, we want the state to encode the location e_l of the agent for step $l \in \mathbb{N}$, the source entity e_c, and the target entity e_d, corresponding to the compound that we aim to repurpose and the target disease, respectively. Thus, a state $S_l \in \mathcal{S}$ for step $l \in \mathbb{N}$ is represented by $S_l := (e_l, e_c, e_d)$. The agent is given no information about the target disease so that the observed part of the state space is given by $(e_l, e_c) \in \mathcal{E}^2$. The set of available actions from a state S_l is denoted by $\mathcal{A}_{S_l}$. It contains all outgoing edges from the node e_l and the corresponding tail nodes. We also include self-loops for each node so that the agent has the possibility to stay at the current node. More formally, $\mathcal{A}_{S_l} := \{(r, e) \in \mathcal{R} \times \mathcal{E} : (e_l, r, e) \in \mathcal{KG}\} \cup \{(\emptyset, e_l)\}$. Further, we denote with $A_l \in \mathcal{A}_{S_l}$ the action that the agent performed in step l. The environment evolves deterministically by updating the state according to the previous action. The transition function is given by $\delta(S_l, A_l) := (e_{l+1}, e_c, e_d)$ with $S_l = (e_l, e_c, e_d)$ and $A_l = (r_l, e_{l+1})$.

Policy. We denote the history of the agent up to step l with $H_l := (H_{l-1}, A_{l-1})$ for $l \geq 1$, with $H_0 := e_c$ and $A_0 := \emptyset$. The agent encodes the transition history via an LSTM [15] by

$$\boldsymbol{h}_l = \mathrm{LSTM}\left(\boldsymbol{h}_{l-1}, \boldsymbol{a}_{l-1}\right) , \tag{1}$$

where $\boldsymbol{a}_{l-1} := [\boldsymbol{r}_{l-1}; \boldsymbol{e}_l] \in \mathbb{R}^{2d}$ corresponds to the vector space embedding of the previous action (or the zero vector for $\boldsymbol{a}_0$), with $\boldsymbol{r}_{l-1}$ and $\boldsymbol{e}_l$ denoting the embeddings of the relation and the tail entity in $\mathbb{R}^d$, respectively. The history-dependent action distribution is given by

$$\boldsymbol{d}_l = \text{softmax}\left(\boldsymbol{A}_l \left(\boldsymbol{W}_2 \text{ReLU}\left(\boldsymbol{W}_1 \left[\boldsymbol{h}_l; \boldsymbol{e}_l\right]\right)\right)\right) , \tag{2}$$

where the rows of $\boldsymbol{A}_l \in \mathbb{R}^{|\mathcal{A}_{S_l}| \times 2d}$ contain the latent representations of all admissible actions from S_l. The matrices $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ are learnable weight matrices. The action $A_l \in \mathcal{A}_{S_l}$ is drawn according to

$$A_l \sim \text{Categorical}\left(\boldsymbol{d}_l\right) . \tag{3}$$

The Eqs. (1)–(3) are repeated for each transition step. In total, L transitions are sampled, where L is a hyperparameter that determines the maximum path length, resulting in a path denoted by

$$P := (e_c \xrightarrow{r_1} e_2 \xrightarrow{r_2} \ldots \xrightarrow{r_L} e_{L+1}) .$$

For each step $l \in \{1, 2, \ldots, L\}$, the agent can also choose to remain at the current location and not extend the reasoning path.

Equations (1) and (2) define a mapping from the space of histories to the space of distributions over all admissible actions. Thus, including Eq. (3), a stochastic policy π_θ is induced, where θ denotes the set of all trainable parameters in Eq. (1) and (2).

Metapaths. Consider the set of metapaths $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$, where each element corresponds to the body of a cyclic rule with *CtD* as rule head. For every metapath M, we denote with $s(M) \in \mathbb{R}_{>0}$ a score that indicates a quality measure of the corresponding rule, such as the confidence or the support with respect to making a correct prediction. Moreover, for a path P, we denote with $\tilde{P}$ the corresponding metapath.

Rewards and Optimization. During training, after the agent has reached its final location, a terminal reward is assigned according to

$$R(S_{L+1}) = \mathbb{1}_{\{e_{L+1}=e_d\}} + b\lambda \sum_{i=1}^{m} s(M_i) \mathbb{1}_{\tilde{P}=M_i} . \tag{4}$$

The first term indicates whether the agent has reached the correct target disease that can be treated by the compound e_c. It means that the agent receives a reward of 1 for a correct prediction. The second term indicates whether the extracted metapath corresponds to the body of a rule and adds to the reward accordingly. The hyperparameter b can either be 1, i. e., the reward is always increased as long as the metapath corresponds to the body of a rule, or b can be set to $\mathbb{1}_{\{e_{L+1}=e_d\}}$, i. e., an additional reward is only applied if the prediction is

also correct. Heuristically speaking, we want to reward the agent for extracting a metapath that corresponds to a rule body with a high score. The hyperparameter $\lambda \geq 0$ balances the two components of the reward. For $\lambda = 0$, we recover the algorithm MINERVA.

We employ REINFORCE [35] to maximize the expected rewards. Thus, the agent's maximization problem is given by

$$\arg\max_{\theta} \mathbb{E}_{(e_c, treats, e_d)\sim\mathcal{D}} \mathbb{E}_{A_1, A_2, \ldots, A_L \sim \pi_\theta} \left[R(S_{L+1}) \,\middle|\, e_c, e_d \right] , \tag{5}$$

where $\mathcal{D}$ denotes the true underlying distribution of $(e_c, \textit{treats}, e_d)$-triples. During training, we replace the first expectation in Eq. (5) with the empirical average over the training set. The second expectation is approximated by averaging over multiple rollouts for each training sample.

4 Experiments

4.1 Dataset Hetionet

Hetionet [13] is a biomedical KG that integrates information from 29 highly reputable and cited public databases, including the Unified Medical Language System (UMLS) [3], Gene Ontology [1], and DrugBank [36]. It consists of 47,031 entities with 11 different types and 2,250,197 edges with 24 different types. Figure 1b illustrates the schema and shows the different types of entities and possible relations between them.

Hetionet differs in many aspects from the standard benchmark datasets that are typically used in the KG reasoning literature. Table 1 summarizes the basic statistics of Hetionet along with the popular benchmark datasets FB15k-237 [31] and WN18RR [7]. One of the major differences between Hetionet and the two other benchmark datasets is the density of triples, i. e., the average node degree in Hetionet is significantly higher than in the other two KGs. Entities of type *Anatomy* are densely connected hub nodes, and in addition, entities of type *Gene* have an average degree of around 123. This plays a crucial role for our application since many relevant paths that connect *Compound* and *Disease* traverse entities of type *Gene* (see Fig. 1b and Table 2). The total counts and the average node degrees according to each entity type are shown in Appendix A. We will discuss in Sect. 4.5 further how particularities of Hetionet impose challenges on existing KG reasoning methods.

We aim to predict edges with type *treats* between entities that correspond to compounds and diseases in order to perform candidate ranking according to the likelihood of successful drug repurposing in a novel treatment application. There are 1552 compounds and 137 diseases in Hetionet with 775 observed links of type *treats* between compounds and diseases. We randomly split these 755 triples into training, validation, and test set, where the training set contains 483 triples, the validation set 121 triples, and the test set 151 triples.

4.2 Metapaths as Background Information

Himmelstein et al. [13] evaluated 1206 metapaths that connect entities of type *Compound* with entities of type *Disease*, which correspond to various pharmacological efficacy mechanisms. They identified 27 effective metapaths that served as features for a logistic regression model that outputs a treatment probability of a compound for a disease. Out of these metapaths, we select the 10 metapaths as background information that have at most path length 3 and exhibit positive regression coefficients, which indicates their importance for predicting drug efficacy. We use the metapaths as the rule bodies and the confidence of the rules as the quality scores (see Sect. 3). The confidence of a rule is defined as the rule support divided by the body support in the data. We estimate the confidence score for each rule by sampling 5,000 paths whose metapaths correspond to the rule body and then computing how often the rule head holds. An overview of the 10 metapaths and their scores is given in Table 2.

Table 1. Comparison of Hetionet with the two benchmark datasets FB15k-237 and WN18RR.

Dataset	Entities	Relations	Triples	Avg. degree
Hetionet	47,031	24	2,250,197	95.8
FB15k-237	14,541	237	310,116	19.7
WN18RR	40,943	11	93,003	2.2

Table 2. All 10 metapaths used in our model and their corresponding scores.

$s(M)$	Metapath M
0.446	$(Compound \xrightarrow{includes^{-1}} Pharmacologic\ Class \xrightarrow{includes} Compound \xrightarrow{treats} Disease)$
0.265	$(Compound \xrightarrow{resembles} Compound \xrightarrow{resembles} Compound \xrightarrow{treats} Disease)$
0.184	$(Compound \xrightarrow{binds} Gene \xrightarrow{associates^{-1}} Disease)$
0.182	$(Compound \xrightarrow{resembles} Compound \xrightarrow{treats} Disease)$
0.169	$(Compound \xrightarrow{palliates} Disease \xrightarrow{palliates^{-1}} Compound \xrightarrow{treats} Disease)$
0.143	$(Compound \xrightarrow{binds} Gene \xrightarrow{binds^{-1}} Compound \xrightarrow{treats} Disease)$
0.058	$(Compound \xrightarrow{causes} Side\ Effect \xrightarrow{causes^{-1}} Compound \xrightarrow{treats} Disease)$
0.040	$(Compound \xrightarrow{treats} Disease \xrightarrow{resembles} Disease)$
0.017	$(Compound \xrightarrow{resembles} Compound \xrightarrow{binds} Gene \xrightarrow{associates^{-1}} Disease)$
0.004	$(Compound \xrightarrow{binds} Gene \xrightarrow{expresses^{-1}} Anatomy \xrightarrow{localizes^{-1}} Disease)$

4.3 Experimental Setup

We apply our method PoLo to Hetionet and calculate the values for hits@1, hits@3, hits@10, and the mean reciprocal rank (MRR) for the link prediction task. All metrics in the paper are filtered [4] and evaluated for tail-sided predictions. During inference, a beam search is carried out to find the most promising paths, and the target entities are ranked by the probability of their corresponding paths. Moreover, we consider another evaluation scheme (PoLo (pruned)) that retrieves and ranks only those paths from the test rollouts that correspond to one of the metapaths in Table 2. All the other extracted paths are not considered in the ranking.

We compare PoLo with the following baseline methods. The rule-based method AnyBURL [19,20] mines logical rules based on path sampling and uses them for inference. The methods TransE [4], DistMult [38], ComplEx [32], ConvE [6], and RESCAL [26] are popular embedding-based models, and we use the implementation from the LibKGE library[2]. To cover a more recent paradigm in graph-based machine learning, we include the graph convolutional approaches R-GCN [30] and CompGCN [34]. We also compare our method with the neuro-symbolic method pLogicNet [27]. The two neuro-symbolic approaches NTP [29] and Neural LP [39] yield good performance on smaller datasets but are not scalable to large datasets like Hetionet. We have also conducted experiments on the two more scalable extensions of NTP (GNTP [22] and CTP [23]), but both were not able to produce results in a reasonable time. More experimental details can be found in Appendix B.

4.4 Results

Table 3 displays the results for the experiments on Hetionet. The reported values for PoLo and MINERVA correspond to the mean across five independent training runs. The standard errors for the reported metrics are between 0.006 and 0.018.

PoLo outperforms all baseline methods with respect to all evaluation metrics. Applying the modified ranking scheme, our method yields performance gains of 27.7% for hits@1, 14.9% for hits@3, 8.1% for hits@10, and 16.2% for the MRR with respect to best performing baseline.

Figure 3a shows the rule accuracy, i. e., the percentage of correct target entities for extracted paths that follow rule metapaths, for PoLo and MINERVA during training. Both lines behave similarly in the beginning, but the rule accuracy of PoLo increases significantly around epoch 20 compared to MINERVA. It seems that giving the agent an extra reward for extracting rules also improves the probability of arriving at correct target entities when applying the rules. We also compare the metric hits@1 (pruned) for the evaluation of the validation set during training (see Fig. 3b). Around epoch 20, where the rule accuracy of PoLo increases compared to MINERVA, hits@1 (pruned) also increases while it decreases for MINERVA. The additional reward for extracting rule paths could

[2] https://github.com/uma-pi1/kge.

Table 3. Comparison with baseline methods on Hetionet.

Method	Hits@1	Hits@3	Hits@10	MRR
AnyBURL	0.229	0.375	0.553	0.322
TransE	0.099	0.199	0.444	0.205
DistMult	0.185	0.305	0.510	0.287
ComplEx	0.152	0.285	0.470	0.250
ConvE	0.100	0.225	0.318	0.180
RESCAL	0.106	0.166	0.377	0.187
R-GCN	0.026	0.245	0.272	0.135
CompGCN	0.172	0.318	0.543	0.292
pLogicNet	0.225	0.364	0.523	0.333
MINERVA	0.264	0.409	0.593	0.370
PoLo	0.314	0.428	0.609	0.402
PoLo (pruned)	**0.337**	**0.470**	**0.641**	**0.430**

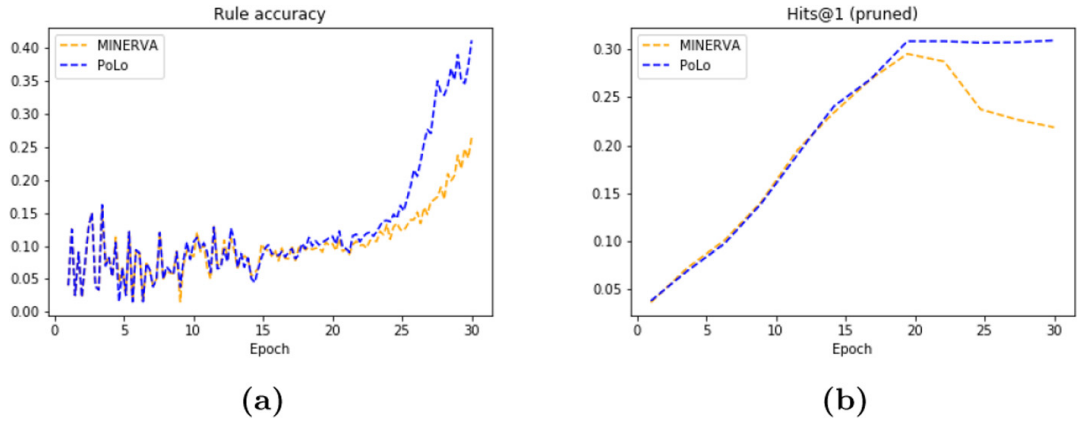


Fig. 3. (a) Rule accuracy during training. (b) Hits@1 (pruned) for the evaluation of the validation set during training.

be seen as a regularization that alleviates overfitting and allows for longer training for improved results.

The metapath that was most frequently extracted by PoLo during testing is

$$(Compound \xrightarrow{causes} Side\ Effect \xrightarrow{causes^{-1}} Compound \xrightarrow{treats} Disease)\ .$$

This rule was followed in 37.3% of the paths during testing, of which 16.9% ended at the correct entity.

During testing, PoLo extracted metapaths that correspond to rules in 41.7% of all rollouts while MINERVA only extracted rule paths in 36.9% of the cases. The accuracy of the rules, i. e., the percentage of correct target entities when rule paths are followed, is 19.0% for PoLo and 17.6% for MINERVA.

4.5 Discussion

We have integrated logical rules as background information via a new reward mechanism into the multi-hop reasoning method MINERVA. The stochastic policy incorporates the set of rules that are presented to the agent during training. Our approach is not limited to MINERVA but can act as a generic mechanism to inject domain knowledge into reinforcement learning-based reasoning methods on KGs [18,37]. While we employ rules that are extracted in a data-driven fashion, our method is agnostic towards the source of background information.

The additional reward for extracting a rule path can be considered as a regularization that induces the agent to walk along metapaths that generalize to unseen instances. In particular, for PoLo (pruned), we consider only extracted paths that correspond to the logical rules. However, the resulting ranking of the answer candidates is not based on global quality measures of the rules (e. g., the confidence). Rather, the ranking is given by the policy of the agent (i. e., metapaths that are more likely to be extracted are ranked higher), which creates an adaptive reweighting of the extracted rules that takes the individual instance paths into account.

Multi-hop reasoning methods contain a natural transparency mechanism by providing explicit inference paths. These paths allow domain experts to evaluate and monitor the predictions. Typically, there is an inherent trade-off between explainability and performance, but surprisingly, our experimental findings show that path-based reasoning methods outperform existing black-box methods on the drug repurposing task. Concretely, we compared our approach with the embedding-based methods TransE, DistMult, ComplEx, ConvE, and RESCAL. These methods are trained to minimize the reconstruction error in the immediate first-order neighborhood while discarding higher-order proximities. However, most explanatory metapaths in the drug repurposing setting have length 2 or more [13]. While MINERVA and PoLo can explicitly reason over multiple hops, our results indicate that embedding-based methods that fit low-order proximities seem not to be suitable for the drug repurposing task, and it is plausible that other reasoning tasks on biomedical KGs could result in similar outcomes.

R-CGN and CompGCN learn node embeddings by aggregating incoming messages from neighboring nodes and combining this information with the node's own embedding. These methods are in principle capable of modeling long-term dependencies. Since the receptive field contains the entire set of nodes in the multi-hop neighborhood, the aggregation and combination step essentially acts as a low-pass filter on the incoming signals. This can be problematic in the presence of many high-degree nodes like in Hetionet where the center node receives an uninformative signal that smooths over the neighborhood embeddings.

The approaches pLogicNet and AnyBURL both involve the learning of rules and yield similar performance on Hetionet, which is worse than PoLo. Most likely, the large amount of high-degree nodes in Hetionet makes the learning and application of logical rules more difficult. Other neuro-symbolic methods such as NTP, its extensions, and Neural LP were not scalable to Hetionet.

To illustrate the applicability of our method, consider the example of the chemical compound sorafenib (see Fig. 2), which is known for treating liver cancer, kidney cancer, and thyroid cancer. The top predictions of our model for new target diseases include pancreatic cancer, breast cancer, and hematologic cancer. This result seems to be sensible since sorafenib already treats three other cancer types. The database ClinicalTrials.gov [33] lists 16 clinical studies for testing the effect of sorafenib on pancreatic cancer, 33 studies on breast cancer, and 6 studies on hematologic cancer, showing that the predicted diseases are meaningful targets for further investigation. Another example of drug repurposing on Hetionet is provided in Appendix C.

4.6 Experiments on Other Datasets

We also conduct experiments on the benchmark datasets FB15k-237 and WN18RR and compare PoLo with the other baseline methods. Since we do not already have logical rules available, we use the rules learned by AnyBURL. We can only apply cyclic rules for PoLo, so we also compare to the setting where we only learn and apply cyclic rules with AnyBURL.

Our method mostly outperforms MINERVA and Neural LP on both datasets. For FB15k-237, PoLo has worse performance than AnyBURL and most embedding-based methods, probably because the number of unique metapaths that occur a large number of times in the graph is lower compared to other datasets [5]. This makes it difficult for PoLo to extract metapaths sufficiently often for good generalization. pLogicNet yields better performance on FB15k-237 than PoLo but worse performance on WN18RR. The results of AnyBURL on FB15k-237 and WN18RR when only using cyclic rules are worse than when also including acyclic rules. It seems that acyclic rules are important for predictions as well, but PoLo cannot make use of these rules. The detailed results for both datasets can be found in Appendix D.

5 Conclusion

Biomedical knowledge graphs present challenges for learning algorithms that are not reflected in the common benchmark datasets. Our experimental findings suggest that existing knowledge graph reasoning methods face difficulties on Hetionet, a biomedical knowledge graph that exhibits both long-range dependencies and a multitude of high-degree nodes. We have proposed the neuro-symbolic approach PoLo that leverages both representation learning and logic. Concretely, we integrate logical rules into a multi-hop reasoning method based on reinforcement learning via a novel reward mechanism. We apply our method to the highly relevant task of drug repurposing and compare our approach with embedding-based, logic-based, and neuro-symbolic methods. The results indicate a better performance of PoLo compared to popular state-of-the-art methods. Further, PoLo also provides interpretability by extracting reasoning paths that serve as explanations for the predictions.

Acknowledgements.. This work has been supported by the German Federal Ministry for Economic Affairs and Energy (BMWi) as part of the project RAKI (01MD19012C).

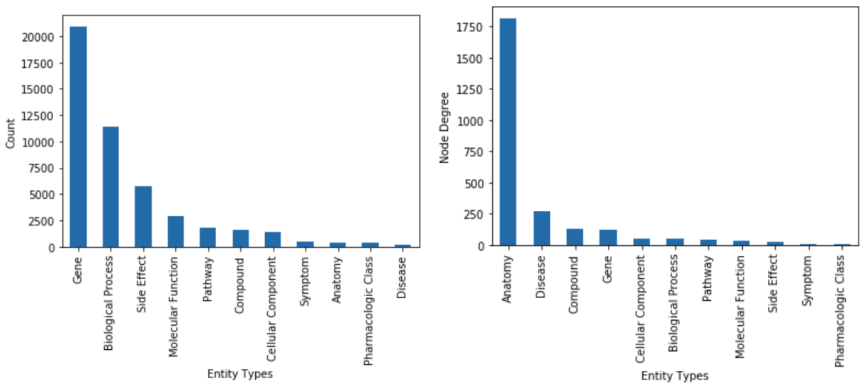
References

1. Ashburner, M., et al.: Gene Ontology: tool for the unification of biology. Nat. Genet. **25**(1), 25–29 (2000)
2. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. J. Biomed. Inform. **41**(5), 706–716 (2008)
3. Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. Nucleic Acids Res. **32**(Database), D267–D270 (2004)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: The 27th Conference on Neural Information Processing Systems (2013)
5. Das, R., et al.: Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In: The 6th International Conference on Learning Representations (2018)
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: The 13th Conference on Neural Information Processing Systems (2016)
7. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: The 32nd AAAI Conference on Artificial Intelligence (2018)
8. Donadello, I., Serafini, L., Garcez, A.: Logic tensor networks for semantic image interpretation. In: The 26th International Joint Conference on Artificial Intelligence (2017)
9. Dörpinghaus, J., Jacobs, M.: Semantic knowledge graph embeddings for biomedical research: data integration using linked open data. In: SEMANTiCS (2019)
10. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. **24**, 707–730 (2015)
11. Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. In: The 36th International Conference on Machine Learning (2019)
12. Hildebrandt, M., Serna, J.A.Q., Ma, Y., Ringsquandl, M., Joblin, M., Tresp, V.: Reasoning on knowledge graphs with debate dynamics. In: The 34th AAAI Conference on Artificial Intelligence (2020)
13. Himmelstein, D.S., et al.: Systematic integration of biomedical knowledge prioritizes drugs for repurposing. Elife **6**, e26726 (2017)
14. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC Textbooks in Computing (2009)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: The 5th International Conference on Learning Representations (2017)
17. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
18. Lin, X.V., Socher, R., Xiong, C.: Multi-hop knowledge graph reasoning with reward shaping. In: The 2018 Conference on Empirical Methods in Natural Language Processing (2018)

19. Meilicke, C., Chekol, M.W., Fink, M., Stuckenschmidt, H.: Reinforced anytime bottom up rule learning for knowledge graph completion. Preprint arXiv:2004.04412 (2020)
20. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime bottom-up rule learning for knowledge graph completion. In: The 28th International Joint Conference on Artificial Intelligence (2019)
21. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In: The 17th International Semantic Web Conference (2018)
22. Minervini, P., Bošnjak, M., Rocktäschel, T., Riedel, S., Grefenstette, E.: Differentiable reasoning on large knowledge bases and natural language. In: The 34th AAAI Conference on Artificial Intelligence (2020)
23. Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., Rocktäschel, T.: Learning reasoning strategies in end-to-end differentiable proving. In: The 37th International Conference on Machine Learning (2020)
24. Mitchell, T.: Machine Learning. McGraw-Hill Series in Computer Science (1997)
25. Muggleton, S.: Inductive logic programming. N. Gener. Comput. **8**(4), 295–318 (1991)
26. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: The 28th International Conference on Machine Learning (2011)
27. Qu, M., Tang, J.: Probabilistic logic neural networks for reasoning. In: The 33rd Conference on Neural Information Processing Systems (2019)
28. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**(1–2), 107–136 (2006)
29. Rocktäschel, T., Riedel, S.: End-to-end differentiable proving. In: The 31st Conference on Neural Information Processing Systems (2017)
30. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The 15th Extended Semantic Web Conference (2018)
31. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: The 2015 Conference on Empirical Methods in Natural Language Processing (2015)
32. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: The 33rd International Conference on Machine Learning (2016)
33. U. S. National Library of Medicine (2000). clinicaltrials.gov
34. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.: Composition-based multi-relational graph convolutional networks. In: The 9th International Conference on Learning Representations (2020)
35. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3–4), 229–256 (1992)
36. Wishart, D.S., et al.: DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res. **34**(Database), D668–D672 (2006)
37. Xiong, W., Hoang, T., Wang, W.Y.: DeepPath: a reinforcement learning method for knowledge graph reasoning. In: The 2017 Conference on Empirical Methods in Natural Language Processing (2017)
38. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: The 3rd International Conference on Learning Representations (2015)

39. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: The 31st Conference on Neural Information Processing Systems (2017)
40. Zitnik, M., Nguyen, F., Wang, B., Leskovec, J., Goldenberg, A., Hoffman, M.M.: Machine learning for integrating data in biology and medicine: principles, practice, and opportunities. Inf. Fusion **50**, 71–91 (2019)

A Dataset Hetionet



(a) **(b)**

Fig. A1. Comparison of the (a) total counts and (b) the average node degrees according to each entity type in Hetionet.

B Experimental Details

B.1 PoLo and MINERVA

For PoLo and MINERVA, we tune the model hyperparameters λ which balances the two rewards, b which defines when the additional reward is added, β which is the entropy regularization constant, the entity and relation embedding size, the number of LSTM layers, the hidden layer size of the LSTM, and the learning rate. The hyperparameter search spaces are shown in Table B1. The path length is set to 3, the number of rollouts is given by 30 for training, and the number of test rollouts at inference time is set to 100. All experiments were conducted on a machine with 16 CPU cores and 32 GB RAM. Training our method on the Hetionet dataset takes around 10 minutes, and testing takes around 10 seconds.

B.2 LibKGE

LibKGE[1] is a library for training, evaluation, and hyperparameter optimization of knowledge graph embedding models. We do a random search for the models TransE, DistMult, ComplEx, ConvE, and RESCAL across the hyperparameter search spaces given in Table B2.

[1] `https://github.com/uma-pi1/kge`

Table B1. Hyperparameter search spaces for PoLo and MINERVA.

Hyperparameter	Search space
Embedding size	{64, 128, 256}
Number of LSTM layers	{1, 2}
Hidden layer size	{128, 256, 512}
Learning rate	[0.0001, 0.01]
λ	$\{1, 2, \ldots, 5\}$
b	$\{1, \mathbb{1}_{\{e_{L+1}=e_d\}}\}$
β	[0.01, 0.1]

Table B2. Hyperparameter search spaces for LibKGE. Models with specific hyperparameters are indicated in parentheses.

Hyperparameter	Search space
Embedding size	$\{64, 128, 256\}$
Training type	Negative sampling
Reciprocal	{True, False}
Number of subject samples	$\{1, 2, \ldots, 1000\}$, log scale
Number of object samples	$\{1, 2, \ldots, 1000\}$, log scale
Loss	Cross-entropy
L^p-norm (TransE)	{1, 2}
Optimizer	{Adam, Adagrad}
Learning rate	$[10^{-4}, 1]$, log scale
Learning rate scheduler patience	$\{1, 2, \ldots, 10\}$
L^p regularization	{None, L^2}
Weight of entity embeddings	$[10^{-20}, 10^{-5}]$
Weight of relation embeddings	$[10^{-20}, 10^{-5}]$
Frequency weighting	{True, False}
Embedding normalization (TransE)	
Entity	{True, False}
Relation	{True, False}
Dropout	
Entity embedding	[0.0, 0.5]
Relation embedding	[0.0, 0.5]
Feature map (ConvE)	[0.0, 0.5]
Projection (ConvE)	[0.0, 0.5]

B.3 AnyBURL

We use the implementation AnyBURL-RE[2] by the authors. The rules are learned for 500 seconds, and the maximum rule length is set to 3. Since AnyBURL does

[2] `http://web.informatik.uni-mannheim.de/AnyBURL/`

not need extensive hyperparameter tuning to achieve good results, we kept the default values of all other hyperparameters.

B.4 R-GCN

We use the implementation provided by the Python library PyG[3]. The hyperparameters are tuned according to the search spaces specified in Table B3.

Table B3. Hyperparameter search spaces for R-GCN.

Hyperparameter	Search space
Embedding size	$\{8, 16, 32\}$
Number of convolutional layers	$\{1, 2, 3\}$
Learning rate	$[0.001, 1]$

B.5 CompGCN

For CompGCN, we use the implementation[4] by the authors. The hyperparameters of CompGCN are tuned according to the search spaces specified in Table B4.

Table B4. Hyperparameter search spaces for CompGCN.

Hyperparameter	Search space
Embedding size	{64, 128, 256}
GCN layer size	{64, 128, 256}
Number of GCN layers	{1, 2}
Scoring function	{TransE, ConvE, DistMult}
Composition operation	Circular-correlation
Dropout rate	0.1
Learning rate	0.001

B.6 pLogicNet

For the experiments on pLogicNet, we use the implementation[5] by the authors and report the results of pLogicNet*, which yields better results than plain pLogicNet. Table B5 shows the hyperparameter search spaces.

[3] `https://github.com/rusty1s/pytorch_geometric`
[4] `https://github.com/malllabiisc/CompGCN`
[5] `https://github.com/DeepGraphLearning/pLogicNet`

Table B5. Hyperparameter search spaces for pLogicNet.

Hyperparameter	Search space
Embedding size	{500, 1000}
Number of negative samples	{256, 512}
α	{0.5, 1}
γ	{6, 12}
λ	{1, 50, 100}
τ_{rule}	{0.1, 0.6}

C Example of Drug Repurposing on Hetionet

The compound ergocalciferol is a type of vitamin D, also called vitamin D_2. It can be found in food and is used to treat rheumatoid athritis and osteoporosis. The disease predictions of our model include hypertension, ulcerative colitis, and psoriasis. The database ClinicalTrials.gov lists 10 clinical studies for testing the effect of ergocalciferol on hypertension, 2 studies on ulcerative colitis, and 2 studies on psoriasis. The following extracted paths from our model support these predictions.

($Ergocalciferol \xrightarrow{resembles} Dihydrotachysterol \xrightarrow{resembles} Cholecalciferol \xrightarrow{treats} Ulcerative\ Colitis$)

($Ergocalciferol \xrightarrow{resembles} Calcidiol \xrightarrow{resembles} Cholecalciferol \xrightarrow{treats} Ulcerative\ Colitis$)

($Ergocalciferol \xrightarrow{causes} Irreversible\ Renal\ Failure \xrightarrow{causes^{-1}} Captopril \xrightarrow{treats} Hypertension$)

($Ergocalciferol \xrightarrow{causes} Polydipsia \xrightarrow{causes^{-1}} Fosinopril \xrightarrow{treats} Hypertension$)

($Ergocalciferol \xrightarrow{resembles} Calcidiol \xrightarrow{resembles} Calcipotriol \xrightarrow{treats} Psoriasis$)

($Ergocalciferol \xrightarrow{causes} Nephrocalcinosis \xrightarrow{causes^{-1}} Calcitriol \xrightarrow{treats} Psoriasis$)

D Experiments on Other Datasets

All metrics are filtered [4] and evaluated for tail-sided predictions. For the experiments with AnyBURL, we learn and apply either only cyclic rules (AnyBURL (cyclic)) or include also acyclic rules of length 1 (AnyBURL).

Table D1. Comparison with baseline methods on FB15k-237. Best model hyperparameters are taken from: [a] [19], [b] [41], [c] [27].

Method	Hits@1	Hits@3	Hits@10	MRR
AnyBURL[a]	0.354	0.479	0.611	0.432
AnyBURL (cyclic)	0.292	0.397	0.539	0.362
TransE[b]	0.321	0.461	0.613	0.418
DistMult[b]	0.340	0.486	0.634	0.439
ComplEx[b]	0.346	0.493	0.639	0.445
ConvE[b]	0.342	0.481	0.630	0.439
RESCAL[b]	0.354	0.498	0.644	0.452
CompGCN	0.356	0.496	0.638	0.452
Neural LP [5]	0.166	0.248	0.348	0.227
pLogicNet[c]	0.327	0.475	0.631	0.429
MINERVA [5]	0.217	0.329	0.456	0.293
PoLo	0.238	0.325	0.438	0.302
PoLo (pruned)	0.256	0.351	0.458	0.321

Table D2. Comparison with baseline methods on WN18RR. Best model hyperparameters are taken from: [a] [19], [b] [41], [c] [27].

Method	Hits@1	Hits@3	Hits@10	MRR
AnyBURL[a]	0.490	0.547	0.609	0.527
AnyBURL (cyclic)	0.445	0.510	0.560	0.482
TransE[b]	0.064	0.387	0.555	0.245
DistMult[b]	0.443	0.495	0.556	0.481
ComplEx[b]	0.455	0.513	0.565	0.494
ConvE[b]	0.426	0.475	0.532	0.461
RESCAL[b]	0.453	0.498	0.536	0.483
CompGCN	0.460	0.512	0.564	0.497
Neural LP [5]	0.376	0.468	0.657	0.463
pLogicNet[c]	0.403	0.455	0.568	0.451
MINERVA [5]	0.413	0.456	0.513	0.448
PoLo	0.430	0.475	0.526	0.461
PoLo (pruned)	0.446	0.495	0.545	0.478

References

41. Ruffinelli, D., Broscheit, S., Gemulla, R.: You can teach an old dog new tricks! On training knowledge graph embeddings. In: The 9th International Conference on Learning Representations (2020)

Chapter 5

TLogic: Temporal Logical Rules for Explainable Link Forecasting on Temporal Knowledge Graphs

This chapter contains the publication

> **Yushan Liu**, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. TLogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *The 36th AAAI Conference on Artificial Intelligence*, volume 36(4), pages 4120-4127, 2022. DOI: 10.1609/aaai.v36i4.20330

TLogic: Temporal Logical Rules for Explainable Link Forecasting on Temporal Knowledge Graphs

Yushan Liu[1,2], **Yunpu Ma**[1,2], **Marcel Hildebrandt**[1], **Mitchell Joblin**[1], **Volker Tresp**[1,2]

[1]Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany
[2]Ludwig Maximilian University of Munich, Geschwister-Scholl-Platz 1, 80539 Munich, Germany
{yushan.liu, yunpu.ma, marcel.hildebrandt, mitchell.joblin, volker.tresp}@siemens.com

Abstract

Conventional static knowledge graphs model entities in relational data as nodes, connected by edges of specific relation types. However, information and knowledge evolve continuously, and temporal dynamics emerge, which are expected to influence future situations. In temporal knowledge graphs, time information is integrated into the graph by equipping each edge with a timestamp or a time range. Embedding-based methods have been introduced for link prediction on temporal knowledge graphs, but they mostly lack explainability and comprehensible reasoning chains. Particularly, they are usually not designed to deal with link forecasting – event prediction involving future timestamps. We address the task of link forecasting on temporal knowledge graphs and introduce TLogic, an explainable framework that is based on temporal logical rules extracted via temporal random walks. We compare TLogic with state-of-the-art baselines on three benchmark datasets and show better overall performance while our method also provides explanations that preserve time consistency. Furthermore, in contrast to most state-of-the-art embedding-based methods, TLogic works well in the inductive setting where already learned rules are transferred to related datasets with a common vocabulary.

Introduction

Knowledge graphs (KGs) structure factual information in the form of triples (e_s, r, e_o), where e_s and e_o correspond to entities in the real world and r to a binary relation, e. g., *(Anna, born in, Paris)*. This knowledge representation leads to an interpretation as a directed multigraph, where entities are identified with nodes and relations with edge types. Each edge (e_s, r, e_o) in the KG encodes an observed fact, where the source node e_s corresponds to the subject entity, the target node e_o to the object entity, and the edge type r to the predicate of the factual statement.

Some real-world information also includes a temporal dimension, e. g., the event *(Anna, born in, Paris)* happened on a specific date. To model the large amount of available event data that induce complex interactions between entities over time, temporal knowledge graphs (tKGs) have been introduced. Temporal KGs extend the triples to quadruples (e_s, r, e_o, t) to integrate a timestamp or time range t, where t indicates the time validity of the static event (e_s, r, e_o), e. g., *(Angela Merkel, visit, China, 2014/07/04)*. Figure 1 visualizes a subgraph from the dataset ICEWS14 as an example of a tKG. In this work, we focus on tKGs where each edge is equipped with a single timestamp.

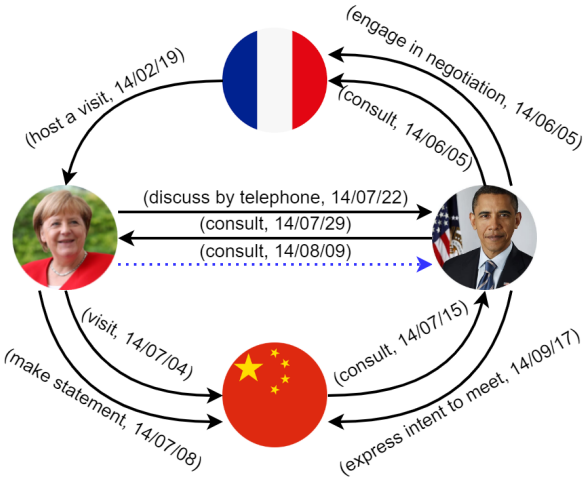


Figure 1: A subgraph from the dataset ICEWS14 with the entities *Angela Merkel, Barack Obama, France*, and *China*. The timestamps are displayed in the format yy/mm/dd. The dotted blue line represents the correct answer to the query *(Angela Merkel, consult, ?, 2014/08/09)*. Previous interactions between *Angela Merkel* and *Barack Obama* can be interpreted as an explanation for the prediction.

One of the common tasks on KGs is link prediction, which finds application in areas such as recommender systems (Hildebrandt et al. 2019), knowledge base completion (Nguyen et al. 2018a), and drug repurposing (Liu et al. 2021). Taking the additional temporal dimension into account, it is of special interest to forecast events for future timestamps based on past information. Notable real-world applications that rely on accurate event forecasting are, e. g., clinical decision support, supply chain management, and extreme events modeling. In this work, we address link forecasting on tKGs, where we consider queries $(e_s, r, ?, t)$ for a timestamp t that has not been seen during training.

Several embedding-based methods have been introduced for tKGs to solve link prediction and forecasting (link prediction with future timestamps), e.g., TTransE (Leblay and Chekol 2018), TNTComplEx (Lacroix, Obozinski, and Usunier 2020), and RE-Net (Jin et al. 2019). The underlying principle is to project the entities and relations into a low-dimensional vector space while preserving the topology and temporal dynamics of the tKG. These methods can learn the complex patterns that lead to an event but often lack transparency and interpretability.

To increase the transparency and trustworthiness of the solutions, human-understandable explanations are necessary, which can be provided by logical rules. However, the manual creation of rules is often difficult due to the complex nature of events. Domain experts cannot articulate the conditions for the occurrence of an event sufficiently formally to express this knowledge as rules, which leads to a problem termed as the knowledge acquisition bottleneck. Generally, symbolic methods that make use of logical rules tend to suffer from scalability issues, which make them impractical for the application on large real-world datasets.

We propose TLogic that automatically mines cyclic temporal logical rules by extracting temporal random walks from the graph. We achieve both high predictive performance and time-consistent explanations in the form of temporal rules, which conform to the observation that the occurrence of an event is usually triggered by previous events. The main contributions of this work are summarized as follows:

- We introduce TLogic, a novel symbolic framework based on temporal random walks in temporal knowledge graphs. It is the first approach that directly learns temporal logical rules from tKGs and applies these rules to the link forecasting task.
- Our approach provides explicit and human-readable explanations in the form of temporal logical rules and is scalable to large datasets.
- We conduct experiments on three benchmark datasets (ICEWS14, ICEWS18, and ICEWS0515) and show better overall performance compared with state-of-the-art baselines.
- We demonstrate the effectiveness of our method in the inductive setting where our learned rules are transferred to a related dataset with a common vocabulary.

Related Work

Subsymbolic machine learning methods, e. g., embedding-based algorithms, have achieved success for the link prediction task on static KGs. Well-known methods include RESCAL (Nickel, Tresp, and Kriegel 2011), TransE (Bordes et al. 2013), DistMult (Yang et al. 2015), and ComplEx (Trouillon et al. 2016) as well as the graph convolutional approaches R-GCN (Schlichtkrull et al. 2018) and CompGCN (Vashishth et al. 2020). Several approaches have been recently proposed to handle tKGs, such as TTransE (Leblay and Chekol 2018), TA-DistMult (García-Durán, Dumančić, and Niepert 2018), DE-SimplE (Goel et al. 2020), TNTComplEx (Lacroix, Obozinski, and Usunier 2020), CyGNet (Zhu et al. 2021), RE-Net (Jin et al. 2019), and xERTE (Han et al. 2021). The main idea of these methods is to explicitly learn embeddings for timestamps or to integrate temporal information into the entity or relation embeddings. However, the black-box property of embeddings makes it difficult for humans to understand the predictions. Moreover, approaches with shallow embeddings are not suitable for an inductive setting with previously unseen entities, relations, or timestamps. From the above methods, only CyGNet, RE-Net, and xERTE are designed for the forecasting task. xERTE is also able to provide explanations by extracting relevant subgraphs around the query subject.

Symbolic approaches for link prediction on KGs like AMIE+ (Galárraga et al. 2015) and AnyBURL (Meilicke et al. 2019) mine logical rules from the dataset, which are then applied to predict new links. StreamLearner (Omran, Wang, and Wang 2019) is one of the first methods for learning temporal rules. It employs a static rule learner to generate rules, which are then generalized to the temporal domain. However, they only consider a rather restricted set of temporal rules, where all body atoms have the same timestamp.

Another class of approaches is based on random walks in the graph, where the walks can support an interpretable explanation for the predictions. For example, AnyBURL samples random walks for generating rules. The methods dynnode2vec (Mahdavi, Khoshraftar, and An 2018) and change2vec (Bian et al. 2019) alternately extract random walks on tKG snapshots and learn parameters for node embeddings, but they do not capture temporal patterns within the random walks. Nguyen et al. (2018b) extend the concept of random walks to temporal random walks on continuous-time dynamic networks for learning node embeddings, where the sequence of edges in the walk only moves forward in time.

Preliminaries

Let $[n] := \{1, 2, \ldots, n\}$.

Temporal knowledge graph Let $\mathcal{E}$ denote the set of entities, $\mathcal{R}$ the set of relations, and $\mathcal{T}$ the set of timestamps.

A *temporal knowledge graph* (tKG) is a collection of facts $\mathcal{G} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$, where each fact is represented by a quadruple (e_s, r, e_o, t). The quadruple (e_s, r, e_o, t) is also called link or edge, and it indicates a connection between the subject entity $e_s \in \mathcal{E}$ and the object entity $e_o \in \mathcal{E}$ via the relation $r \in \mathcal{R}$. The timestamp $t \in \mathcal{T}$ implies the occurrence of the event (e_s, r, e_o) at time t, where t can be measured in units such as hour, day, and year.

For two timestamps t and $\hat{t}$, we denote the fact that t occurs earlier than $\hat{t}$ by $t < \hat{t}$. If additionally, t could represent the same time as $\hat{t}$, we write $t \leq \hat{t}$.

We define for each edge (e_s, r, e_o, t) an inverse edge (e_o, r^{-1}, e_s, t) that interchanges the positions of the subject and object entity to allow the random walker to move along the edge in both directions. The relation $r^{-1} \in \mathcal{R}$ is called the inverse relation of r.

Link forecasting The goal of the *link forecasting* task is to predict new links for future timestamps. Given a query with a previously unseen timestamp $(e_s, r, ?, t)$, we want to identify a ranked list of object candidates that are most likely

to complete the query. For subject prediction, we formulate the query as $(e_o, r^{-1}, ?, t)$.

Temporal random walk A *non-increasing temporal random walk* W of length $l \in \mathbb{N}$ from entity $e_{l+1} \in \mathcal{E}$ to entity $e_1 \in \mathcal{E}$ in the tKG $\mathcal{G}$ is defined as a sequence of edges

$$((e_{l+1}, r_l, e_l, t_l), (e_l, r_{l-1}, e_{l-1}, t_{l-1}), \ldots, (e_2, r_1, e_1, t_1)) \quad \text{with } t_l \geq t_{l-1} \geq \cdots \geq t_1, \tag{1}$$

where $(e_{i+1}, r_i, e_i, t_i) \in \mathcal{G}$ for $i \in [l]$.

A non-increasing temporal random walk complies with time constraints so that the edges are traversed only backward in time, where it is also possible to walk along edges with the same timestamp.

Temporal logical rule Let E_i and T_i for $i \in [l+1]$ be variables that represent entities and timestamps, respectively. Further, let $r_1, r_2, \ldots, r_l, r_h \in \mathcal{R}$ be fixed.

A *cyclic temporal logical rule* R of length $l \in \mathbb{N}$ is defined as

$$((E_1, r_h, E_{l+1}, T_{l+1}) \leftarrow \wedge_{i=1}^{l}(E_i, r_i, E_{i+1}, T_i))$$

with the temporal constraints

$$T_1 \leq T_2 \leq \cdots \leq T_l < T_{l+1}. \tag{2}$$

The left-hand side of R is called the rule head, with r_h being the head relation, while the right-hand side is called the rule body, which is represented by a conjunction of body atoms (E_i, r_i, E_{i+1}, T_i). The rule is called cyclic because the rule head and the rule body constitute two different walks connecting the same two variables E_1 and E_{l+1}. A temporal rule implies that if the rule body holds with the temporal constraints given by (2), then the rule head is true as well for a future timestamp T_{l+1}.

The replacement of the variables E_i and T_i by constant terms is called grounding or instantiation. For example, a grounding of the temporal rule

$$((E_1, \textit{consult}, E_2, T_2) \leftarrow (E_1, \textit{discuss by telephone}, E_2, T_1))$$

is given by the edges *(Angela Merkel, discuss by telephone, Barack Obama, 2014/07/22)* and *(Angela Merkel, consult, Barack Obama, 2014/08/09)* in Figure 1. Let rule grounding refer to the replacement of the variables in the entire rule and body grounding refer to the replacement of the variables only in the body, where all groundings must comply with the temporal constraints in (2).

In many domains, logical rules are frequently violated so that confidence values are determined to estimate the probability of a rule's correctness. We adapt the standard confidence to take timestamp values into account. Let $(r_1, r_2, \ldots, r_l, r_h)$ be the relations in a rule R. The body support is defined as the number of body groundings, i. e., the number of tuples $(e_1, \ldots, e_{l+1}, t_1, \ldots, t_l)$ such that $(e_i, r_i, e_{i+1}, t_i) \in \mathcal{G}$ for $i \in [l]$ and $t_i \leq t_{i+1}$ for $i \in [l-1]$. The rule support is defined as the number of body groundings such that there exists a timestamp $t_{l+1} > t_l$ with $(e_1, r_h, e_{l+1}, t_{l+1}) \in \mathcal{G}$. The confidence of the rule R, denoted by conf(R), can then be obtained by dividing the rule support by the body support.

Algorithm 1: Rule learning

Input: Temporal knowledge graph $\mathcal{G}$.
Parameters: Rule lengths $\mathcal{L} \subset \mathbb{N}$, number of temporal random walks $n \in \mathbb{N}$, transition distribution $d \in \{\text{unif}, \exp\}$.
Output: Temporal logical rules $\mathcal{TR}$.

for relation $r \in \mathcal{R}$ **do**
  for $l \in \mathcal{L}$ **do**
    for $i \in [n]$ **do**
      $\mathcal{TR}_r^l \leftarrow \emptyset$
      According to transition distribution d, sample a temporal random walk W of length $l+1$ with $t_{l+1} > t_l$. ▷ See (4).
      Transform walk W to the corresponding temporal logical rule R. ▷ See (5).
      Estimate the confidence of rule R.
      $\mathcal{TR}_r^l \leftarrow \mathcal{TR}_r^l \cup \{(R, \text{conf}(R))\}$
  $\mathcal{TR}_r \leftarrow \cup_{l \in \mathcal{L}} \mathcal{TR}_r^l$
$\mathcal{TR} \leftarrow \cup_{r \in \mathcal{R}} \mathcal{TR}_r$
return $\mathcal{TR}$

Our Framework

We introduce TLogic, a rule-based link forecasting framework for tKGs. TLogic first extracts temporal walks from the graph and then lifts these walks to a more abstract, semantic level to obtain temporal rules that generalize to new data. The application of these rules generates answer candidates, for which the body groundings in the graph serve as explicit and human-readable explanations. Our framework consists of the components rule learning and rule application. The pseudocode for rule learning is shown in Algorithm 1 and for rule application in Algorithm 2.

Rule Learning

As the first step of rule learning, temporal walks are extracted from the tKG $\mathcal{G}$. For a rule of length l, a walk of length $l+1$ is sampled, where the additional step corresponds to the rule head.

Let r_h be a fixed relation, for which we want to learn rules. For the first sampling step $m = 1$, we sample an edge $(e_1, r_h, e_{l+1}, t_{l+1})$, which will serve as the rule head, uniformly from all edges with relation type r_h. A temporal random walker then samples iteratively edges adjacent to the current object until a walk of length $l+1$ is obtained.

For sampling step $m \in \{2, \ldots, l+1\}$, let $(e_s, \tilde{r}, e_o, t)$ denote the previously sampled edge and $\mathcal{A}(m, e_o, t)$ the set of feasible edges for the next transition. To fulfill the temporal constraints in (1) and (2), we define

$$\mathcal{A}(m, e_o, t) := \begin{cases} \{(e_o, r, e, \hat{t}) \mid (e_o, r, e, \hat{t}) \in \mathcal{G},\ \hat{t} < t\} & \text{if } m = 2, \\ \{(e_o, r, e, \hat{t}) \mid (e_o, r, e, \hat{t}) \in \tilde{\mathcal{G}},\ \hat{t} \leq t\} & \text{if } m \in \{3, \ldots, l\}, \\ \{(e_o, r, e_1, \hat{t}) \mid (e_o, r, e_1, \hat{t}) \in \tilde{\mathcal{G}},\ \hat{t} \leq t\} & \text{if } m = l+1, \end{cases}$$

where $\tilde{\mathcal{G}} := \mathcal{G} \setminus \{(e_o, \tilde{r}^{-1}, e_s, t)\}$ excludes the inverse edge to avoid redundant rules. For obtaining cyclic walks, we sample in the last step $m = l+1$ an edge that connects

the walk to the first entity e_1 if such edges exist. Otherwise, we sample the next walk.

The transition distribution for sampling the next edge can either be uniform or exponentially weighted. We define an index mapping $\hat{m} := (l+1) - (m-2)$ to be consistent with the indices in (1). Then, the exponentially weighted probability for choosing edge $u \in \mathcal{A}(m, e_{\hat{m}}, t_{\hat{m}})$ for $m \in \{2, \ldots, l+1\}$ is given by

$$\mathbb{P}(u; m, e_{\hat{m}}, t_{\hat{m}}) = \frac{\exp(t_u - t_{\hat{m}})}{\sum_{\hat{u} \in \mathcal{A}(m, e_{\hat{m}}, t_{\hat{m}})} \exp(t_{\hat{u}} - t_{\hat{m}})} \quad (3)$$

where t_u denotes the timestamp of edge u. The exponential weighting favors edges with timestamps that are closer to the timestamp of the previous edge and probably more relevant for prediction.

The resulting temporal walk W is given by

$$((e_1, r_h, e_{l+1}, t_{l+1}), (e_{l+1}, r_l, e_l, t_l), \ldots, (e_2, r_1, e_1, t_1)). \quad (4)$$

W can then be transformed to a temporal rule R by replacing the entities and timestamps with variables. While the first edge in W becomes the rule head $(E_1, r_h, E_{l+1}, T_{l+1})$, the other edges are mapped to body atoms, where each edge (e_{i+1}, r_i, e_i, t_i) is converted to the body atom $(E_i, r_i^{-1}, E_{i+1}, T_i)$. The final rule R is denoted by

$$((E_1, r_h, E_{l+1}, T_{l+1}) \leftarrow \wedge_{i=1}^{l} (E_i, r_i^{-1}, E_{i+1}, T_i)). \quad (5)$$

In addition, we impose the temporal consistency constraints $T_1 \leq T_2 \leq \cdots \leq T_l < T_{l+1}$.

The entities $(e_1, \ldots, e_{l+1})$ in W do not need to be distinct since a pair of entities can have many interactions at different points in time. For example, Angela Merkel made several visits to China in 2014, which could constitute important information for the prediction. Repetitive occurrences of the same entity in W are replaced with the same random variable in R to maintain this knowledge.

For the confidence estimation of R, we sample from the graph a fixed number of body groundings, which have to match the body relations and the variable constraints mentioned in the last paragraph while satisfying the condition from (2). The number of unique bodies serves as the body support. The rule support is determined by counting the number of bodies for which an edge with relation type r_h exists that connects e_1 and e_{l+1} from the body. Moreover, the timestamp of this edge has to be greater than all body timestamps to fulfill (2).

For every relation $r \in \mathcal{R}$, we sample $n \in \mathbb{N}$ temporal walks for a set of prespecified lengths $\mathcal{L} \subset \mathbb{N}$. The set $\mathcal{TR}_r^l$ stands for all rules of length l with head relation r with their corresponding confidences. All rules for relation r are included in $\mathcal{TR}_r := \cup_{l \in \mathcal{L}} \mathcal{TR}_r^l$, and the complete set of learned temporal rules is given by $\mathcal{TR} := \cup_{r \in \mathcal{R}} \mathcal{TR}_r$.

It is possible to learn rules only for a single relation or a set of specific relations of interest. Explicitly learning rules for all relations is especially effective for rare relations that would otherwise only be sampled with a small probability. The learned rules are not specific to the graph from which they have been extracted, but they could be employed in an inductive setting where the rules are transferred to related datasets that share a common vocabulary for straightforward application.

Algorithm 2: Rule application

Input: Test query $q = (e^q, r^q, ?, t^q)$, temporal logical rules $\mathcal{TR}$, temporal knowledge graph $\mathcal{G}$.
Parameters: Time window $w \in \mathbb{N} \cup \{\infty\}$, minimum number of candidates k, score function f.
Output: Answer candidates $\mathcal{C}$.

$\mathcal{C} \leftarrow \emptyset$
▷ Apply the rules in $\mathcal{TR}$ by decreasing confidence.
if $\mathcal{TR}_{r^q} \neq \emptyset$ **then**
  for rule $R \in \mathcal{TR}_{r^q}$ **do**
    Find all body groundings of R in $\mathcal{SG} \subset \mathcal{G}$, where $\mathcal{SG}$ consists of the edges within the time window $[t^q - w, t^q)$.
    Retrieve candidates $\mathcal{C}(R)$ from the target entities of the body groundings.
    for $c \in \mathcal{C}(R)$ **do**
      Calculate score $f(R, c)$. ▷ See (6).
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{(c, f(R, c))\}$
    if $|\{c \mid \exists R : (c, f(R, c)) \in \mathcal{C}\}| \geq k$ **then**
      break
return $\mathcal{C}$

Rule Application

The learned temporal rules $\mathcal{TR}$ are applied to answer queries of the form $q = (e^q, r^q, ?, t^q)$. The answer candidates are retrieved from the target entities of body groundings in the tKG $\mathcal{G}$. If there exist no rules $\mathcal{TR}_{r^q}$ for the query relation r^q, or if there are no matching body groundings in the graph, then no answers are predicted for the given query.

To apply the rules on relevant data, a subgraph $\mathcal{SG} \subset \mathcal{G}$ dependent on a time window $w \in \mathbb{N} \cup \{\infty\}$ is retrieved. For $w \in \mathbb{N}$, the subgraph $\mathcal{SG}$ contains all edges from $\mathcal{G}$ that have timestamps $t \in [t^q - w, t^q)$. If $w = \infty$, then all edges with timestamps prior to the query timestamp t^q are used for rule application, i. e., $\mathcal{SG}$ consists of all facts with $t \in [t_{\min}, t^q)$, where $t_{\min}$ is the minimum timestamp in the graph $\mathcal{G}$.

We apply the rules $\mathcal{TR}_{r^q}$ by decreasing confidence, where each rule R generates a set of answer candidates $\mathcal{C}(R)$. Each candidate $c \in \mathcal{C}(R)$ is then scored by a function $f : \mathcal{TR}_{r^q} \times \mathcal{E} \rightarrow [0, 1]$ that reflects the probability of the candidate being the correct answer to the query.

Let $\mathcal{B}(R, c)$ be the set of body groundings of rule R that start at entity e^q and end at entity c. We choose as score function f a convex combination of the rule's confidence and a function that takes the time difference $t^q - t_1(\mathcal{B}(R, c))$ as input, where $t_1(\mathcal{B}(R, c))$ denotes the earliest timestamp t_1 in the body. If several body groundings exist, we take from all possible t_1 values the one that is closest to t^q. For candidate $c \in \mathcal{C}(R)$, the score function is defined as

$$f(R, c) = a \cdot \text{conf}(R) + (1 - a) \cdot \exp(-\lambda(t^q - t_1(\mathcal{B}(R, c)))) \quad (6)$$

with $\lambda > 0$ and $a \in [0, 1]$.

The intuition for this choice of f is that candidates generated by high-confidence rules should receive a higher score.

Adding a dependency on the timeframe of the rule grounding is based on the observation that the existence of edges in a rule become increasingly probable with decreasing time difference between the edges. We choose the exponential distribution since it is commonly used to model interarrival times of events. The time difference $t^q - t_1(\mathcal{B}(R,c))$ is always non-negative for a future timestamp value t^q, and with the assumption that there exists a fixed mean, the exponential distribution is also the maximum entropy distribution for such a time difference variable. The exponential distribution is rescaled so that both summands are in the range $[0,1]$.

All candidates are saved with their scores as $(c, f(R,c))$ in $\mathcal{C}$. We stop the rule application when the number of different answer candidates $|\{c \mid \exists R : (c, f(R,c)) \in \mathcal{C}\}|$ is at least k so that there is no need to go through all rules.

Candidate Ranking

For the ranking of the answer candidates, all scores of each candidate c are aggregated through a noisy-OR calculation, which produces the final score

$$1 - \Pi_{\{s|(c,s)\in\mathcal{C}\}}(1-s). \tag{7}$$

The idea is to aggregate the scores to produce a probability, where candidates implied by more rules should have a higher score.

In case there are no rules for the query relation r^q, or if there are no matching body groundings in the graph, it might still be interesting to retrieve possible answer candidates. In the experiments, we apply a simple baseline where the scores for the candidates are obtained from the overall object distribution in the training data if r^q is a new relation. If r^q already exists in the training set, we take the object distribution of the edges with relation type r^q.

Experiments

Datasets

We conduct experiments on the dataset Integrated Crisis Early Warning System[1] (ICEWS), which contains information about international events and is a commonly used benchmark dataset for link prediction on tKGs. We choose the subsets ICEWS14, ICEWS18, and ICEWS0515, which include data from the years 2014, 2018, and 2005 to 2015, respectively. Since we consider link forecasting, each dataset is split into training, validation, and test set so that the timestamps in the training set occur earlier than the timestamps in the validation set, which again occur earlier than the timestamps in the test set. To ensure a fair comparison, we use the split provided by Han et al. (2021)[2]. The statistics of the datasets are summarized in the supplementary material.

Experimental Setup

For each test instance (e_s^q, r^q, e_o^q, t^q), we generate a list of candidates for both object prediction $(e_s^q, r^q, ?, t^q)$ and subject prediction $(e_o^q, (r^q)^{-1}, ?, t^q)$. The candidates are ranked by decreasing scores, which are calculated according to (7).

The confidence for each rule is estimated by sampling 500 body groundings and counting the number of times the rule head holds. We learn rules of the lengths 1, 2, and 3, and for application, we only consider the rules with a minimum confidence of 0.01 and minimum body support of 2.

We compute the mean reciprocal rank (MRR) and hits@k for $k \in \{1, 3, 10\}$, which are standard metrics for link prediction on KGs. For a rank $x \in \mathbb{N}$, the reciprocal rank is defined as $\frac{1}{x}$, and the MRR is the average of all reciprocal ranks of the correct query answers across all queries. The metric hits@k (h@k) indicates the proportion of queries for which the correct entity appears under the top k candidates.

Similar to Han et al. (2021), we perform time-aware filtering where all correct entities at the query timestamp except for the true query object are filtered out from the answers. In comparison to the alternative setting that filters out all other objects that appear together with the query subject and relation at any timestamp, time-aware filtering yields a more realistic performance estimate.

Baseline methods We compare TLogic[3] with the state-of-the-art baselines for static link prediction DistMult (Yang et al. 2015), ComplEx (Trouillon et al. 2016), and AnyBURL (Meilicke et al. 2019, 2020) as well as for temporal link prediction TTransE (Leblay and Chekol 2018), TA-DistMult (García-Durán, Dumančić, and Niepert 2018), DE-SimplE (Goel et al. 2020), TNTComplEx (Lacroix, Obozinski, and Usunier 2020), CyGNet (Zhu et al. 2021), RE-Net (Jin et al. 2019), and xERTE (Han et al. 2021). All baseline results except for the results on AnyBURL are from Han et al. (2021). AnyBURL samples paths based on reinforcement learning and generalizes them to rules, where the rule space also includes, e. g., acyclic rules and rules with constants. A non-temporal variant of TLogic would sample paths randomly and only learn cyclic rules, which would presumably yield worse performance than AnyBURL. Therefore, we choose AnyBURL as a baseline to assess the effectiveness of adding temporal constraints.

Results

The results of the experiments are displayed in Table 1. TLogic outperforms all baseline methods with respect to the metrics MRR, hits@3, and hits@10. Only xERTE performs better than Tlogic for hits@1 on the datasets ICEWS18 and ICEWS0515.

Besides a list of possible answer candidates with corresponding scores, TLogic can also provide temporal rules and body groundings in form of walks from the graph that support the predictions. Table 2 presents three exemplary rules with high confidences that were learned from ICEWS14. For the query *(Angela Merkel, consult, ?, 2014/08/09)*, two walks are shown in Table 2, which serve as time-consistent explanations for the correct answer *Barack Obama*.

Inductive setting One advantage of our learned logical rules is that they are applicable to any new dataset as long as the new dataset covers common relations. This might be relevant for cases where new entities appear. For example, Donald Trump, who served as president of the United States

[1] https://dataverse.harvard.edu/dataverse/icews

[2] https://github.com/TemporalKGTeam/xERTE

[3] Code available at https://github.com/liu-yushan/TLogic.

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**

Table 1: Results of link forecasting on the datasets ICEWS14, ICEWS18, and ICEWS0515. All metrics are time-aware filtered. The best results among all models are displayed in bold.

Confidence	Head	Body
0.963	$(E_1, \textit{demonstrate or rally}, E_2, T_4)$	$(E_1, \textit{riot}, E_2, T_1) \wedge (E_2, \textit{make statement}, E_1, T_2) \wedge (E_1, \textit{riot}, E_2, T_3)$
0.818	$(E_1, \textit{share information}, E_2, T_2)$	$(E_1, \textit{express intent to ease sanctions}^{-1}, E_2, T_1)$
0.750	$(E_1, \textit{provide military aid}, E_3, T_3)$	$(E_1, \textit{provide military aid}, E_2, T_1) \wedge (E_2, \textit{intend to protect}^{-1}, E_3, T_2)$
0.570	*(Merkel, consult, Obama, 14/08/09)*	*(Merkel, discuss by telephone, Obama, 14/07/22)*
0.500	*(Merkel, consult, Obama, 14/08/09)*	*(Merkel, express intent to meet, Obama, 14/05/02)* $\wedge$ $(\textit{Obama}, \textit{consult}^{-1}, \textit{Merkel}, \textit{14/07/18}) \wedge (\textit{Merkel}, \textit{consult}^{-1}, \textit{Obama}, \textit{14/07/29})$

Table 2: Three exemplary rules from the dataset ICEWS14 and two walks for the query *(Angela Merkel, consult, ?, 2014/08/09)* that lead to the correct answer *Barack Obama*. The timestamps are displayed in the format yy/mm/dd.

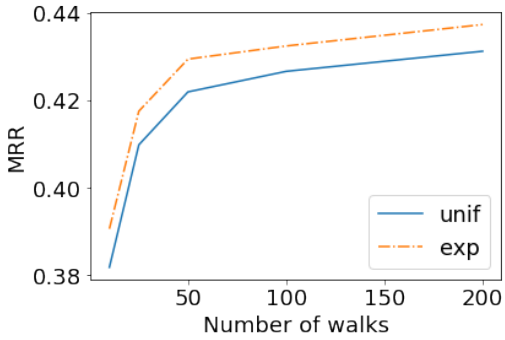


Figure 2: MRR performance on the validation set of ICEWS14. The transition distribution is either uniform or exponentially weighted.

from 2017 to 2021, is included in the dataset ICEWS18 but not in ICEWS14. The logical rules are not tied to particular entities and would still be applicable, while embedding-based methods have difficulties operating in this challenging setting. The models would need to be retrained to obtain embeddings for the new entities, where existing embeddings might also need to be adapted to the different time range.

For the two rule-based methods AnyBURL and TLogic, we apply the rules learned on the training set of ICEWS0515 (with timestamps from 2005/01/01 to 2012/08/06) to the test set of ICEWS14 as well as the rules learned on the training set of ICEWS14 to the test set of ICEWS18 (see Table 3). The performance of TLogic in the inductive setting is for all metrics close to the results in Table 1, while for AnyBURL, especially the results on ICEWS18 drop significantly. It seems that the encoded temporal information in TLogic is essential for achieving correct predictions in the inductive setting. ICEWS14 has only 7,128 entities, while ICEWS18 contains 23,033 entities. The results confirm that temporal rules from TLogic can even be transferred to a dataset with a large number of new entities and timestamps and lead to a strong performance.

Analysis

The results in this section are obtained on the dataset ICEWS14, but the findings are similar for the other two datasets. More detailed results can be found in the supplementary material.

Number of walks Figure 2 shows the MRR performance on the validation set of ICEWS14 for different numbers of walks that were extracted during rule learning. We observe a performance increase with a growing number of walks. However, the performance gains saturate between 100 and 200 walks where rather small improvements are attainable.

$\mathcal{G}_{\text{train}}$	$\mathcal{G}_{\text{test}}$	Model	MRR	h@1	h@3	h@10
ICEWS0515	ICEWS14	AnyBURL	0.2664	0.1800	0.3024	0.4477
		TLogic	0.4253	0.3291	0.4780	0.6122
ICEWS14	ICEWS18	AnyBURL	0.1546	0.0907	0.1685	0.2958
		TLogic	0.2915	0.1987	0.3330	0.4795

Table 3: Inductive setting where rules learned on $\mathcal{G}_{\text{train}}$ are transferred and applied to $\mathcal{G}_{\text{test}}$.

Transition distribution We test two transition distributions for the extraction of temporal walks: uniform and exponentially weighted according to (3). The rationale behind using an exponentially weighted distribution is the observation that related events tend to happen within a short timeframe. The distribution of the first edge is always uniform to not restrict the variety of obtained walks. Overall, the performance of the exponential distribution consistently exceeds the uniform setting with respect to the MRR (see Figure 2).

We observe that the exponential distribution leads to more rules of length 3 than the uniform setting (11,718 compared to 8,550 rules for 200 walks), while it is the opposite for rules of length 1 (7,858 compared to 11,019 rules). The exponential setting leads to more successful longer walks because the timestamp differences between subsequent edges tend to be smaller. It is less likely that there are no feasible transitions anymore because of temporal constraints. The uniform setting, however, leads to a better exploration of the neighborhood around the start node for shorter walks.

Rule length We learn rules of lengths 1, 2, and 3. Using all rules for application results in the best performance (MRR on the validation set: 0.4373), followed by rules of only length 1 (0.4116), 3 (0.4097), and 2 (0.1563). The reason why rules of length 3 perform better than length 2 is that the temporal walks are allowed to transition back and forth between the same entities. Since we only learn cyclic rules, a rule body of length 2 must constitute a path with no recurring entities, resulting in fewer rules and rule groundings in the graph. Interestingly, simple rules of length 1 already yield very good performance.

Time window For rule application, we define a time window for retrieving the relevant data. The performance increases with the size of the time window, even though relevant events tend to be close to the query timestamp. The second summand of the score function f in (6) takes the time difference between the query timestamp t^q and the earliest body timestamp $t_1(\mathcal{B}(R,c))$ into account. In this case, earlier events with a large timestamp difference receive a lesser weight, while generally, as much information as possible is beneficial for prediction.

Score function We define the score function f in (6) as a convex combination of the rule's confidence and a function that depends on the time difference $t^q - t_1(\mathcal{B}(R,c))$. The performance of only using the confidence (MRR: 0.3869) or only using the exponential function (0.4077) is worse than the combination (0.4373), which means that both the information from the rules' confidences and the time differences are important for prediction.

Variance The variance in the performance due to different rules obtained from the rule learning component is quite small. Running the same model with the best hyperparameter settings for five different seeds results in a standard deviation of 0.0012 for the MRR. The rule application component is deterministic and always leads to the same candidates with corresponding scores for the same hyperparameter setting.

Training and inference time The worst-case time complexity for learning rules of length l is $\mathcal{O}(|\mathcal{R}|nlDb)$, where n is the number of walks, D the maximum node degree in the training set, and b the number of body samples for estimating the confidence. The worst-case time complexity for inference is given by $\mathcal{O}(|\mathcal{G}| + |\mathcal{TR}_{r^q}|D^L|\mathcal{E}|\log(k))$, where L is the maximum rule length in $\mathcal{TR}_{r^q}$ and k the minimum number of candidates. For large graphs with high node degrees, it is possible to reduce the complexity to $\mathcal{O}\left(|\mathcal{G}| + |\mathcal{TR}_{r^q}|KLD|\mathcal{E}|\log(k)\right)$ by only keeping a maximum of K candidate walks during rule application.

Both training and application can be parallelized since the rule learning for each relation and the rule application for each test query are independent. Rule learning with 200 walks and exponentially weighted transition distribution for rule lengths $\{1,2,3\}$ on a machine with 8 CPUs takes 180 sec for ICEWS14, while the application on the validation set takes 2000 sec, with $w = \infty$ and $k = 20$. For comparison, the best-performing baseline xERTE needs for training one epoch on the same machine already 5000 sec, where an MRR of 0.3953 can be obtained, while testing on the validation set takes 700 sec.

Conclusion

We have proposed TLogic, the first symbolic framework that directly learns temporal logical rules from temporal knowledge graphs and applies these rules for link forecasting. The framework generates answers by applying rules to observed events prior to the query timestamp and scores the answer candidates depending on the rules' confidences and time differences. Experiments on three datasets indicate that TLogic achieves better overall performance compared to state-of-the-art baselines. In addition, our approach also provides time-consistent, explicit, and human-readable explanations for the predictions in the form of temporal logical rules.

As future work, it would be interesting to integrate acyclic rules, which could also contain relevant information and might boost the performance for rules of length 2. Furthermore, the simple sampling mechanism for temporal walks could be replaced by a more sophisticated approach, which is able to effectively identify the most promising walks.

Acknowledgements

This work has been supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of the project RAKI under grant number 01MD19012C and by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS18036A. The authors of this work take full responsibility for its content.

References

Bian, R.; Koh, Y. S.; Dobbie, G.; and Divoli, A. 2019. Network embedding and change modeling in dynamic heterogeneous networks. In *Proceedings of the Forty-Second International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the Twenty-Sixth International Conference on Neural Information Processing Systems*.

Galárraga, L.; Teflioudi, C.; Hose, K.; and Suchanek, F. M. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, 24: 707–730.

García-Durán, A.; Dumančić, S.; and Niepert, M. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Goel, R.; Kazemi, S. M.; Brubaker, M.; and Poupart, P. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Han, Z.; Chen, P.; Ma, Y.; and Tresp, V. 2021. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *Proceedings of the Ninth International Conference on Learning Representations*.

Hildebrandt, M.; Sunder, S. S.; Mogoreanu, S.; Joblin, M.; Mehta, A.; Thon, I.; and Tresp, V. 2019. A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context. In *Proceedings of the Sixteenth Extended Semantic Web Conference*.

Jin, W.; Zhang, C.; Szekely, P.; and Ren, X. 2019. Recurrent event network for reasoning over temporal knowledge graphs. Workshop paper at the Seventh International Conference on Learning Representations.

Lacroix, T.; Obozinski, G.; and Usunier, N. 2020. Tensor decompositions for temporal knowledge base completion. In *Proceedings of the Eighth International Conference on Learning Representations*.

Leblay, J.; and Chekol, M. W. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the Web Conference 2018*.

Liu, Y.; Hildebrandt, M.; Joblin, M.; Ringsquandl, M.; Raissouni, R.; and Tresp, V. 2021. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In *Proceedings of the Eighteenth Extended Semantic Web Conference*.

Mahdavi, S.; Khoshraftar, S.; and An, A. 2018. dynnode2vec: scalable dynamic network embedding. In *Proceedings of the 2018 IEEE International Conference on Big Data*.

Meilicke, C.; Chekol, M. W.; Fink, M.; and Stuckenschmidt, H. 2020. Reinforced anytime bottom-up rule learning for knowledge graph completion. arXiv:2004.04412.

Meilicke, C.; Chekol, M. W.; Ruffinelli, D.; and Stuckenschmidt, H. 2019. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.

Nguyen, D. Q.; Nguyen, T. D.; Nguyen, D. Q.; and Phung, D. 2018a. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the Sixteenth Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nguyen, G. H.; Lee, J. B.; Rossi, R. A.; Ahmed, N. K.; Koh, E.; and Kim, S. 2018b. Dynamic network embeddings: from random walks to temporal random Walks. In *Proceedings of the 2018 IEEE International Conference on Big Data*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*.

Omran, P. G.; Wang, K.; and Wang, Z. 2019. Learning temporal rules from knowledge graph streams. In *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering*.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the Fifteenth Extended Semantic Web Conference*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *Proceedings of the Thirty-Third International Conference on Machine Learning*.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. 2020. Composition-based multi-relational graph convolutional networks. In *Proceedings of the Eighth International Conference on Learning Representations*.

Yang, B.; Yih, W.-T.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the Third International Conference on Learning Representations*.

Zhu, C.; Chen, M.; Fan, C.; Cheng, G.; and Zhang, Y. 2021. Learning from history: modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Supplementary Material

Dataset statistics Table 4 shows the statistics of the three datasets ICEWS14, ICEWS18, and ICEWS0515. $|\mathcal{X}|$ denotes the cardinality of a set $\mathcal{X}$.

Dataset	$\lvert\mathcal{G}_{\text{train}}\rvert$	$\lvert\mathcal{G}_{\text{valid}}\rvert$	$\lvert\mathcal{G}_{\text{test}}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$
14	63,685	13,823	13,222	7,128	230	365
18	373,018	45,995	49,545	23,033	256	304
0515	322,958	69,224	69,147	10,488	251	4,017

Table 4: Dataset statistics with daily time resolution for all three ICEWS datasets.

Experimental details All experiments were conducted on a Linux machine with 16 CPU cores and 32 GB RAM. The set of tested hyperparameter values and best values for TLogic are displayed in Table 5. Due to memory constraints, the time window w for ICEWS18 is set to 200 and for ICEWS0515 to 1000. The best hyperparameter values are chosen based on the MRR on the validation set. Due to the small variance of our approach, the shown results are based on one algorithm run. A random seed of 12 is fixed for the rule learning component to obtain reproducible results.

Hyperparameter	Values	Best
Number of walks n	$\{10, 25, 50, 100, 200\}$	200
Transition distribution d	$\{\text{unif}, \exp\}$	exp
Rule lengths $\mathcal{L}$	$\{\{1\}, \{2\}, \{3\}, \{1, 2, 3\}\}$	$\{1, 2, 3\}$
Time window w	$\{30, 90, 150, 210, 270, \infty\}$	∞
Minimum candidates k	$\{10, 20\}$	20
α (score function f)	$\{0, 0.25, 0.5, 0.75, 1\}$	0.5
λ (score function f)	$\{0.01, 0.1, 0.5, 1\}$	0.1

Table 5: Tested hyperparameter values and best values.

All results in the appendix refer to the validation set of ICEWS14. However, the observations are similar for the test set and the other two datasets. All experiments use the best set of hyperparameters, where only the analyzed parameters are modified.

Object distribution baseline We apply a simple object distribution baseline when there are no rules for the query relation or no matching body groundings in the graph. This baseline is only added for completeness and does not improve the results in a significant way.

The proportion of cases where there are no rules for the test query relation is 15/26,444 = 0.00056 for ICEWS14, 21/99,090 = 0.00021 for ICEWS18, and 9/138,294 = 0.00007 for ICEWS0515. The proportion of cases where there are no matching body groundings is 880/26,444 = 0.0333 for ICEWS14, 2,535/99,090 = 0.0256 for ICEWS18, and 2,375/138,294 = 0.0172 for ICEWS0515.

Number of walks and transition distribution Table 6 shows the results for different choices of numbers of walks and transition distributions. The performance for all metrics increases with the number of walks. Exponentially weighted transition always outperforms uniform sampling.

Walks	Transition	MRR	h@1	h@3	h@10
10	Unif	0.3818	0.2983	0.4307	0.5404
10	Exp	0.3906	0.3054	0.4408	0.5530
25	Unif	0.4098	0.3196	0.4614	0.5803
25	Exp	0.4175	0.3270	0.4710	0.5875
50	Unif	0.4219	0.3307	0.4754	0.5947
50	Exp	0.4294	0.3375	0.4837	0.6024
100	Unif	0.4266	0.3315	0.4817	0.6057
100	Exp	0.4324	0.3397	0.4861	0.6092
200	Unif	0.4312	0.3366	0.4851	0.6114
200	Exp	0.4373	0.3434	0.4916	0.6161

Table 6: Results for different choices of numbers of walks and transition distributions.

Rule lengths Table 7 indicates that using rules of all lengths for application results in the best performance. Learning only cyclic rules probably makes it more difficult to find rules of length 2, where the rule body must constitute a path with no recurring entities, leading to fewer rules and body groundings in the graph.

Rule length	MRR	h@1	h@3	h@10
1	0.4116	0.3168	0.4708	0.5909
2	0.1563	0.0648	0.1776	0.3597
3	0.4097	0.3213	0.4594	0.5778
1,2,3	0.4373	0.3434	0.4916	0.6161

Table 7: Results for different choices of rule lengths.

Time window Generally, the larger the time window, the better the performance (see Table 8). If taking all previous timestamps leads to a too high memory usage, the time window should be decreased.

Time window	MRR	h@1	h@3	h@10
30	0.3842	0.3080	0.4294	0.5281
90	0.4137	0.3287	0.4627	0.5750
150	0.4254	0.3368	0.4766	0.5950
210	0.4311	0.3403	0.4835	0.6035
270	0.4356	0.3426	0.4892	0.6131
∞	0.4373	0.3434	0.4916	0.6161

Table 8: Results for different choices of time windows.

Score function Using the best hyperparameters values for α and λ, Table 9 shows in the first row the results if only the rules' confidences are used for scoring, in the second row if only the exponential component is used, and in the last row the results for the combined score function. The combination yields the best overall performance. The optimal balance between the two terms, however, depends on the application and metric prioritization.

α	λ	MRR	h@1	h@3	h@10
1	arbitrary	0.3869	0.2806	0.4444	0.5918
0	0.1	0.4077	0.3515	0.4820	0.6051
0.5	0.1	0.4373	0.3434	0.4916	0.6161

Table 9: Results for different parameter values in the score function f.

Rule learning The figures 3 and 4 show the number of rules learned under the two transition distributions. The total number of learned rules is similar for the uniform and exponential distribution, but there is a large difference for rules of lengths 1 and 3. The exponential distribution leads to more successful longer walks and thus more longer rules, while the uniform distribution leads to a better exploration of the neighborhood around the start node for shorter walks.

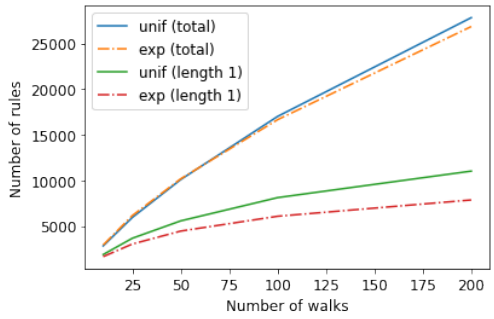


Figure 3: Total number of learned rules and number of rules for length 1.

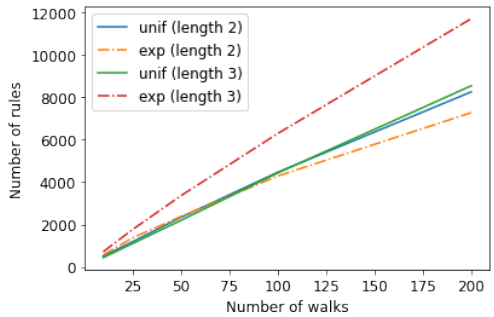


Figure 4: Number of rules for lengths 2 and 3.

Training and inference time The rule learning and rule application times are shown in the figures 5 and 6, dependent on the number of extracted temporal walks during learning.

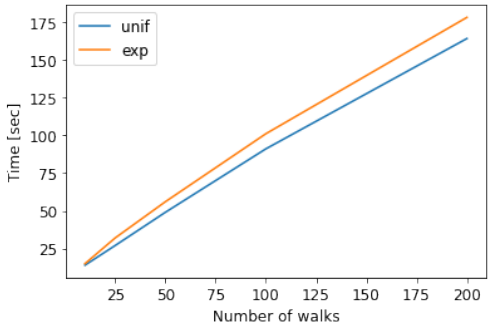


Figure 5: Rule learning time.

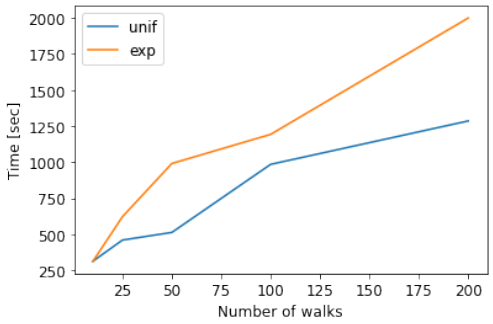


Figure 6: Rule application time.

The worst-case time complexity for learning rules of length l is $\mathcal{O}(|\mathcal{R}|nlDb)$, where n is the number of walks, D the maximum node degree in the training set, and b the number of body samples for estimating the confidence. The worst-case time complexity for inference is given by $\mathcal{O}(|\mathcal{G}| + |\mathcal{TR}_{rq}|D^L|\mathcal{E}|\log(k))$, where L is the maximum rule length in $\mathcal{TR}_{rq}$ and k the minimum number of candidates. More detailed steps of the algorithms for understanding these complexity estimations are given by Algorithm 3 and Algorithm 4.

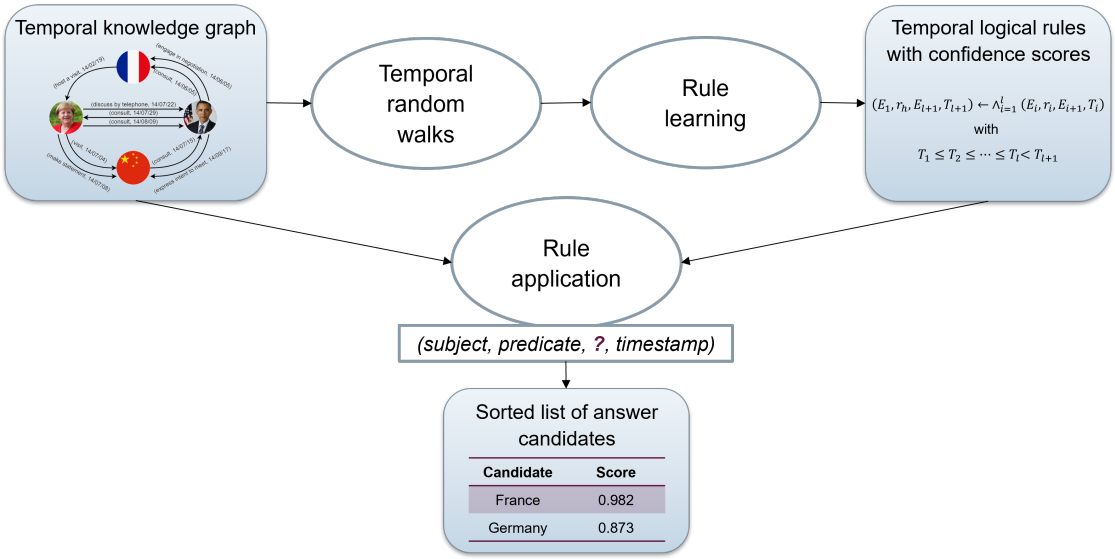


Figure 7: Overall framework.

Algorithm 3: Rule learning (detailed)

Input: Temporal knowledge graph $\mathcal{G}$.
Parameters: Rule lengths $\mathcal{L} \subset \mathbb{N}$, number of temporal random walks $n \in \mathbb{N}$, transition distribution $d \in \{\text{unif}, \exp\}$.
Output: Temporal logical rules $\mathcal{TR}$.

for relation $r \in \mathcal{R}$ **do**
  for $l \in \mathcal{L}$ **do**
    for $i \in [n]$ **do**
      $\mathcal{TR}_r^l \leftarrow \emptyset$
      According to transition distribution d**, sample a temporal random walk** W **of length** $l+1$ **with** $t_{l+1} > t_l$**.** ▷ See (4).
      Sample uniformly a start edge (e_s, r, e_o, t) with edge type r.
      for step $m \in \{2, \ldots, l+1\}$ **do**
        Retrieve adjacent edges of current object node.
        if $m = 2$ **then**
          Filter out all edges with timestamps greater than or equal to the current timestamp.
        else
          Filter out all edges with timestamps greater than the current timestamp.
          Filter out the inverse edge of the previously sampled edge.
        if $m = l+1$ **then**
          Retrieve all filtered edges that connect the current object to the source of the walk.
        Sample the next edge from the filtered edges according to distribution d.
        break if there are no feasible edges because of temporal or cyclic constraints.
      Transform walk W **to the corresponding temporal logical rule** R**.** ▷ See (5).
      Save information about the head relation and body relations.
      Define variable constraints for recurring entities.
      Estimate the confidence of rule R**.**
      Sample b body groundings. For each step $m \in \{1, \ldots, l\}$, filter the edges according to the correct body relation and the timestamps required to fulfill the temporal constraints.
      For successfully sampled body groundings, check the variable constraints.
      For each unique body, check if the rule head exists in the graph.
      Calculate rule support / body support.
      $\mathcal{TR}_r^l \leftarrow \mathcal{TR}_r^l \cup \{(R, \text{conf}(R))\}$
  $\mathcal{TR}_r \leftarrow \cup_{l \in \mathcal{L}} \mathcal{TR}_r^l$
$\mathcal{TR} \leftarrow \cup_{r \in \mathcal{R}} \mathcal{TR}_r$
return $\mathcal{TR}$

Algorithm 4: Rule application (detailed)

Input: Test query $q = (e^q, r^q, ?, t^q)$, temporal logical rules $\mathcal{TR}$, temporal knowledge graph $\mathcal{G}$.
Parameters: Time window $w \in \mathbb{N} \cup \{\infty\}$, minimum number of candidates k, score function f.
Output: Answer candidates $\mathcal{C}$.

$\mathcal{C} \leftarrow \emptyset$
▷ Apply the rules in $\mathcal{TR}$ by decreasing confidence.
Retrieve subgraph $\mathcal{SG} \subset \mathcal{G}$ with timestamps $t \in [t^q - w, t^q)$.
▷ Only done if the timestamp changes. The queries in the test set are sorted by timestamp.
Retrieve edges with timestamps $t \in [t^q - w, t^q)$.
Store edges for each relation in a dictionary.
if $\mathcal{TR}_{r^q} \neq \emptyset$ **then**
  for rule $R \in \mathcal{TR}_{r^q}$ **do**
    Find all body groundings of R in $\mathcal{SG}$.
    Retrieve edges that could constitute walks that match the rule's body. First, retrieve edges whose subject matches e^q and the relation the first relation in the rule body. Then, retrieve edges whose subject match one of the current targets and the relation the next relation in the rule body.
    Generate complete walks by merging the edges on the same target-source entity.
    Delete all walks that do not comply with the time constraints.
    Check variable constraints, and delete the walks that do not comply with the variable constraints.
    Retrieve candidates $\mathcal{C}(R)$ from the target entities of the walks.
    for $c \in \mathcal{C}(R)$ **do**
      Calculate score $f(R, c)$. ▷ See (6).
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{(c, f(R, c))\}$
    if $|\{c \mid \exists R : (c, f(R, c)) \in \mathcal{C}\}| \geq k$ **then**
      break
return $\mathcal{C}$

Chapter 6

A Knowledge Graph Perspective on Supply Chain Resilience

This chapter contains the publication

> **Yushan Liu**, Bailan He, Marcel Hildebrandt, Maximilian Buchner, Daniela Inzko, Roger Wernert, Emanuel Weigel, Dagmar Beyer, Martin Berbalk, and Volker Tresp. A knowledge graph perspective on supply chain resilience. In *The 2nd International Workshop on Linked Data-Driven Resilience Research, Extended Semantic Web Conference*, volume urn:nbn:de:0074-3401-3, pages 1-11, 2023. URL: `https://ceur-ws.org/Vol-3401/paper3.pdf`

A Knowledge Graph Perspective on Supply Chain Resilience

Yushan Liu[1,4,*], Bailan He[1,4], Marcel Hildebrandt[1], Maximilian Buchner[1], Daniela Inzko[1], Roger Wernert[3], Emanuel Weigel[2], Dagmar Beyer[1], Martin Berbalk[1] and Volker Tresp[1,4]

[1]*Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany*
[2]*Siemens AG, Östliche Rheinbrückenstraße 50, 76187 Karlsruhe, Germany*
[3]*Siemens Schweiz AG, Theilerstraße 1a, 6300 Zug, Switzerland*
[4]*Ludwig-Maximilians-Universität München, Geschwister-Scholl-Platz 1, 80539 Munich, Germany*

Abstract

Global crises and regulatory developments require increased supply chain transparency and resilience. Companies do not only need to react to a dynamic environment but have to act proactively and implement measures to prevent production delays and reduce risks in the supply chains. However, information about supply chains, especially at the deeper levels, is often intransparent and incomplete, making it difficult to obtain precise predictions about prospective risks. By connecting different data sources, we model the supply network as a knowledge graph and achieve transparency up to tier-3 suppliers. To predict missing information in the graph, we apply state-of-the-art knowledge graph completion methods and attain a mean reciprocal rank of 0.4377 with the best model. Further, we apply graph analysis algorithms to identify critical entities in the supply network, supporting supply chain managers in automated risk identification.

Keywords

Supply Chain Resilience, Knowledge Graphs, Machine Learning, Graph Analytics

1. Introduction

Global crises such as pandemics, natural disasters, and economic events as well as political and regulatory developments lead to increasing requirements regarding supply chain transparency and resilience. To ensure smooth procurement and production processes, it is essential for companies to react timely and flexibly to dynamic conditions and incidents to prevent production delays and bottlenecks within the supply network.

Usually, only direct (tier-1) suppliers of a company are tracked in supply chain management tools. The knowledge of subsuppliers is often limited and disregarded for decision making. In a survey, almost 80% of the companies cannot even name the number of their tier-n ($n \geq 2$) suppliers [1], let alone their names and locations. Intransparent supply chains make it highly challenging to achieve precise forecasts and react in the best way in case of disruptions.

Second International Workshop on Linked Data-driven Resilience Research (D2R2'23) co-located with ESWC 2023, May 28th, 2023, Hersonissos, Greece
*Corresponding author.
yushan.liu@siemens.com (Y. Liu)

Besides the intransparency of supply chains, another challenge is posed by the decentralized storage of relevant data and their incompleteness. The data come from different sources and are stored in various formats and locations. The disconnectedness makes it difficult to get a good overview of the situation and available information. Some information is also generally hard to retrieve, e. g., the exact production location of a material. Even if the supplier that delivers the material is known, the exact production site is often unknown.

Supply chain management involves monitoring supply chains to ensure their operability. Due to the inherent domain complexity and the high volume of data, significant blind spots at deeper levels of the supply chains remain, which matters because many of today's most pressing supply shortages (e. g., in the semiconductor industry) happen at these deeper tiers. Therefore, possible risks in the supply chains need to be identified early, i. e., constellations in the supply chains that lack the resistance to withstand disruptive events. For example, constellations can be critical if many suppliers are located in the same region, multiple tier-1 suppliers buy from the same subsupplier, only one supplier is related to a specific business scope, etc. After identifying possible criticalities, strategic decisions and mitigation measures can be derived within the organization. Risk identification is often based on domain knowledge and manual efforts. 75% of the companies in a survey see a need for improvement with respect to risk identification methods, where the potential of machine learning approaches is valued highly [1].

In this paper, we aim at increasing supply chain resilience, based on data from Siemens, by addressing the challenges mentioned above in the following ways:

- Supply chain intransparency and data disconnectedness: We collect and connect supply chain-related data from different sources and create a knowledge graph, which contains information from Siemens suppliers up to tier 3.
- Data incompleteness: We apply state-of-the-art knowledge graph completion methods for link prediction in the knowledge graph to predict missing information.
- Identification of criticalities: We use graph analysis algorithms to identify critical entities in the supply network, where we focus on centrality measures to derive an importance score for each supplier.

The remainder of this paper is organized as follows. Section 2 outlines related work, and Section 3 describes the supply chain knowledge graph. In Section 4, we apply knowledge graph completion methods to the data, while in Section 5, we use graph analytics to find criticalities in the supply chains. The conclusion and further research directions follow in Section 6.

2. Related Work

The application of machine learning for supply chain management is becoming an increasingly active field of research [2]. While many supervised machine learning methods (e.g., decision trees, support vector machines, and neural networks) were successfully applied to tasks related to supply chain design, planning, and execution [2, 3], not many works exist in the area of knowledge graphs and graph machine learning.

In 2018, Brintrup et al. [4] published the first work to apply link prediction to a supply network. They modeled the supply network as a homogeneous graph (i. e., containing one

relation type) with handcrafted embeddings and defined a binary classification task to predict new links in the graph. A follow-up work used graph neural networks to predict the supplier relationship between companies [5]. Gopal and Chang [6] also used graph neural networks to predict new supplier relationships, where they included external information about companies, e. g., industry classification and revenue segmentation, as features. Lu and Chen [7] discovered potential partnerships between companies based on graph projections and connectivity patterns. Aziz et al. [8] represented supply networks as heterogeneous graphs (i. e., containing several relation types, also commonly referred to as knowledge graphs) and applied a relational graph convolutional network for link prediction.

3. Knowledge Graph Dataset

The supply chain knowledge graph is constructed from both Siemens-internal and external sources to reflect both internal knowledge such as tier-1 suppliers, business scopes, and Siemens parts and external knowledge such as public data about smelters and substances. The information about tier-2 and tier-3 suppliers of Siemens is obtained mainly from public customs data, and a small part is obtained from private customs data and public media. There are in total 16,910 tier-1, 43,759 tier-2, and 49,775 tier-3 suppliers of Siemens, where the suppliers at different tier levels are not mutually exclusive. The graph is modeled via the graph data platform Neo4j.

We define a knowledge graph (KG) as a collection of triples $\mathcal{G} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ denotes the set of entities and $\mathcal{R}$ the set of relation types. Elements in $\mathcal{E}$ correspond to supply chain-related entities, e. g., suppliers, smelters, and components, and are represented as nodes in the graph. Every entity has a unique entity type, which is defined by the mapping $t : \mathcal{E} \to \mathcal{T}$, where $\mathcal{T}$ stands for the set of entity types. The entities are connected via relation types specified in $\mathcal{R}$, represented as directed edges in the graph. All entity and relation types and corresponding

Table 1
Entity and relation type statistics. In the graph, there are 8 entity types, where most nodes are suppliers, and 11 relation types, where most edges are from the type *supplies_to*.

Entity type	Nodes	Relation type	Edges
Supplier	61,234	supplies_to	138,197
Manufacturer Part	1,650	related_to	59,894
Siemens Part	1,295	belongs_to	56,663
Smelter	340	located_in	30,107
Substance	321	includes	10,088
Component	233	produces	7,831
Country	172	produced_in	4,381
Business Scope	32	same_as	1,847
		manufactured_by	1,564
		contains	764
		refines	340
Total	65,277	Total	311,676

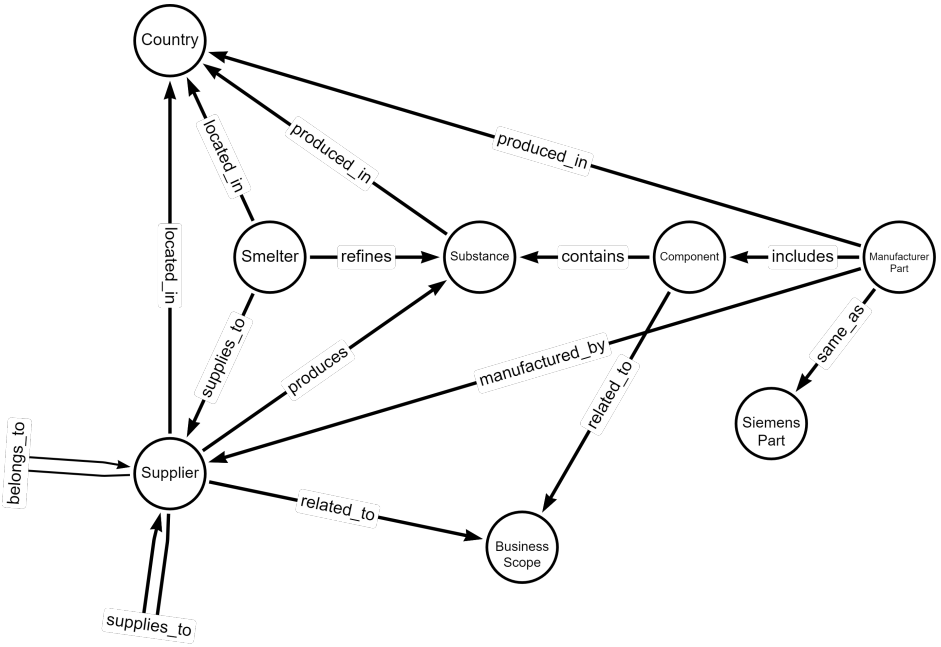


Figure 1: Knowledge graph schema. There are 8 entity types and 11 relation types.

numbers of nodes and edges are listed in Table 1.

Each relation type $r \in \mathcal{R}$ connects entities from a fixed set of source entity types $\mathcal{T}_{source}(r)$ to a fixed set of target entity types $\mathcal{T}_{target}(r)$. For example, $\mathcal{T}_{source}(\text{supplies\_to}) = \{\text{Supplier, Smelter}\}$ and $\mathcal{T}_{target}(\text{supplies\_to}) = \{\text{Supplier}\}$. The schema of the graph depicts the possible connections between the entity types and is shown in Figure 1.

A fact from the graph is represented by a triple $(\text{subject}, \text{predicate}, \text{object}) \in \mathcal{G}$, where the subject and object are entities and the predicate is the relation type that connects them, directed at the object. Triples in the graph are assumed to be true facts, while the truth value of non-existing triples could either be wrong or unknown (since the data are highly incomplete).

4. Knowledge Graph Completion

4.1. Object prediction task

Many KGs suffer from incompleteness, so a common reasoning task in graph machine learning is KG completion or link prediction. We formulate the link prediction problem as an object prediction task. Given a query of the form $(\text{subject}, \text{predicate}, ?)$, the goal is to predict a ranked list of entity candidates that are most likely the correct object of the query.

To measure the quality of the predictions, the mean reciprocal rank (MRR) and hits@k for $k \in \mathbb{N}$ are standard metrics used for link prediction on KGs. For a rank $x \in \mathbb{N}$, i. e., the position in the ranked list of entity candidates, the reciprocal rank is defined as $\frac{1}{x}$, and the MRR is the average of all reciprocal ranks of the correct query objects over all queries. The metric

hits@k represents the proportion of queries for which the correct object appears under the top k candidates.

4.2. Knowledge graph completion methods

There exists a variety of methods for KG completion [9]. In this paper, we focus on graph representation learning, where the underlying idea is to learn low-dimensional embeddings (i. e., vectors or matrices) for the entities and relation types in the graph that capture their semantic meanings. Based on these embeddings, a score can be calculated for each entity, indicating its likelihood to be the correct object of a query.

We apply the following traditional and state-of-the-art methods to the supply chain KG:

- RESCAL [10] was the first method to be published for learning KG embeddings. It models the KG as a three-way tensor and performs tensor factorization for relational learning tasks such as link prediction.
- ComplEx [11] was the first KG embedding method that learns embeddings in the complex vector space. It is based on tensor factorization and the Hermitian dot product.
- TuckER [12] is a tensor factorization method based on the Tucker decomposition. It can be seen as a generalized version of RESCAL and ComplEx.
- TransE [13] was the first translational method, which models the relations between two entities as translations in the vector space.
- RotatE [14] is a roto-translational method, which models the relations between two entities as rotations in the complex vector space.
- ConvE [15] was the first method that uses convolutional neural networks to model the interactions between entities.
- RGCN [16] consists of a relational graph convolutional network for encoding the entities and a tensor factorization method for scoring.
- CompGCN [17] incorporates composition operators to learn joint embeddings for entities and relation types. It is a generalized version of RGCN.

4.3. Experimental setup

For all KG completion methods, we use the implementations provided by the Python library PyKEEN [18]. We split the graph into training, validation, and test dataset, where we operate under the transductive setting, i. e., all entities and relation types from the validation and test set are also included in the training set. The training set consists of 65,277 nodes and 249,340 triples, while both the validation and test set have 31,168 triples. The number of nodes for the validation and test set is 22,212 and 22,213, respectively. All three datasets include all entity and relation types. We use the optimizer Adam and optimize the margin ranking loss with a margin of 1, where one negative triple is sampled for each training triple. We tune the hyperparameters embedding size in the range $\{16, 32, 64, 256, 512, 1024\}$ and learning rate in the range $\{0.0001, 0.001, 0.01\}$. The training of the model is stopped early if there is no improvement regarding the metric hits@10 on three subsequent evaluations on the validation set, where the evaluation takes place every 10 epochs.

Table 2
Results on the test dataset for the object prediction task. The best results are displayed in bold.

Method	MRR	Hits@1	Hits@3	Hits@10
RESCAL	0.1476	0.0684	0.1809	0.2772
ComplEx	0.2535	0.1793	0.2850	0.3949
TuckER	0.1738	0.0749	0.1878	0.4033
TransE	0.1595	0.0873	0.1733	0.3164
RotatE	**0.4377**	**0.3686**	**0.4733**	**0.5627**
ConvE	0.2289	0.1549	0.2438	0.3875
RGCN	0.2911	0.1784	0.3379	0.5195
CompGCN	0.2223	0.1271	0.2486	0.4229

4.4. Results

Table 2 displays the results of the best models for the selected KG completion methods. All models were able to learn useful embeddings from the training set, while RotatE performed best with respect to all metrics. For more than 36% of the test queries, RotatE predicts the correct object as the highest-ranked entity in the candidates list, and for almost half of the queries, RotatE is able to predict the correct object under the top three entities. Out of the three tensor factorization methods (RESCAL, ComplEx, and TuckER), ComplEx, which learns embeddings in the complex vector space, performs best. Out of the three neural network-based methods (ConvE, RGCN, and CompGCN), RGCN performs best, while ConvE and CompGCN have similar performance. Before materializing the results in the graph, domain experts should check the predicted triples for plausibility.

In Figure 2, the results of the best models are shown for each relation type. For every model, the performance with respect to the MRR is colored from best (green) to worst (red). RotatE performs best for all relation types except for *locate_in* (where all neural network-based models have higher MRR) and *refines* (where ComplEx and RGCN are better). The models show varying degrees of performance for the different relation types, where most models tend to have good results on *related_to*, *includes*, and *belongs_to* and bad performance on *same_as*, *contains*, and *refines*. Especially *same_as* is worst for all models. The reason lies in the graph schema and structure, where the *same_as* relation type connects manufacturer parts and Siemens parts, which do not have any other connections. Without further information, it is difficult to predict the correct Siemens part as query object.

5. Graph Analytics

5.1. Graph analysis algorithms

Supply chain managers mainly decide based on tier-1 supplier data whether there are risks in the supply chains, and they are usually directly informed by tier-1 suppliers if there are already existing problems. If additional data about tier-*n* suppliers are available, more detailed and precise decisions can be made, and mitigation measures in an earlier phase of the supply chain

MRR	RESCAL	ComplEx	TuckER	TransE	RotatE	ConvE	RGCN	CompGCN
supplies_to	0.1422	0.2661	0.0539	0.0740	0.3499	0.1574	0.2116	0.1718
related_to	0.2900	0.2756	0.4317	0.3539	0.7256	0.4876	0.5025	0.3291
belongs_to	0.0039	0.3411	0.3428	0.6671	0.6675	0.0261	0.0548	0.4000
located_in	0.0003	0.0653	0.1500	0.0909	0.1526	0.1726	0.2241	0.1973
includes	0.0004	0.5259	0.4176	0.5084	0.8682	0.0734	0.4607	0.4390
produces	0.0006	0.2923	0.0341	0.0984	0.3975	0.0113	0.1845	0.2122
produced_in	0.0003	0.1939	0.2907	0.1131	0.3539	0.0813	0.2043	0.1668
same_as	0.0002	0.0022	0.0011	0.0003	0.0490	0.0001	0.0049	0.0020
manufactured_by	0.0005	0.5646	0.3831	0.1420	0.9564	0.0707	0.3648	0.1791
contains	0.0005	0.0212	0.0180	0.0024	0.2106	0.0015	0.1143	0.1362
refines	0.0014	0.4593	0.0152	0.0337	0.1501	0.0166	0.4567	0.0402

Best

Worst

Figure 2: Results of the best models for each relation type. For each model, the performance with respect to the MRR is colored from best (green) to worst (red).

can be enabled. However, this kind of approach is mainly reactive, and manual decision making is not scalable to complex supply networks. Therefore, we propose to use graph analytics to support supply chain managers by automatically identifying critical suppliers so that they can be monitored more closely and mitigation strategies can be derived together.

For the graph analysis, we use the Neo4j Graph Data Science library and concentrate on the subgraph consisting of supplier entities and the relation type *supplies_to*. We calculate the following centrality and community detection metrics, which serve as a basis for deriving the importance or criticality of a supplier:

- The **degree centrality** measures the number of incoming and outgoing edges for each node. The number of incoming edges represents the number of suppliers and the number of outgoing edges the number of customers for each company. Companies with high in- or out-degree might be affected by disruptive events more often.
- The **betweenness centrality** for each node is based on the number of shortest paths between all node pairs that the node lies on. A company with high betweenness connects many companies and is more likely to cause a bottleneck.
- The **closeness centrality** measures the average length of the shortest paths between a node and all other nodes. A company with high closeness is a central customer for many suppliers.
- The **triangle count** is a community detection measure that calculates the number of adjacent triangles of a node. A company with a high triangle count is part of an interconnected supply network.

To make the suppliers better comparable, we normalize the metrics in-degree, out-degree, betweenness, closeness, and triangle count to be between 0 and 10 and sum them up to obtain an aggregated importance score for each supplier in the graph.

5.2. Results

When comparing the aggregated scores of the suppliers, Siemens is obviously the center of the supply network and has an aggregated score of 37.25. The next supplier has a score of only 16.66, and there are only 3 suppliers with a score above 15. There are in total 988 suppliers with

Table 3
Correlation matrix. There is a high correlation between in-degree, betweenness, and triangle count.

Correlation	in-degree	out-degree	betweenness	closeness	triangle count
in-degree	1.0000				
out-degree	0.1969	1.0000			
betweenness	0.8816	0.3928	1.0000		
closeness	0.0686	0.2792	0.0859	1.0000	
triangle count	0.9809	0.2048	0.8774	0.0542	1.0000

a score above 10, which might be critical entities in the supply network and should be examined by domain experts. Since an aggregated score loses information, the suppliers with the highest values for each metric should be analyzed in more detail. Table 3 shows the correlation matrix of the five metrics. There is a high correlation between in-degree, betweenness, and triangle count. That means, companies with many suppliers often lie on a large number of shortest paths (supply chains) and are part of a highly interconnected supply network.

Figure 3 illustrates a possible way to visualize the results in order to identify critical paths in the supply network. The subgraph contains yellow and red nodes, which represent suppliers, and purple nodes, which represent business scopes. The red suppliers have aggregated scores above 10 and might be more critical than the yellow suppliers. Any supply chain containing a critical supplier might have a higher risk. The size of the purple nodes corresponds to the number of suppliers related to the corresponding business scopes. The orange edges represent the edge type *supplies_to* and the blue edges the edge type *related_to*. If there are business scopes to which only one supplier is related, then the supplier might be critical since a delay of this supplier would not be compensated easily by another supplier within the same business scope. In the figure, three such business scopes can be identified (purple nodes with blue circles), where two of the corresponding suppliers also have a critical score.

6. Conclusion and Further Research Directions

Challenges for supply chain management include supply chain intransparency, data disconnectedness and incompleteness, and the scalable identification of criticalities in the supply network. We addressed these challenges by modeling supply chain-related information as a knowledge graph. We used state-of-the-art knowledge graph completion methods to predict missing links and applied graph analysis algorithms to compute importance scores for all suppliers. Based on the importance scores and the graph structure, critical supply chains could be identified, which is an essential step towards more resilient supply networks.

For further research, we propose the following possible directions:

- Integration of node and edge properties: In this paper, we only focused on the graph structure for link prediction and graph analysis. For some entity and relation types, however, there exist properties that could be helpful for prediction. For example, the

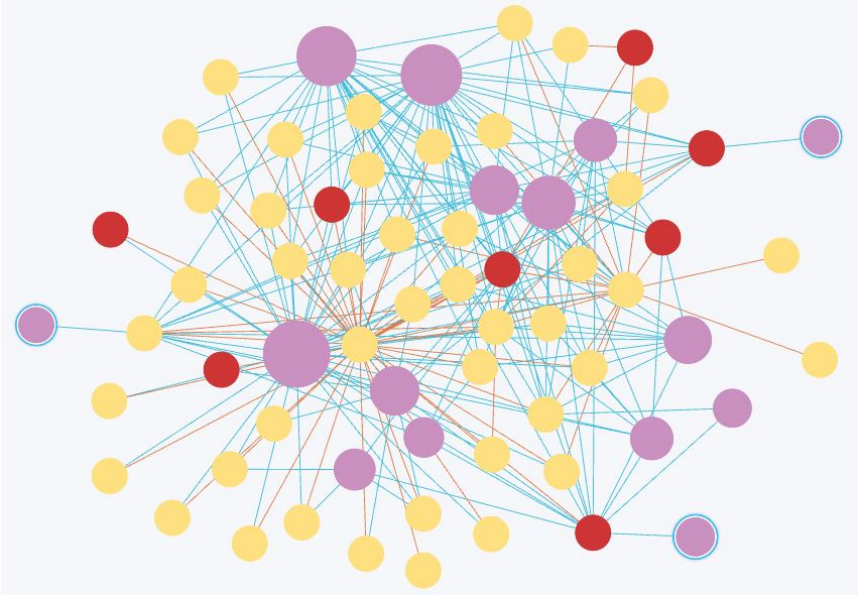


Figure 3: Visualization of a subgraph. The yellow nodes and red nodes represent suppliers, where the red suppliers have aggregated scores above 10. Business scopes are represented as purple nodes. The size of the purple nodes correlates to the number of suppliers related to the corresponding business scopes. The orange edges represent the edge type *supplies_to* and the blue edges the edge type *related_to*.

relation type *produced_in* between a substance and a country has the property Herfindahl-Hirschmann-Index, a measure of market concentration. For the prediction of the relation type *located_in*, the company name could be a good indicator. These properties could be integrated when learning embeddings or calculating importance scores.

- Node regression or classification: Besides link prediction, node regression or classification are common tasks on knowledge graphs. Given, e.g., risk scores or categories for a subset of companies, one could learn risk scores or categories for companies that are missing this information in the graph.
- Analysis of the complete graph: We conducted the graph analysis based on a subgraph containing the suppliers and the *supplies_to* relation type. To calculate the importance scores, more information from the graph could be included. For example, a supplier that is located in a country with a high sustainability risk might also have a higher risk, or a supplier that manufactures many Siemens parts would be more critical for Siemens.

Acknowledgments

This work has been supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of the project CoyPu under grant number 01MK21007K.

References

[1] M. Brylowski, M. Schröder, W. Kersten, Machine Learning im Supply Chain Risk Management: Studie, Technical Report, Technische Universität Hamburg, 2021.

[2] M. Brylowski, M. Schröder, S. Lodemann, W. Kersten, Machine learning in supply chain management: A scoping review, in: Proceedings of the Hamburg International Conference of Logistics, volume 31, 2021.

[3] E. B. Tirkolaee, S. Sadeghi, F. M. Mooseloo, H. R. Vandchali, S. Aeini, Application of machine learning in supply chain management: A comprehensive overview of the main areas, in: Mathematical Problems in Engineering, volume 2021, 2021.

[4] A. Brintrup, P. Wichmann, P. Woodall, D. McFarlane, E. Nicks, W. Krechel, Predicting hidden links in supply networks, in: Complexity, volume 2018, 2018.

[5] E. E. Kosasih, A. Brintrup, A machine learning approach for predicting hidden links in supply chain with graph neural networks, in: International Journal of Production Research, volume 60, 2021.

[6] A. Gopal, C. Chang, Discovering supply chain links with augmented intelligence, in: The Second ACM International Conference on AI in Finance, Workshop Natural Language Processing and Network Analysis in Financial Applications, 2021.

[7] Z.-G. Lu, Q. Chen, Discovering potential partners via projection-based link prediction in the supply chain network, in: International Journal of Computational Intelligence Systems, volume 13, 2020.

[8] A. Aziz, E. E. Kosasih, R.-R. Griffiths, A. Brintrup, Data considerations in graph representation learning for supply chain networks, in: The Thirty-Eighth International Conference on Machine Learning, Workshop Machine Learning for Data: Automated Creation, Privacy, Bias, 2021.

[9] M. Wang, L. Qiu, X. Wang, A survey on knowledge graph embeddings for link prediction, in: Symmetry, volume 13, 2021.

[10] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: The Twenty-Eighth International Conference on Machine Learning, 2011.

[11] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: The Thirty-Third International Conference on Machine Learning, 2016.

[12] I. Balažević, C. Allen, T. M. Hospedales, TuckER: Tensor factorization for knowledge graph completion, in: The 2019 Conference on Empirical Methods in Natural Language Processing and the Ninth International Joint Conference on Natural Language Processing, 2019.

[13] A. Bordes, N. Usunier, A. Garcia-Durán, Translating embeddings for modeling multi-relational data, in: The Twenty-Seventh Conference on Neural Information Processing Systems, 2013.

[14] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, RotatE: Knowledge graph embedding by relational rotation in complex space, in: The Seventh International Conference on Learning Representations, 2019.

[15] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2D knowledge graph embeddings, in: The Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[16] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: The Fifteenth Extended Semantic Web Conference, 2018.
[17] S. Vashishth, S. Sanyal, V. Nitin, P. Talukdar, Composition-based multi-relational graph convolutional networks, in: The Eighth International Conference on Learning Representations, 2020.
[18] M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann, PyKEEN 1.0: A Python library for training and evaluating knowledge graph embeddings, in: Journal of Machine Learning Research, volume 22, 2021.

Chapter 7

On Calibration of Graph Neural Networks for Node Classification

This chapter contains the publication

> Tong Liu*, **Yushan Liu***, Marcel Hildebrandt, Mitchell Joblin, Hang Li, and Volker Tresp. On calibration of graph neural networks for node classification. In *The 2022 International Joint Conference on Neural Networks (IJCNN)*, July 2022. *Equal contribution. DOI: 10.1109/IJCNN55064.2022.9892866

On Calibration of Graph Neural Networks for Node Classification

1st Tong Liu
LMU Munich
Munich, Germany
tong.liu@physik.uni-muenchen.de

1st Yushan Liu
Siemens AG, LMU Munich
Munich, Germany
yushan.liu@siemens.com

2nd Marcel Hildebrandt
Siemens AG
Munich, Germany
marcel.hildebrandt@siemens.com

3rd Mitchell Joblin
Siemens AG
Munich, Germany
mitchell.joblin@siemens.com

4th Hang Li
Siemens AG, LMU Munich
Munich, Germany
hang.li@siemens.com

5th Volker Tresp
Siemens AG, LMU Munich
Munich, Germany
volker.tresp@siemens.com

***Abstract*—Graphs can model real-world, complex systems by representing entities and their interactions in terms of nodes and edges. To better exploit the graph structure, graph neural networks have been developed, which learn entity and edge embeddings for tasks such as node classification and link prediction. These models achieve good performance with respect to accuracy, but the confidence scores associated with the predictions might not be calibrated. That means that the scores might not reflect the ground-truth probabilities of the predicted events, which would be especially important for safety-critical applications. Even though graph neural networks are used for a wide range of tasks, the calibration thereof has not been sufficiently explored yet. We investigate the calibration of graph neural networks for node classification, study the effect of existing post-processing calibration methods, and analyze the influence of model capacity, graph density, and a new loss function on calibration. Further, we propose a topology-aware calibration method that takes the neighboring nodes into account and yields improved calibration compared to baseline methods.**

***Index Terms*—Graph neural networks, calibration, node classification**

I. Introduction

Learning graph representations for relational data structures has been gaining increasing attention in the machine learning community [1], [2]. A graph is able to model real-world, complex systems by representing entities as nodes and interactions between them as edges. Since the information in graphs is often incomplete, e.g., missing node attributes or edges, relevant graph-related tasks for attaining new knowledge include node classification and link prediction. A variety of graph neural network (GNN) models have been developed [3]–[5], which learn node and edge embeddings in a low-dimensional vector space. Subsequently, these embeddings can be used to solve downstream tasks like node classification. Usually, the focus here lies on maximizing the accuracy – the proportion of nodes that are classified correctly. GNNs achieve good performance with respect to accuracy but are also black boxes and lack interpretability.

Most machine learning models output confidence scores associated with the predictions, and the concept of calibration captures the idea that the score should reflect the ground-truth probability of the prediction's correctness. For example, if 100 instances have a score of 0.6 for a specific class k, then 60 instances are expected to actually be of class k. A real-world application is autonomous driving, where the model should not only be aware that the object in front of the car is more likely to be a plastic bag than a pedestrian but also know how much more likely it is. A score distribution of 0.99 for plastic bag and 0.01 for pedestrian or 0.51 for plastic bag and 0.49 for pedestrian could have a huge influence on the next action of the car. Generally, calibrated scores lead to a better interpretation of the results and increase the trustworthiness of machine learning models, which is especially important in safety-critical domains.

The calibration of deep neural networks has been addressed in several works [6]–[9]. The calibration of GNNs, however, has not been sufficiently explored yet, and existing calibration methods do not exploit the graph structure. Due to the different architectures of GNNs compared to neural networks, GNNs might exhibit different calibration characteristics. In this work, we are interested in the following research questions:

R1. How are GNNs calibrated for the node classification task, and are existing calibration methods sufficient to calibrate GNNs?

R2. How do model capacity (width and depth) and graph density influence the calibration?

R3. Can a calibration error term be added to the loss function in a straightforward way to improve the calibration without hurting the accuracy?

R4. Can the incorporation of topological information improve calibration?

To better understand the calibration properties of GNNs, we conduct an empirical analysis of several GNN models in a node classification setting. Based on our experimental finding that the nodes in the graph express different levels of over- and underconfidence, we propose a topology-aware calibration method that takes the neighboring nodes into account. Our contributions are summarized as follows:

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

- We inspect the calibration of five representative GNN models on three benchmark citation datasets for node classification.
- We analyze the influence of model capacity, graph density, and a new loss function on the calibration of GNNs.
- We propose a calibration method that takes the graph topology into account and yields improved calibration compared to state-of-the-art post-processing calibration methods.

In Section II, we define the necessary concepts and summarize related work. The existing GNNs and calibration methods used in this work are also described briefly. An experimental study on the calibration of GNNs is presented in Section III ($\rightarrow$ **R1**, **R2**, **R3**). In Section IV ($\rightarrow$ **R1**, **R4**), we propose a topology-aware calibration method and show experimental results compared to state-of-the-art calibration baselines. The results are discussed in Section V.

II. Background

A. Definitions

1) Node classification on graphs: An undirected graph is defined as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges. An edge $e = \{i, j\} \in \mathcal{E}$ connects the two nodes i and j in the graph. The information about the edges can be encoded in an adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$. With $\mathbf{A}_{ij}$ being the entry in the i-th row and j-th column of $\mathbf{A}$, we define $\mathbf{A}_{ij} = 1$ if $\{i, j\} \in \mathcal{E}$ and $\mathbf{A}_{ij} = 0$ otherwise[1]. Moreover, we define $\mathcal{N}(i)$ as the set of neighbors of node i. For attributed graphs, where each node i is associated with a d-dimensional feature vector $\mathbf{X}_i \in \mathbb{R}^d$, we denote the feature matrix by $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$.

The goal of the node classification task is to assign each node $i \in \mathcal{V}$ a class label $\hat{y}_i \in \mathcal{K} := \{1, 2, \ldots, K\}$, where K stands for the total number of classes.

2) Calibration: Let $\mathbf{H}_i \in \mathbb{R}^h$ denote the node embedding and $y_i \in \mathcal{K}$ the ground-truth label of sample (or node) $i \in \mathcal{V}$. Let $g : \mathbb{R}^h \rightarrow [0,1]^K$ be a function that takes $\mathbf{H}_i$ as input and outputs a probability vector $g(\mathbf{H}_i)$, where $g(\mathbf{H}_i)_k$ represents the k-th element. The predicted class label for sample i is given by $\hat{y}_i = \arg\max_{k\in\mathcal{K}} g(\mathbf{H}_i)_k$, where $\hat{p}_i = \max_{k\in\mathcal{K}} g(\mathbf{H}_i)_k$ is called the corresponding confidence score for $\hat{y}_i$. Perfect calibration is defined as $\mathbb{P}(\hat{y}_i = y_i \mid \hat{p}_i = p) = p$ for all $p \in [0,1]$ and any sample i [6].

A reliability diagram [10] plots accuracy against confidence to visualize the calibration of the model (see Fig. 1). More formally, the samples are grouped into $M \in \mathbb{N}$ equally-spaced interval bins according to their confidences $\hat{p}_i$. For each bin B_m, $m \in \{1, 2, \ldots, M\}$, the accuracy and average confidence are calculated according to

$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \mathbb{1}[\hat{y}_i = y_i] \quad \text{and} \tag{1}$$

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \hat{p}_i \ , \tag{2}$$

[1]We identify nodes and indices to ease the notation.

respectively, where $|B_m|$ denotes the number of samples in bin B_m and $\mathbb{1}$ the indicator function. In case of perfect calibration, the equation $\mathrm{acc}(B_m) = \mathrm{conf}(B_m)$ holds for all m. Reliability diagrams also present a way to identify if the model is over- or underconfident. If the bars are above the diagonal line, it implies that the accuracy is higher than the average confidence, and the model is called underconfident. If the bars are below the diagonal, the model is overconfident.

The expected calibration error (ECE) [11] measures the miscalibration by averaging the gaps in the reliability diagram and is given by

$$\sum_{m=1}^{M} \frac{|B_m|}{N} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \ , \tag{3}$$

where N is the total number of samples.

The marginal ECE (MECE) approximates the marginal calibration error [12], which takes all classes into account. For each bin and every class k, it compares the average confidence of samples for class k to the proportion of samples that has as ground-truth label class k. The MECE is defined as

$$\sum_{k=1}^{K} w_k \sum_{m=1}^{M} \frac{1}{N} \left| \sum_{i\in B_m} \mathbb{1}[y_i = k] - \sum_{i\in B_m} g(\mathbf{H}_i)_k \right| \ , \tag{4}$$

where w_k is a class-dependent weight factor, which is set to $1/K$ if all classes are equally important.

B. Related work

Guo et al. [6] showed that modern neural networks are miscalibrated and tend to be overconfident, i. e., the confidence scores are higher than the proportions of correct predictions. They proposed temperature scaling, a single-parameter variant of Platt scaling [13], to calibrate the results. Several other methods were introduced to improve the calibration of deep neural networks (e. g., mixup training [7] and FALCON [8]). Methods that improve calibration by preventing overconfidence include label smoothing [9] and focal loss [14], [15]. In GNNs, calibration issues have only been studied recently. A first evaluation of GNNs was done by Teixeira et al. [16], who performed experiments on multiple node classification datasets and concluded that GNNs are miscalibrated and existing calibration methods are not always able to improve the calibration to the desired extent.

C. Methods

1) Graph neural networks: Given an adjacency matrix $\mathbf{A}$ and a feature matrix $\mathbf{X}$, the idea of all GNNs is to learn node embeddings $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times h}$. The embedding for node i is denoted by $\mathbf{H}_i \in \mathbb{R}^h$, which can be fed to a task-specific decoder g. For example, since we are concerned with node classification, we use a single-layer perceptron with softmax activation as decoder. For our experiments, we select the widely used models graph convolutional network (GCN) [3], graph attention network (GAT) [4], and simple graph convolution (SGC) [17]. Further, we consider graph filter neural network (gfNN) [18], a straightforward extension of SGC, and

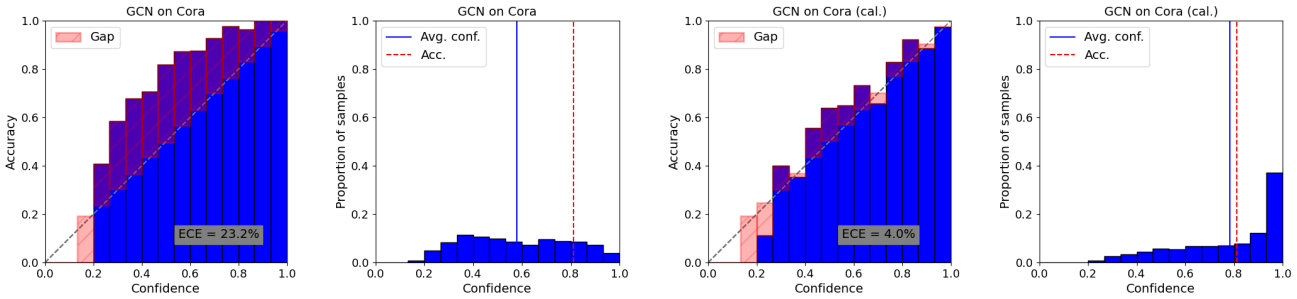


Fig. 1. Reliability diagrams and corresponding confidence histograms for GCN on Cora. The two left plots show the results before calibration, while the two right plots show the results after calibration with temperature scaling. The diagonal line indicates perfect calibration.

approximate personalized propagation of neural predictions (APPNP) [5], a model with state-of-the-art performance.

GCN applies a normalized adjacency matrix with self-loops $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\hat{\mathbf{D}}^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{D}}$ the degree matrix of $\mathbf{A}+\mathbf{I}$. Concretely, the hidden layer of a GCN is formed according to $\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$, where $\mathbf{W}^{(l)}$ is a trainable weight matrix, σ an activation function, and $\mathbf{H}^{(0)} := \mathbf{X}$. GCN aggregates information from a node's neighbors by computing the normalized sum of adjacent node embeddings.

GAT differs from GCN in the neighbor aggregation function by introducing an attention mechanism that scales the importance of neighbors when summing over their embeddings.

SGC is a GCN without nonlinear activation functions between the layers, resulting from the authors' conjecture that the good performance of GCNs comes from the aggregation of local neighborhood information and not from the application of nonlinear feature maps.

gfNN extends SGC with a nonlinear layer σ so that the node embeddings for layer l are obtained from $\mathbf{H}^{(l)} = \sigma(\tilde{\mathbf{A}}^{l}\mathbf{X}\mathbf{W})$, where $\mathbf{W}$ is a trainable weight matrix.

APPNP is based on the personalized PageRank (PPR) algorithm [19]. The node embeddings in layer $l+1$ are calculated via $\mathbf{H}^{(l+1)} = (1-\alpha)\tilde{\mathbf{A}}\mathbf{H}^{(l)} + \alpha\mathbf{H}^{(0)}$, where $\alpha \in (0,1]$ is a hyperparameter and $\mathbf{H}^{(0)} := f(\mathbf{X})$, with f being a trainable neural network.

2) Calibration methods: We consider the classical post-processing methods histogram binning [20], isotonic regression [21], and Bayesian binning into quantiles (BBQ) [11], which is a refinement of histogram binning. Further, we include temperature scaling [6] as a multiclass calibration method and Meta-Cal [22], a recently introduced approach with state-of-the-art performance.

Histogram binning divides the confidence scores $\hat{p}_i$ into M bins and assigns a new score $\hat{q}_m$ to each bin to represent the calibrated confidences. The scores $\hat{q}_m$ are learned by minimizing $\sum_{m=1}^{M}\sum_{i\in B_m}(\hat{q}_m - y_i)^2$.

Isotonic regression learns a piecewise constant function f by minimizing $\sum_{m=1}^{M}\sum_{i\in B_m}(f(\hat{p}_i) - y_i)^2$. It is a generalization of histogram binning where the bin boundaries and scores are jointly optimized.

BBQ extends histogram binning and learns a distribution $\mathbb{P}(\hat{q}_i \mid \hat{p}_i, \mathcal{D}_{\text{val}})$ by marginalizing out all possible binnings, where $\mathcal{D}_{\text{val}}$ is the validation set.

Temperature scaling is a single-parameter extension of Platt scaling [13] for multiple classes. Given the output logit vector $\mathbf{z}$ before the softmax activation, a rescaling $\mathbf{z}/T$ depending on a temperature $T > 0$ is applied.

Meta-Cal combines temperature scaling as a base model with a bipartite ranking model to weaken the limitation of accuracy-preserving calibration methods. By investigating two practical constraints (miscoverage rate control and coverage accuracy control), the goal is to improve calibration depending on the bipartite ranking while controlling the accuracy.

III. Experimental Study

A. Setup

Experiments We first inspect the calibration of GNN models on benchmark citation datasets, where we take the best hyperparameter and training settings from the corresponding original papers.

Then, we empirically analyze the influence of model capacity (width and depth) on calibration. It has been observed that stacking too many GCN layers drastically worsens the performance, which is partly attributed to a phenomenon called oversmoothing [23]. Oversmoothing happens when repeated neighbor aggregation leads to similar node embeddings in the graph, and various methods have been proposed to tackle this problem [5], [24]. In the following, we investigate if increasing model depth also affects calibration.

One of the core mechanisms of GNNs is the message aggregation from neighboring nodes. We examine how graph density, i. e., the ratio of the number of edges in the graph to the number of maximum possible edges, influences the calibration performance.

Finally, we also test a new loss function (5) that combines the standard cross-entropy loss L_{ce} with an ECE-inspired term L_{cal} for optimizing the calibration. We define L_{cal} as the cross entropy between the confidence of the sample and the accuracy of its corresponding bin, where the idea is that the confidence should stay close to the accuracy. Given the original cross-entropy loss L_{ce}, we define the new loss as

TABLE I
DATASET STATISTICS.

Dataset	K	d	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Label rate
Cora	7	1,433	2,708	5,429	0.052
Citeseer	6	3,703	3,327	4,732	0.036
Pubmed	3	500	19,717	44,338	0.003

$$L = \alpha L_{\text{ce}} + (1-\alpha) L_{\text{cal}} \quad \text{with} \tag{5}$$

$$L_{\text{cal}} = -\sum_{i=1}^{N} \text{acc}(B_m(i)) \cdot \log(\hat{p}_i) \ , \tag{6}$$

where $\alpha \in (0, 1)$ and $B_m(i)$ denotes the bin that sample i belongs to.

For the experiments on width, depth, graph density, and the new loss function, we focus on GCN and GAT, two of the basic and most widely used GNN models.

Datasets Cora, Citeseer, and Pubmed[2] are three commonly used benchmark datasets for node classification. They are citation networks, where nodes represent scientific publications and edges between pairs of nodes correspond to one publication citing the other. Each node comes with a d-dimensional feature vector that indicates the presence of words from a predefined vocabulary. The class label of a node is the topic of the corresponding publication. Similar to previous works [3], [4], we operate under a semi-supervised setting, where only a small amount of labeled data is available during training. The statistics of the datasets are summarized in Table I.

Implementation The GNN models are implemented using the PyTorch-Geometric library[3]. The bin number for calculating the ECE and MECE is set to 15. More information about the hyperparameters and experimental settings can be found in the supplementary material[4].

B. Results

Uncalibrated results We run all GNNs on the three citation datasets and show the uncalibrated performance with respect to accuracy, ECE, and MECE in Table II. The method APPNP is best on Cora and Pubmed in terms of accuracy (second-best on Citeseer), and it is also best calibrated on the datasets Citeseer and Pubmed. For Cora, gfNN has the lowest ECE and MECE. All models except for gfNN[5] have stable calibration values with small standard deviations. The worst method with respect to the calibration performance is SGC, which is, apart from the softmax activation for normalization, the only linear model. Adding a nonlinear layer as in gfNN results in better calibration. Moreover, we find that GAT outperforms GCN in terms of ECE and MECE in two of three datasets.

Influence of width We compare the calibration of GCN and GAT for varying model width, i. e., the number of hidden dimensions per layer. While the accuracy basically stays constant, the ECE and MECE decrease with increasing number of hidden dimensions initially (see Fig. 2). When a certain width is reached, the calibration values stagnate or slightly increase again. Generally, wider networks tend to be better calibrated.

[2]https://github.com/kimiyoung/planetoid

[3]https://github.com/pyg-team/pytorch_geometric

[4]Source code and supplementary material available at https://github.com/liu-yushan/calGNN.

[5]The original paper trains for 50 epochs without early stopping. A different training setting might stabilize the results more.

TABLE II
UNCALIBRATED PERFORMANCE WITH RESPECT TO ACCURACY, ECE, AND MECE (MEAN±SD OVER 100 INDEPENDENT RUNS). THE BEST RESULTS ARE DISPLAYED IN BOLD.

Dataset	Model	Acc.	ECE	MECE
Cora	GCN	81.43±0.60	23.51±1.89	7.01±0.46
	GAT	83.14±0.39	17.26±1.09	5.15±0.30
	SGC	81.19±0.05	26.03±0.16	7.78±0.08
	gfNN	78.73±5.04	**6.45±2.44**	**3.16±1.45**
	APPNP	**83.68±0.36**	14.90±0.69	4.73±0.17
Citeseer	GCN	71.32±0.70	21.80±1.21	8.57±0.32
	GAT	70.99±0.60	18.92±1.05	7.66±0.30
	SGC	**72.46±0.15**	53.59±0.14	19.15±0.00
	gfNN	67.33±6.58	15.50±4.47	8.40±1.26
	APPNP	72.10±0.38	**11.93±0.80**	**5.37±0.28**
Pubmed	GCN	79.23±0.43	10.62±1.28	7.29±0.84
	GAT	79.05±0.38	14.37±0.48	9.89±0.23
	SGC	78.72±0.04	22.40±0.04	14.98±0.02
	gfNN	77.94±2.32	6.04±2.90	5.23±2.65
	APPNP	**80.09±0.25**	**4.38±0.74**	**3.59±0.41**

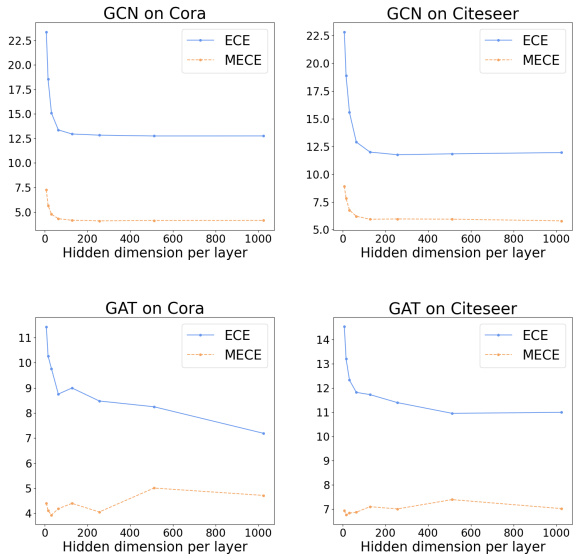


Fig. 2. Varying model width (hidden dimension per layer).

Influence of depth We investigate the influence of model depth, i. e., the number of layers, on the calibration performance (see Fig. 3). Oversmoothing becomes particularly pronounced when the test accuracy decreases significantly with increasing number of layers. The ECE first improves when changing from two to three layers, then it increases again until five or six layers. Using an even larger model depth, the ECE eventually decreases again.

Influence of graph density For this experiment, we remove

TABLE III
UNCALIBRATED PERFORMANCE OF GCN AND GAT UNDER THE STANDARD AND THE NEW LOSS FUNCTION (MEAN±SD OVER 10 INDEPENDENT RUNS). THE BETTER RESULTS WHEN COMPARING THE TWO LOSS FUNCTIONS ARE UNDERLINED.

Dataset	Model	Acc. (L_{ce})	ECE (L_{ce})	MECE (L_{ce})	Acc. (L)	ECE (L)	MECE (L)
Cora	GCN	81.43±0.60	23.51±1.89	7.01±0.46	81.81±0.85	14.91±2.7	4.64±0.56
	GAT	83.14±0.39	17.26±1.09	5.15±0.30	82.73±0.40	5.29±1.31	2.41±0.32
Citeseer	GCN	71.32±0.70	21.80±1.21	8.57±0.32	71.67±0.50	14.65±0.42	6.43±0.26
	GAT	70.99±0.60	18.92±1.05	7.66±0.30	71.13±0.44	9.39±0.97	4.58±0.40
Pubmed	GCN	79.23±0.43	10.62±1.28	7.29±0.84	79.00±0.37	7.50±0.91	5.60±0.73
	GAT	79.05±0.38	14.37±0.48	9.89±0.23	78.88±0.40	9.58±0.79	6.91±0.63

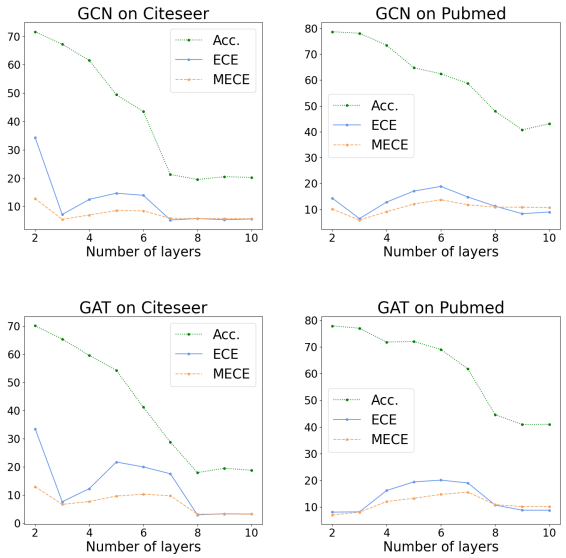


Fig. 3. Varying depth (number of layers).

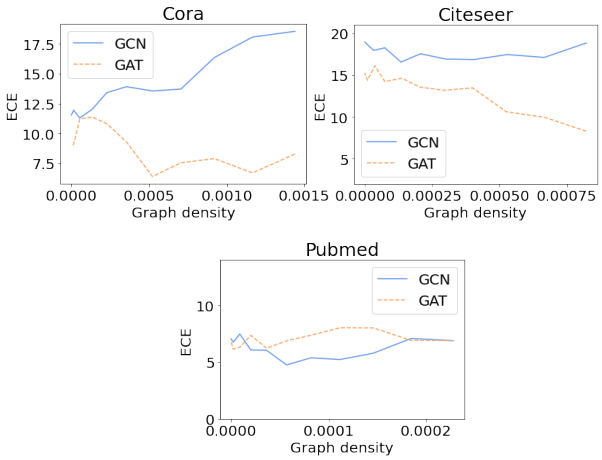


Fig. 4. Influence of graph density. The graph density is the ratio of the number of edges in the graph to the number of maximum possible edges.

different proportions of edges randomly from the dataset, ranging from 0% (original dataset) to 100% (no graph structure at all). The models GCN and GAT only differ in the aggregation mechanism, i. e., GAT introduces attention coefficients to weigh the importance of neighbors. The results are shown in Fig. 4. Similar to Table II, the ECE of GAT is consistently lower than the ECE of GCN on Cora and Citeseer, while on Pubmed, GCN expresses partly better calibration. Generally, the graph density of Pubmed is the lowest. It might be that the attention weights in GAT are beneficial for calibration and especially useful when enough edges exist in the graph.

Influence of new loss function Table III compares the results of the standard cross-entropy loss L_{ce} and the new loss function L from (5), which contains a calibration error term. The new loss L improves the model calibration in all cases while keeping the accuracy at the same level or even slightly increasing the accuracy.

Underconfidence vs. overconfidence Taking the best hyperparameter and training settings from their corresponding publications, all GNNs exhibit underconfidence on all three datasets, i. e., the confidence scores are lower than the accuracy of the predictions (see Fig. 1 and figures in the supplementary material). In some cases, however, we find that the model changes from underconfidence to overconfidence if it is trained without early stopping.

The left plot in Fig. 5 shows the test accuracy, scaled test negative log-likelihood (NLL), and scaled test ECE for a 4-layer GCN on Cora during training, with a weight decay set to 5e-4. Around epoch 150, the NLL and accuracy become stable, while the ECE is still improving. At this point, GCN is underconfident, as shown in the left-most reliability diagram in Fig. 6. In the epochs between 200 and 300, the ECE gains the best performance when the model changes from underconfidence to overconfidence (see the two diagrams in the middle of Fig. 6). After epoch 300, GCN starts to overfit with respect to the ECE, while the NLL and accuracy remain rather unchanged. During this process, overconfidence aggravates, and the ECE increases to 7.8% in epoch 400, which is displayed in the right-most diagram in Fig. 6.

In summary, GCN first optimizes NLL and accuracy during training, then fits the confidence scores, and eventually starts to overfit regarding the ECE without influencing the NLL and accuracy. When we slightly increase the weight decay to 7.5e-4 (see right plot in Fig. 5), the ECE stabilizes after reaching the optimal value, and the values of NLL and accuracy also stay in a smaller range compared to the left plot.

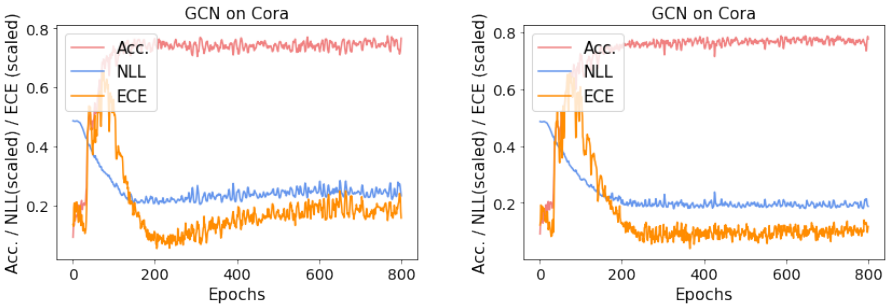


Fig. 5. Test accuracy, scaled test NLL, and scaled test ECE for GCN on Cora, with a weight decay of 5e-4 (left) and 7.5e-4 (right).

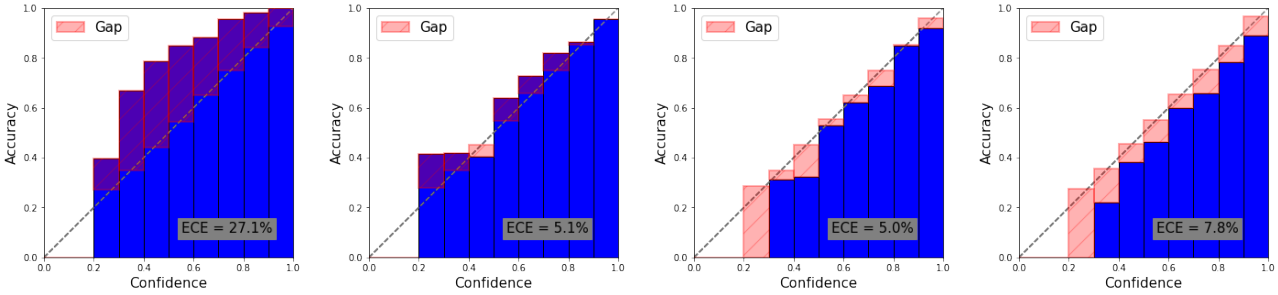


Fig. 6. Reliability diagrams for GCN on Cora during the training process. From left to right, the corresponding number of epochs is 100, 200, 300, and 400.

IV. Ratio-binned scaling for calibrating GNNs

A. Same-class-neighbor ratio

From our experiments, we find that GNN models tend to be underconfident. Even though the overall model exhibits underconfidence, there might be differences depending on node-level properties, which have not been considered before. Especially for graph data, the topology could provide structural information that is useful for calibration. For node classification, the class labels and properties of a node's neighbors have a significant influence on the classification. We calculate for each node i the same-class-neighbor ratio, i. e., the proportion of neighbors that have the same class as node i, and develop a new binning scheme that groups samples into bins based on the same-class-neighbor ratio for calibration.

To evaluate the correlation between the same-class-neighbor ratio and the confidence of a model, we calculate the ratio for each node based on the ground-truth labels. In Fig. 7, we group the nodes into 5 equally-spaced interval bins according to their ratios. Employing a trained GNN model, we compute the output of the classifier g for each node and draw the average confidence of the samples in each bin as a blue bar. The gap illustrates the difference between the average confidence and the accuracy in each bin. We observe that the average confidence increases with the same-class-neighbor ratio, where bins with higher ratios express underconfidence and bins with lower ratios overconfidence. Consequently, a binning scheme that groups samples depending on their same-class-neighbor ratios would take the graph structure into account and allow for an adaptive calibration depending on the confidence level of each bin.

B. Ratio-binned scaling

We propose ratio-binned scaling (RBS), a topology-aware method, which first approximates the same-class-neighbor ratio for each sample, then groups the samples into M bins, and finally learns a temperature for each bin to rescale the confidence scores.

In the semi-supervised setting, we only know the labels of a small number of nodes and therefore cannot use the true labels for binning. One natural option is to replace the nodes' ground-truth labels with their confidence scores for estimating the same-class-neighbor ratio. More precisely, we define the estimated ratio for node i as

$$\hat{r}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} g(\mathbf{H}_j)_{\hat{y}_i} \quad \in [0,1] \,, \tag{7}$$

where $\mathbf{H}_j$ is the node embedding of node j, which is learned by a GNN model, and g is the classifier. $g(\mathbf{H}_j)_{\hat{y}_i}$ denotes the confidence score of node j corresponding to the class $\hat{y}_i$ that is predicted for the central node i.

For the ratio-based binning scheme, let $\{B_m \mid 1 \leq m \leq M\}$ be a set of bins that partitions the interval $[0,1]$ uniformly. After calculating the output for all nodes, each node i is assigned to a bin according to its estimated same-class-neighbor ratio $\hat{r}(i)$, i. e., $B_1 = \{i \in \mathcal{V} \mid \hat{r}(i) \in [0, \frac{1}{M}]\}$ and $B_m = \{i \in \mathcal{V} \mid \hat{r}(i) \in (\frac{m-1}{M}, \frac{m}{M}]\}$ for $m \in \{2, \ldots, M\}$.

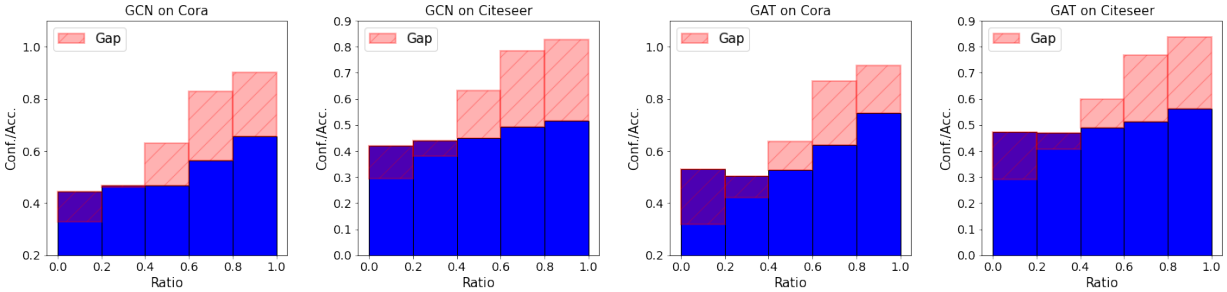


Fig. 7. The nodes are grouped according to their same-class-neighbor ratios. The blue bar represents the average confidence and the gap the difference between average confidence and accuracy in each bin.

TABLE IV
CALIBRATED PERFORMANCE WITH RESPECT TO ECE (MEAN±SD OVER 100 INDEPENDENT RUNS). THE BEST GNN MODEL FOR EACH CALIBRATION METHOD IS UNDERLINED. THE BEST CALIBRATION METHOD FOR EACH GNN MODEL IS DISPLAYED IN BOLD.

Dataset	Model	Uncal.	His. bin.	Iso. reg.	BBQ	Tem. scal.	Meta-Cal	RBS	RRBS
Cora	GCN	23.51±1.89	4.50±0.76	3.94±0.66	4.53±0.64	**3.82±0.60**	4.09±0.65	3.90±0.61	3.10±0.63
	GAT	17.26±1.09	4.86±0.62	4.04±0.59	4.18±0.60	3.53±0.66	**3.28±0.65**	3.34±0.63	2.67±0.54
	SGC	26.03±0.16	4.38±0.30	4.21±0.42	4.35±0.21	4.05±0.11	4.02±0.35	**3.55±0.07**	2.57±0.11
	gfNN	6.45±2.44	3.80±0.86	**3.72±0.78**	4.15±1.22	3.74±1.34	4.07±1.52	3.77±0.96	3.39±1.22
	APPNP	14.90±0.69	4.20±0.62	3.43±0.60	3.90±0.57	3.14±0.50	3.48±0.57	**3.01±0.53**	2.68±0.44
Citeseer	GCN	21.80±1.21	4.61±0.82	**4.46±0.93**	5.30±1.09	4.86±0.76	5.04±0.90	4.99±0.75	4.11±0.88
	GAT	18.92±1.05	5.00±0.73	4.90±0.69	5.04±0.71	5.92±0.58	6.08±0.61	**4.45±0.73**	4.71±0.67
	SGC	53.59±0.14	7.55±0.23	6.93±0.19	7.43±0.13	4.47±0.19	4.17±0.29	**4.04±0.17**	2.97±0.20
	gfNN	15.50±4.47	**4.74±1.00**	4.92±1.03	5.06±1.37	5.43±1.27	5.45±1.32	5.19±1.31	4.34±1.27
	APPNP	11.93±0.80	4.75±0.85	**4.50±0.67**	5.10±1.02	4.98±0.67	5.29±0.67	5.08±0.69	3.97±0.58
Pubmed	GCN	10.62±1.28	4.69±0.78	4.76±0.77	4.69±0.74	4.27±0.61	4.99±1.11	**4.16±0.60**	3.28±0.89
	GAT	14.37±0.48	4.85±0.89	4.93±0.78	5.70±1.03	3.94±0.67	4.45±0.74	**3.61±0.75**	2.56±0.56
	SGC	22.40±0.04	4.40±0.29	4.29±0.21	5.04±0.22	4.13±0.12	4.64±0.58	**4.07±0.17**	3.01±0.12
	gfNN	6.04±2.90	4.98±1.06	5.00±0.83	5.15±1.26	4.97±1.67	6.06±1.95	**4.91±1.65**	3.78±1.03
	APPNP	4.38±0.74	4.86±0.75	4.72±0.57	4.79±0.84	3.98±0.59	4.34±0.72	**3.80±0.60**	2.60±0.47

Let the output of g be in the form $g(\mathbf{H}_i) = \sigma(\mathbf{Z}_i) \in \mathbb{R}^K$ for node i, where σ is the softmax function and $\mathbf{Z}_i$ the logits before normalization. For each bin B_m, $m \in \{1, \dots, M\}$, a temperature $T_m > 0$ is learned on the validation dataset.

The calibrated confidence for a test node j is then given by

$$\hat{\mathbf{q}}_j = \sigma(\mathbf{Z}_j / T_m) \in [0,1]^K \quad \text{if } j \in B_m. \tag{8}$$

We apply temperature scaling for calibrating the nodes in each bin, but it would also be possible to apply other post-processing calibration methods for obtaining calibrated scores.

C. Results

Table IV summarizes the calibration performance of all considered post-processing calibration methods and our proposed method RBS in terms of ECE. All methods can improve the calibration of GNN models on Cora and Citeseer. In particular, the obtained ECE for a specific dataset and calibration method is rather similar for all GNNs regardless of the uncalibrated ECE. On Pubmed, most methods have difficulties improving calibration of APPNP, which already has low ECE. RBS gains the best performance in the majority of the cases and outperforms classical temperature scaling in 11 out of 15 experiments.

Next to a good calibration performance, accuracy preservation is desirable for calibration methods. RBS and temperature scaling do not change the ranking of the classes and thus the accuracy stays unchanged. Meta-Cal trades good calibration for lower accuracy, while the other methods yield comparable or even slightly improved accuracy in some cases (see supplementary material).

D. Effectiveness of real-ratio-binned scaling

Table IV also shows the calibrated results of real-ratio-binned scaling (RRBS), where we assume that the ground-truth labels are available for all nodes. RRBS outperforms the best calibration method in 14 out of 15 experiments. Although the correct labels are not accessible in the semi-supervised setting, the results still indicate the effectiveness of the intuition of our proposed method.

V. DISCUSSION

In general, the calibration performance depends on the specific GNN model and dataset, where all models perform best on Pubmed (see Table IV). When using the hyperparameter and training settings from the original publications, all GNNs tend to be underconfident on all three datasets, in contrast to the finding that deep neural network models rather exhibit

overconfidence [6], [7]. However, when plotting the reliability diagrams for a varying number of epochs, we observe that in some cases, underconfidence changes to overconfidence when the number of epochs increases. It seems that underconfidence or overconfidence is not necessarily a property of the model architecture but is also dependent on the training setting.

Most GNNs suffer from oversmoothing, which becomes apparent when increasing the number of layers in the model [5], [23], [24]. We observe that for large numbers of layers, the accuracy drops significantly, while the ECE improves. Oversmoothing results in similar node embeddings, which might be uninformative for the model. In this case, the model would most likely learn the distribution of classes in the training data as confidence scores. Therefore, all samples would be grouped into one bin, resulting in low ECE if the test distribution is close to the training distribution. However, such a model does not make use of the underlying graph structure and is probably not useful for application.

The results for RBS and RRBS show the potential of a calibration method that takes the graph structure into account, where the binning scheme is constructed depending on node-level properties. RRBS almost always outperforms RBS, which suggests that RBS might be especially helpful for cases where the estimated ratios are close to the real ratios, i.e., for models with relatively high accuracy. The number of bins for RBS was chosen from $\{2, 3, 4\}$, and it seems that even a small number of bins can lead to improved calibration compared to classical temperature scaling. It would further be interesting to apply RBS to other kinds of datasets, e.g., heterophilic graphs, where nodes from different classes are likely to be connected.

VI. Conclusion

We investigated the calibration of graph neural networks for node classification on three benchmark datasets. Graph neural networks seem to be miscalibrated, where the exact calibration depends on both the dataset and the model. Existing post-processing calibration methods are able to alleviate the miscalibration but do not consider the graph structure. Based on our experimental finding that the nodes in the graph express different levels of over- or underconfidence depending on their same-class-neighbor ratios, we proposed the topology-aware calibration method ratio-binned scaling. It takes the predictions of neighboring nodes into account and shows better performance compared to state-of-the-art baselines. For future work, it would be interesting to gain a more theoretical understanding of the calibration properties and conduct experiments on larger and a wider variety of datasets.

Acknowledgment

This work has been supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of the project RAKI under grant number 01MD19012C.

References

[1] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *arXiv:2005.03675*, 2021.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 40, pp. 52–74, 2017.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the Fifth International Conference on Learning Representations*, 2017.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the Sixth International Conference on Learning Representations*, 2018.

[5] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proceedings of the Seventh International Conference on Learning Representations*, 2019.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the Thirty-Fourth International Conference on Machine Learning*, 2017.

[7] S. Thulasidasan, G. Chennupati, J. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, 2019.

[8] C. Tomani and F. Buettner, "Towards trustworthy predictions from deep neural networks with fast adversarial calibration," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

[9] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" in *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, 2019.

[10] A. H. Murphy and R. L. Winkler, "Reliability of subjective probability forecasts of precipitation and temperature," *Applied Statistics*, vol. 26, pp. 41–47, 1977.

[11] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using Bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[12] A. Kumar, P. Liang, and T. Ma, "Verified uncertainty calibration," in *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems*, 2019.

[13] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, 1999.

[14] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. S. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *Proceedings of the Thirty-Fourth Conference on Neural Information Processing Systems*, 2020.

[15] N. Charoenphakdee, J. Vongkulbhisal, N. Chairatanakul, and M. Sugiyama, "On focal loss for class-posterior probability estimation: A theoretical perspective," in *Proceedings of the 2021 Conference on Computer Vision and Pattern Recognition*, 2021.

[16] L. Teixeira, B. Jalaian, and B. Ribeiro, "Are graph neural networks miscalibrated?" *arXiv:1905.02296*, 2019.

[17] F. Wu, T. Zhang, Jr. Souza, A. H. d., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the Thirty-Sixth International Conference on Machine Learning*, 2019.

[18] H. NT and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv:1905.09550*, 2019.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," *Stanford InfoLab*, 1998.

[20] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[21] ——, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the Eighth ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.

[22] X. Ma and M. B. Blaschko, "Meta-Cal: Well-controlled post-hoc calibration by ranking," in *Proceedings of the Thirty-Eighth International Conference on Machine Learning*, 2021.

[23] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI conference on Artificial Intelligence*, 2018.

[24] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, 2018.

SUPPLEMENTARY MATERIAL

GNN models We follow the best hyperparameter and training settings given by the corresponding graph neural network papers, using the PyTorch Geometric implementation[6] of GCN [3], GAT [4], SGC [17], and APPNP [5]. For SGC, we set the learning rate to 0.2, train for 100 epochs without early stopping, and tune weight decay for 60 iterations using Hyperopt[7], according to the original paper. We implement gfNN [18] by ourselves and follow the setting in the original paper. In order to capture the variance across different training runs, each model is run for 100 times, and we report the averaged results with standard deviations.

Width We conduct experiments on the influence of width on the models GCN and GAT, where we use the best hyperparameter settings and vary the hidden dimensions per layer in the parameter space given by $\{2^i \mid 3 \leq i \leq 10\}$. Dropout layers are removed, and the number of epochs is set to 200 with early stopping after 10 epochs without improvement of the validation loss. Each model is run for 10 times.

Depth We conduct experiments on the influence of depth on the models GCN and GAT, where we follow the experimental setting in Appendix B by Kipf and Welling [3]. The number of layers is in the parameter space $\{1, 2, \ldots, 10\}$. Each model is run for 10 times.

Graph density We conduct experiments on the influence of graph density on the models GCN and GAT. Different proportions of edges are removed randomly from 0% (original dataset) to 100% (no graph structure at all). Each model is run for 10 times.

New loss function We follow the setting by Tomani and Buettner [8], who also introduced an ECE-inspired loss function. An annealing coefficient is specified for the calibration error term since the early epochs are usually more focused on reaching the cross-entropy minimum. More precisely, we define

$$\text{anneal\_coef} = \lambda \cdot \min\left\{1, \frac{\text{epoch}}{\text{EPOCHS} \cdot \text{anneal\_max}}\right\}, \tag{9}$$

$$\tilde{L}_{\text{cal}} = \text{anneal\_coef} \cdot L_{\text{cal}}\,, \quad \text{and} \tag{10}$$

$$L = \alpha \cdot L_{\text{ce}} + (1 - \alpha) \cdot \tilde{L}_{\text{cal}}\,, \tag{11}$$

where epoch and EPOCHS are the current training epoch and the total number of epochs, respectively, and λ, anneal_max, and α are hyperparameters. We set $\text{anneal\_max} = 1$, $\lambda = 1$, and tune α on the validation set in the range $\{0.95, 0.96, 0.97, 0.98, 0.99\}$. Each model (fixed hyperparameter setting) is run for 10 times.

TABLE V
CALIBRATED ACCURACY (MEAN±SD OVER 100 INDEPENDENT RUNS). TEMPERATURE SCALING AND RBS DO NOT CHANGE THE ACCURACY.

Dataset	Model	Uncal.	His	Iso	BBQ	Meta
Cora	GCN	81.43±0.60	80.38±0.82	81.80±0.57	81.34±0.67	79.23±1.61
	GAT	83.14±0.39	81.39±0.48	84.05±0.51	83.52±0.59	79.99±1.70
	SGC	81.19±0.05	79.91±0.13	81.16±0.11	79.83±0.24	78.77±1.88
	gfNN	78.73±5.04	79.06±1.16	80.21±1.28	79.96±1.31	76.30±5.51
	APPNP	83.68±0.36	82.52±0.46	83.45±0.45	83.20±0.53	81.33±1.79
Citeseer	GCN	71.32±0.70	71.93±0.84	72.39±0.66	71.79±0.99	68.22±4.13
	GAT	70.99±0.60	71.78±0.56	72.21±0.52	71.81±0.70	68.28±2.63
	SGC	72.46±0.15	74.13±0.05	73.81±0.12	73.32±0.13	69.19±2.08
	gfNN	67.33±6.58	71.74±1.22	71.98±1.15	71.10±1.22	64.63±6.61
	APPNP	72.10±0.38	72.94±0.48	72.90±0.48	72.63±0.83	69.64±2.29
Pubmed	GCN	79.23±0.43	79.01±0.55	79.03±0.46	78.85±0.67	76.99±4.62
	GAT	79.05±0.38	78.50±0.61	78.85±0.38	78.19±0.56	78.00±1.43
	SGC	78.72±0.04	79.05±0.08	79.30±0.03	79.88±0.19	77.83±1.57
	gfNN	77.94±2.32	77.92±1.11	78.16±0.98	77.76±1.33	75.66±3.05
	APPNP	80.09±0.25	80.12±0.44	80.15±0.30	79.50±0.47	78.37±1.64

[6]https://github.com/pyg-team/pytorch_geometric/tree/master/benchmark/citation
[7]https://github.com/hyperopt/hyperopt

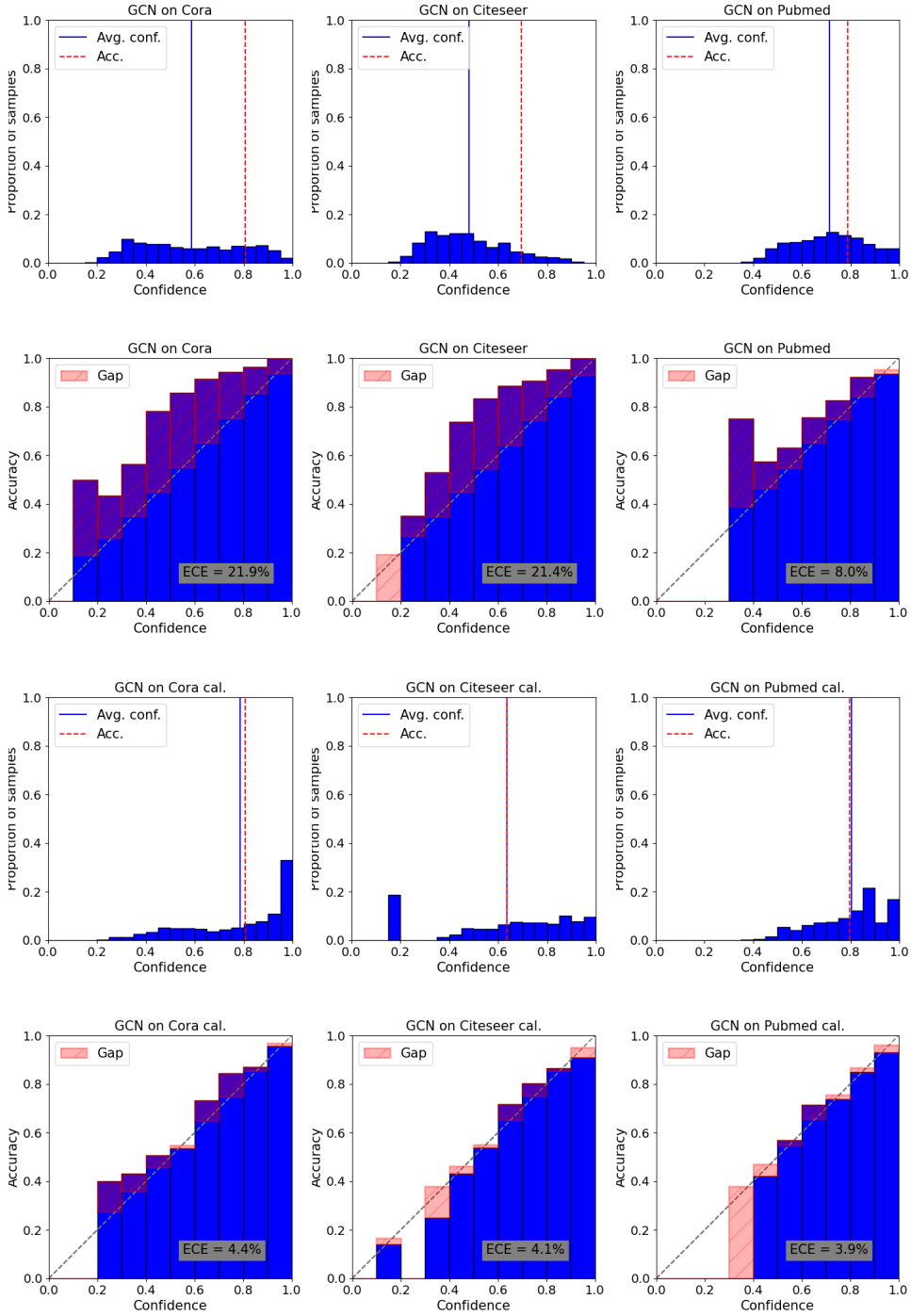


Fig. 8. Histograms and reliability diagrams for GCN.

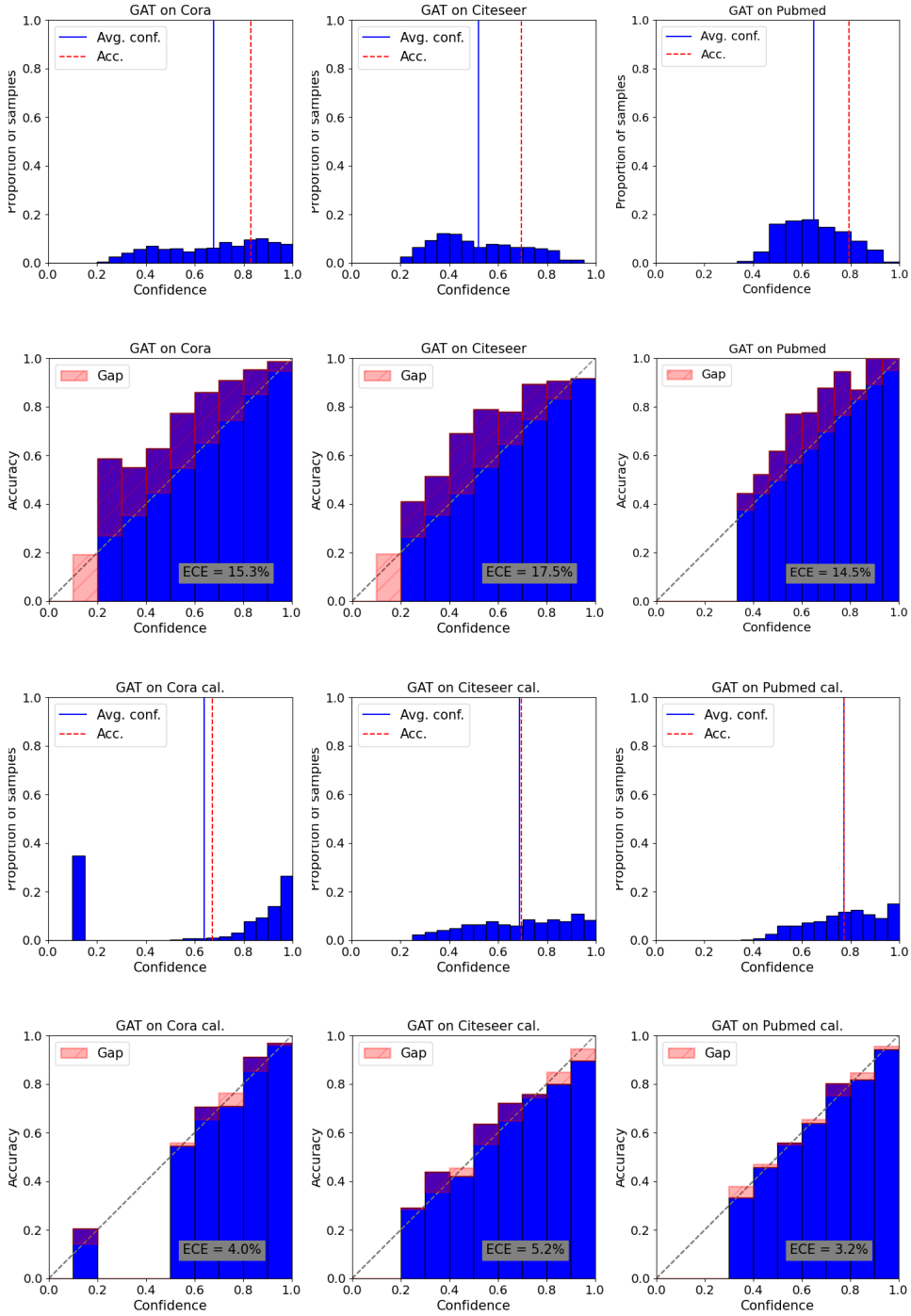


Fig. 9. Histograms and reliability diagrams for GAT.

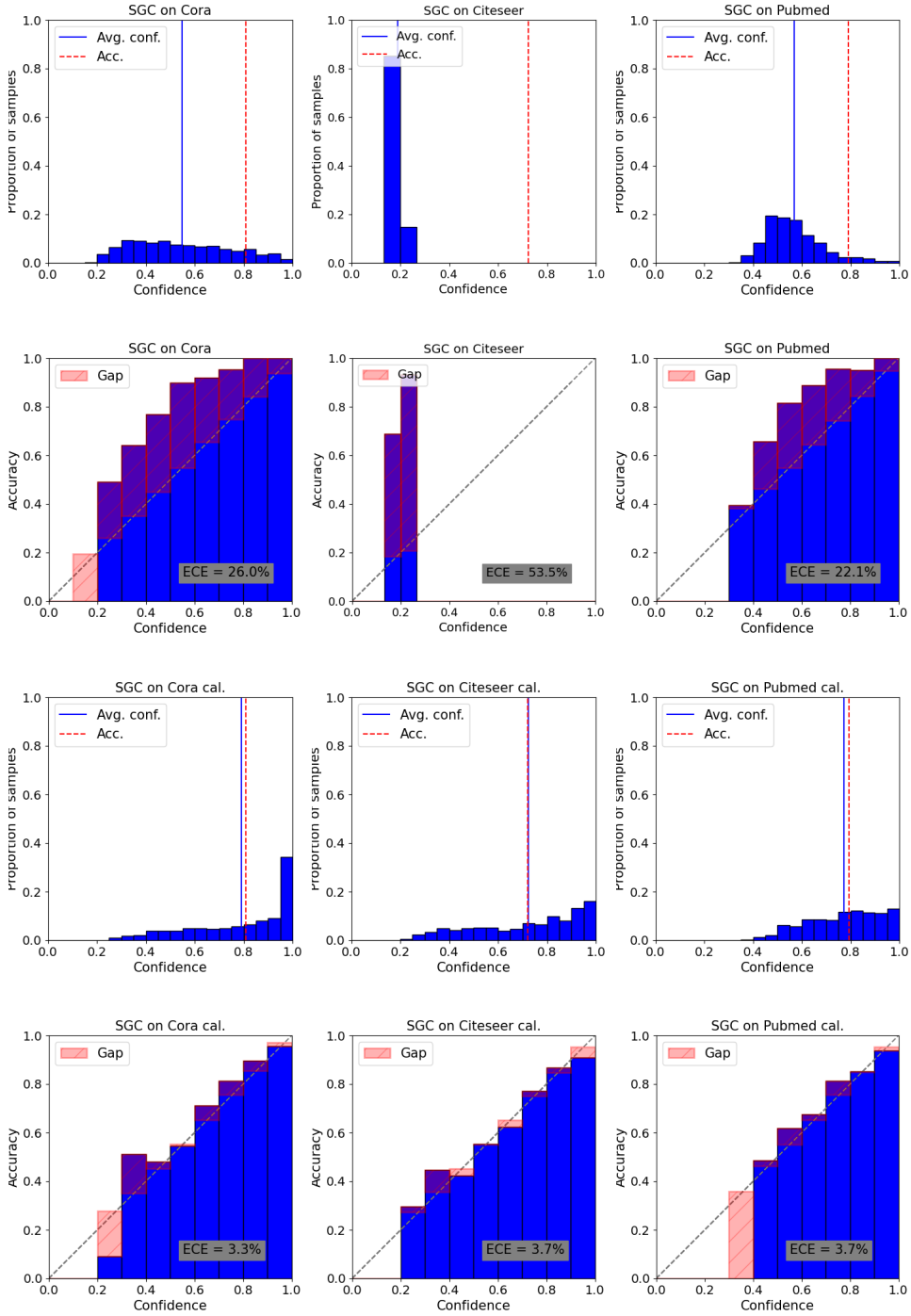


Fig. 10. Histograms and reliability diagrams for SGC.

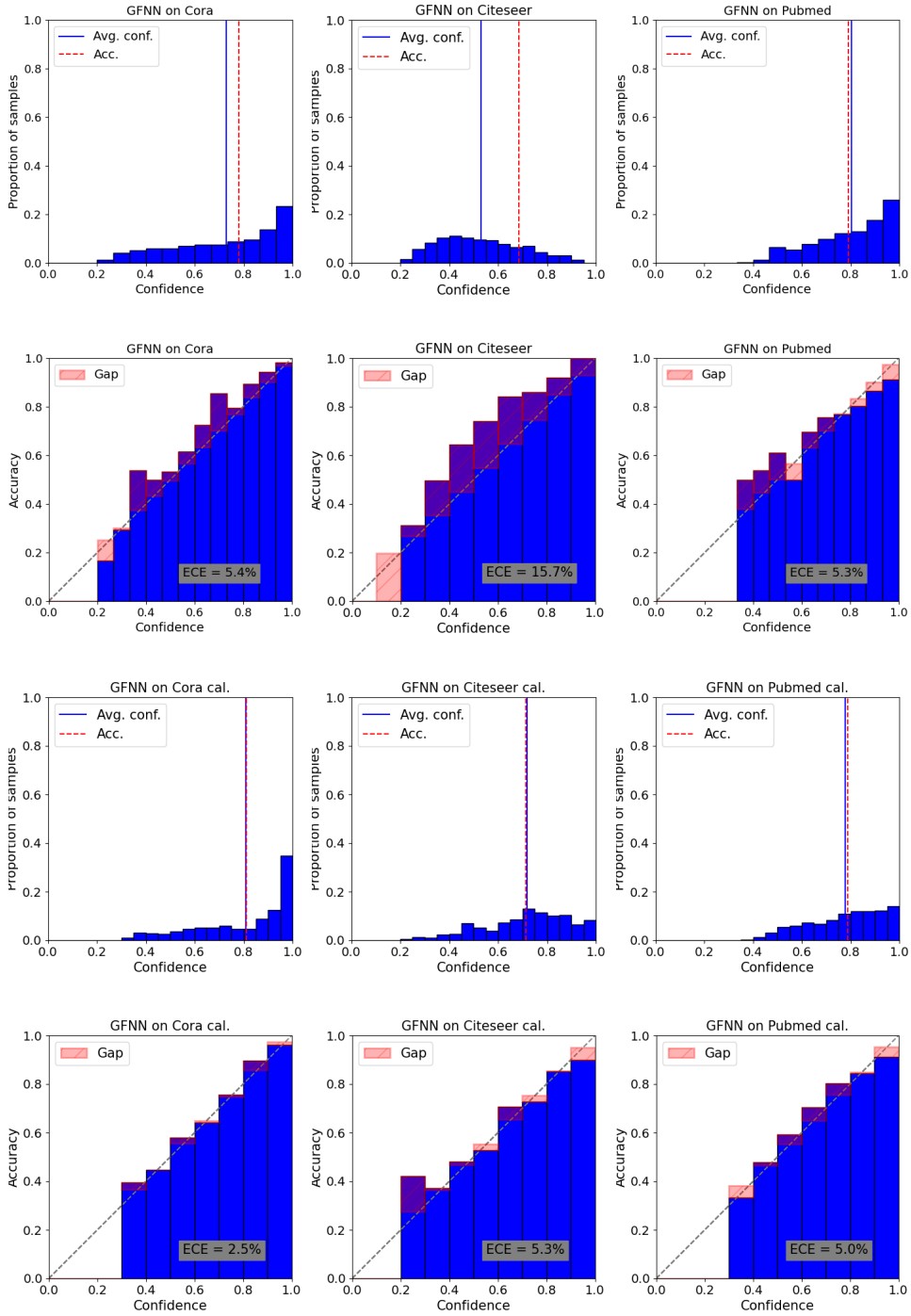


Fig. 11. Histograms and reliability diagrams for gfNN.

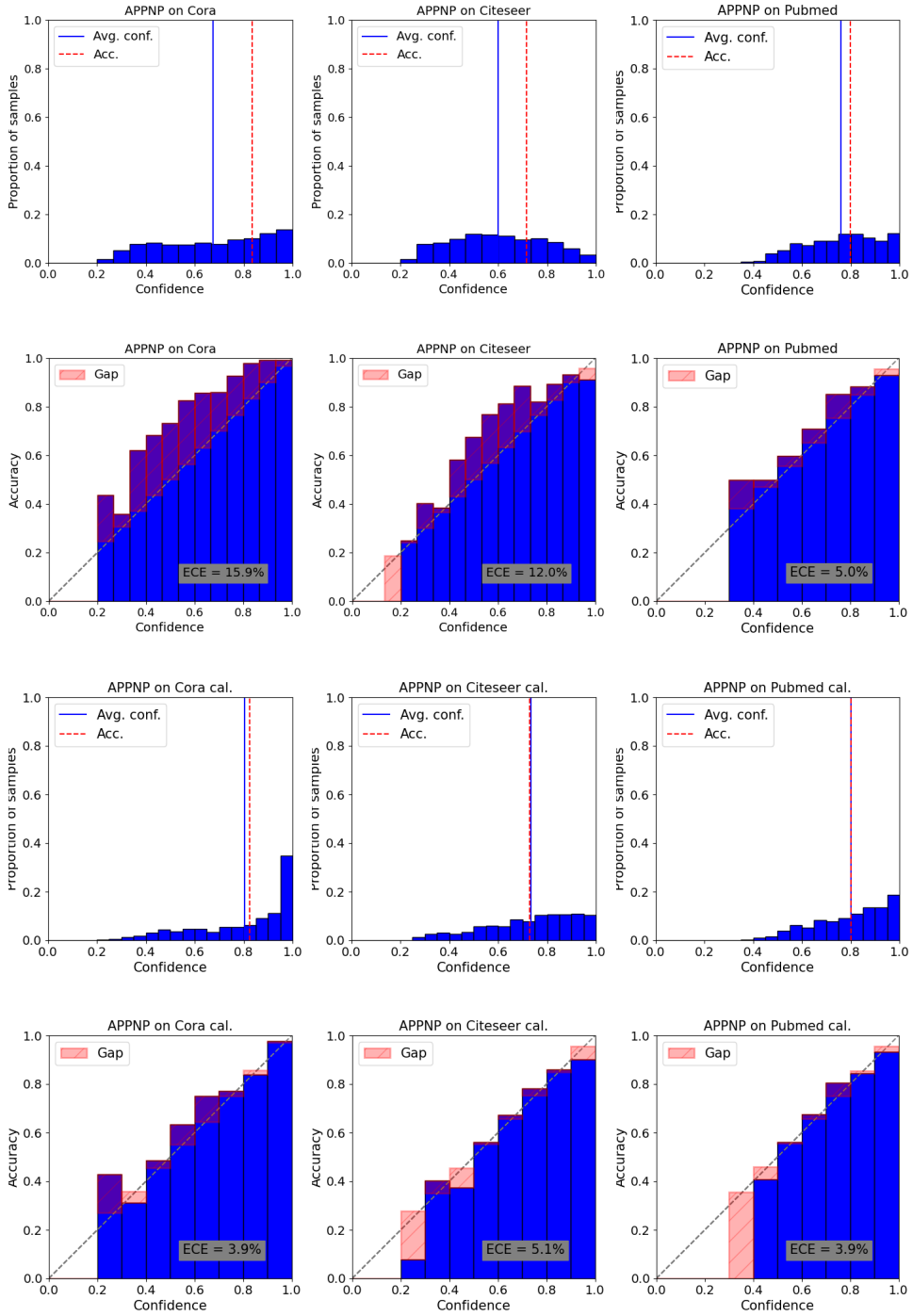


Fig. 12. Histograms and reliability diagrams for APPNP.

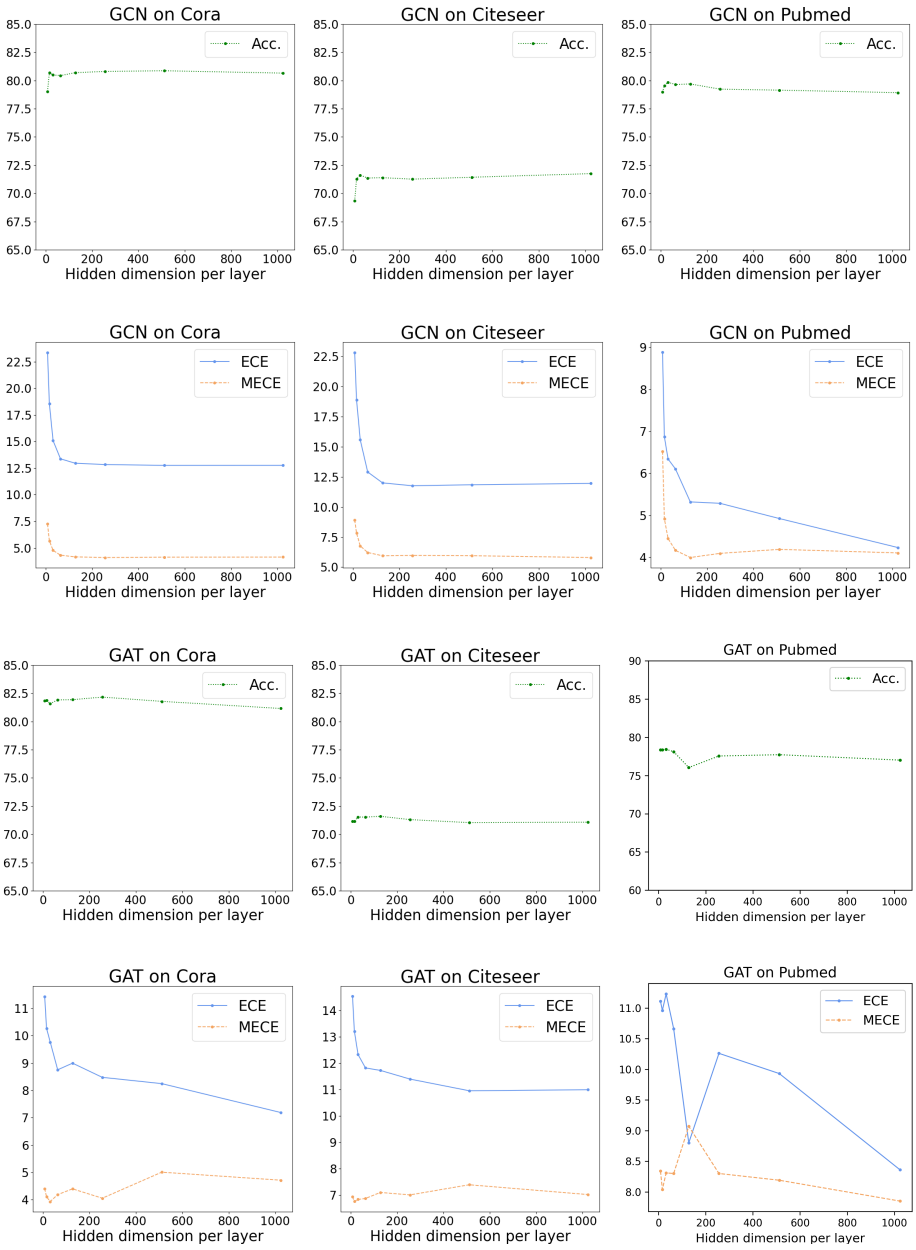


Fig. 13. Influence of width.

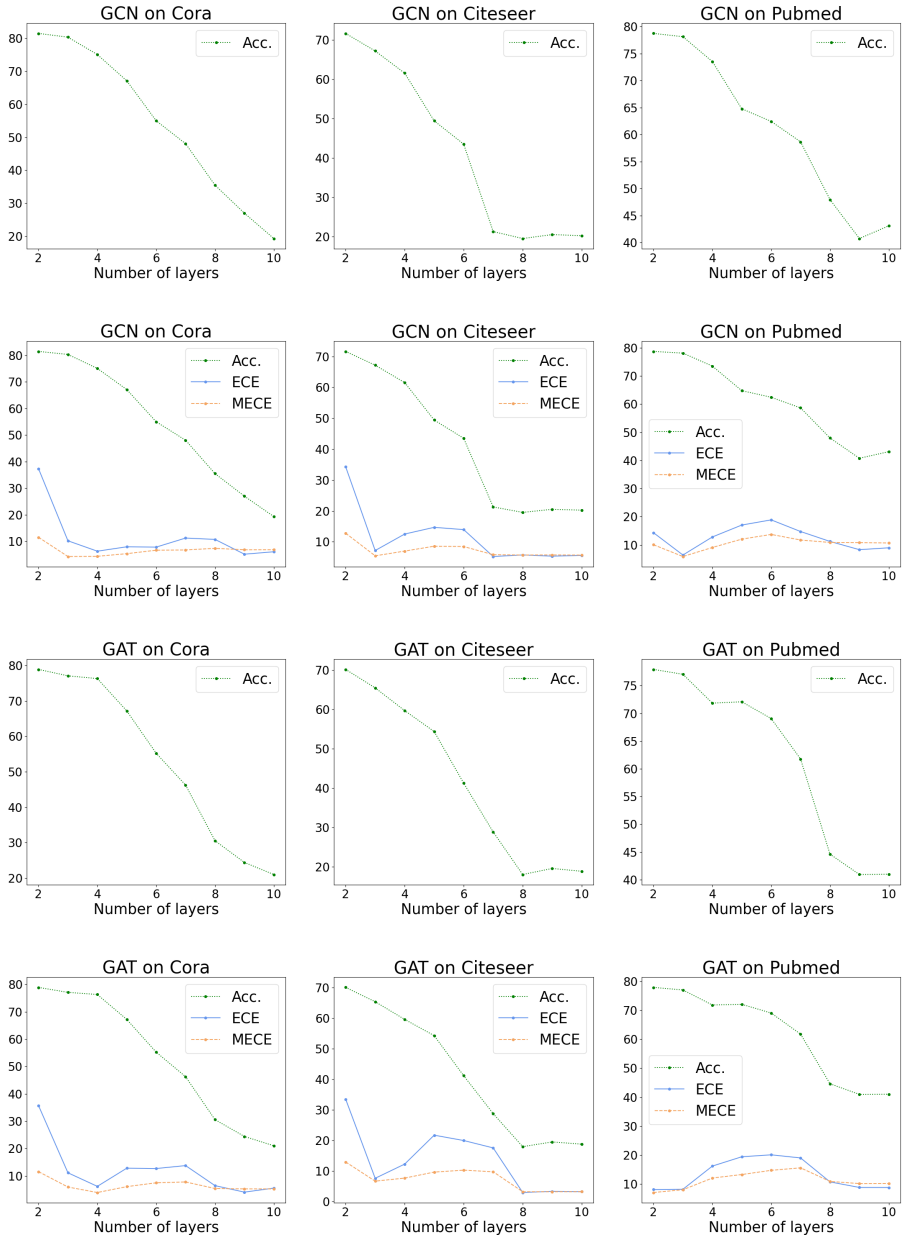


Fig. 14. Influence of depth.

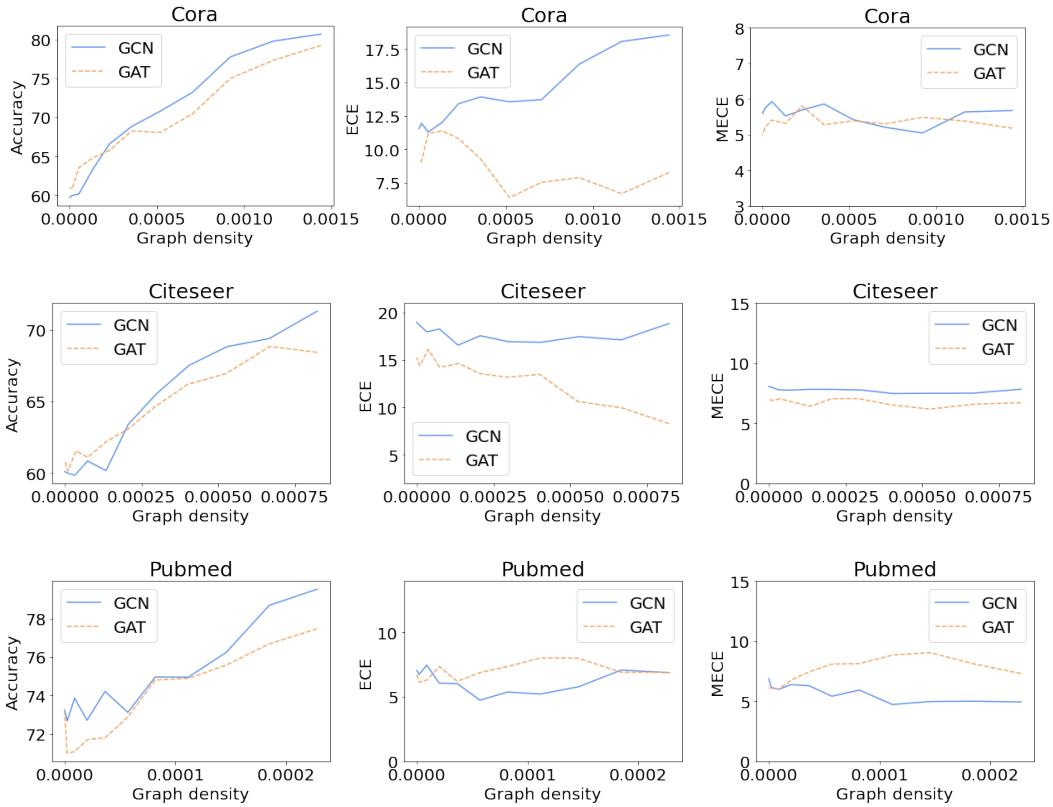


Fig. 15. Influence of graph density.

Chapter 8

Conclusion

Machine learning (ML) with graph-structured data has gained significant attention in recent years since graphs are ubiquitous in diverse domains and ML techniques are well-suited for learning and reasoning on graphs, leading to state-of-the-art performance. In this dissertation, we introduced novel graph ML methods for several use cases and attempted to make the results better understandable by humans.

In Chapter 4, we integrated domain knowledge in the form of logical rules to predict new treatment targets in the context of drug repurposing. We showed that the inclusion of the metapaths in the reward function indeed helps the reinforcement learning agent to navigate to the correct target disease. The metapaths were extracted in a data-driven manner, but it is also possible to let domain experts define rules explicitly to guide the graph traversal mechanism. The traversed paths help us understand why a particular disease is a suitable candidate for treatment. In future works, we will investigate the setting in which logical rules are used to actively guide the agent during graph traversal instead of only integrating the metapaths into the reward function.

In Chapter 5, we learned and applied temporal logical rules on a temporal knowledge graph (tKG) to predict events with future timestamps. Our approach can be scaled to large datasets by sampling relevant paths and subgraphs. Another advantage is the transferability of the rules to related datasets without the need for retraining, while many subsymbolic methods cannot easily handle previously unseen entities and timestamps in an inductive setting. The corresponding groundings of the logical rules in the graph, which also preserve time consistency, provide explanations of why a certain event is likely to happen. In the paper, we only learned cyclic rules, so we are interested in learning acyclic rules as well in future works. Further, extending the simple sampling mechanism for temporal random

walks, e.g., by using reinforcement learning, might lead to a more efficient identification of promising logical rules.

In Chapter 6, we applied KG completion methods to an industrial KG about supply chains. Due to the structure of the KG, it might not be meaningful to perform link prediction on all relation types. The methods that achieve overall strong performance could complement a supply chain risk management tool by updating the data on a regular basis. Additionally, we applied graph analytics to identify critical suppliers in the supply network, after discussing the definition of criticalities and relevant metrics with domain experts. An extension of the adopted metrics and integration of existing node features in future works might create an even more holistic view on a supplier's importance.

In Chapter 7, we examined the calibration of graph neural networks for the node classification task. We observed that there is no special neural network model or post-hoc calibration method that works best on all citation datasets. Incorporating topological information based on the predictions of neighboring nodes seems to be beneficial for the calibration. Generally, further theoretical insights should be acquired to get a deeper understanding of the calibration properties. For future works, it would be interesting to evaluate our method on a wider variety of datasets, e.g., heterophilic graphs.

In summary, we considered a range of diverse graph ML tasks and use cases. We demonstrated that integrating background and topological knowledge can boost a model's performance and that symbolic methods can be sufficiently scalable to outperform sub-symbolic methods on forecasting tasks. By extracting explicit paths from the graphs and utilizing domain expertise, the results can be presented in a more understandable and convincing way to the user, while calibrated probabilities can enhance the trustworthiness of the models. Our work contributes to the set of graph ML approaches that provide increased explainability as well as competitiveness with black-box models. Since our work only covers a small fraction within the fields of graph ML and explainable artificial intelligence (XAI), we outline in the following interesting directions for future exploration:

- The experimental outcomes in individual papers usually depend on a restricted set of training settings, baselines, datasets, and use cases. A more comprehensive and consistent evaluation of ML methods is necessary to analyze their performance under varying circumstances. For instance, Ruffinelli et al. showed that traditional methods like RESCAL [70] can beat newer methods when trained under the same setting [82]. The development, maintenance, and usage of such evaluation frameworks facilitate a fairer comparison and assessment of various models.

- The proposed methods in the thesis did not take possibly existent node and edge features into account. Some KGs also have connections to multi-modal information, e.g., images, textual descriptions, or audio recordings. The integration of specific features and modalities might improve the prediction performance and increase the interpretability of the results.

- The term XAI is not uniquely defined, so the separation between concepts such as explainability, interpretability, transparency, understandability, and comprehensibility is not clear. Explainability might have different meanings for distinct target groups, and there are no standard measures to quantify the quality of explanations. Logical rules, extracted subgraphs, and background knowledge can increase the explainability of the models, but especially when contemplating the integration of ML methods into existing tools, an in-depth validation of the outputs should be conducted by the prospective users.

- With the emergence of large language models (LLMs) and their remarkable capabilities, efforts are being made to unify LLMs and KGs [75]. LLMs could be applied to graph ML tasks, using their background knowledge to support the predictions and outputting natural language verbalizations for better understandability. LLMs, however, are black-box models and prone to hallucinations. A combination with KGs could provide validated evidence for the generation process, creating more reliable and trustworthy LLMs.

Bibliography

[1] Hemn Barzan Abdalla. A brief survey on big data: Technologies, terminologies and data-intensive applications. *Journal of Big Data*, 9:107, 2022.

[2] Grigoris Antoniou and Frank van Harmelen. Web Ontology Language: OWL. *Handbook on Ontologies*, pages 67–92, 2004.

[3] Alejandro Barredo Arrieta, Natalia Díaz, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference*, volume 4825, pages 722–735, 2007.

[5] Ajmal Aziz, Edward Elson Kosasih, Ryan-Rhys Griffiths, and Alexandra Brintrup. Data considerations in graph representation learning for supply chain networks. In *Workshop on Machine Learning for Data, International Conference on Machine Learning*, 2021.

[6] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[7] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022.

[8] Ivana Balažević, Carl Allen, and Timothy M. Hospedales. TuckER: Tensor factorization for knowledge graph completion. In *The 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5185–5194, 2019.

[9] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[10] Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge graph embeddings and explainable AI. *Knowledge graphs for explainable artificial intelligence: Foundations, applications and challenges*, pages 49–72, 2020.

[11] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112:859–877, 2017.

[12] Olivier Bodenreider. The Unified Medical Language System (UMLS): Integrating biomedical terminology. *Nucleic Acids Research*, 32:D267–D270, 2004.

[13] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, 37:1719–1778, 2023.

[14] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *The 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

[15] Stephen Bonner, Ian P. Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Andreas Bender, Charles Tapley Hoyt, and William L. Hamilton. A review of biomedical datasets relating to drug discovery: A knowledge graph perspective. *Briefings in Bioinformatics*, 23(6):bbac404, 2022.

[16] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *The 27th Conference on Neural Information Processing Systems*, volume 26, pages 2787–2795, 2013.

[17] Matthew Botvinick, Sam Ritter, Jane X. Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in Cognitive Sciences*, 23(5):408–422, 2019.

[18] Martin Brylowski, Meike Schröder, and Wolfgang Kersten. Machine learning im supply chain risk management: Studie. Technical report, Technische Universität Hamburg, 2021.

[19] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

[20] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *Journal of Machine Learning Research*, 23:3840–3903, 2022.

[21] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *The 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[22] Alain Colmerauer. An introduction to Prolog III. *Computational Logic*, pages 37–79, 1990.

[23] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *The 6th International Conference on Learning Representations*, 2018.

[24] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[25] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *The 32nd AAAI Conference on Artificial Intelligence*, pages 1811–1818, 2018.

[26] Finale Doshi-Velez and Been Kim. Considerations for evaluation and generalization in interpretable machine learning. *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 3–17, 2018.

[27] Michael Färber and Adam Jatowt. Citation recommendation: Approaches and datasets. *International Journal on Digital Libraries*, 21:375–405, 2020.

[28] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *The 22nd International World Wide Web Conference*, pages 413–422, 2013.

[29] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal*, 24:707–730, 2015.

[30] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *The 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, 2018.

[31] Michael R. Genesereth and Matthew L. Ginsberg. Logic programming. *Communications of ACM*, 28(9):933–941, 1985.

[32] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. In *The 34th AAAI Conference on Artificial Intelligence*, volume 34(4), pages 3988–3995, 2020.

[33] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *The 34th International Conference on Machine Learning*, volume 70, pages 1321–1330, 2017.

[34] William L. Hamilton. *Graph Representation Learning*. Springer International Publishing, 2020.

[35] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *The 9th International Conference on Learning Representations*, 2021.

[36] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L. Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E. Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6:e26726, 2017.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[38] Alfred Horn. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(1):14–21, 1951.

[39] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning with Applications in Python*. Springer International Publishing, 2023.

[40] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent Event Network: Autoregressive structure inference over temporal knowledge graphs. pages 6669–6683, 2020.

[41] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126, 2021.

[42] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21:2648–2720, 2020.

[43] Seyed Mehran Kazemi and David Poole. SimplE embedding for link prediction in knowledge graphs. In *The 32nd Conference on Neural Information Processing Systems*, volume 31, pages 4284–4295, 2018.

[44] Shristi Shakya Khanal, P.W.C. Prasad, Abeer Alsadoon, and Angelika Maag. A systematic review: Machine learning based recommendation systems for e-learning. *Education and Information Technologies*, 25:2635–2664, 2019.

[45] Shima Khoshraftar and Aijun An. A survey on graph representation learning methods. arXiv:2204.01855, 2022.

[46] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *The 5th International Conference on Learning Representations*, 2017.

[47] Chanchal Kumar, Taran Singh Bharati, and Shiv Prakash. Online social network security: A comparative review using machine learning and deep learning. *Neural Processing Letters*, 53:843–861, 2021.

[48] Timothee Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *The 8th International Conference on Learning Representations*, 2020.

[49] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. Fast and exact rule mining with AMIE 3. In *The 17th Extended Semantic Web Conference*, volume 12123, pages 36–52, 2020.

[50] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81:53–67, 2010.

[51] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *The Web Conference 2018*, pages 1771–1776, 2018.

[52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[53] Kalev Leetaru and Philip A. Schrodt. GDELT: Global data on events, location, and tone. In *The International Studies Association Annual Convention*, 2013.

[54] Ke Liang, Lingzuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, Xinwang Liu, and Fuchun Sun. A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal. arXiv:2212.05767, 2023.

[55] Tong Liu, Yushan Liu, Marcel Hildebrandt, Mitchell Joblin, Hang Li, and Volker Tresp. On calibration of graph neural networks for node classification. In *The 2022 International Joint Conference on Neural Networks*, 2022.

[56] Yushan Liu, Bailan He, Marcel Hildebrandt, Maximilian Buchner, Daniela Inzko, Roger Wernert, Emanuel Weigel, Dagmar Beyer, Martin Berbalk, and Volker Tresp. A knowledge graph perspective on supply chain resilience. In *The 2nd International Workshop on Linked Data-Driven Resilience Research, Extended Semantic Web Conference*, 2023.

[57] Yushan Liu, Marcel Hildebrandt, Mitchell Joblin, Martin Ringsquandl, Rime Raissouni, and Volker Tresp. Neural multi-hop reasoning with logical rules on biomedical knowledge graphs. In *The 18th Extended Semantic Web Conference*, volume 12731, pages 375–391, 2021.

[58] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. TLogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *The 36th AAAI Conference on Artificial Intelligence*, volume 36(4), pages 4120–4127, 2022.

[59] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Workshop on Deep Learning for Audio, Speech, and Language Processing, International Conference on Machine Learning*, 2013.

[60] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *The 2018 IEEE International Conference on Big Data*, pages 3762–3765, 2018.

[61] Sunil Malviya, Arvind Kumar Tiwari, Rajeev Srivastava, and Vipin Tiwari. Machine learning techniques for sentiment analysis: A review. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, 12(2):72–78, 2020.

[62] Christian Meilicke, Melisachew Wudage Chekol, Patrick Betz, Manuel Fink, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for large-scale knowledge graph completion. *The VLDB Journal*, 2023.

[63] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *The 28th International Joint Conference on Artificial Intelligencee*, pages 3137–3143, 2019.

[64] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Workshop of the International Conference on Learning Representations*, 2013.

[65] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[66] Mohammad Nagahisarchoghaei, Nasheen Nur, Logan Cummins, Nashtarin Nur, Mirhossein Mousavi Karimi, Shreya Nandanwar, Siddhartha Bhattacharyya, and Shahram Rahimi. An empirical survey on explainable AI technologies: Recent trends,

use-cases, and categories from technical and application perspectives. *Electronics*, 12(5):1092, 2023.

[67] Meike Nauta, Jan Trienes Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. An objective metric for explainable AI: How and why to estimate the degree of explainability. *ACM Computing Surveys*, 55(135):1–42, 2023.

[68] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer International Publishing, 1996.

[69] H.D. Nguyen, K.P. Tran, S. Thomassey, and M. Hamad. Forecasting and anomaly detection approaches using LSTM and LSTM autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, 57:102282, 2021.

[70] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *The 28th International Conference on Machine Learning*, pages 809–816, 2011.

[71] Nils J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[72] Seam P. O'Brien. Crisis early warning and decision support: Contemporary approaches and thoughts on future research. *International Studies Review*, 12:87–104, 2010.

[73] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. Learning temporal rules from knowledge graph streams. In *The AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering*, 2019.

[74] Simon Ott, Christian Meilicke, and Matthias Samwald. SAFRAN: An interpretable, rule-based link prediction method outperforming embedding models. In *The 3rd Conference on Automated Knowledge Base Construction*, 2021.

[75] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. arXiv:2306.08302, 2023.

[76] Karl Pearson. The problem of the random walk. *Nature*, 72:294, 1905.

[77] John C. Platt. Probabilities for sv machines. *Advances in Large Margin Classifiers*, pages 61–74, 2000.

[78] Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *The 33rd Conference on Neural Information Processing Systems*, volume 32, pages 7680–7690, 2019.

[79] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[80] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[81] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *The 31rd Conference on Neural Information Processing Systems*, volume 30, pages 3791–3803, 2017.

[82] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! On training knowledge graph embeddings. In *The 8th International Conference on Learning Representations*, 2020.

[83] Tara Safavi, Danai Koutra, and Edgar Meij. Evaluating the calibration of knowledge graph embeddings for trustworthy link prediction. In *the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8308–8321, 2020.

[84] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer International Publishing, 2019.

[85] Iqbal H. Sarker, A.S.M. Kayes, Shahriar Badsha, Hamed Alqahtani, Paul Watters, and Alex Ng. Cybersecurity data science: An overview from machine learning perspective. *Journal of Big Data*, 7:41, 2020.

[86] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The 15th Extended Semantic Web Conference*, volume 10843, pages 593–607, 2018.

[87] E.W. Schneider. Course modularization applied: The interface system and its implications for sequence control and data analysis. Technical report, Human Resources Research Organization, 1973.

[88] Raymond M. Smullyan. *First-Order Logic*. Dover Publications, 1995.

[89] Francesco Sovrano and Fabio Vitali. An objective metric for explainable AI: How and why to estimate the degree of explainability. *Knowledge-Based Systems*, 278:110866, 2023.

[90] Claudio Stamile, Aldo Marzullo, and Enrico Deusebio. *Graph Machine Learning: Take graph data to the next level by applying machine learning techniques and algorithms*. Packt Publishing, 2012.

[91] Wenjuan Sun, Paolo Bocchini, and Brian D. Davison. Applications of artificial intelligence for disaster management. *Natural Hazards*, 103:2631–2689, 2020.

[92] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *The 7th International Conference on Learning Representations*, 2019.

[93] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[94] Pedro Tabacof and Luca Costabello. Probability calibration for knowledge graph embedding models. In *The 8th International Conference on Learning Representations*, 2020.

[95] Leanardo Teixeira, Brian Jalaian, and Bruno Ribeiro. Are graph neural networks miscalibrated? In *Workshop on Learning and Reasoning with Graph-Structured Representation, International Conference on Machine Learning*, 2019.

[96] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *The 33rd Conference on Neural Information Processing Systems*, volume 32, pages 13888–13899, 2019.

[97] Christian Tomani and Florian Buettner. Towards trustworthy predictions from deep neural networks with fast adversarial calibration. In *The 35th AAAI Conference on Artificial Intelligence*, volume 35(11), pages 9886–9896, 2021.

[98] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *The 33rd International Conference on Machine Learning*, volume 48, pages 2071–2080, 2016.

[99] Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[100] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *The 8th International Conference on Learning Representations*, 2020.

[101] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *The 6th International Conference on Learning Representations*, 2018.

[102] I.N. Vos, I.R. Bhat, B.K. Velthuis, Y.M. Ruigrok, and H.J. Kuijf. Calibration techniques for node classification using graph neural networks on medical image data. In *The International Conference on Medical Imaging with Deep Learning*, 2023.

[103] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[104] Jiapu Wang, Boyue Wang, Meikang Qiu, Shirui Pan, Bo Xiong, Heng Liu, Linhao Luo, Tengfei Liu, Yongli Hu, Baocai Yin, and Wen Gao. A survey on temporal knowledge graph completion: Taxonomy, progress, and prospects. arXiv:2308.02457, 2023.

[105] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141:61–67, 2021.

[106] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *The 36th International Conference on Machine Learning*, volume 97, pages 6861–6871, 2019.

[107] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *The 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, 2017.

[108] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *The 3rd International Conference on Learning Representations*, 2015.

[109] Fan Yang, Qi Zhang, Xiaokang Ji, Yanchun Zhang, Wentao Li, Shaoliang Peng, and Fuzhong Xue. Machine learning applications in drug repurposing. *Interdisciplinary Sciences: Computational Life Sciences*, 14:15–21, 2022.

[110] B. Yegnanarayana. *Artificial Neural Networks*. Prentice-Hall of India, 2004.

[111] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *The 33rd Conference on Neural Information Processing Systems*, volume 32, pages 9240–9251, 2019.

[112] L.A. Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, 1988.

[113] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *The 18th International Conference on Machine Learning*, pages 609–616, 2001.

[114] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699, 2002.

[115] Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35, 2021.

[116] Wen Zhang, Jiaoyan Chen, Juan Li, Zezhong Xu, Jeff Z. Pan, and Huajun Chen. Knowledge graph reasoning with logics and embeddings: Survey and perspective. arXiv:2202.07412, 2022.

[117] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *The 35th AAAI Conference on Artificial Intelligence*, volume 35(5), pages 4732–4740, 2021.

[118] Esra Zihni, Vince Istvan Madai, Michelle Livne, Ivana Galinovic, Ahmed A. Khalil, Jochen B. Fiebach, and Dietmar Frey. Opening the black box of artificial intelligence for clinical decision support: A study predicting stroke outcome. *PLoS One*, 15(4):e0231166, 2020.

[119] Karl Johan Åström. Optimal control of Markov processes with incomplete state information I. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.