TU WIEN Informatics

Causality Prediction in a Cyber-Physical-Energy-System using Knowledge Graph Embeddings

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Data Science

eingereicht von

Katrin Schreiberhuber
Matrikelnummer 01503964

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dr.techn. Peter Knees
Mitwirkung: Univ.Prof. Marta Sabou, PhD.
Dr.techn. Mag. Fajar Ekaputra
Projektass. Mag. Peb Aryan

Wien, 25. Jänner 2023

[redacted]	[redacted]
Katrin Schreiberhuber	Peter Knees

Informatics

Causality Prediction in a Cyber-Physical-Energy-System using Knowledge Graph Embeddings

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Data Science

by

Katrin Schreiberhuber
Registration Number 01503964

to the Faculty of Informatics

at the TU Wien

Advisor: Associate Prof. Dipl.-Ing. Dr.techn. Peter Knees
Assistance: Univ.Prof. Marta Sabou, PhD.
Dr.techn. Mag. Fajar Ekaputra
Projektass. Mag. Peb Aryan

Vienna, 25th January, 2023 [redacted] [redacted]

Katrin Schreiberhuber Peter Knees

Erklärung zur Verfassung der Arbeit

Katrin Schreiberhuber

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. Jänner 2023

[redacted]

Katrin Schreiberhuber

Kurzfassung

Mit fortschreitender Digitalisierung werden immer mehr Cyber-Physical Systems eingesetzt, womit bestehende physische Systeme ergänzt und verbessert werden können. Smart Grids sind ein Beispiel für ein Cyber-Physical System. Diese bestehen aus einer physischen Ebene, die die Produzenten und Verbraucher des Stromnetzes enthält, sowie einer Cyber-Ebene, die das Netz mithilfe von digitalen Prozessen verbessert. Durch die Energiewende befinden sich immer mehr kleine und volatile Energieerzeuger in einem System, wodurch die Komplexität weiter steigt. Diese Situation führt zu einem Bedarf an leistungsfähigeren und komplexeren Steuerungsmechanismen im Stromnetz. Leider ist es in heutigen Stromnetzen grundsätzlich nicht möglich, die Herkunft elektrischer Energie zu messen. Daher bedarf es neuer Ansätze, um Ereignisse im System besser erklären zu können.

Im Zuge eines Forschungsprojektes haben Aryan et al. [Ary21] einen Knowledge Graph entwickelt, der ein Cyber-Physical Energy System (CPES) darstellt. Da Knowledge Graphs in der Lage sind, komplexe und heterogene Daten zu modellieren bringen sie ideale Voraussetzungen mit, um die Beziehungen und Merkmale eines Smart Grids darzustellen und Kausalitäten zwischen Ereignissen im System zu finden. Derzeit wird ein ein regelbasierter Ansatz verwendet, um relevante Kausalitäten in einem bestehenden System zu finden.

Basierend auf diesem Knowledge Graph wendet diese Masterarbeit Knowledge Graph Embeddings (KGE) an, um das Potenzial von Machine Learning Ansätzen für die Prognose von Kausalitäten zwischen Ereignissen in einem Smart Grid zu untersuchen. Basierend auf Literaturrecherchen wurden vier KGE-Modelle ausgewählt, um diese zu trainieren - TransE, TransH, ComplEx, TTransE. In einem ersten Schritt wurden die Modelle allgemein evaluiert, um zu testen wie gut die Modelle den Knowledge Graph und die Beziehungen zwischen Instanzen im Graph repräsentieren können. In einem zweiten Schritt wurde jedes der vier Modelle hinsichtlich der Prognose von Kausalitäten zwischen Ereignissen evaluiert. In einem Evaluation-Workshop wurden die Modellprognosen weiters manuell analysiert. Dort wurden die Ergebnisse analysiert und beurteilt, ob diese echte Kausalitätszusammenhänge beinhalten, die im aktuellen Knowledge Graph (noch) nicht vorhanden sind.

Diese Arbeit kann keine eindeutige Antwort darauf geben, welches KGE-Modell am besten für die Prognose von Kausalitäten in einem CPES geeignet ist. Sie stellt jedoch ein Framework vor, um die Anwendung von KGEs für die Prognose von Kausalitäten in einem CPES Knowledge Graph zu testen. Darüber hinaus zeigte die Untersuchung des Einsatzes von KGEs in diesem Use Case, dass der Einsatz von Hybrid-AI Potenzial zur Verbesserung der Erklärbarkeit in einem Smart Grid bietet.

Abstract

With the emergence of digitalisation to increase the capabilities and efficiency of systems, Cyber-Physical Systems have emerged at the intersection of physical and computational systems. Smart grids are an example of such a Cyber-Physical System as it is comprised of a physical layer containing the producers and consumers of a system as well as a computational layer, which makes the power grid "smart". An increasing number of smaller producers in a system due to a transition to more renewable energy sources sparked the need for a more capable and complex control mechanism of a power grid. Unfortunately, it is not inherently possible to measure the source of electrical energy in power grids as they exist today. Therefore, a need for new approaches exists in order to make power grids more explainable.

In recent research [Ary21], Aryan et al. introduced a Knowledge Graph which can represent a Cyber-Physical Energy System. As Knowledge Graphs are able to model complex and heterogeneous data, they seem ideal to represent the relations and features of a smart grid and to find causalities between events in the system. A rule-based approach currently aims to find relevant causalities in an existing Knowledge Graph.

Based on this research, this thesis applies Knowledge Graph Embeddings (KGE) on the Knowledge Graph to investigate the possibilities of using machine learning approaches for causality link prediction between events in a smart grid. Upon literature research, four KGE models were chosen to train on the Knowledge Graph - TransE, TransH, ComplEx, TTransE. In a first step, the model performance was evaluated based on their ability to represent the Knowledge Graph and the relations between various entities in the graph. In the second step, each model was evaluated on predicting causality links between two events in the system. In this process, an evaluation workshop was held where model predictions were analysed by knowledge experts in order to determine whether there may be true causality links in the predictions which are not present in the current Knowledge Graph.

While no certain conclusion can be drawn on which KGE model is best suited for causality link prediction in a Cyber-Physical Energy System, this thesis provides a framework to test the application of KGEs for causality link prediction on a Knowledge Graph representing a smart grid. Additionally, the exploration of using KGEs on this use case showed that there is potential on the use of hybrid AI for improving explainability in a smart grid.

Contents

CHAPTER 1

Introduction

1.1 Context and Motivation

Currently, major changes are happening in the energy domain. Due to the emergence of smarter entities which can communicate with each other, power grids are gaining new opportunities, which come with challenges. New requirements are arising which should be addressed by the power grid, its functionalities and management. Especially the transition to an increased amount of renewable energy sources brings a shift from few major suppliers offering a constant flow of energy to many small and volatile energy producers. Smart grids are transforming the power grid to a Cyber-Physical System by incorporating software components. These smart components help managing energy systems by using automated processes and computational power. They enable the system to consider the capabilities of all types of actors in the area and find a solution to meet the demands of consumers at any point in time.

Cyber-Physical Systems (CPS) have emerged some years ago, and are designed to support humans in their interaction with machines. In this thesis I refer to the definition of a CPS as "the integration of computational and physical capabilities which are able to interact with humans, allowing for better communication and new functionalities" [BG11]. Cyber Physical Systems are already used in many domains, ranging from Event Planning over Production Systems to Energy Systems (CPES). In the energy domain, CPS are used to cope with and use the increased interaction between sensors and actuators in a system to provide new and better functionalities. CPES are able to monitor and manage information provided by devices in a power grid, enabling the communication between two entities [Kar11].

Even though a system may be "smart" by enabling interaction and adaption between users and devices, this does not yet mean that all events and decisions in a system are explainable. Especially for unexpected situations and decisions by the system, users who interact with a "smart" system want to know the reason behind a system's decision. Therefore, a new requirement arises for smart systems: adding explainability to their capabilities, thus building an Explainable Cyber Physical System (expCPS). An expCPS should be able to "explain certain aspects of interest about the system, both in human-comprehensible and machine-processable format" [GLV19].

In the context of Explainable Cyber-Physical Energy Systems (expCPES), increasing volatility of electrical energy producers poses new and unprecedented challenges to the electrical power grid. The increase in volatility of energy production is caused by more renewable energy sources as well as new and more volatile consumers such as electrical vehicles. The use of CPS in the form of smart grids enables the regulation of constantly changing demands to ensure the stability of the network (power grid). Knowing the reasons behind certain decisions made by a system is crucial to increase trust in the system and to find bottlenecks and error causes within the system.

As an example, if a user notices that their electrical vehicle fails to load at full capacity, they will want to know why. An explainable Cyber Physical Energy System (expCPES) should then be able to analyse the causes leading to this behaviour, whether there is a defect in the loading station, an increased energy demand in the area due to low temperatures or reduced energy production due to low solar radiation.

To achieve this goal, a system needs to be aware of the connections and implications between devices as well as their features within the system. One well-suited approach to model heterogeneous data from many different kinds of devices are Knowledge Graphs (KG) [AES+21]. There are many definitions of Knowledge Graphs, where some are quite narrow, while others are very broad. The term was first coined by Google in 2012, describing Knowledge Graphs as follows:

> “A graph that understands real-world entities and their relationships to one another: things, not strings ” [Sin12].

While this is a very vague definition, it has been cited many times, resulting in a vast amount of possible definitions. To this day, there is not a single definition of a Knowledge Graph that is universally accepted by the research community. In section 2.1, a deeper analysis of the definition of Knowledge Graphs is conducted. For now, a Knowledge Graph can be seen as a data storage method, which can connect entities to each other using relations. Due to the simple, yet flexible structure of Knowledge Graphs, they are scalable and can deal with very heterogeneous data, as is the case in a CPES with many different types of devices and connections. As a new development in the research area of Knowledge Graphs, their applications and capabilities, Knowledge Graph Embeddings (KGE) emerged as a technique to map entities and relations of a knowledge graph into a continuous vector space, called embeddings, to capture their semantic meaning [JPC+22]. As Knowledge Graphs aim to model relations between real-world entities, they are highly incomplete. By capturing the semantic meaning, a KGE should be able to find new connections which potentially exist in the semantic context, but are missing in an incomplete Knowledge Graph. Additionally, the representation in a continuous vector space allows for simpler manipulation of the Knowledge Graph while preserving its inherent structure [WMWG17]. Therefore, KGEs improve performance of existing tasks and provide new solutions to tasks in the context of Knowledge Graphs. Some examples of typical use cases for KGEs are link prediction, knowledge base completion or entity classification.

1.2 Problem Definition

Based on previous work focusing on using Knowledge Graphs for modelling a smart grid to derive causality information [Ary21], this thesis aims to improve existing causality predictions by using Knowledge Graph Embeddings. The ExpCPS Knowledge Graph, which is used as the basis of this thesis, contains the topology of a smart grid as well as events happening within a certain time period. This KG has been constructed based on data generated by the BIFROST simulation, a smart grid simulation platform. As mentioned in [MDE$^+$19], using simulation data for testing and demonstrating new approaches fosters many benefits. Testing new approaches and solutions in real-time settings poses unacceptable risks to critical infrastructure and may impact the comfort of residents. For example if a heating device is wrongly activated in summer due to bugs in the tested algorithm, residents of the building are faced with an overheated apartment as well as wasted energy and high costs. Moreover, data access to real-life data is limited due to privacy laws, increasing the need for simulated scenarios, which can model the energy consumption of a typical household.

The simulation data for this analysis have been generated by the BIFROST platform. BIFROST is “a persistent, shared design tool and simulation environment for Smart Cities, with a strong focus on powergrid infrastructure” [MDE$^+$19]. Supported by the European Union’s Horizon 2020 research and innovation program, it aims to provide an integrated tool to simulate energy communities with the option to manipulate and control scenarios in the system. The tool provides the possibility to design and test scenarios which would be difficult if not impossible to create on purpose in a real setting. Depending on the use case, it is possible to build a community featuring any devices and buildings needed in a scenario. Figure 1.1 shows an electrical vehicle charging use case inside the BIFROST simulation user interface.

Utilizing the potential of this tool, [ADES21] exports the data produced by a simulation run of BIFROST to build a Knowledge Graph representing the devices, physical connections and time-dependent data measurements in the scenario. Predefined events are then derived from the KG by querying those events based on rules defined in SPARQL queries. Additionally, causal relations between components in the smart grid are added to the existing Knowledge Graph by domain experts by defining causality rules [ADES21].

The resulting Knowledge Graph contains the topology of a simulated village (devices

Figure 1.1: Use case shown in the user interface of the BIFROST simulation environment as in [AES+20]

in the simulation, including their location, connection-type and other features), the device measurements for multiple subsequent timestamps as well as annotated events and general causalities between devices. Until now, a rule-based approach is applied to derive concrete causalities between events from the Knowledge Graph and its data.

As this method is able to capture many causal connections based on concrete rules, some connections are potentially missed due to the restricted nature of rule-based querying. The goal of this thesis is to apply Knowledge Graph Embeddings to this event-explainability use case. Potentially, new information may be derived from the data through KGE which may not yet have been captured by a rule-based approach.

The thesis will address two major goals:

G1 Finding a Knowledge Graph Embedding method which works well with the Knowledge Graph for this use case and

G2 Using promising Knowledge Graph Embedding Methods to find new insights into the ExpCPS Knowledge Graph.

Based on these goals, the following research questions should be answered in the scope of the master thesis:

Q1 Which Knowledge Graph Embedding Methods are most suitable for causality detection in a Cyber Physical System?

Q2 How well do the chosen Knowledge Graph Embedding Methods from **Q1** perform in representing the knowledge base captured by the ExpCPS Knowledge Graph?

Q3 How well can the embedding space be used to uncover causality relations in a system?

Q1 aims to find a suitable embedding method, considering the features and specific requirements of the ExpCPS Knowledge Graph. Answering this question requires thorough literature research to find promising embedding methods which fit the requirements of the use case. Additionally, the requirements, advantages and drawbacks of each embedding method need to be evaluated. Considering the task of causality prediction in a simulated smart grid, the most fitting embedding methods for this use case will be decided upon.

In **Q2**, the best suited models based on literature research (**Q1**) will be implemented and trained on the training data. By checking Hits@10 and mean rank, the model which represents the Knowledge Graph most accurately will be determined. The models will not only be analysed by comparing the performance metrics overall, but also a detailed analysis of strengths and weaknesses of each model concerning different areas of the Knowledge Graph will be conducted. Even though the main goal of the thesis is causality prediction, checking whether the main semantics are captured by a model allows for a better understanding of the performance of the model and whether overfitting on certain relations might be an issue.

The final Research Question **Q3** focusses solely on causality relations. While the general performance of KGE methods was analysed in **Q1** and their overall performance on the ExpCPS Knowledge Graph by [ADES21] was evaluated in **Q2**, this research question aims to analyse the performance specifically focussed on causality relations between events. Furthermore, new relations which have not been captured by the original Knowledge Graph will be analysed whether they are plausible causes, which should be added to the KG. If so, the chosen KGE can assist in the task of knowledge base completion for a CPES by suggesting missing facts and relations in the data.

1.3 Simulation Scenario and Data Structure

This section aims to explain the simulation scenario used in this thesis. The section will focus on the data source and the data processing pipeline, which precedes the experiment. Understanding the data and its intricacies is crucial for choosing the right tools and methods for its analysis. As already mentioned, the data in the ExpCPS Knowledge Graph is sourced from a simulation platform for a smart power grid. In the first part of this section, the data pipeline to build the Knowledge Graph is discussed in detail. The second part will then focus on the Knowledge Graph itself and its structure.

1.3.1 Data Pipeline

The Knowledge Graph containing a smart grid and its causal relations as it is used in this thesis was created using multiple data sources and computational layers to manipulate the data. The simulation platform, which the ExpCPS Knowledge Graph is based on is called BIFROST [MDE+19]. In this framework, a user can build an energy village and connect multiple components, such as residential houses, transformers, electric vehicle (EV) charging stations, etc. with each other using underground cables.

The physical setup and the connections between two devices in the village is represented in its topology. It forms the basis of the ExpCPS Knowledge Graph by providing the structural setup of a village. In the next step, the simulation is run for a certain amount of time (24h in this thesis) to obtain measurements over time. Additionally, the granularity of time steps can be configured (1h timestep in this thesis). The simulation run is producing a list of system states, which indicate measurements of sensors for each time step in the simulation. These measurements are stored as observations in the ExpCPS Knowledge Graph.

Using the observations, an Event Annotator is defined to derive specific events from the observations which have been measured during the simulation. Based on some predefined rules, events are labelled by analysing the data of observations over time. As an example, an OverloadingEvent is registered if a loading observation at a transformer has a particularly high value.

For the creation of the ExpCPS Knowledge Graph, the causal relationships between sensors and events are still missing. The concept of causality will be explained in more detail in section 2.3.1, but for now there are three different layers of causality:

Layer 1. generally known causality between two types of objects. As an example, a general causality is that an increase in energy production in a solar panel can cause an increase in available electricity in a transformer.

Layer 2. potential causality between two concrete objects, derived from general causality. For example, there is a `solarPanelA` located at a specific location in a smart grid setup and a `TransformerC`, which is physically connected to `solarPanelA`. Using the information about generally known causality, we can derive a potential cause between these two objects. So, `solarPanelA` potentially causes changes at `TransformerC`.

Layer 3. concrete causality between two events. Based on information about potential causality between two objects, a concrete causal relationship between two events happening at two objects at the same time or in close proximity can be derived. As an example, a sensor at `TransformerC` registers `OverloadEventX` at `timestamp t`. Additionally, an `IncreasedProductionEventY` is registered at `solarPanelA`. As we know there is a potential causality between `TransformerC` and `solarPanelA`, a causality between `OverloadEventX` at

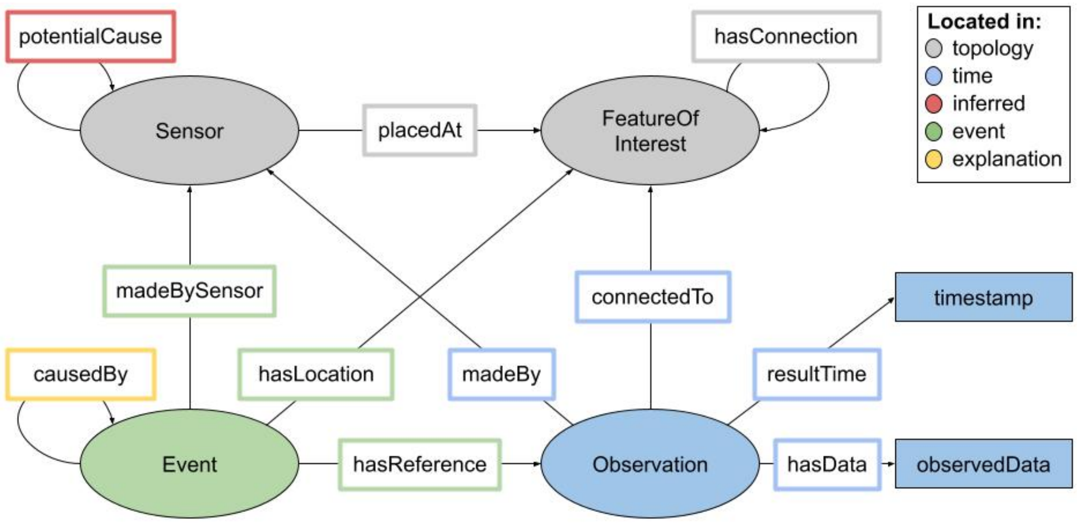


Figure 1.2: Schematic overview of ExpCPS Knowledge Graph entities and relations

TransformerC and IncreasedProductionEventY at solarPanelA can be derived.

Based on this knowledge, generally known causality rules (layer 1) from domain experts were used to derive potentialCause relations between specific sensors in the system (layer 2). In the final step, this information was used to find concrete causality explanations between events in the simulation (layer 3). When looking for an explanation of EventA, the idea is to check the sensor which measured the events, such as SensorK and to find sensors, which are connected to SensorK through a potentialCause - relation. For each sensor, which could be a potentialCause of SensorK, the algorithm is looking for an EventX happening at the sensor at the same time or in close temporal proximity to EventA. If such an event exists, a causedBy relation is created between EventA and EventX. For ranking a growing number of potential causes, a reliability measure of the Event Annotator is used. This measure can be interpreted as the uncertainty of the connection between the causal event and the effect event.

1.3.2 Knowledge Graph Structure

Based on the data pipeline above, a Knowledge Graph containing all information necessary to derive causalities is created. For better structure in the storage of the Knowledge Graph, each part of the Knowledge Graph is stored in smaller sub-graphs based on the topic of the data and the stage of the pipeline where it was created. In figure 1.2, a schematic overview of the sub-graphs and their elements is shown. Moreover, each sub-graph in ExpCPS Knowledge Graph is described here:

Topology In this graph, the physical structure of the energy village is stored. Any physical component in the power grid is called `FeatureofInterest`. This can be an electric vehicle charging station, a solar panel, an underground cable or a weather station. Any connection between two components of the power grid is stored as a `hasConnection` relation. Additionally, sensors are placed at features of interest, which can measure various things, such as voltage, cable length or capacity of a component. Sensor measurements can be static in time, such as cable length, or dynamic, such as voltage or power.

Time The time graph contains all the observations, which are made by a sensor in the topology. Each observation contains information about the sensor which made the observation and the feature of interest it is connected to. Each observation has a timestamp of when the observation was registered. In case an observation is time-invariant, the start of the simulation is set as the respective timestamp. Finally, the data which was measured is stored in an `observedData` entity. This could be a value of the cable length, the voltage of a specific transformer at a specific time or a GPS location of a feature of interest.

Inferred Based on expert knowledge about the potential causes between two objects(layer 1), the connections between two sensors in the energy villages are inferred based on SPARQL queries (layer 2). Additional information about using SPARQL queries for causality annotations in the ExpCPS Knowledge Graph are described in detail in [ADES21]. Important to note here is that this `potentialCause` is not a causality between two events. This relation only shows a potential causal relation between two sensors, which is connected to a specific object in the energy village, but it is not connected to any observation or event. The `potentialCause` relation is a way to represent common knowledge about directions of a causal relation. For example, generally low solar intensity regisered by `WeatherStationA` causes low power production in `SolarPanelH` and not the other way around.

Event The event sub-graph is created by an event annotator. It uses data from observations to find specific types of events, such as an overload in a transformer or a peak demand in a residential building.

Explanation This sub-graph is the most crucial part of the Knowledge Graph for this thesis as this is where the ground truth is stored on which potential causal relations will be evaluated. Based on annotated events, their associated sensors and the `potentialCause` relation, causalities between two events are derived. The aim of this thesis is to train a good embedding model to be able to predict these `causeEvents` and to find potential new events which could have caused a specific event.

In figure 1.2, the sub-graphs are color-coded and the entities and relations in each sub-graph are represented in the schema. Therefore, this schematic overview can serve as assistance for the reader.

1.4 Structure of the Work

To answer the research questions posed, the thesis will start by introducing the concepts needed to understand the tools and methods used in the thesis and analysing related work in the field in chapter 2. This chapter aims to explain the concepts of Knowledge Graphs, Knowledge Graph Embeddings and Event Explainability. The reader will be introduced to the ideas behind the keywords, but also recent advances in research will be discussed. At the end of this chapter, the research questions should not only be understood at a superficial level, but also the concepts involved will be covered to allow the reader to comprehend the research questions which should be answered in the course of this thesis. This chapter will be the basis for any future design and methodology decisions of this work.

In chapter 3, the goal is to discuss and explain the methodology as a basis of the empirical analysis conducted in the thesis. Two main topics will be discussed: possible Knowledge Graph Embedding methods which can be applied as well as architecture choices of the experiment design. In the first section, an in-depth analysis of currently available methods, their designs, assumptions as well as benefits and drawbacks of each method will be conducted. The second topic focusses on the experiment architecture. As the ExpCPS Knowledge Graph seems to be complete in itself, an adapted version of the experiment architecture will be constructed and used to evaluate the performance of an embedding model on the ExpCPS Graph.

Following the explanation of the methodology, chapter 4 continues by explaining the experimental setup. This includes an in-depth analysis of the data structure, necessary preprocessing work for the thesis, libraries and frameworks used for all steps and a schematic overview of the pipeline to produce the final results. Finally, the evaluation metrics used to analyse the performance results will be discussed and explained.

Chapter 5 will show the results obtained by running the experiment as it was described in chapter 4. There are three main types of performance results to show:

- performance metrics of various KGE models as chosen in section 3.1
- a detailed performance analysis of the trained embedding models considering performance on different subgraphs of the data.
- quantitative and qualitative analysis of link prediction results obtained by applying the embedding models for predicting causality relations.

In the final chapter 6, the results will be discussed and analysed. By recapping the research questions, checking their answers and providing possible conclusions from the experiment results, all relevant findings and answers provided by this thesis will be summarised. Moreover, limitations and future work will be discussed as well as the contributions this thesis has made to the research domain.

CHAPTER 2

Background and Related Work

Based on a Knowledge Graph as the primary source of information containing details about the entities, properties and connections in a simulated smart energy system, the goal in this thesis is to use a prediction model to improve existing knowledge in the graph. Since this task requires a combination of semantic understanding of the knowledge represented in a knowledge graph as well as the ability to integrate a lot of data using a flexible and adaptive approach, Knowledge Graph Completion methods are mostly staged between symbolic and sub-symbolic AI. While there has been a contest between these two methods, Knowledge Graph completion tasks show that there are benefits to both approaches and best results can be achieved by integrating them. The most commonly used and powerful method for Knowledge Graph Completion is Knowledge Graph Embeddings [IK20]. They aim to create a representation of the KG in a low-dimensional vector space, reducing the complexity of the graph to gain new insights. While a lot of research has been conducted in this field, no application of KGE for complex relations, such as causalities in a network, could be found in the literature. Especially in the area of event explainability in the energy domain, this thesis aims to test the performance of KGEs as a combination of symbolic and sub-symbolic AI approaches for improving the known causal relations in the graph. In the following sections, the main research topics connected to the thesis will be elaborated. Current research and trends are discussed and their relevance to the thesis will be analysed. The three main topics to be investigated are Knowledge Graphs, Knowledge Graph Embeddings and Event Explainability.

In the first section, the concept of Knowledge Graphs will be discussed in more detail. As there is no universal definition of a Knowledge Graph, there will be a discussion of the most relevant features defining a Knowledge Graph. Additionally, ideas, origins and applications as well as current trends and promising ideas for the future application of Knowledge Graphs will be addressed.

In section 2.2, Knowledge Graph Embeddings as a tool to improve Knowledge Graphs will be investigated. Knowledge Graph Embeddings can be used as a means to apply Machine Learning concepts to Knowledge Graphs, opening new opportunities for advanced solutions for various tasks, such as link prediction, entitiy classification or clustering. As the application of Knowledge Graph Embeddings on the ExpCPS Knowledge Graph forms the core of the thesis, various methods, their advantages and drawbacks as well as assumptions on which they are based will be discussed.

Finally, section 2.3 will focus on Event Explainability. As the goal of the thesis is to apply KGE methods for causality link prediction, this section aims to analyse methods explored in research to achieve explainability of an event in a system. The methods should not be restricted to semantic technologies, but the aim is to give a broad overview of possibilities for event explainability and their application domains.

2.1 Knowledge Graphs

As the data used in this thesis consists of a Knowledge Graph representing a Cyber-Physical Energy System, this section will focus on the definition of a Knowledge Graph, its applications and how to build it. As mentioned in [FSA$^+$20], Knowledge Graphs have experienced a huge trend since their introduction by Google, yet there is not a single, precise definition of a Knowledge Graph to date.

2.1.1 Definition

The original definition of a Knowledge Graph by Google was "A graph that understands real-world entities and their relationships to one another" [Sin12]. However, this definition does not give a proper explanation of what a knowledge graph is supposed to be. Therefore, the term has been used for many types of graphs and data representations as the concept became more popular. In an extensive survey of potential definitions for the term "Knowledge Graph", [EW16] has identified two main issues connected to the ambiguous usage of the term. First, Google's initial definition in a blog entry is commonly cited as a definition even though it does not provide a good explanation of what constitutes a Knowledge Graph, resulting in very different interpretations of the term. Second, there is no clear differentiation between the terms knowledge base, knowledge graph and ontology as they are sometimes just used as synonyms for each other [EW16]. Some definitions of a Knowledge Graph could equally be used to describe an ontology, which makes it hard to differentiate the two terms.

By analysing the differences between the three terms Knowledge Graph, Knowledge Base and ontology, [EW16] has suggested a new definition of Knowledge Graphs:

> "A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge."

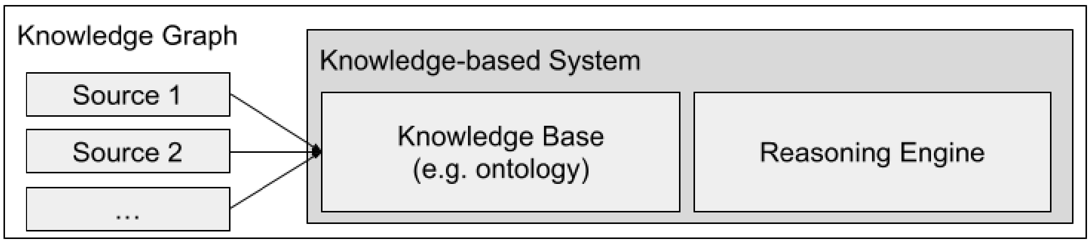


Figure 2.1: Architecture of a Knowledge Graph, adapted from [EW16]

In this definition, a Knowledge Base is a simple collection of knowledge independent of where the knowledge is coming from and how it was collected. An Ontology is a possible representation schema of a Knowledge Base, while a Knowledge Graph contains the acquisition and integration of data into an ontology, the ontology itself as well as the use of a reasoner to derive new knowledge. An ontology is further defined as a formally represented conceptualization of a domain. As ontologies allow for semantic modeling of any knowledge, they are often used as Knowledge Bases, especially for knowledge-based systems. The definition is visualised in figure 2.1. The schema shows the differences between the three terms mentioned above, where an ontology is a concrete implementation of a Knowledge Base. If a Reasoning Engine is applied to extend a Knowledge Base, it is called a knowledge-based system. Only when the integration and acquisition of data is included as well, the concept is called a Knowledge Graph.

The ExpCPS Knowledge Graph, which is the data source in this thesis, contains all three aspects of a KG as defined by [EW16]. An ontology was defined as an initial structure, containing the features and definitions of the data points. Using a data processing pipeline, data from the BIFROST simulation was integrated and could be used for data acquisition in the Knowledge Graph. Finally, a reasoning engine was applied to extend the Knowledge Base with all the missing information derived from the definition of the ontology.

On a final note, the definition quoted in this section, even though considering various important aspects of a Knowledge Graph, is only one of many possible definitions. This definition was chosen as it considers various sources of definitions and based on a meta analysis, it aimed to include the most important aspects of each individual definition. There is just not one commonly accepted definition of a Knowledge Graph. Many other researchers have tried to find a definition, but do not always agree with each other, resulting in a vast amount of proposed definitions. Additionally, the idea of graph-based knowledge representation has been around for decades and is not a completely new invention by Google. However, the introduction by Google has started a new golden era for this concept and sparked new research in the domain.

2.1.2 Implementation

In this section, terms commonly used for working with Knowledge Graphs will be explained. Additionally, the most commonly used method to build and store a Knowledge Graph, using RDF triples, will be explained.

Even though Knowledge Graphs are not restricted to any type of implementation, RDF (Resource Description Framework) triples are primarily used to represent data in a Knowledge Graph based on a consensus in the semantic web community. [WHCMS18]. In [FEM+18], a Knowledge Graph is even defined as an RDF Graph, making the use of RDF triples the main discriminative feature of a Knowledge Graph in their definition (even though this definition seems very restrictive and does not capture many important aspects of a KG). An RDF graph is a directed graph, which is built by using RDF triples, also called statements or facts. An RDF triple is an ordered set of three terms - head, relation and tail. An example of an RDF triple could be `(Vienna, locatedIn, Austria)`, where `Vienna` is the head, `locatedIn` is the relation and `Austria` is the tail of the triple. Each RDF term can have one of three different forms. It is either a IRI - Internationalized Resource Identifier (allowed for head, relation or tail), a blank node (allowed for head and tail) or a literal (allowed for tail). RDF triples are used to create a labelled directed relationship (relation) between two entities (head and tail). The terms head, relation, tail can be used interchangeably with the terms subject, predicate, object respectively, which can be observed in many research papers. By using a set of multiple triples, a graph is created [WHCMS18]. A collection of triples can be stored in a named graph, offering data managers the possibility to create multiple sets of triples. An RDF dataset can consist of multiple named graphs.

For querying data in an RDF graph, SPARQL is recommended by the W3C as the standard querying language. In a SPARQL query, a certain graph pattern is specified which is then matched against the RDF graph to be queried [WHCMS18]. It has some similarities with the query-language SQL for relational databases, which is where the name is derived from.

2.2 Knowledge Graph Embeddings

As discussed in the section above, Knowledge Graphs have gained a lot of attention in recent years as they are very effective in representing structured data. However, the inherent structure of Knowledge Graphs and their representation in triples makes it difficult to analyse the knowledge systematically as well as to manipulate the graph. Therefore, Knowledge Graph Embeddings have emerged as a new research field. The main idea of KGEs is to embed the components of a KG into a continuous vector space. This facilitates the manipulation of a KG, while its original structure stays untouched. Using these embeddings, various tasks can be tackled, such as KG completion, relation extraction or entity classification [WMWG17].

Knowledge Graphs tend to be incomplete, which means that there are connections between entities which are not present in the Knowledge Graph. The task of identifying these missing connections is called Knowledge Graph Completion. The goal of Knowledge Graph Completion, also called Link Prediction, is to find new relations in an existing Knowledge Graph, which are not present in the original graph but depict true facts. This task is especially interesting as the aim of the thesis is to find new causality links between events for the ExpCPS use case. Even without using Knowledge Graph Embeddings, new links can be derived from a Knowledge Graph by using defined rules and constraints in a reasoning engine (as mentioned in section 2.1). For example, if a Knowledge Graph contains the triple (`Vienna, capitalOf, Austria`), based on defined rules, we can additionally derive new triples. Considering that any capital needs to be a city and every capital is located inside the country it is the capital of, we can add the following triples to our Knowledge Graph: (`Vienna, type, city`) and (`Vienna, locatedIn, Austria`).

Unfortunately, these rules need to be defined in advance, usually by imposing constraints and rules in the underlying ontology. Considering the size of some Knowledge Graphs, this process may be labour-intensive or just infeasible. Additionally, some connections are unknown beforehand or are not clear enough to be defined by rules. Big Knowledge Graphs have a lot of information stored, but lack a coherent set of rules to derive new knowledge. Therefore, Knowledge Graph Embeddings have been developed to target this issue, as they aim to derive similarities in entities and relations which are not directly derived from rules and constraints. This means that new relations can be predicted even without having to define rules in advance. Thus, Knowledge Graph Embeddings open a whole new possibility of analysing or extending an existing Knowledge Graph. [BRC+20]

In the next paragraphs, the general setup of a Knowledge Graph Embedding will be explained before explaining in more detail the separate components needed for a KGE. Moreover, the most common embedding methods are discussed, describing their approach as well as pros and cons of each method.

2.2.1 General Implementation

The main idea of Knowledge Graph Embeddings is to create a representation of a specific Knowledge Graph in one or more low-dimensional vector spaces. This means that for each entity and each relation a vector is created based on their connections to other entities and their types of relations. Therefore, similar entities should be represented as similar vectors within the vector space. For analysis, the entities/relations can be compared by using vector similarity measures. [BRC+20].

There are three main components which compose the core of any Knowledge Graph Embedding. In Figure 2.2, these core layers are shown in the center. They are called the Lookup Layer, Scoring Layer and Loss Function. In addition to these three core components, for each KGE, an optimizer needs to be defined and negative

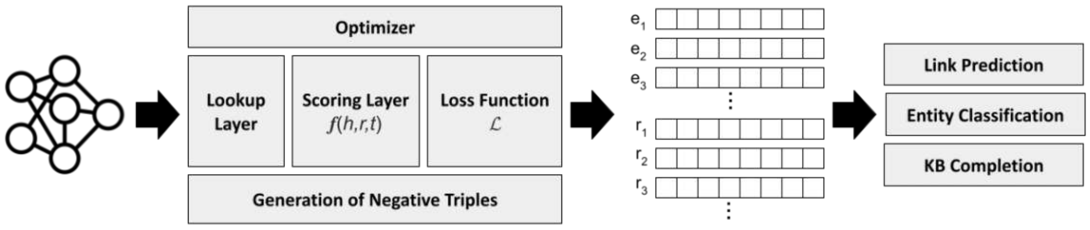


Figure 2.2: Schematic overview of a generic Knowledge Graph Embedding, adapted from [CPMJ20]

triples need to be generated for training. The choice of all these components is what defines the performance differences between various embedding methods. Based on this general architecture, a Knowledge Graph Embedding is trained like a gradient learning algorithm: by passing the training set through the algorithm multiple times for incremental performance increases in each epoch. The result of a trained Knowledge Graph Embedding is a vector representation of each entity and relation in the Knowledge Graph. This representation can then be further used to complete downstream tasks on the Knowledge Graph, such as Link Prediction, Entity Classification or Question Answering, as shown in figure 2.2.

In the following sections, each part of the KGE architecture (Lookup Layer, Scoring Layer, Generation of Negative Triples, Loss Function, Optimizer) will be explained in more detail and possible choices for each component are presented. Even though papers on different KGE methods mostly focus only on the Scoring Layer, the choice of the other components is equally important. A lack of transparency considering all the components of the KGE architecture has caused the emergence of many embedding methods claiming their performance is better than the performance of traditional methods, while they just used a different framework around their scoring function which caused the actual performance increase. Possible differences in the overall experiment architecture not attributed to the scoring layer may be different optimizers or loss functions. This lack of comparability results in a need for benchmark tests on the performance of new embedding methods. A new method can only be truly compared to existing methods when testing the two methods under the same overall framework ceteri paribus. Recently, several papers have tried to offer a unified framework to compare existing and new embedding methods [HZMT21]. In addition, some papers have conducted a meta-analysis, trying to compare embedding methods using the exact same setup and data for all methods [DWXG20]. These analyses show that the performance differences change quite a lot when controlling for outside factors and comparing performances even shows reverse effects in the new setups.

Additionally, I want to mention that due to the high research interest in Knowledge Graph Embeddings and the vast amount of possible embedding methods, it is not possible to explain or mention all available Embedding Methods in this section. Hence, I will focus on the most relevant ones to the research domain and most cited methods.

Variable	**Explanation**
E	the set of entities present in the Knowledge Graph
R	the set of relations present in the Knowledge Graph
`(h,r,t)`	a triple representing a fact of the form `(head, relation, tail)`, where $\{(h,r,t)\|h,t \in E, r \in R\}$
S	the set of triples in the Knowledge Graph
S'	a set of corrupted triples, see formula 2.12
$\mathbf{h},\mathbf{r},\mathbf{t}$	embedding vectors corresponding to the entities and relation of `(h,r,t)`
$\mathcal{L}$	A loss function as defined in an embedding method
$f(h,r,t)$	A scoring function as defined in an embedding method
$\mathbf{M}$	a numerical matrix
$\bar{\mathbf{a}}$	the complex conjugate of $\mathbf{a}$
$Re(\mathbf{x})$	the real part of a complex value $\mathbf{x}$
$\|\|\mathbf{a}\|\|_{\ell_1}$	Manhattan length of the vector $\mathbf{a}$, defined as $L_{\ell_1} = \|a_1\| + ... + \|a_n\|$
$\|\|\mathbf{a}\|\|_{\ell_2}$	Euclidean length of vector $\mathbf{a}$, defined as $L_{\ell_2} = \sqrt{a_1^2 + ... + a_n^2}$
n	number of distinct entities in a KG
m	number of distinct relations in a KG
o	number of distinct timestamps in a KG
d	dimension size of an embedding space
k	dimension size of a relational embedding space, if relevant
h', t'	corrupted head/tail

Table 2.1: Variable notations and explanations for Knowledge Graph Embedding definitions

For further information, an extensive list of possible embedding methods sorted by traditions, information types, augmentations and emergents, including the respective papers and, if available, code of each method is available here[1] .

For better understanding, table 2.1 shows a list of variables and their explanations used in the following section for the discussion of KGE methods and their components.

2.2.2 Lookup Layer

The first layer of the Knowledge Graph Embedding is the Lookup Layer. This layer is simply assigning an embedding vector to each entity and relation in the Knowledge Graph. As the name suggests, for each triple, the entities and relation are looked up in the embedding space and for new entries, new embedding vectors are created. For

[1] https://github.com/xinguoxia/KGE

initialization of embedding vectors, there are multiple possibilities, such as uniform, normalized, or Xavier initialization. While it seems unimportant what kind of initialization is chosen as weights are updated and overwritten again and again over the whole training process, bad initialization can have an impact on the performance of the whole model. For example, if initializing each weight with zero, or the same random number for each weight, it is hard to learn big differences between variables as the update magnitude tends to be similar for each variable. Also, if initial weights are either too small or too large, this may have an impact on the learning process. Therefore, random initialization based on a distribution function such as uniform or normal distribution is commonly used. Xavier initialization aims to find a good variance of weights without explicitly defining any type of distribution [GB10, Tri19].

2.2.3 Scoring Layer

The scoring layer consists of a scoring function, which assigns a score to each triple in a Knowledge Graph. The goal is to produce a high score for true triples (meaning a triple exists in a Knowledge Graph). For negative triples or wrong facts, a low score should be produced. The scoring function is the core of each embedding method and therefore a lot of different methods have been produced in research. Most traditional embedding methods use structure-based approaches. This means that they focus on the triples present in a Knowledge Graph without considering other information which might be available, such as temporal dynamics or semantic meanings of the entities. As triples are sometimes called facts in literature, these models are also called fact alone methods [BRC$^+$20]. The two most widely used types of structure-based methods are translational models and semantic matching models. Recently, Neural Networks for Knowledge Graph Embeddings have emerged, introducing deeper networks to improve the embedding process.

As opposed to embedding methods focusing only on triples present in the Knowledge Graph, there are methods which are aiming to leverage additional information to build a good representation of a Knowledge Graph, such as temporal information, entity types or textual descriptions. These methods are called enhanced KGEs. Temporal Knowledge Graph Embeddings are the most relevant enhanced KGEs for this thesis. Consequently, there will be a detailed explanation of this type of enhanced embedding while other possible enhancements will only be discussed briefly.

There are multiple papers conducting a meta-analysis of currently available Knowledge Graph Embedding Models, their implications, assumptions and performances. [WWMK20], [BRC$^+$20] and [DWXG20] form the basis of the following discussion about different embedding methods.

Translational Models

Translational Distance Models are using distance-based scoring functions to build a Knowledge Graph Embedding. Based on an input triple `(head, relation, tail)`,

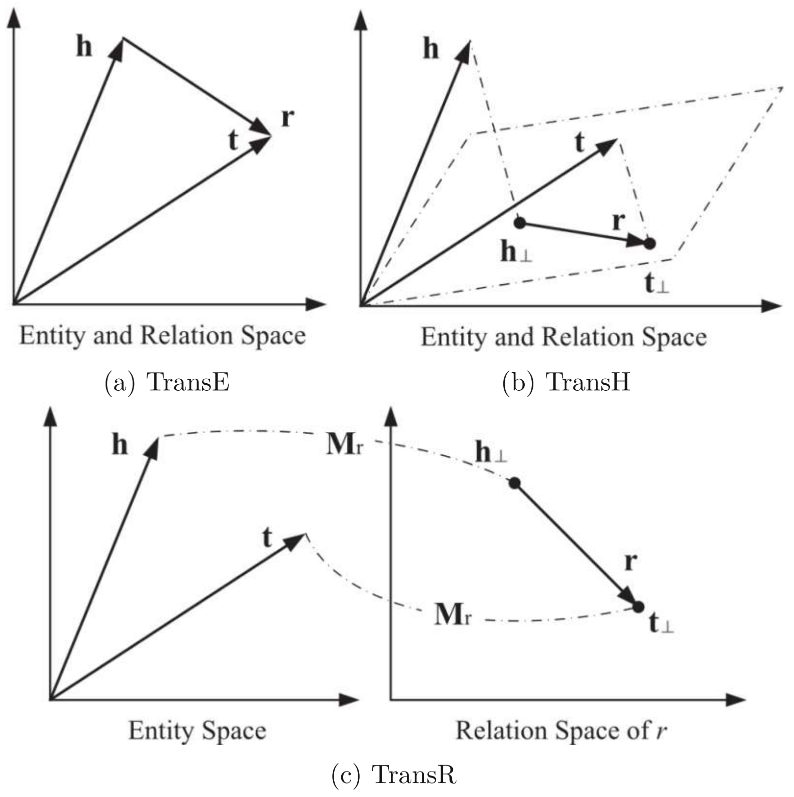


Figure 2.3: Simple Examples of Translational Knowledge Graph Embeddings [WWMK20]

they are learning the translation from the head to the tail, carried out by the relation of the triple. The intuition behind this translation is to connect head and tail through the relation with low error, i.e. $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if the triple (h, r, t) exists in a Knowledge Graph. Figure 2.3a shows an example of a very simple translation based on the triple (h, r, t) as defined in the TransE model.

TransE [BUGD+13] is the most representative translational distance model and has been one of the earliest KGE models in general. Its main contribution was the simplicity and efficiency of the embedding, which made it one of the first scalable embeddings, applicable even for big Knowledge Graphs. The authors of TransE use a common k-dimensional vector space to represent entities and relations. The idea is to use the relation as a translation from the head to the tail, as shown in figure 2.3a. As depicted, a common vector space is used in the TransE model to represent entities and relations. As a scoring function $f(\mathbf{h} + \mathbf{r}, \mathbf{t})$ is proposed, where f is either the $L1$-norm(ℓ_1, Manhattan Distance) or $L2$-norm (ℓ_2, Euclidean Distance) between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$ (in practice, the euclidean distance is used most of the time):

$$f(\mathbf{h},\mathbf{r},\mathbf{t})_{TransE} = ||\mathbf{h}+\mathbf{r}-\mathbf{t}||_{\ell_1/\ell_2} \tag{2.1}$$

TransE shows a very simple yet efficient representation of KG facts in a low-dimensional vector space. However, there are some limitations to the TransE embedding, especially when dealing with 1-N, N-1 or N-N relations (for definition of 1-N relations, see section 2.2.3). This comes from the definition of the scoring function and the shared vector space for entities and relations. As an example, let's assume the triples `(Neubau, locatedIn, Vienna)` and `(Neubau,locatedIn,Austria)` are present in a Knowledge Graph. Since the scoring function aims to accomplish $\mathbf{h}+\mathbf{r}\approx\mathbf{t}$, the elements `Vienna` and `Austria` would be represented by the same vector according to the scoring definition. However, there is obviously a differences between these entities. `Vienna` is a city and `Austria` is a country. They only have some common relations, but this does not make them the same entity.

Therefore, novel extensions of the TransE model have been developed to address these flaws, such as TransH [WZFC14b] and TransR [LLS+15]. Both models follow a similar general idea, as they allow entities to have different representations based on different types of relations. While `Vienna` and `Austria` may be similar based on the `locatedIn` relation, they may be very different for another type of relation, such as `numberOfInhabitants`.

TransH [WZFC14b] introduces relation-specific hyperplanes to create different representations of entities based on the type of relation. Each relation r is represented as a vector on a relation-specific hyperplane $\mathbf{r}$ with the normal vector of the hyperplane being $\mathbf{w}_r$. Therefore, the head and tail entities of a triple need to be projected onto the relation's hyperplane depending on the type of the relation to get their projections $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$. Assuming a triple of the form `(h,r,t)`, the projections are calculated as follows:

$$\mathbf{h}_{\perp\mathbf{r}} = \mathbf{h}-\mathbf{w}_r^\top\mathbf{h}\mathbf{w}_r \quad , \quad \mathbf{t}_{\perp\mathbf{r}} = \mathbf{t}-\mathbf{w}_r^\top\mathbf{t}\mathbf{w}_r \tag{2.2}$$

These projections are then fed into the scoring function, calculating the squared euclidean distance:

$$f(\mathbf{h},\mathbf{r},\mathbf{t})_{TransH} = ||(\mathbf{h}-\mathbf{w}_r^\top\mathbf{h}\mathbf{w}_r)+\mathbf{r}-(\mathbf{t}-\mathbf{w}_r^\top\mathbf{t}\mathbf{w}_r)||_2^2 = ||\mathbf{h}_{\perp r}+\mathbf{r}-\mathbf{t}_{\perp r}||_2^2 \tag{2.3}$$

With the use of relation-specific projections on hyperplanes, entities can have different roles for different relation types in the TransH model. Figure 2.3b visually shows the projection of head and tail entities onto a relation-specific hyperplane as defined in the TransH model.

TransR [LLS+15] While sharing the general idea of relation-specific embeddings, TransR uses relation-specific spaces rather than hyperplanes. For each type of relation, a projection matrix $\mathbf{M}_r$ is defined to project entities from the entity space to a relation-specific vector space:

$$\mathbf{h}_r = \mathbf{h}\mathbf{M}_r \quad , \quad \mathbf{t}_r = \mathbf{t}\mathbf{M}_r \tag{2.4}$$

The scoring function is then equal to the scoring function of TransH, considering the projected entities:

$$f(\mathbf{h}, \mathbf{r}, \mathbf{t})_{TransR} = -||\mathbf{h}\mathbf{M}_r + \mathbf{r} - \mathbf{t}\mathbf{M}_r||_2^2 = -||\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r||_2^2 \tag{2.5}$$

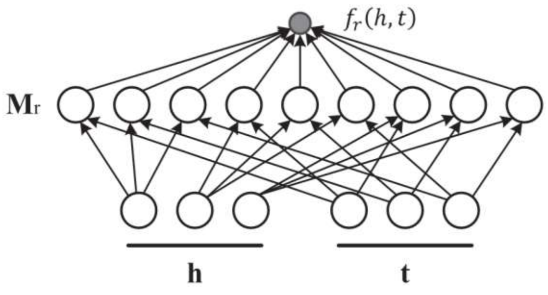
The projection of entities to the relation space as defined in TransR is shown in figure 2.3c. The introduction of a projection matrix for each type of relation increases the number of parameters which need to be learned. Therefore, model performance becomes more complex and less efficient. In later approaches, the complexity of TransR is reduced to increase computational efficiency. In TransD, the projection matrices are decomposed into a product of two vectors to reduce model complexity from $\mathcal{O}(nd + mdk)$ in TransR to $\mathcal{O}(nd + mk)$ in TransD. However, all methods mentioned above have one big issue in common: None of them takes into account the heterogeneity of relations, as some relations may form a lot of connections between entities while others do not. This can cause overfitting on very general, simple relations and underfitting on more specific or complex relations [DWXG20]. The TranSparse model aims to use sparse projection matrices to reduce complexity and account for the number of occurrences of a relation to reduce overfitting of common relations.

KG2E [HLJZ15] and **TransG** [XHHZ17] aim to incorporate uncertainties of relations in the model training process. They both use Gaussian distributions to effectively model uncertainties of entities and relations in a Knowledge Graph [WMWG17]. While KG2E merely focusses on the uncertainty of relations, TransG also considers multiple semantic meanings of a relation by using a mixture of Gaussian distributions. This leads to the result that TransG outperforms most models in the benchmark Knowledge Graphs for link prediction [DWXG20].

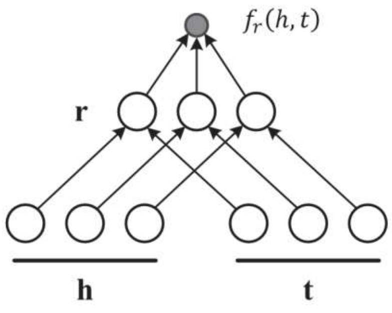
There are many more translational approaches for KGEs which perform better than the presented approaches in specific tasks. However, these approaches are also becoming increasingly complex, which requires high computational resources to apply them to bigger Knowledge Graphs.

Semantic Matching Models

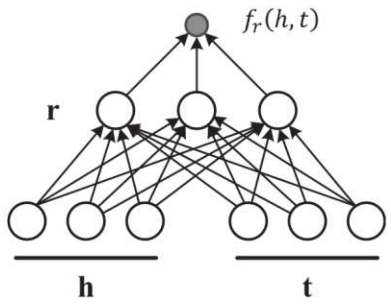
Semantic Matching Models are leveraging latent semantics present in the vector embeddings of entities and relations to measure the plausibility of triples. Compared to the distance-based scoring of translational models, semantic matching models apply



(a) RESCAL



(b) DistMult



(c) HolE

Figure 2.4: Examples of Semantic Matching KGEs, sourced from [WWMK20]

similarity-based scoring functions. In general, semantic matching models, also called bilinear models, achieve good results in link prediction tasks according to [KP18].

RESCAL [NTK11] was one of the earliest embedding models, developed in 2011. It uses a full-rank matrix for each relationship. Interaction between elements is modelled by matrix-vector products. The use of a full-rank matrix allows for high expressive power. The scoring function of RESCAL is defined as follows:

$$f(\mathbf{h},\mathbf{r},\mathbf{t})_{RESCAL} = \mathbf{h}^\top \mathbf{M}_r \mathbf{t} = \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} [\mathbf{M}_r]_{ij} \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_j \tag{2.6}$$

$\mathbf{M}_p$ is the matrix associated with the relation of the triple. Each type of relation has their own interaction matrix, which should represent the interaction between two entities through this relation. Figure 2.4a shows the interaction of **h** and **t** through relation **r** in the RESCAL scoring function. In this function, pairwise interaction between all components of the head and tail entities is captured through the relation matrix. Due to the high expressivity, RESCAL is prone to overfitting, offering a big opportunity for new approaches trying to address this issue and improve prediction performance [WWMK20].

DistMult [YYH+15] introduced a simplified version of RESCAL by restricting the interaction matrix $\mathbf{M}_r$ to diagonal matrices. Instead of a full-rank matrix, a vector embedding r is defined for each relation and the interaction matrix is defined as the diagonal matrix of vector r. The scoring function of DistMult is thus defined as:

$$f(\mathbf{h},\mathbf{r},\mathbf{t})_{DistMult} = \mathbf{h}^\top diag(\mathbf{r})\mathbf{t} = \sum_{i=0}^{d-1} [\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_i \qquad (2.7)$$

In this scoring function, pairwise interactions are only captured between components of the two entities along the same dimension, as shown in figure 2.4b. The massive reduction in complexity causes the model to only be able to represent symmetric relations, which is a big limitation in its capabilities. [WWMK20] However, the reduction in complexity also reduces requirements in computational efforts and memory. Unfortunately, application on large-scale Knowledge Graphs with different types of relations of the DistMult model typically is not very performant.

HolE [NRP16] introduces a circular operation between head and tail to capture interactions not only in the same dimension, but also compositional interactions over multiple dimensions. Therefore, interactions caused by multiple dimensions of a relation can be captured, which makes it possible to model complex relations between entities. Figure 2.4c shows the increased number of captured interactions, while only using a single vector $\mathbf{r}$ for the transformation. The circular correlation operation is denoted as $\star$ and is defined using the fast Fourier transformation $\mathcal{F}(\cdot)$ [DWXG20]:

$$h \star t = \mathcal{F}^{-1}(\overline{\mathcal{F}(h)} \odot \mathcal{F}(h)) \qquad (2.8)$$

Using this transformation, the scoring function of HolE is defined as:

$$f(\mathbf{h},\mathbf{r},\mathbf{t})_{HolE} = \mathbf{r}^\top (h \star t) \qquad (2.9)$$

As this circular correlation operation is not commutative, HolE is able to model non-symmetric relations, adding an important capacity to bilinear models.

ComplEx [TWR+16] aims to extend the DistMult model by embedding the entities and relations in a complex vector space. This means that the vectors **h,r,t** are not restricted to the real space, but can also lie in the complex space. This introduces the possibility to model asymmetric relations, similar to HolE. In fact, [HS17] proved that ComplEx and HolE are isomorphic. This means that any ComplEx model has an equivalent HolE model. The scoring function for the ComplEx model is defined as:

$$f(\mathbf{h},\mathbf{r},\mathbf{t})_{ComplEx} = Re\left(\mathbf{h}^\top diag(\mathbf{r})\bar{\mathbf{t}}\right) = Re\left(\sum_{i=0}^{d-1}[\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\bar{\mathbf{t}}]_i\right) \tag{2.10}$$

where $\bar{\mathbf{t}}$ is the conjugate of $\mathbf{t}$ and $Re(x)$ is the real part of the complex value of x. Extending the vectors of entities and relations to the complex space allows for different scores of a triple depending on the order of the entities, which is the definition of an antisymmetric relation.

Neural Network Models

As deep learning has become a popular and extensively used research topic in many applications, neural network-based Knowledge Graph Embeddings have been proposed as well. They show promising performances for link prediction and seem to be very effective in finding good embeddings for a Knowledge Graph. However, they suffer from low explainability due to the black-box nature of neural networks. Additionally, a huge number of parameters needs to be trained due to multiple layers in the architecture. This results in high computational complexity, which further results in poor scalability of the models [DWXG20]. Proposed methods are Semantic Matching Energy (**SME**) [BGWB14], Neural Tensor Networks (**NTN**) [SCMN13] and **ConvKB** [NNNP18], introducing convolutional layers to Knowledge Graph Embeddings.

Enhanced Embedding Methods

As mentioned before, new embedding approaches aim to use not only triples present in a Knowledge Graph, but also try to leverage additional information. The following list aims to give an overview of possible information which can be exploited to improve embedding performance. Since most of the additional information is irrelevant or not available for the Knowledge Graph used in this thesis, these possibilities will only be discussed briefly, while relevant embedding enhancements (temporal embeddings) will be addressed in more detail.

Textual Description For many Knowledge Graphs, there is additional text information about entities and relations available. Some embedding approaches try to leverage this information to create a better performing embedding of the Knowledge Graph. In [WZFC14b], jointly embedding words and Knowledge Graph entities in the same vector space is proposed, claiming to show promising results for increased prediction performance. Additionally, out-of-KG facts can be predicted by using joint embeddings of facts and textual descriptions. This is not possible in facts-alone embeddings as there is no possibility to learn any information which is not contained in the trained KG.

Relation Path Another idea to improve Knowledge Graph Embeddings is to focus on relation paths. While single triples only contain one hop of a relation, [LLL+15] claims that important patterns can only be detected by considering multi-step relation paths as well. Additionally, a reliability measure of each relation path is calculated to find the most reliable paths. This approach could potentially yield good results when applied to the ExpCPS KG, as an event explanation should be seen as a path of causality relations. However, exploring this option is not within the scope of this thesis.

Image Enhancement Arguing that images of entities provide rich information to improve the embedding of entities, [XLLS17] proposes to add image representations constructed by a neural image encoder to include visual information into a Knowledge Graph representation via attention-based methods.

Logic Enhancement Addressing the fact that traditional embedding methods only use facts (triples of the form `(head, relation, tail)`), [WWG15] proposes to use logical rules by formulating them as an integer linear programming (ILP) problem. This method converts logical rules into triples, which can be used to train any traditional embedding method, such as TransE.

Temporal Information Arguing that most Knowledge Graphs are changing over time, researchers have started to investigate the inclusion of temporal information in Knowledge Graph Embeddings. Especially relations are potentially time-variant. As an example, the triple `(Bob, livesIn, Vienna)` is only valid until Bob moves to another city. To include temporal information in a Knowledge Graph, triples are extended to quadruples of the form `(head, relation, tail, timestamp)`. The timestamp is describing a point in time for which the triple is valid [HZMT21].

As temporal embedding methods are currently highly researched, many different approaches have been proposed. There are two main components of a temporal KGE method which can be addressed in a new embedding approach - the temporal embedding (how the timestamps are included in an embedding) and the embedding model (which model to use for the Knowledge Graph Embedding). To compare the performance differences for each component separately, a meta-analysis of temporal Knowledge Graph Embeddings was conducted by [HZMT21]. In a grid search analysis, six different temporal embeddings as well as four different embedding methods were tested for a wide range of hyperparameters. Interestingly, the most simple, naive timestamp embedding approach performs as well as the more advanced embedding approach, provided that it was trained properly.

The so-called "naive" model in this analysis is called TTransE, which is an adaption of the conventional TransE model to include temporal facts in the embedding. The main intuition is to embed the timestamp in the same vector space as entities and relations. The adapted scoring function for TTransE is defined as follows:

Model	Score $f(\mathbf{h},\mathbf{r},\mathbf{t} <,\mathbf{z} >)$	Space Complexity	Symmetry	Antisymmetry	Inversion	Composition	1-N Relations
TransE	$\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|_2$	$\mathcal{O}(nd+md)$	✗	✓	✓	✓	✗
TransH	$\|\|\mathbf{h}_{\perp r}+\mathbf{r}-\mathbf{t}_{\perp r}\|\|_2^2$	$\mathcal{O}(nd+md)$	✓	✓	✓	✗	✓
TransR	$\|\|\mathbf{h}_r+\mathbf{r}-\mathbf{t}_r\|\|_2^2$	$\mathcal{O}(nd+mdk)$	✓	✓	✓	✗	✓
RESCAL	$\mathbf{h}^\top \mathbf{M}_r \mathbf{t}$	$\mathcal{O}(nd+md^2)$	✓	✓	✓	✗	✓
DistMult	$\mathbf{h}^\top diag(\mathbf{r})\mathbf{t}$	$\mathcal{O}(nd+md)$	✓	✗	✗	✗	✓
HolE	$\mathbf{r}^\top (h \star t)$	$\mathcal{O}(nd+md)$	✓	✓	✓	✗	✓
ComplEx	$Re\left(\mathbf{h}^\top diag(\mathbf{r})\bar{\mathbf{t}}\right)$	$\mathcal{O}(nd+md)$	✓	✓	✓	✗	✓
TTransE	$\|\|\mathbf{h}+\mathbf{r}+\mathbf{z}-\mathbf{t}\|\|_2$	$\mathcal{O}(nd+md+od)$	✗	✓	✓	✓	✗

Table 2.2: Summary of Embedding Models and their capabilities, according to [WWMK20]

$$f(\mathbf{h},\mathbf{r},\mathbf{t},\mathbf{z})_{TTransE} = ||\mathbf{h}+\mathbf{r}+\mathbf{z}-\mathbf{t}||_2 \tag{2.11}$$

where **z** denotes the timestamp of the triple.

Summary of Models

Table 2.2 shows a summary of each model, which was discussed in detail in this section. It shows the scoring function of each model, its space complexity as well as which types of relations a model is able to learn. The types of relations are defined as follows:

Symmetry: a relation r is symmetric if
$\forall x, y : r(x, y) \Rightarrow r(y, x)$

Antisymmetry: a relation r is antisymmetric if
$\forall x, y : r(x, y) \Rightarrow \neg r(y, x)$

Inversion: relation r_1 is inverse to relation r_2 if
$\forall x, y : r_1(x, y) \Rightarrow r_2(y, x)$

Composition: a relation r_3 is composed of relations r_1 and r_2 if
$\forall x, y, z : r_1(x, y) \wedge r_2(y, z) \Rightarrow r_3(x, z)$

1-N-Relation: a 1-N relation (also called one-to-many relation) exists if for a relation $r(x, y)$, each entity x is allowed multiple relations r to different entities y, while each y is only allowed to be related to one x through relation r.

Depending on the scoring function of a model, different types of relations can be modelled in the embedding space. In table 2.2, we can see that none of the models is able to encode all of the types of relations discussed in this section (more types of relations exist than shown here, but these are some of the more important ones). Depending on the use case, some types of relations are more important, so the choice of an embedding model should always be guided by the characteristics of the Knowledge Graph one wants to analyse.

Additionally, the space complexity of the models is depicted in table 2.2. n, m, o denote the number of entities, relations, and timestamps (if applicable) of a Knowledge Graph, while d and k are the number of dimensions of the entity and relation embedding space, respectively (mostly $d = k$). While most models have a linear space complexity regarding the number of entities and relations, TransR and RESCAL need a lot more computational memory than the others as they need to learn more parameters during training. Furthermore, when including temporal information, more memory is required for training as timestamps need to be embedded as well.

2.2.4 Generation of Negative Triples

Another part of the KGE framework (as explained in figure 2.2) is the generation of negative triples. In common machine learning tasks, there is a clear set of positive and negative facts which an algorithm should learn to distinct. As Knowledge Graphs are generally incomplete, meaning that true facts are potentially missing from the data, the generation of negative facts needs some separate consideration. Generally, there are two widely used assumptions guiding the generation of negative triples for Knowledge Graph Embedding training: Open World Assumption (OWA) and Closed World Assumption (CWA). [WWMK20]

In the Closed World Assumption, all facts not present in a Knowledge Graph are assumed to be false. In this case, a simple loss function as known in Machine Learning can be used, such as the squared error, to draw a distinction between true and false triples. However, CWA does not hold for incomplete Knowledge Graphs (as most KGs are). This results in bad prediction performance when using the Closed World Assumption in a Knowledge Graph Embedding. [WWMK20]

The Open World Assumption claims that all triples present in a KG are true facts, and all triples not present in the KG may be either false or missing facts. Therefore, it is not possible to just use any non-observed facts as negative facts. [BUGD+13] proposed the creation of negative triples by randomly replacing either the head or the tail of a triple with any random entity present in the Knowledge Graph. The formula to create a set of corrupted triples S' is defined as follows:

$$S' = \{(h', r, t)|h' \in E\} \cup \{(h, r, t')|t' \in E\} \tag{2.12}$$

where E is the set of entities present in the Knowledge Graph. While this is an effective way to create negative triples with low computational effort, there is a high chance of creating false negatives.

In the paper introducing TransH [WZFC14a], the authors argued that the probability of a false negative is depending on the type of relation in a triple. As an example, in the triple `(Bob, gender, male)`, there is a higher probability of creating a false negative triple when replacing the head (i.e. `(Max, gender, male)` than when replacing the tail(i.e. `(Bob, gender, female)`. The reason is that `gender` is a N-1 relation, so each head can only have one tail, while each tail can have multiple heads. Therefore, [WZFC14a] proposed to introduce a Bernoulli distribution to assign different weights to the probability of replacement for the head or tail in a triple based on the number of head entities per tail and vice versa for each relation in the Knowledge Graph in order to reduce false negatives in the embedding.

2.2.5 Loss Function

The loss function is the final layer of the Embedding Model, where the error of the embedding is calculated based on the scores of negative and positive triples. Loss functions are what converts the scores from any embedding model to calculate the performance of the scores. They are necessary to update weights of a model. There are multiple types of loss functions available. The main distinction of loss functions can be drawn between pointwise loss functions and pairwise loss functions. In the first, each triple is calculated separately by evaluating the score of the triple against the label of a triple (true or false). Examples of pointwise loss functions are the squared loss, pointwise hinge loss or pointwise logistic loss [ABH+21].

A pairwise loss function is applied to a positive triple $\mathbf{x}$ and a negative triple $\mathbf{x}'$ simultaneously and computes a value for the pair of triples. The idea is to maximize the difference in scores for positive and negative triples. The most relevant pairwise loss functions are the margin ranking loss (pairwise hinge loss) and the pairwise logistic loss. The margin ranking loss is defined as follows [ABH+21]:

$$\mathcal{L}(\mathbf{x},\mathbf{x}')_{hinge} = max\,(0, f(\mathbf{x}) + \gamma - f(\mathbf{x}')) \tag{2.13}$$

where γ is the margin parameter separating the two triples.

The pairwise logistic loss is defined as follows [ABH+21]:

$$\mathcal{L}(\mathbf{x},\mathbf{x}')_{logistic} = \log(1 + \exp\,(f(\mathbf{x}) - f(\mathbf{x}') + \gamma) \tag{2.14}$$

Additionally, a regularizer can be applied to the loss function, such as an L1, L2 or L3 regularizer. For convolutional approaches, dropout can also be used. [CPMJ20]

2.2.6 Optimizer

Once all components, which make up the core of a Knowledge Graph Embedding, are defined, there is only one final piece missing: the optimizer. Even though not directly related to the embedding model itself, the optimizer is used to define the learning rate throughout model training. Here, the same optimizers can be used as in usual machine learning models. Therefore, either a fixed learning rate can be defined or any learning rate scheduler can be used, such as SGD, Adam or AdaGrad. These schedulers define the learning rate dynamically for each training epoch based on previous performance improvements.

Now, based on the definition of all five elements - Lookup Layer, Scoring Layer, Loss Function, Generation of Negative Triples and Optimizer - the embedding is ready to be trained. For this, the training data is repeatedly passed through the training architecture, aiming to improve the performance in each epoch. By using multiple epochs, in which the training data is fully passed through the training algorithm, training loss should be minimized to optimize prediction performance of the embedding.

2.3 Event Explainability

The goal of using KGE on the ExpCPS Knowledge Graph is to improve link prediction on causality links between events in the simulated smart grid system, which is a Cyber-Physical Energy System. In essence, causality prediction is a tool to find explanations of events in a system. Finding these new `causedBy` connections helps to improve the explainability of the energy system overall. Currently, existing literature contains a vast amount of papers on KGE for link prediction and Knowledge Graph completion, as well as on attempts to make cyber-physical systems more transparent and explainable. However, extensive research has not uncovered any research focussed on applying KGE to improve the event explainability of a running CPS. In this section, a definition of event explainability is given as well as current methods to achieve it.

At first, the concept of an event should be properly defined for future use in this thesis. In [WMT$^+$07], the definition of an event is as follows: "an event refers to a real-world occurrence that unfolds over space and time". This definition shows the importance of the two dimensions (space and time) for any event. Therefore, any analysis of events needs to consider these two layers as any event is staged during a certain time at a certain place.

Event Explainability as such is a trickier term to define. For this thesis, the definition of [Ary21] is used as this definition was also used when creating the ExpCPS Knowledge Graph:

> "Explainability is the ability of a (software) system to provide explanations about its states or behaviors in terms of a set of facts."

Event though causality is not explicitly mentioned in this definition, it is a crucial aspect of being able to explain how a system's state happened to occur. Therefore, causality between a set of facts and the system's behaviors is implicitly assumed in this definition. Hence, the term causality should be investigated further as well. Even though causality seems to be a trivial concept at first, its definition has occupied many philosophers and researchers for thousands of years. Nobel Price Winner Clive Granger has defined the Granger-Causality, which is based on prediction. It addresses the differences between correlations between events which happen by chance and actual causality, which he calls "Granger-Causality".

> "If a signal X_1 "Granger-causes" (or "G-causes") a signal X_2, then past values of X_1 should contain information that helps predict X_2 above and beyond the information contained in past values of X_2 alone." [Set07]

This definition shows the importance of a connection between two events to assume causality. The connection needs to be backed by data to be valid, especially considering that humans tend to find connections to prove causalities even though they are non-existent in data.

2.3.1 Finding Causality in connected environments

As systems are becoming more complex, there is an increasing need for explanations of why any given event is happening in a system. To achieve this goal, events need to be analysed to find their potential causes. Researchers aim to find causalities between entities and events in many different research domains as causality is what is needed to derive suggested changes in a system. Randomized controlled trials are the gold standard to discover causal relationships between two events. They require a controlled experiment setup, where a single change can be made in a setup to measure its impact. These experiments are however very difficult to set up, costly and for some situations not possible to conduct. Therefore, causal analysis by inferring causal relationships through modelling cause-effect relationships from observational data have been developed in recent years [JGC+21]. The idea is to use computational models, such as graphical models or deep neural networks to derive causalities from event sequences.

For any model aiming to find causalities in observational data, two main problems need to be addressed: event detection and event explanation. Event detection implies the task of finding events in a system based on a set of observations. Event explanation, on the other hand, is concerned with finding explanations for an existing set of events based on observations that happened during or right before an event. While a lot of researchers work on solutions for event detection, less research has been published on finding event explanations. The two main approaches mentioned in [GSL+21] are using machine learning models to enrich video sources to get event explanations during a video sequence (which requires access to relevant videos) or working with heterogeneous data sources from different sensors to interpret events.

As this thesis is working with a simulated Cyber-Physical System, the first approach, which is fostering video material for causality detection, is not applicable to the available data. However, the latter approach uses the same kind of data as the data in the ExpCPS use case and will therefore be investigated further. The core problems addressed by any framework considering event explanation using heterogenous data sources is the integration process of multiple data sources to make the data usable for analysis and, further, to facilitate interpretability [GSL+21, WZL06]. All frameworks mentioned in [GSL+21], which work on integrating multiple heterogeneous data sources, are applying semantic technologies in some way or another. As storing complex heterogeneous data sources and their connections in one system is what semantic technologies are designed for, they seem to be a good fit for this use case. These features can be exploited in the next steps as well, where I am aiming at finding explanations of a system's behaviour. Semantic research also enables researchers to use existing general-purpose knowledge about a system for analysing its behaviour [Ste18].

2.3.2 Explainability in Cyber-Physical Systems

Even though the process of integrating information to make it centrally available is crucial for any further use of the data collected by sensors, explainability is not automatically achieved by just making data available. An additional step is needed to create an explainable system. [BGCG+19] proposed a framework called MAB-EX (Monitor, Analyze, Build, Explain) to build self-explainable Cyber-Physical Systems. The proposed framework uses four steps to find explanations when needed:

1. Monitor the system and its environment at run-time
2. Analyse the monitored data and identify situations which request an explanation
3. Build an explanation for the given situation
4. Provide the explanation to stakeholders in an appropriate form.

While all of these four steps are needed to provide a fully self-explainable system, step 3 - building an explanation for a given situation - will be the core problem addressed by this thesis. Steps 1 and 2 have already been addressed in the development of the ExpCPS Knowledge Graph [AES+20, AES+21]. Processing new-found explanations further to make them understandable to stakeholders, as suggested in step 4, will be an interesting topic for further research.

CPS have benefitted a lot from machine learning models. Compared to their predecessors, smart systems have better capabilities for "self-healing, situational awareness, information interaction and stability" [XLL+22]. However, with increasing performance of machine learning and AI models, the complexity of decision-making within a system has increased a lot in recent years. By using Machine Learning (ML) algorithms, new relationships can be found, as complex mechanisms in the system can be modelled and learned. This means that ML models can break through the barrier of existing knowledge, uncovering possibly important new knowledge. However, with more complex models and increased prediction performance interpretability has become worse. This is a problem especially for domains with critical infrastructure, where interpretability of the decision-making process is crucial [XLL+22]. When a model is deployed in a working environment, faulty decisions can have a direct impact on users' well-being, comfort, and security.

While many explainability approaches focus on the decision-making process based on the input data, explaining what has happened, what is currently going on, and what will occur in the future [GSC+19], there are additional factors which need to be considered to get a truly explainable system. First, the type of explanation needed for a system is highly dependent on the domain it is operating in. A system applied in the legal domain needs to focus on different types of explanations than a system deployed at a production site. Second, there are many different types of users interacting with a system, at different points in time. Explainability should consider this by giving adequate explanations catered to the needs of a user. For example, in a smart grid a system operator may want explanations on system failures to know which components are at fault or which parts of the system need further investigations on possible improvements. In contrast, an energy consumer needing to charge an electric vehicle may want to know how much energy will be produced and available in the next hours.

Shifting from Explainable AI systems to Explainable CPS, considering the context and users is even more important for expCPS than for XAI. As XAI models aim to explain the workings of an algorithm, they often overlook the impact of physical and virtual influences on the system decisions. The goal of expCPS is to take external influences on an event into account as well. While the process of a decision-making model in the background is only one part a user is interested it, the main goal is to find an explanation based on changes in the system context, such as increased solar intensity leading to increased energy production. Therefore, model-agnostic explanation systems aim to retrieve explanations from a system's historical data without analysing or interrupting the CPS itself [Jha22].

2.3.3 Semantic Technologies for Explainability

Semantic Technologies are designed to store complex and heterogeneous data and their connections. These features can be exploited when aiming to find explanations of a system behaviour in CPS. Semantic Research enables researchers to use existing general-purpose knowledge about a system for finding causality relationships between events [Ste18]. Especially when faced with heterogeneous data sources, all papers describing event explainability in connected environments use semantic technologies to model the data. Ontologies, which are used to build a semantic data model, are designed to capture complex relationships, different types of sensor data and their connections. In 2012, an ontology was proposed as a basic data model to store sensor data called Semantic Sensor Network Ontology [CBB+12]. The proposed ontology defines how sensors, measurements and observations as well as sensor deployments should be stored. Based on this work, many additional extensions have been developed to fit this ontology to specific use cases, such as smart building diagnosis [PSL14] or connected environments [Man19]. Based on the work of [PSL14], [AES+21] investigated the use of Knowledge Graphs for solving the problem of system explainability in a Cyber-Physical Energy System. The framework merges data from multiple sources, such as simulation data, system setup and known causality relations.

All these ontologies help to collectively store heterogeneous data sources in one data model to enable causality analysis of a system. However, none of them has yet suggested a solution to actually derive causality from the data model. Currently, causalities are extracted from a data model by using domain expert knowledge to learn potential causalities and querying them in the data. In [ADES21], connections and causal relations between components of the smart grid are added to the existing Knowledge Graph by domain experts. For this, domain expert knowledge was broken down into rules to be able to query causalities in the Knowledge Graph using SPARQL. Even though using SPARQL queries assists in capturing events with the same setup efficiently, this approach is still labour-intensive and is dependent on known rules which are defined by domain experts. Therefore, a new approach is tested in this thesis: using Knowledge Graph Embeddings for embedding the ExpCPS Knowledge Graph in a low-dimensional vector space to derive new information based on the existing data.

CHAPTER 3

Methodology

In this chapter, the tools and methods used in the thesis will be explained in detail. Decisions on the choice of models, experiment architecture and evaluation metrics will be presented and the reasons behind these choices will be elaborated and discussed. The first section is based on the embedding methods presented in section 2.2. The focus will be on defining the most suitable KGE models to be implemented to address the research questions, depending on the use case and their features. In the process, reasons for design and methodology decisions will be given. In the second section, the training architecture used in the experiments will be explained in more detail, accompanied with reasons on why this architecture was chosen as well as its advantages and necessary assumptions to be made. In the final section of this chapter, the evaluation metrics used to evaluate the models will be explained. This includes the formulas of the metrics as well as what aspect of performance they represent.

3.1 Knowledge Graph Embedding Methods

In section 2.2, a detailed analysis of well-known Knowledge Graph Embedding techniques and their intuitions was given. An analysis of prediction performances for all models on benchmark Knowledge Graphs (based on Wordnet and Freebase) was conducted in [DWXG20]. In this analysis, TransG was one of the best performing models in the analysis over both datasets. As already mentioned in the previous chapter, the main improvement of TransG and KG2E compared to other translational models is their ability to model uncertainty in semantics. This feature can improve performance drastically for the benchmark KGs, such as Wordnet, as these KGs contain a lot of entities and relations based on natural language, resulting in ambiguous and uncertain relations. However, this is not a major issue with the ExpCPS Knowledge Graph used in this thesis as no data is derived from natural language data. The ExpCPS KG is modelling a simulation of a smart grid, where observations made by sensors are modelled in a Knowledge Graph. This simulation is explicitly defined to have clear connections with unambiguous names and is complete in itself. Therefore, uncertainty should not be an issue. Furthermore, the increased complexity which is introduced by considering multiple Gaussian distributions requires high computational efforts which can not be addressed in this thesis.

To chose the right model for the link prediction task of ExpCPS Knowledge Graph, I want to analyse the types of relations present in the KG and which models are able to represent them in an embedding. Based on the schematic model of the ExpCPS Knowledge Graph in figure 1.2, the types of relations present in the graph can be analysed. A detailed introduction to the Knowledge Graph and the data will be available later in the thesis (section 1.3).

Symmetric relations are present through the relation `hasConnection`, which indicates that two features of interest are physically connected to each other. Symmetry cannot be modelled by a TransE model based on table 2.2.

Antisymmetric relations are an important part of the graph, as the `causedBy` relation is antisymmetric. As already mentioned before, the causedBy relation is a crucial relation in the experiment as this is the relation which should be predicted by the models. Additionally, there is only a very limited number of `causedBy` relations in the data, compared to the other relations, making it even more important to find a model which can correctly represent this type of relation. All models except DistMult can model antisymmetry.

Inversion is not present in the Knowledge Graph, so the ability of a model to represent inversion is negligible. Composition, on the other hand, plays an important role in the KG. The connections between entities, which are composed of multiple triples are relevant to understand the overall picture in a Knowledge Graph. Especially in the ExpCPS KG, the goal is not to model simple relations, but to understand the entirety of the graph and its connections. For example, no event is directly connected to a timestamp, but it is connected to an observation, which contains a timestamp. This composition can be important for model performance. Unfortunately, only TransE can inherently model compositional relations.

Finally, **1-N relations** are also crucial in the KG as multiple observations are connected to a single feature of interest. Moreover, an event can potentially have multiple causes to form a full explanation. This is a feature of the Knowledge Graph which needs to be addressed, as the inherent reason to use a Knowledge Graph for modelling the data is the possibility to connect observations to their context. The importance and the setting around an observation can only be comprehended by including the sensor which measured the observation and the location of the feature of interest. Except TransE, all KGE models can process 1-N relations.

After having considered various types of relations and their importance in the ExpCPS Knowledge Graph, there is not a single model which can model all important types of relations in the graph. As no model fits the job perfectly, I decided to use four different models to test their performance and check how well they can predict relations in the Knowledge Graph. The idea was to use models based on different intuitions to check which one works best.

The first model to be tested is TransE, as it is the only model which can work with compositional relations. Additionally, it is one of the first KGE models which has been proposed and it uses a very simple scoring function. Hence, it can be used as a baseline model for more complex setups. TransE will also be used in this experiment as a representation of the translational KGE models.

As an improved version of a translational model, TransH shows promising results in a performance analysis. Since TransH requires less memory than TransR with similar performance results, TransH is the second model added to the analysis. Generally, TransH is expected to better model the relations between various entities as is uses separate hyperplanes for each type of relation.

As a representative for semantic matching models, ComplEx is applied as well. ComplEx is the model which is recommended to be used for initial analysis when starting with KGE on any Knowledge Graph, according to [DWXG20], as it generally shows very good results even when using default parameters. Moreover, ComplEx shows the best results of semantic matching models in the benchmark KGs.

Finally, TTransE (model four) will be used to test the effect of using temporal information directly in a KGE. Since the model is a very simple extension of TransE, the direct comparison of TransE and TTransE can show the direct effect of timestamps added to each triple.

Overall, a set of four KGE models will be tested in the thesis. Three models, which are based on a translational approach (TransE, TransH, TTransE) and one model based on a semantic matching approach (ComplEx). TTransE will be used to check the effect of using an enhanced embedding method, as it incorporates temporal information into the embedding, which is not directly considered in the other models.

3.2 Training Data Architecture

Now that the embedding methods to be used in the experiment are defined, the next topic to be considered is the data setup for conducting the experiment. As in all Machine Learning projects, the data needs to be split into a training set, a test set, and a validation set. The training set will be used to train the embedding models, while the test set is used to determine the performance of the embeddings. Only in the last step, when the training process is finished and the final state of the model is determined, the validation set is used to measure the model performance on new, unseen data. The validation set aims to imitate new data which has not been seen by the model before.

While this approach seems reasonable and straight-forward, the setup of the ExpCPS Knowledge Graph makes it difficult to implement this approach. Since the goal of using

	topology	inferred	time	events	explanation
Village A	train/test	train/test	train/test	train/test	train/test
Village B	train/test	train/test	train/test	train/test	validation

Table 3.1: Training, test and validation split of the simulation data for model training

embedding models was to find new `causedBy` relations, multiple splits containing this relation are necessary. However, only 85 `causedBy` relations are present in the first data set. While this is a good start, splitting these events into training, test, and validation would have resulted in very small subsets. Possibly, some events would only occur in one of the three subsets, which would result in the model not knowing what to do with unknown entities. Additionally, splitting the events randomly instead of timewise would distort the idea of finding a model which can be used for new events happening after the training data.

As the target relation, which we aim to predict, contains only a very small number of objects, splitting this relation into three subsets does not seem reasonable. The problem is that the data set is already very small for a machine learning setup and there are only very few examples available for a model to learn the correct embedding. However, there is one key advantage of the ExpCPS data, which can be used to tackle these issues. As the data set is not a static data set, which was collected once, but is based on simulation data, running the simulation multiple times is very cheap and fast compared to collecting real data. Therefore, the simulation could either be run multiple times, or the simulation could be run for a longer time period, creating more data and more events. However, just running the same village setup for a longer period of time would probably not create a lot of new data. More likely, it would repeat the same daily simulation multiple times with only minor differences between the days by introducing some noise. While this approach may still provide some valuable input in the future, the huge amount of data needed to achieve improvements in the embedding model with this approach was not feasible in this thesis.

As an alternative approach, I propose to create two independent energy villages in the BIFROST simulation framework, which contain similar buildings and devices, but are still distinct from each other. These two villages will be called VillageA and VillageB from now on. The main benefit of this approach is that the trained models can be tested on a new village. The goal would be to see whether the models overfit on village-specific characteristics, or whether they managed to learn the semantics of relations between devices and derive causalities based on the connections between devices in the system.

As KGEs cannot work with unseen entities, such as buildings, which are only present in one of the two villages, the proposed idea is to train the embedding model on data from both villages, while withholding any `causedBy` relations from VillageB. With this setup, an embedding model has the opportunity to learn the setup of both villages (connections between features, registered observation, etc.), but only learns the causality relations of VillageA. For evaluation of a model, we then ask the model to predict `causedBy` relations for events of VillageB to check whether the events and their connections are embedded correctly.

In table 3.1, the proposed split of data sets is shown. For the training process, all the information available from Village A is included, as well as the topology, general causalities, observations and events from Village B. The event explanations in Village B are not used for training at all. The training data is then randomly split into a training and test set in an 80/20 ratio. This approach may seem to include a lot of information in the training data. However, looking at the data in detail, only information which would also be available in a real-time setting is used in the training data. The topology contains information about the location of buildings and their connections. This information is known for any village, which is modelled by a Knowledge Graph. The inferred data contains generally known potential causalities, which are not time-dependent and can be derived even without knowing anything about actual observations in a village. The data in the time and events data set are trickier to argue. In a live-setting, only observations and events from the past would be available, compared to all the events and observations of one day for this setup. However, live-explanations would require to re-train the models for each timestamp (currently 1h). Training any of the chosen models takes more than one hour, so this approach would not be possible in a live-setting for now. However, the data could be trained at the end of each day, which would result in an explanation being available with a delay of one day. In this case, the data in the current training split would be fully available in a live-setting as well.

While this approach was not found as such in the literature, it leverages ideas from several concepts. Semi-supervised learning was introduced for problems with too few labelled data and aimed to leverage unlabeled data to improve prediction performance. As this approach assumes that the distributions in labelled and unlabelled data are the same, semi-supervised models can only be applied for very similar problems. Transfer learning was introduced to address this issue, aiming to use previous knowledge for any task which is vaguely related [PY10].

In the use case of ExpCPS, the issue is not the difference in domains, but the difference in village setups. The proposed approach also does not really train any model on a single village as both village setups are used in one training phase. The idea of simulating two independent villages may have more in common with data augmentation tasks as the setup is creating new events and causalities, which are different to those in the training data.

Finally, applying a transfer learning approach for Knowledge Graph embeddings could be investigated in further research. This would reduce the computational effort needed to make predictions on a new village.

3.3 Evaluation

The performance of the chosen KGE models using the training architecture described above needs to be measured. There are three stages of performance evaluation in this experiment. In the first stage, the performance of the models on the test split of the training data will be evaluated. This evaluation will give a very brief overview on performance differences of the four trained models. In the second stage, the models will be evaluated on four target categories of Village B:

- **inferred** the model performance when trying to predict potential causalities between sensors
- **explanation** model performance when predicting causal relationships between events (this is the most crucial category for the experiment, but also the category containing the least data. Hence, the performance will probably be low). Also, this is the only data, which was not in the training and test data during the training process of the models.
- **events** based on observations and locations, the identification of events is predicted
- **all** this category contains the full data set, measuring the overall performance of the model on all triples in the Village B Knowledge Graph

In the final stage of evaluation, the ability of any trained embedding model to predict causality links between events will be evaluated. In this stage, each model will be analysed in detail, checking their strengths and weaknesses on different types of events and for various timestamps. Additionally, their performance on previously unknown causalities will be evaluated. To identify previously unknown facts, a manual evaluation of the top 5 predictions of each model was conducted by knowledge engineers. For each prediction, the likelihood of it being a true fact, which was missed by the original KG, is defined.

3.3.1 Evaluation Metrics

Evaluation Metrics of any model for Knowledge Graphs need to evaluate performance by using the Open World Assumption (OWA) as Knowledge Graphs are potentially highly incomplete. OWA means that any facts not existing in the original Knowledge Graph are not automatically negative facts. They might be negative facts, or positive facts missing from the original KG. This is why novel link prediction can be performed on a Knowledge Graph and it is even one of the major tasks for which embedding methods aim to offer a solution.

As there are no explicit negative facts in a Knowledge Graph, traditional performance metrics in Machine Learning, such as Precision and Recall, cannot be applied. However, as already used in NLP algorithms, rank-based performance metrics are suitable for performance evaluation of KGEs. Rank-based metrics work by considering the rank of true facts assigned by a model. For a set of test triples, either the head, relation or tail of a triple is removed. Then, the model tries to predict which entity or relation may be missing from a triple. As tail prediction is the focus of this experiment, all performance metrics will be tested on the prediction of the tail of a triple. For the missing tail, all potential entities in the graph will be assigned a score. A high score is assigned to tails, which the model thinks is likely true, while a low score is assigned to unlikely tails. Based on this score, the potential tails can be ranked from likeliest to unlikeliest. Rank-based metrics are then considering the ranking of a set of facts and measure the position of known true facts in the ranking. The two rank-based metrics used for evaluation in this thesis are *hits@k* and *arithmetic mean rank*, as they are most-used in other papers on the performance of KGEs as well.

hits@k One of the simplest rank-based metrics is *hits*@*k*, where *k* is usually a number between 1 and 10. For each triple in the test set, the tail is removed to calculate the score of each possible entity. Then, all the potential candidates are ranked by their score, and the top *k* candidates are checked whether they form a valid/true fact. The fraction of true entities in the first *k* entries of the ranking list is the value of the *hits*@*k* metric. Therefore, the *hits*@*k* metric is always a value between 0 and 1, where a higher value represents better model performance. The intuition behind the *hits@k* metric is coming from the recommender-system domain, as only very few recommendations will be considered by a user when looking for a relevant recommendation.

arithmetic mean rank The second metric used to measure the performance of the four KGE models is mean rank. It computes the arithmetic mean of all ranks of positive entities. Compared to *hits*@*k*, mean rank takes all performance changes into account and does not contain a cutoff point. Therefore, the mean rank is representing the average performance of a model. The arithmetic mean rank is always a value in the interval $[1, \infty]$, where a smaller mean rank indicates better overall performance of a model.

Based on these two metrics, the model performances will be evaluated. Both metrics should be analysed in detail, as they show different angles of the suitability and performance of a KGE model.

CHAPTER 4

Experiment Design

At this point, the reader should know all the theoretical concepts and tools involved in running the experiments of the thesis. This chapter will focus on the specific settings, data and evaluation methods used to conduct the experiments. In the first section, the data (Knowledge Graph) will be explained in detail. While previous sections have already explained the general setup of the ExpCPS KG and the idea of using two villages for conducting the experiments, this section will show the types of features and events present in each village. A detailed analysis of the ground truth present in the simulated data will be conducted to get a feeling of the possible causal relations between two events and to be able to interpret the model results in the next chapter. Following the explanation of the data, section 4.2 will focus on the experimental setup, including the tools and libraries used in the experiments as well as steps involved to reach the final results. In the final section of this chapter, the evaluation to measure model performance will be explained. While the formula and meaning of the evaluation metrics have already been explained in the previous chapter, the evaluation setup in this specific experiment will be covered and explained in-depth.

4.1 Data Structure

The data used in the following experiment is created by running two simulation runs in BIFROST for two distinct villages, called Village A and Village B. By running the BIFROST simulation and sending the resulting data through the data pipeline as it was explained in section 1.3.1, a set of knowledge graphs was created for each of the village setups. The simulation in Village A was run for 23 hours (01:00 until 23:00), while the simulation in Village B was run for 24 hours (01:00 until 00:00 the next day). Each village has a slightly different setup, which can be analysed in detail in table 4.1. In this table, the features present in each energy village are listed as well as the prevalence of each feature type in village A and B. Each feature has different properties, which can be

Feature	A	B	Feature	A	B
airship	1	1	grid-node	95	110
battery-storage-small	1	1	household-battery	1	1
billboard	1	1	lv-transformer	1	1
cable-underground-sd	118	140	powergrid-connector	22	30
charging-pole	1	1	powerswitch	1	1
commercial-battery-small	1	1	residential-multi-medium	1	1
commercial-factory	4	4	residential-single	7	9
commercial-single	5	9	router	1	1
data-connection-wired	51	51	solar-panel	4	4
data-node	41	41	trafo-building	1	1
datagrid-connector	17	17	vertical-farm-big	2	2
ev-station	2	2	grocery-store	-	2
			Total	**379**	**432**

Table 4.1: features of interest present in Village A and Village B

measured. As an example there are sensors connected to an underground cable, which measure cable length, current and loading of the cable. An airship, which is responsible for creating weather data features sensors concerning a climate model, precipitation and temperature measurements. Each village setup contains exactly one airship for measuring weather data, one billboard, which is responsible for administering flexibility requests, and one transformer, monitoring loadings in the village overall.

By adding two additional residential houses and four commercial buildings in Village B compared to Village A, additional underground cables, grid nodes and power grid-connectors are included in the simulated village. There is an overview of the two villages shown in table 4.2. For each sub graph of the Knowledge Graph setup, a general overview of the most important entities of each graph is given. During the simulation run of 23 and 24 hours respectively, 25 067 and 30 075 observations were measured in Village A and Village B respectively. Based on these observations, 107 and 191 events were created by the Event Annotator. There are ten different types of events occuring in the villages, which can be inspected in table 4.3. Two types of events did only occur in Village B. In Village A, 12 events can be explained by at least one cause event through the `causedBy` relation. A set of cause events, which explain an event is called an explanation. In Village B, 17 events contain an explanation. In Village A, each event in the explanation graph contains 7.1 cause events on average, while there are 7.8 cause events on average in Village B. In total, there are 85 cause events in village A and 133 of them in village B. Conclusively, village B is slightly bigger than village A, as there are more buildings in the setup and more events were registered during the simulation period. Interestingly, there are two types of events in village B, which were not registered at any point in time in village A. These events will be especially interesting to analyse as there is no direct information on event causality

subset	Village A	Village B
topology	1,004 sensors for 379 features of interest, 23 types of features	1,203 sensors for 432 features of Interest, 24 types of features
timestamps	25,067 observations for 23 hours, on 1973-12-25, 01:00:40 – 23:00:40	30,075 observations for 23 hours, on 1973-12-25, 01:00:40 – 00:00:40
inferred	57 potential causes between sensors	77 potential causes between sensors
events	107 events of 8 unique types	191 events of 10 unique types
explanations	85 explanations for 12 different events	133 explanations for 17 different events

Table 4.2: general facts about the simulated village setup

connections of these event types available during model training.

4.1.1 Event Analysis

The meanings and connections of events are a core subject of the thesis. Table 4.3 contains a detailed overview of the ten event types occurring in the simulations, their defining properties and the number of occurrences in each village. The events can be grouped into four categories: Charging Events, Flexibility Events, Demand Events and Loading Events. While Charging events are events depending on the charging state of a battery, flexibility events are events related to a flexibility request, which is posed from outside the village. Demand and Loading events are defined depending on the relation between energy production and demand in the village. The exact definition of each type of event is given in the following description.

Charging Event Based on the State-of-Charge of a commercial battery, this event occurs when the battery level is increasing, indicating that the battery is currently charging. This event can only occur at a commercial battery.

Discharging Event Based on the State-of-Charge of a commercial battery, this event occurs when the battery level is decreasing, indicating that a battery is currently discharging. This event can only occur at a commercial battery.

FlexContributedEvent This event indicates a flexibility request from the billboard to a residential or commercial building. It shows a request to provide additional electricity, or to consume more energy within the village to balance the fluctuations in a connected network. This request can happen in both directions, either reducing or increasing power consumption.

FlexRequestApprovedEvent and FlexRequestRejectedEvent Following a Flex-ContributedEvent, the request can either be approved or rejected. Based on the

Event	**Defining Property**	**A**	**B**
ChargingEvent	Battery State-of-Charge	10	7
DischargingEvent	Battery State-of-Charge	12	17
FlexContributedEvent	Power Distribution	51	87
FlexRequestApprovedEvent	Available Flexibility	7	6
FlexRequestRejectedEvent	Available Flexibility	0	2
FlexUnavailableState	Available Flexibility	0	4
LoweringDemandEvent	State of Active Power	5	9
PeakingDemandEvent	State of Active Power	5	9
OverloadingEvent	Loading Rate	9	25
NormalizingEvent	Loading Rate	8	25

Table 4.3: A list of event types, their definitions and their number of occurences in Village A and B

available flexibility in the requested building, the request will be evaluated. If enough flexibilities are available, the request can be approved. Otherwise, it will be rejected.

FlexUnavailableState If the flexibility state of a building is not available or cannot be determined, the response to a FlexContributedEvent is FlexUnavailableState. This means that the request can neither be approved nor rejected as the required data is not available.

LoweringDemandEvent based on the active power at a certain location, a LoweringDemandEvent is registered if the active power is decreasing.

PeakingDemandEvent based on active power, exceptionally high power demand is registered as a PeakingDemandEvent.

OverloadingEvent The loading rate of a feature is measured as a percentage and indicates the load factor. A loading rate above 60% will trigger an OverloadingEvent.

NormalizingEvent After an OverloadingEvent, the reduction of the loading rate to below 60% will be registered as a NormalizingEvent. This event shows that the overloading event is not relevant anymore.

4.1.2 Causality Data Analysis

Following the definition of events, the causal relations between events in the data are the next important step to analyse. While the causal connection between two events is very much dependent on the situation, analysing recurring patterns can help in understanding the possible relations better. The main idea behind this analysis is

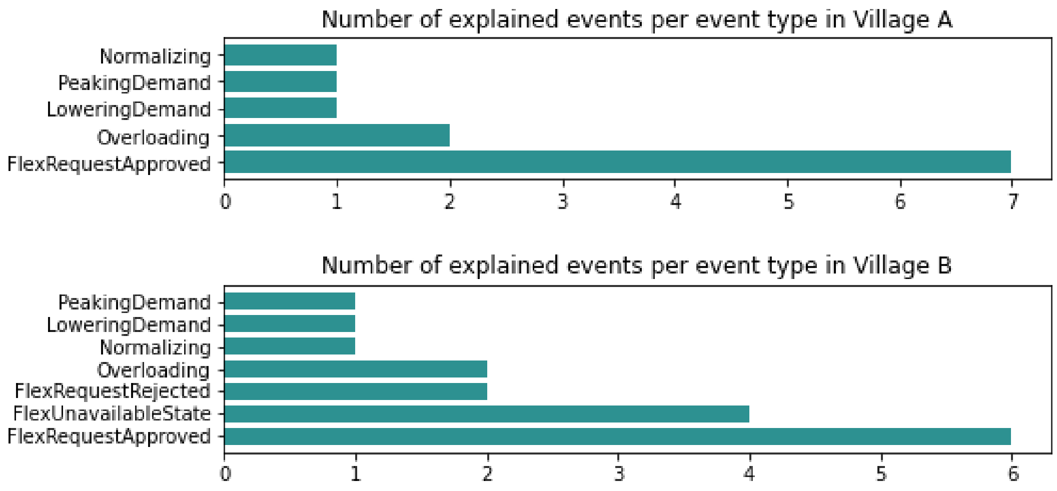


Figure 4.1: Number of events containing an explanation for each event type per village

to find out whether there is a certain type of event, which is the main cause in any explanation of another type of event.

In the validation data, there is a set of 17 events for which explanations exist, containing at least one event which is known to be a true cause of the event. There are 7 types of events, for which an explanation exists. In figure 4.1, the distribution of event types for events which hold explanations is shown. The event type, which is by far the most represented in the explained event data is FlexRequestApproved events. There is a total of 6 explanations for FlexRequestApproved events. While FlexUnavailableState has four explained events, all other event types have a maximum of two event explanations.

In the training data (Village A), there are a lot less explanations present in general. First, there is no explanation for any FlexRequestRejected or FlexUnavailableState Event in village A. This means that there is no explicit training on the explanation of these two event explanations at all. Additionally, there is a big difference between the number of explained events per event type. The general tendencies are very similar in both villages. In Village A, there is only one explained event for the majority of event types. Looking at the results, it will be interesting if the number of explanations in the training data will have an effect on model performance on different event types.

In table 4.5, the distribution of causal events for any of the 17 events containing an explanation is shown. At the top of the table, a list of the seven event types is shown. These are all types of events, for which an explanation exists in any of the two villages. On the left side, all seven event types are listed, which are causes in any explanation. Each event explanation contains multiple cause events, which are of different event types. In this table, the average distribution of event types for each event explanation is shown separately for village A and village B.

Looking at the table more closely, a few interesting characteristics can be observed. Generally, one would expect to find the same cause events for an event type in each village, as seen for PeakingDemand. Here, even though not having the same distribution of event types, the event explanations contain the same three event types (FlexContributed, LoweringDemand, PeakingDemand). However, for some events, there is no similarity between the two villages at all, as can be seen for the Normalizing event. In village A, explanations contain FlexRequestApproved and PeakingDemand events, while in village B, only FlexRequestRejected and LoweringDemand events are present in the event explanation. Moreover, two event types did not have an explanation in village A, but they do in village B. FlexRequestRejected as well as FlexUnavailableState are the two event types, which did not occur in village A at all (as can be seen in table 4.3 and figure 4.1 as well).

A visual representation of causality flows is shown in figure 4.2. Based on the data in table 4.5, an arrow shows any occurrence of causal explanation between two types of events in either village A or village B. This flowchart shows that there are basically three hierarchy levels in the causality relations between events. On level one (left side of the graph), the event types are never explained, but they provide explanations for other events. The three events on this level are Charging, Discharging and FlexContributed. On level two, events can be explained and, further, provide an explanation for other events, creating an explanation path. In the final level, there are events which can be explained, but do not provide further explanations of other events.

This graph will be useful for analysing the prediction results of KGE models, as it shows whether or not a causal relation exists in the data. For example, if a model predicts that an overloading event causes a discharging event, this graph shows that this causal relation is very unlikely according to the ground truth data.

Effect Event	FlexRequest Approved		FlexRequest Rejected		Flex Unavailable State		Lowering Demand		Normalizing		Overloading		Peaking Demand	
Cause Event	A	B	A	B	A	B	A	B	A	B	A	B	A	B
Charging	7%	14%	-	-	-	-	-	-	-	-	-	-	-	-
Discharging	13%	-	-	50%	-	13%	-	-	-	-	-	-	-	-
FlexContributed	80%	86%	-	50%	-	87%	80%	78%	-	-	-	-	21%	58%
FlexRequestApproved	-	-	-	-	-	-	-	-	50%	-	50%	33%	-	-
FlexRequestRejected	-	-	-	-	-	-	-	-	-	50%	-	33%	-	-
LoweringDemand	-	-	-	-	-	-	-	22%	-	50%	-	33%	26%	17%
PeakingDemand	-	-	-	-	-	-	20%	-	50%	-	50%	-	53%	25%

Table 4.5: Distribution of event types causing effect events in the explanation data

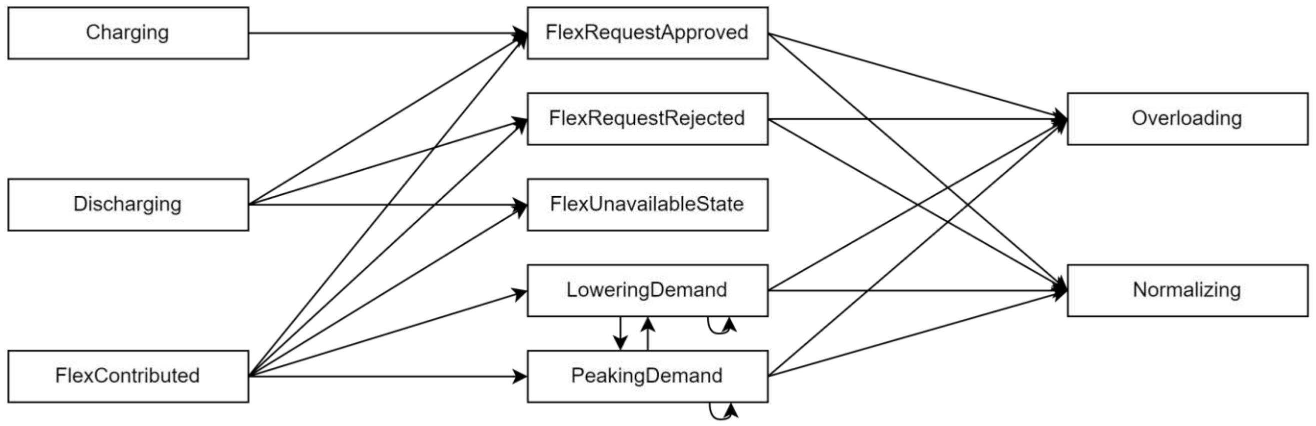


Figure 4.2: Flow chart of causality paths in explanation data of village A and B

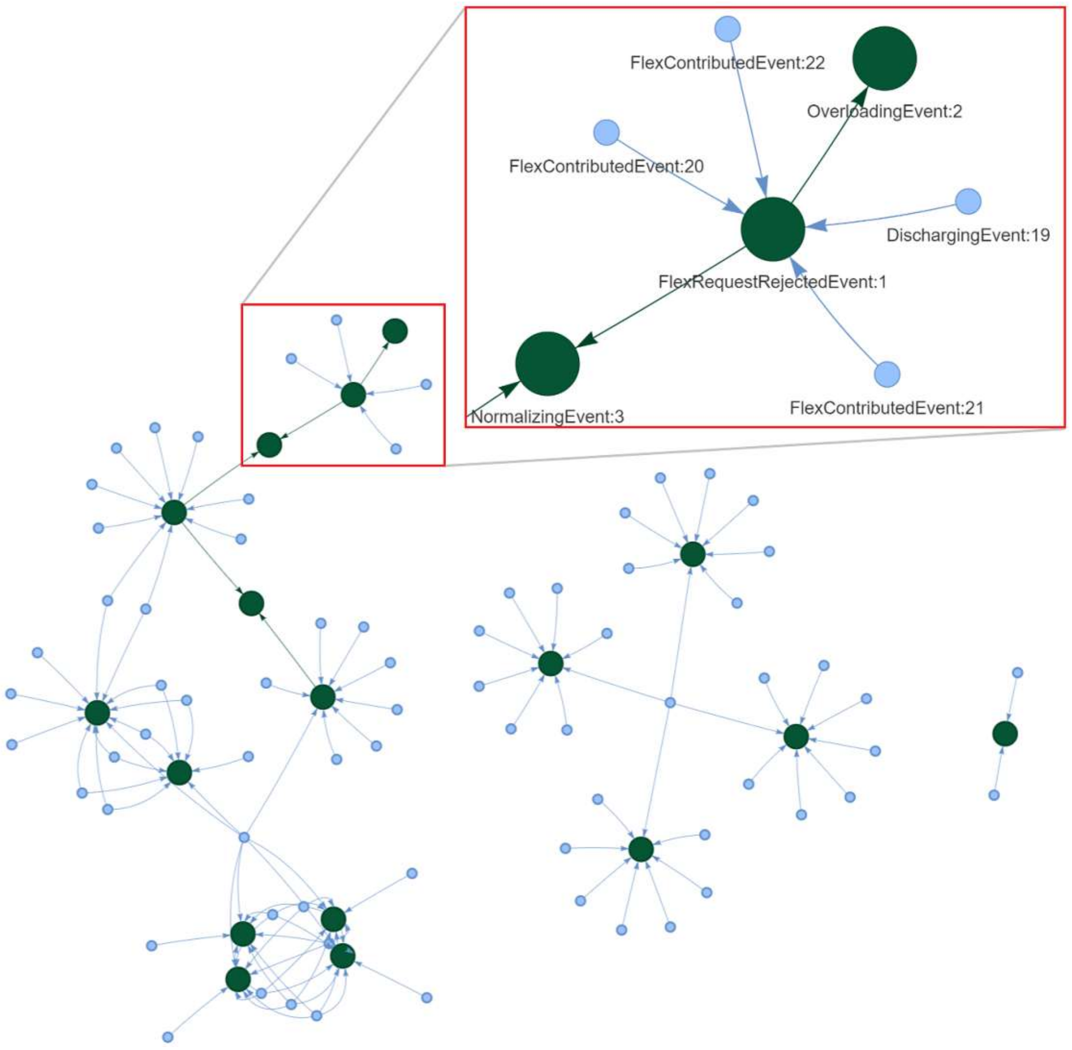


Figure 4.3: Causality network in village B

In figure 4.3, the causality network in village B is shown. Each green node represents an event for which an explanation exists, while the blue dots represent the cause events. Directed edges show the causality relations between events (EventA –causes–> EventB).

This network graph shows that there are three connected graphs in the data, which form a set of related causalities each. Some explanations are very independent from others, while some of them are highly interconnected. Additionally, some of the effect events are causes of other effect events, creating a multi-step causality path. In the ground truth data, only direct causes are listed as true causes.

In the top section of figure 4.3, a more detailed view of the section marked in red is shown. There, an example of a causality path is shown. While according to the ground truth only one event - `FlexRequestRejectedEvent:1` - is the cause of `OverloadingEvent:2`, there are four events, which are causing `FlexRequestRejectedEvent:1`. Therefore, they are indirectly causing `OverloadingEvent:2` as well. This fact expands the number of true causes from one direct cause to five causes.

4.1.3 Preprocessing steps

As a last step before implementing the KGE models and feeding the ExpCPS Knowledge Graph into the embedding models for training, some minor preprocessing steps were conducted to fit the data to the models. Some observation-timestamps did not comply with the time frame of the simulation. These timestamps marked a time before the start of the simulation. This is not a big error as these observations are time-invariant, such as cable length. Nonetheless, for consistency, these timestamps were reset to the time of the start of the simulation.

For traditional KGE methods, a set of triples is fed to a model for training the embedding. However, as not only traditional Knowledge Graph Embedding methods are used, but also a Temporal KGE, the triples needed to be extended to quadruples for the temporal embeddings. [1]

After inital preprocessing steps, the data was split into training and validation set. As already described in section 3.2, the training set consists of all the data available from Village A and all the data available from Village B, except the explanation graph, which contains the `causedBy` relations. The explanation data from Village B serves as the validation data set. For the training process, only training data was used, which was split into training and test data randomly with a split of 80% training data and 20% test data.

4.2 Experimental Setup

Following the explanation of the data structure of the ExpCPS Knowledge Graph that is used in the experiment, this section will focus on the experimental setup. All the Knowledge Graph Embedding Methods were implemented in python using pytorch, which is a highly flexible deep learning framework for python. The pykeen package [ABH+21] is a framework built for reproducible knowledge graph embeddings. The pykeen implementations of TransE, TransH and ComplEx were used in this thesis. As TTransE is using quadruples instead of triples, another framework is needed to run the TTransE model on the data. For this model, the implementation of [HZMT21] was used. The initial Knowledge Graph was analysed in a local instance of GraphDB. For preprocessing and result analysis, additional python packages, such as pandas, matplotlib and seaborn were used. The whole experiment setup is available on GitHub, including the preprocessing steps, model training and evaluation. A full list of required

[1]Detailed preprocessing steps are available at `https://github.com/Kat-rin-sc/ExpCPSKGE/src/code/01_Preprocessing.ipynb`

PARAMETER	TYPE	POTENTIAL VALUES
model embedding dimensions	INT	32, 64, 128
model scoring function norm	INT_+	1,2
loss function	STR	margin ranking
loss margin	$FLOAT_+$	1.0,2.0
optimizer	STR	Adam, SGD
optimizer learning rate	$FLOAT_{+,<1}$	0.001, 0.01, 0.1
negative sampler	STR	basic, bernoulli
negative samples per positive	INT_+	32
max number of epochs	INT_+	100
early stopper	$BOOL$	True, False

Table 4.6: Parameter settings for hyper-parameter optimization(HPO) of KGE models

packages and their versions to run the experiments is provided as well as a detailed explanation on how to run the full experiment.[2]

Each of the four KGE models was trained using the default values of the corresponding implementation setup first. Then, each model was trained using hyperparameter optimization(HPO) to find the best-performing hyper-parameter settings for each model. In table 4.6, the tested parameters as well as potential values for each variable are shown without making a claim to be exhaustive. While the models may perform differently depending on the model-specific definition of the score function, the overall framework was tested on the same parameters, such as lookup layer, loss functions, generation of negative triples and optimizers. This way, the setup allows for enough flexibility for a model to find the ideal settings for good performance, while ensuring comparability and providing the same starting conditions for each model before training.

4.2.1 Workflow

In this section, the workflow pipeline of the experiment will be explained in detail. In figure 4.4, a schematic representation of the workflow is presented. The whole workflow can be split into four main tasks - preprocessing, model training, model evaluation and link prediction.

1. **Preprocessing** The initial data provided for the training process is the ExpCPS Knowledge Graph. Both villages are stored in a local GraphDB instance, where they can be accessed through a user interface or python command. As described in section 4.1.3, the Knowledge Graphs were preprocessed and stored in csv files as triples and quadruples for further use.

[2]Full experiment is available at `https://github.com/Kat-rin-sc/ExpCPSKGE/`

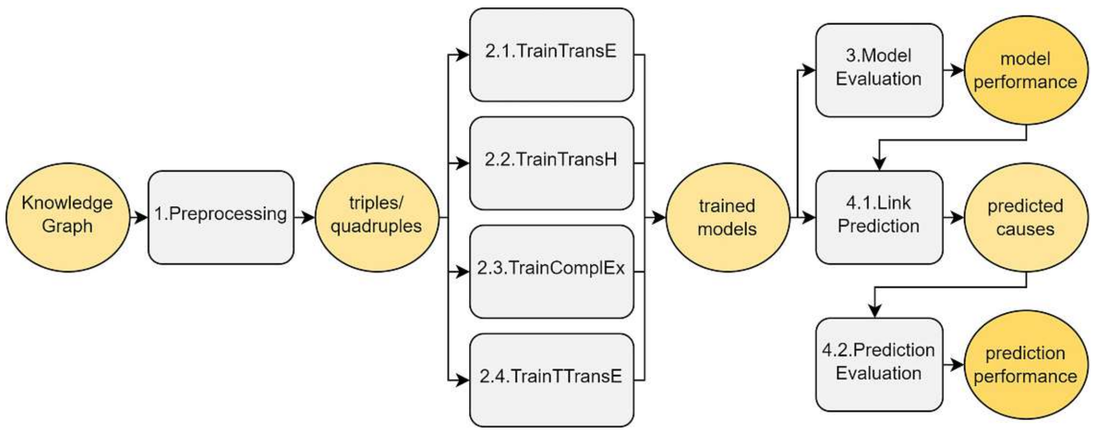


Figure 4.4: Workflow representation of the Experimental Setup

2. **Model Training** In the next step, all four Knowledge Graph Embedding models chosen in section 3.1 were trained on the training and test data defined in section 4.1.3. Each model was trained with the default values of the implementation setup as well as using a hyper-parameter optimization algorithm for finding the optimal model embedding dimensions, loss margin and optimizer. Training the default models and conducting a hyperparameter optimization takes approximately half a day per model using CUDA. As a result, eight trained models are stored for further evaluation (one default training run and one optimized run for each of the four chosen models).

3. **Model Evaluation** Each model from the model-training section is then further evaluated on multiple sections of the Village B data. Model performance is tested on event data (checking whether the model knows which events are found by the Knowledge Graph), inferred data (analysing the embedding quality of `potentialCause` relations) and explanation data (checking the performance on `causedBy` relations). The `causedBy` relations are the relations which were not present in the training data at all. For reference, the mean performance for any relations in the Village B KG was added. Using this setup of different evaluation metrics for different sections of the Knowledge Graph aims to analyse the model performance not only on general performance, or performance on the target relations, but to find out where a model is performing particularly badly or exceptionally well. This might help to find weaknesses of specific models as well as finding out what characteristics are underrepresented in the embeddings. The detailed model evaluation will be presented in chapter 5.

4. **Link Prediction and Evaluation** Following general model evaluation, each of the models is used to predict potential causality links in the ExpCPS Knowledge Graph. For each event, which contains a known effect event, the trained KGE models are used to predict a potential tail for the triple `(effectEvent, causedBy, ?)`. Each model will score all existing entities in the Knowledge

Graph based on their scoring function and the trained embedding space. Based on this score, the entities are ranked according to their likelihood of being a true cause. The ranked model predictions are then evaluated manually in a human annotation process. In the next section a more detailed explanation will be provided on how the evaluation of the predictions will be conducted.

4.3 Link Prediction Evaluation Setup

In a link prediction task, the goal is to find a set of potential links, which could be true links, but which may not be part of the original Knowledge Graph. In the ExpCPS use case, where the goal is to find cause events for a set of effect events, the prediction task is focussed on tail prediction. In a tail prediction, the tail of a triple is omitted and entities are scored on their potential to be a fitting tail for this triple. For example, if potential causes of `effectEventA` should be predicted, a fitting tail is searched for the triple `(effectEventA, causedBy, ?)`. After feeding this triple to a prediction model, it will provide a list of potential tails. Each potential tail is an entity of the Knowledge Graph, which has a score based on the model's scoring function as well as a ranking, which shows the rank of the entity score compared to other entities in the KG. For evaluation, the true label of each entity is required. For evaluating the experiment, the top five ranked entities are chosen to be evaluated, based on the ground truth data and manual evaluation. Therefore, a *hits@5* evaluation metric can be calculated and analysed in the results.

As all entities in the KG are compared in the link prediction task, some preprocessing steps are conducted before evaluating the top five results per model. A few rules are applied to the set of predictions to filter out irrelevant predictions from the set of entities before looking at the results in more detail:

Defined rules for filtering prediction results:

1. **Is the predicted tail of type event?**
   - No: drop entity

2. **Is the predicted event happening in Village B?**
   - No: drop entity

3. **Did the predicted event happen after the effect event?**
   - Yes: drop entity

Based on these rules, a lot of predictions can already be filtered out as they are irrelevant entities and are therefore identified as false positives. As a result, only a set of potentially relevant events in Village B is left to analyse. Based on the filtered set of predictions, the top five predictions of each model are analysed in more detail. For any predictions which are still potential candidates, the sensor connected to the event, the physical location of the feature where the event occured, the reference observation

Value	Definition	Example
4	definitely true	ground truth, level 2 ground truth
3	probably true	same sensor as ground truth, time intervals reasonable
2	probably false	none of the above, but possibly true, potential causality between sensors exists
1	definitely false	same event as effect-event or no connection, no potential causality

Table 4.7: Definition of the Evaluation Scale

as well as the timestamp of the event are analysed in more detail. Additionally, the shortest physical distance to the effect event as well as the temporal distance are calculated for each predicted cause event.

This process results in a set of 17 prediction files, one set of entities for each event for which an explanation exists. Each file contains a set of predicted causes as well as the ground truth causes for the event.

4.3.1 Human Annotation

Following the preprocessing of the evaluation (step 4.2. as shown in figure 4.4), these prediction files are annotated manually in an annotation workshop by Knowledge Engineers. The goal of the evaluation is to determine whether the predicted causes are true causes or false positives. While some labels are very clear to find (when a ground truth event is predicted, it is obviously true), for some labels the decision is not so clear. To tackle this problem, a 4-point Likert scale is used to label the likelihood of a prediction to be true. Thereby, a distinction can be drawn between events which are definitely true and those, which are probably true. A verbal definition of the four points on the evaluation scale as well as examples, which would be labelled by the respective value, is depicted in table 4.7.

As can be seen in table 4.7, the four categories are chosen based on general rules. An evaluation workshop with Knowledge Engineers was held to determine the labels of all 416 causality predictions. A set of rules was developed to determine the category of each prediction on the evaluation scale. For this evaluation, a very conservative approach was used to allow for potential causalities, which are not present in the ground truth. In further research, the effect of more liberal evaluations could be analysed. In the following definitions, any entity connected to the effect event, such as the sensor of the event or the event itself, will be marked by the appendix X_E, while the appendix X_C will be used for entities connected to the predicted cause event. For each point on the scale, a set of rules was defined to make sure each event is allocated to one of the four potential likelihoods. Only one of the rules has to be fulfilled for an event to be allocated to the respective score.

[1] definitely false

- $Sensor_C$ does not have a potential causality connection to $Sensor_E$
- $Event_C$ is equal to $Event_E$

[2] probably false

- potential causality between $Sensor_C$ and $Sensor_E$ exists, but $Sensor_C$ does not exist in the ground truth
- $Sensor_C$ does exist in the ground truth, but the time difference between $Event_C$ and $Event_E$ is $> 1h$

[3] probably true

- $Event_C$ does not exist in the ground truth, but $Sensor_C$ exists in the ground truth and time difference between $Event_C$ and $Event_E$ is $<= 1h$

[4] definitely true

- $Event_C$ is in ground truth
- $Event_C$ is a level 2 - ground truth ($Event_C$ is a true cause of an event, which caused $Event_E$)

Based on these allocation-rules, all causality predictions should be labelled successfully and the evaluation of the prediction results can be finalized. Based on this evaluation, evaluation metrics are calculated using these labels. As only the top five predictions of each model are analysed, the evaluation is limited to the *hits@5* metric. In the evaluation, there will be a distinction between *defHits@5*, which only includes level-4 true values, and *probHits@5*, which includes level-4 and level-3 true values from the Evaluation Scale as true positives.

Additionally, there will be a set of evaluation metrics called $hits@5_{adj}$. This metric is adjusting the *defHits@5* metric based on the number of ground truth data available. As there are some explanations, which contain less than five ground truth-causalities, false positives will be ignored if all of the potential true causes have already been predicted by a model.

CHAPTER 5

Results

In this chapter, the results of the experiment as explained in chapter 4 will be shown. The experiment results are split into three main sections. First, the results of the model training phase will be discussed. For each of the trained models, the optimal hyperparameters are discussed and development of the loss during model training will be shown. In the second section, model performance will be compared based on general representation capabilities of each model. For this section, the performance of the models will be compared on different metrics and test sets to find the best-performing model. In the final section, the models will be used to predict causality links between events in the Knowledge Graph. Based on the evaluation setup in section 4.3, the causality prediction of each model is evaluated on a four-point Likert scale. Based on this evaluation, the performance of each of the four models on a causality link prediction task is measured.

5.1 Model Training

Based on literature research on potential KGE models to represent the ExpCPS Knowledge Graph in a low-dimensional embedding space, four models have been chosen to be trained and evaluated (TransE, TransH, ComplEx, TTransE). Each model was first trained using the default parameters of the KGE framework and then a hyperparameter optimization was conducted. In table 5.1, some of the optimized parameters are shown for each model. Interestingly, most parameters ended up being very similar for the different models. Only the early stopping setting led to very different numbers of training epochs for the various models.

In figure 5.1, the development of the training loss per epoch is shown for each model. In the top row, the default training runs are shown, while the optimized runs are shown in the lower section of the figure. Potentially, each model was trained up to 100 epochs, but due to early stopping, many training runs stopped earlier as there was no

Parameter	TransE	TransH	ComplEx	TTransE
dimensions	64	64	64	32
loss function	margin ranking	margin ranking	softplus	cross entropy loss
optimizer	Adam	Adam	Adam	Adam
learning rate	0.001	0.001	0.001	0.008011243
epochs	100	30	80	45
early stopping	True	True	True	True

Table 5.1: model settings after Hyperparameter Optimization

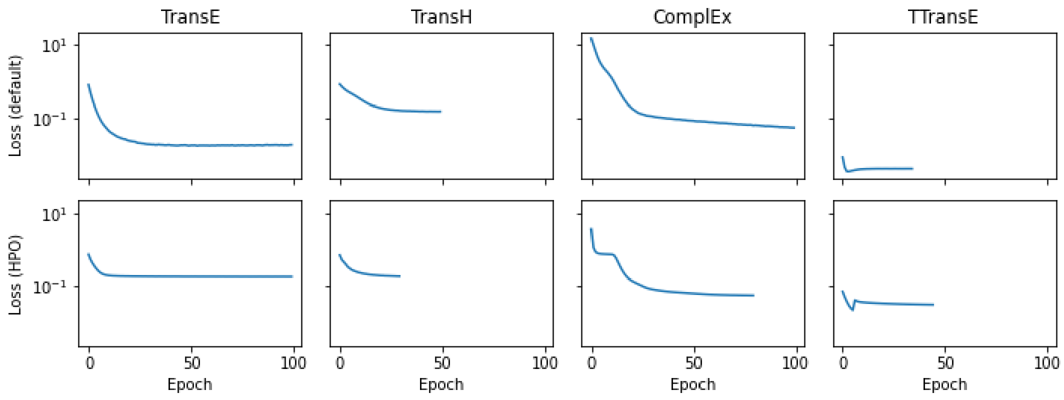


Figure 5.1: Training Loss per Epoch for all models, shown on a logarithmic scale

	default	HPO model	
	training	model search	training
TransE	0:41:42	14:37:25	1:45:30
TransH	0:26:47	12:46:47	0:33:31
ComplEx	2:28:30	14:54:24	1:01:37
TTransE	7:49:54	11D 21:30:15	17:29:37

Table 5.2: training times of models

significant improvement in training the models for several epochs. Only TransE never stopped training early, even though the development of training loss suggests a very low improvement for any epoch after epoch 50. Figure 5.1 shows the training loss on a logarithmic scale, which may let small improvements seem to disappear in the graph.

Training the embedding models took between 30 minutes and eight hours, while optimizing the hyperparameters took between 12 hours and 12 days. The detailed comparison of training times can be examined in detail in table 5.2. Training of the TTransE model took by far the longest to complete. The reason therefore might be the additional computational effort needed due to the temporal parameter to be embedded

		default			optimized		
data	**model**	**hits@10**	**hits@3**	**MR**	**hits@10**	**hits@3**	**MR**
inferred	TransE	0.740	0.545	37	0.539	0.429	53
	TransH	0.591	0.513	240	0.610	0.571	114
	ComplEx	0.227	0.123	2639	0.727	0.526	185
	TTransE	0.965*	0.825*	3*	0.870*	0.760*	10*
explanation	TransE	0.036	0.013*	233*	0.214*	0.040*	61*
	TransH	0.112*	0	256	0.125	0.009	160
	ComplEx	0.000	0.000	49069	0.004	0.000	34315
	TTransE	0.031	0.008	14229	0.075	0.000	487
events	TransE	0.887*	0.808*	79*	0.658	0.515	1076
	TransH	0.742	0.639	296	0.673	0.547	591
	ComplEx	0.530	0.445	5502	0.773	0.628	918
	TTransE	0.880	0.801	559	0.809*	0.694*	21*
all	TransE	0.872*	0.832*	1125*	0.728	0.655	1080*
	TransH	0.781	0.734	1217	0.748	0.682	1237
	ComplEx	0.778	0.701	1813	0.819*	0.777*	1322
	TTransE	0.704	0.572	1776	0.777	0.735	1503

* best result in this category

Table 5.3: Performance Metrics of all trained models, measured in four categories

in the model. Another explanation might be the different embedding framework used for TTransE. This framework seems to not have implemented some of the performance enhancements, which are used in pykeen for the other models.

5.2 General Performance Results

In a first evaluation step, the ability of a model to represent various parts of the ExpCPS Knowledge Graph is examined. Performance metrics are calculated individually for the subgraphs of inferred data, events data and explanation data. Inferred data is the subgraph containing potential causalities between sensors. Events data contains all the events, which are connected to observations from the simulation data. The explanation graph covers the set of causality relations between events. This graph is not known by the embedding models during the training process at all. Further analysis of prediction performance on explanation triples is thoroughly investigated in the next section. Moreover, the overall performance when considering the whole Knowledge graph is added to the evaluation. All of the evaluation results are based on the Village B Knowledge Graph.

Table 5.3 shows an overview of all performance results for each model on each data type,

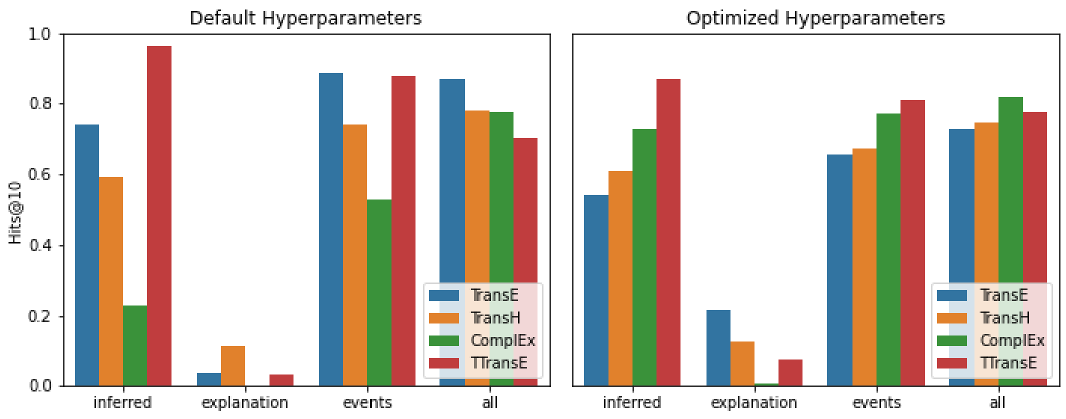


Figure 5.2: Model Performances of all trained models measured in Hits@10

comparing the default settings and the optimized settings. For each category and metric, the best score is marked by a star. The table shows that different models are better at representing different sections of the Knowledge Graph. TTransE performs best in representing inferred data, which is interesting as inferred data is time-independent, meaning that is should not benefit from temporal embeddings. Event data is best represented by TransE when using default settings, but TTransE outperforms TransE in the optimized settings. Explanation data is best represented by TransE, while overall performance for the whole Knowledge Graph is best represented by ComplEx, when considering *hits@k* metrics. Mean Rank performs best for TransE.

In figure 5.2, the performance differences between models are visualised for the *hits@10* metric. The figure shows that hyperparameter optimization does not result in better performance for models over all categories. For inferred data, model performance actually decreased. But most importantly, prediction performance improved with hyperparameter optimization for explanation triples, which are the main focus of the thesis. Unsurprisingly, performance is worst on explanation data overall as this section of the Knowledge Graph is not present in the training data and the triples are completely new to the models. It seems like TTransE is overfitting on the data categories, which are in the training set as TTransE performs extremely well on existing data, such as inferred and events data, but it is the second worst model to represent explanation data. Optimized TransE, on the other hand, is performing worst on the known data categories but performs best on explanation data. Without jumping to any conclusions, it seems like TransE does manage to understand general connections and semantics in the ExpCPS Knowledge Graph better than other models as it manages to apply its embeddings on new, unseen data with the best results.

Performance of models measured in Arithmetic Mean Rank (MR) is shown in figure 5.3 (low mean rank respresents better performance). MR shows a more general view of the performance of a model, as it considers all ranked results in the evaluation compared

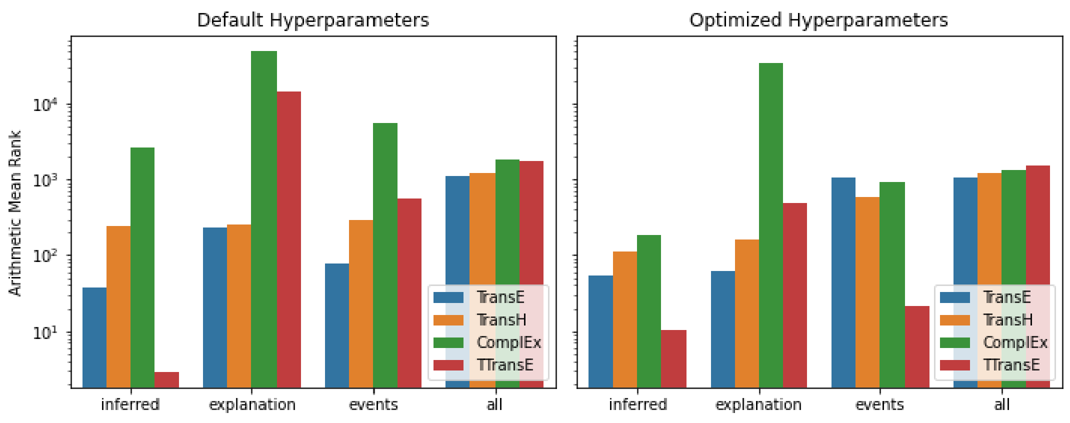


Figure 5.3: Model Performances of all trained models measured in Arithmetic Mean Rank

to *hits@k*, where only the top k results are evaluated. As the Mean Rank can be any number in the interval of $[1, \#EntitiesInKG]$, the y-axis in figure 5.3 is shown on a log-scale. Therefore, comparison of height differences between bars in the figure has to be done with care. While models seem to perform similarly in terms of Mean Rank overall, there are big differences when investigating the three categories. ComplEx performs worst on the categories inferred and explanation. Since ComplEx does not perform much worse than others in the rest of the KG, it seems like ComplEx has difficulties modelling causal relation in the data. TTransE performs exceptionally well on inferred and events data, but it is drastically outperformed by TransE and TransH on explanation data. Comparing the improvements from default hyperparameters to optimized hyperparameters, it looks like ComplEx did not benefit from the optimization at all. While mean rank was reduced for most categories when applying optimized hyperparameters to ComplEx, mean ranks of explanation data improved by 30% and remained at an unacceptably high level.

As TransE was already investigated in detail for *hits@10*, looking at MR seems to give a more complete view on the performance of the model. While TransE did not manage to rank as many true values in the top 10 predictions as the other models for most categories, it performed best when considering all rankings of ground truth values. Therefore, it also seems to be more consistent in its performance.

TransH has not been mentioned in the analysis of performance data much, as it is never the top-performing model, but it also does not perform too badly. Overall, TransH seems to be a solid, robust model, but is always slightly outperformed by another model, especially TransE.

5.3 Link Prediction Results

In this section the model performances for causality prediction will be evaluated. There are 17 events in the validation data which contain an explanation that is unknown to the trained models. Each explanation contains between one and twelve events, which are known to cause the event. These events are used as the ground truth when testing model performances. Additionally, potential cause events not present in the ground truth will be investigated on whether they are true causes, which are missing in the current Knowledge Graph. There are 7 types of events, for which an explanation exists.

The causality relation is depicted in the Knowledge Graph as a triple of the form `(EffectEventA, causedBy, CauseEventX)`. In this triple, the entity `CauseEventX` is unknown. Each of the four trained models in the experiment is tasked to find the five best matches to fit the `causedBy` relation for an effect event. In the analysis of general model performances, the models trained with optimized hyperparameter settings all achieved better results on the explanation data than the default values. Therefore, the hyperparameter-optimized models are used for causality link prediction.

For each model, the top 5 predictions are derived and evaluated as described in section 4.3. This means that predictions are filtered according to the filtering rules in section 4.3 and the top five results remaining are further evaluated. Each prediction is labelled according to its likelihood to be true on a scale between 1 and 4. As a result, each of the 17 events will have a set of top 5 predictions per model and each prediction is labelled on a four-point Likert scale. Based on these labels, there are three evaluation metrics to be calculated and analysed: *defHits@5* (number of predictions, which are labelled to be definitely true), *adjusted defHits@5* (*defHits@5*, adjusted by the number of possible truth values in the data) and *probHits@5* (number of predictions, which are labelled probably or definitely true).

The results of the evaluation are shown in table 5.4. In figure 5.4, the comparison between *definite* and *probable hits@5* is visualised overall and per event type. Overall, the TransE model performed best on link prediction for *definite* and *probable hits@5*. TransH and ComplEx benefitted the most from including probable hits in the performance metric as their result improved by 0.09 from *definite* to *probable hits@5*.

The event types FlexRequestRejected and FlexUnavailableState did not occur at all in the training data of the models. Therefore, there was no explanation of these event types known to the trained KGE models. As expected, all models perform very poorly when trying to find and explanation for FlexUnavailableState events. None of the models found any event, which was contained in the ground truth data. Only TransE and TransH came close by finding some events which are probably true. Conversely, predictions on FlexRequestRejected events are extremely good. TransE and TransH performed best on this type of event. As there was no FlexRequestRejected event present in Village A at all, it is impressive that performance on this event type is among the best when comparing to other event types. As this type of causality was definitely not learned by the models explicitly, it seems like the models could pick up on the

event type	**model**	**defHits@5**	**defHits@5 (adjusted)**	**probHits@5**
FlexRequest Approved	**TransE**	0.17*	0.17*	0.23*
	TransH	0.07	0.07	0.17
	ComplEx	0.10	0.10	0.23
	TTransE	0.03	0.03	0.03
FlexRequest Rejected**	**TransE**	0.50*	0.50*	0.70*
	TransH	0.50*	0.50*	0.70*
	ComplEx	0.10	0.10	0.10
	TTransE	0.30	0.30	0.40
Flex Unavailable State**	TransE	0.00	0.00	0.07
	TransH	0.00	0.00	0.13*
	ComplEx	0.00	0.00	0.00
	TTransE	0.00	0.00	0.00
Lowering Demand	TransE	0.20	0.20	0.20
	TransH	0.40*	0.40*	0.40*
	ComplEx	0.00	0.00	0.40*
	TTransE	0.40*	0.40*	0.40*
Normalizing	**TransE**	0.60*	0.60*	0.60*
	TransH	0.40	0.40	0.60*
	ComplEx	0.20	0.20	0.20
	TTransE	0.20	0.20	0.20
Overloading	**TransE**	0.40*	0.40*	0.40*
	TransH	0.40*	0.40*	0.40*
	ComplEx	0.00	0.00	0.10
	TTransE	0.10	0.10	0.10
Peaking Demand	**TransE**	0.20*	0.20*	0.20*
	TransH	0.20*	0.20*	0.20*
	ComplEx	0.00	0.00	0.00
	TTransE	0.20*	0.20*	0.20*
Overall	**TransE**	0.24*	0.24*	0.30*
	TransH	0.20	0.20	0.29
	ComplEx	0.06	0.06	0.15
	TTransE	0.11	0.18	0.13

* best result in this category
** event type is not present in the training data

Table 5.4: Link Prediction Performance over all event types and models

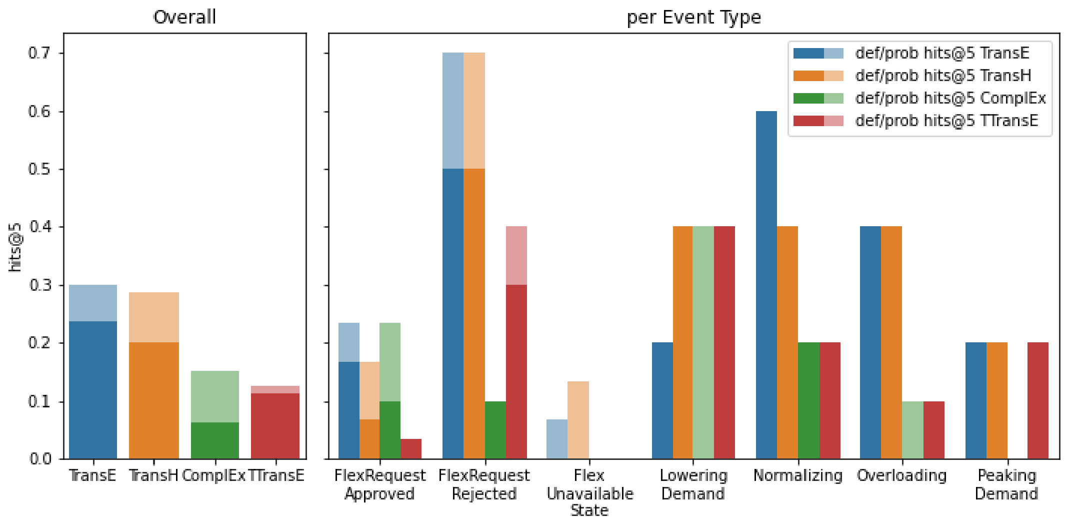


Figure 5.4: Link Prediction Performance of all four models overall and per event type

semantic implications of causality well enough to be able to predict causal events of an event which was unseen before. However, this conclusion cannot be drawn without hesitation as applying the models for predicting FlexUnavailableState did not work as well. On the other hand, causal explanations of FlexUnavailableState may be very hard to capture as this event only shows that the state of the building is not available. This could either be a coincidence or it could be caused by a lack of communication.

While *probable hits@5* increase model performances for some types of events, general performance rankings between the four models hardly ever change between definite and *probable hits@5*. Only for LoweringDemand events, ComplEx does not predict any causal events from the ground truth, while it outperforms TransE when including events which are probably true.

In table 5.4, the best-performing models are marked per event type by printing the model names in bold font. Even though TransE does not perform best over all types of models, it still is one of the best performing models for most event types. The biggest achievement of TransE is that it always managed to predict at least a probable cause for each event type. When looking at the individual events, it achieves *probHits@5*>0 for each of the 17 explanations except for two out of four UnavailableState events.

There was no significance test conducted on the statistical significance of the results as the data set is too small to achieve further insights. For the majority of event types, less than three explanations were tested. Checking for statistically significant performance differences would require a longer time period to be tested, where more events can be explained and evaluated. This is subject to future work.

	TransE	**TransH**	**ComplEx**	**TTransE**
		timestamp		
defHits@5	-0.79	-0.73	-0.29	-0.76
probHits@5	-0.85	-0.72	-0.49	-0.79

Table 5.5: Pearson Correlation Coefficient between timestamp and performance

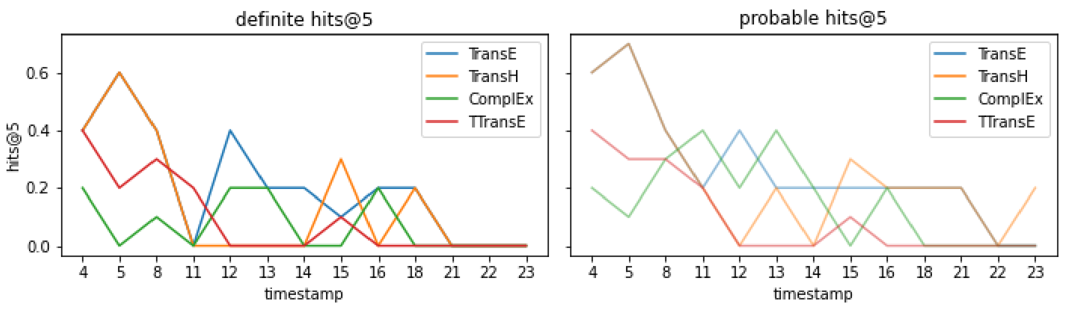


Figure 5.5: Prediction performance over time

5.3.1 Timeseries Analysis

In an attempt to analyse prediction results in more detail, model performance over time is depicted in figure 5.5. On the left side, *definite hits@5* are shown, while *probable hits@5* can be observed on the right figure. Obviously, *probable hits@5* generally performs better than *definite hits@5*. However, the same trend is visible for all models for both performance metrics. Models perform worse on effect events occuring later in the time interval. It seems like finding the right causal event gets more difficult over time. While visual plots over time give a good idea of general trends in the data, it is not enough to jump to conclusions yet. Therefore, Pearson correlation statistics are displayed in table 5.5. For each model, correlation between the timestamp of an event and its performance (*defHits@5*, *probHits@5*) is calculated. The magnitude of correlation varies between models, but with no exception all of the performances correlate negatively with time.

While there could also be other explanations of this behavior, the experimental setup may be responsible for this temporal correlation. As only previous events from Village B are considered for link prediction, the set of potential events which can cause an event grows over time as there are more events which happened in the past the longer a simulation has already been running. This makes it harder for models to predict the right causal events at a later point in time.

Additionally, the correlation between time and performance may suggest that performance differences between event types are not only due to different abilities of a model to understand types of events and their causal relations, but due to different events occuring at different timestamps. To analyse this theory in more detail, figure 5.6 shows

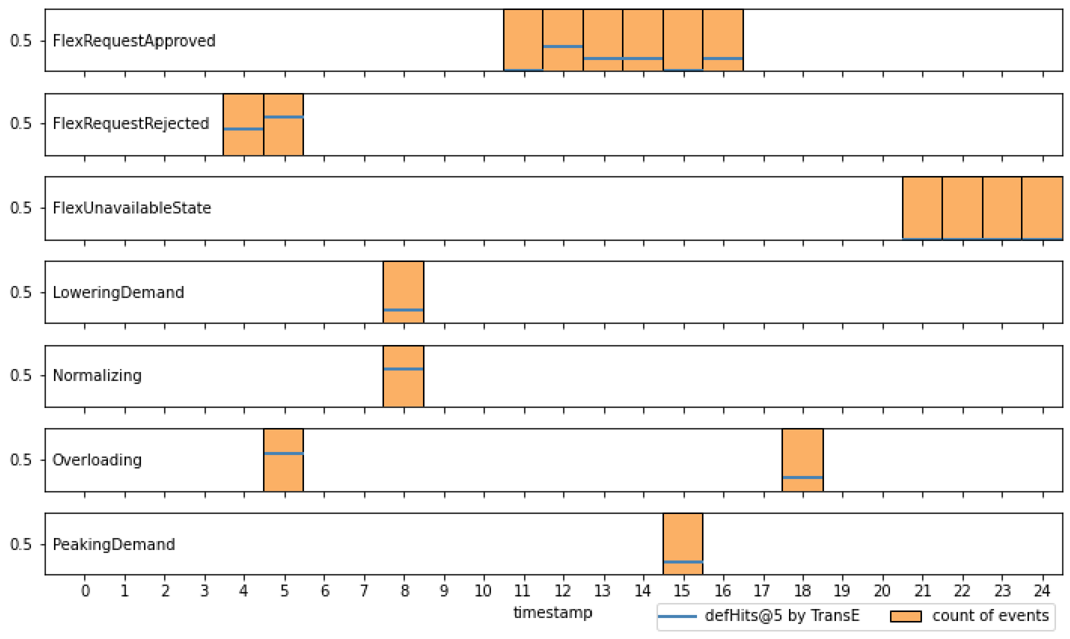


Figure 5.6: Prediction performance of TransE on different event types over time

the distribution of all seven event types on the time interval of 24h as orange bars. An orange bar always has a height of one as one type of effect event only occurs maximum once per timestamp. One can see that FlexRequestApproved, FlexRequestRejected, FlexUnavailableState and Overloading events occur across a larger time interval. For all other event types only a single event happens over the whole time frame, so no analysis of time development can be conducted. The blue bars in figure 5.6 show the prediction performance of TransE for each event at the respective timestamp. While there is a big difference between prediction performances over time at Overloading events, there is no clear correlation over time for any of the other events. While a different data set size of potential causal events over time may have an influence, there may be a correlation over time merely due to different types of events (which have different model performances generally) happening at different timestamps.

5.3.2 Ground Truth Analysis

In a next step, the effect of the number of ground truth data available in an explanation will be investigated in detail as this may also have an influence on the performance of a prediction model. For example, if only two ground truth events exist in the validation data, only a maximum of two events can be predicted correctly according to the ground truth, while up to five events can be predicted correctly if five or more ground truth events are available in the test data. While the metric *hits@5 (adjusted)* should account for this fact, table 5.4 shows that there is no difference between the *hits@5* metric and *hits@5 (adjusted)*. Since the adjusted metric only has an effect if all ground truth

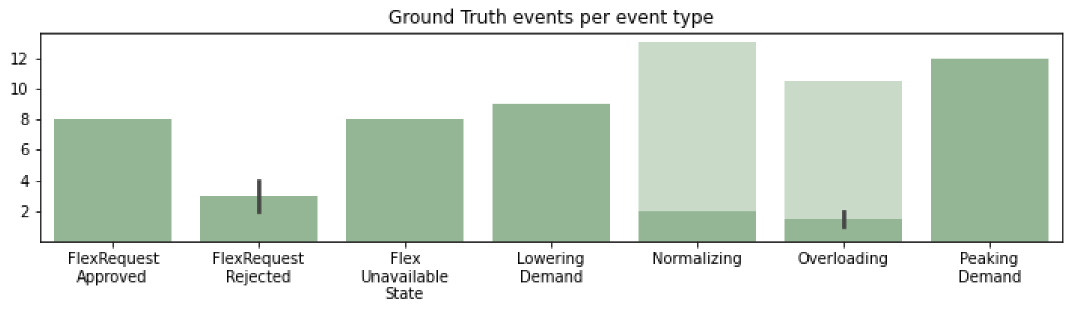


Figure 5.7: Number of Ground Truth events for each type of effect event

events are already predicted, even for events with only a small amount of ground truth data, not all ground truth values were predicted. If only one event is missing, the metric is the same as the unadjusted one. This shows that the adjusted *hits@5* metric does not properly take into account the different amount of ground truth values for the evaluation of different events.

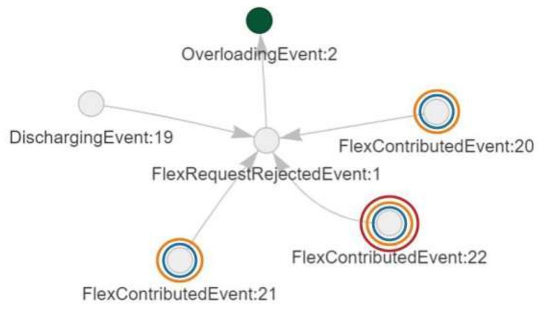
As the number of ground truth events for an explanation may still influence the metrics, a comparison of ground truth events in an explanation of different event types is shown in figure 5.7. The low number of events in the ground truth for Normalizing and Overloading events is misleading as these two types of events profit from additional level2 ground truth events, which can potentially be predicted by a model and will be counted as definite hits in the *hits@5* metric. The transparent bar for these two events shows the number of level 2-ground truth events for these two event types. For the remaining types of events, all event types contain more than five ground truth values, except for the FlexRequestRejected event type. As already mentioned before, the FlexRequestRejected event is one of the best-performing event types in the data, which shows that there is no influence of the number of ground truth values on the performance of event types visible in the data.

5.3.3 Level 2 Prediction Analysis

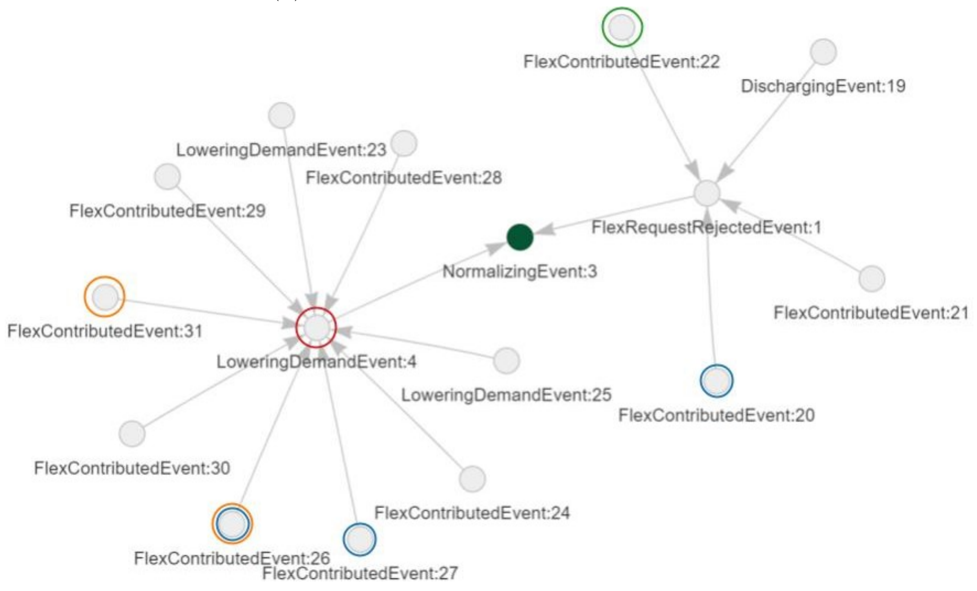
One of the most interesting achievements of using Knowledge Graph Embedding models for causality link prediction was the fact that these models were able to predict ground-truth data which was never directly connected to the events to be explained. This behavior shows that the models managed to represent the type of connection of a causality relation and could apply these semantics to the data. There are three events, which contain level 2 ground-truth in the data. In figure 5.8, these three events and their causality network is visualised. The dark green nodes of the graphs represent the event for which an explanation exists. The grey nodes represent the events, which existed in the ground truth as causes of the event. If an event was correctly predicted by an embedding model, a circle of the corresponding color is drawn around the predicted node.

OverloadingEvent, depicted in figure 5.8(a) shows the most interesting predictions, as none of the models could predict the immediate cause event, but three of the indirect causes of the OverloadingEvent were predicted by TransE and TransH. This means that 75% of the indirect causes were predicted correctly, while no direct cause could be predicted by these two models. As the other two events have a bigger causality network, the percentage of predicted events decreases (Only the top five predictions per model are considered). Interestingly, only TTransE managed to predict a direct cause of the NormalizingEvent in figure 5.8(b). For all other events and models, prediction of direct causes was extremely low.

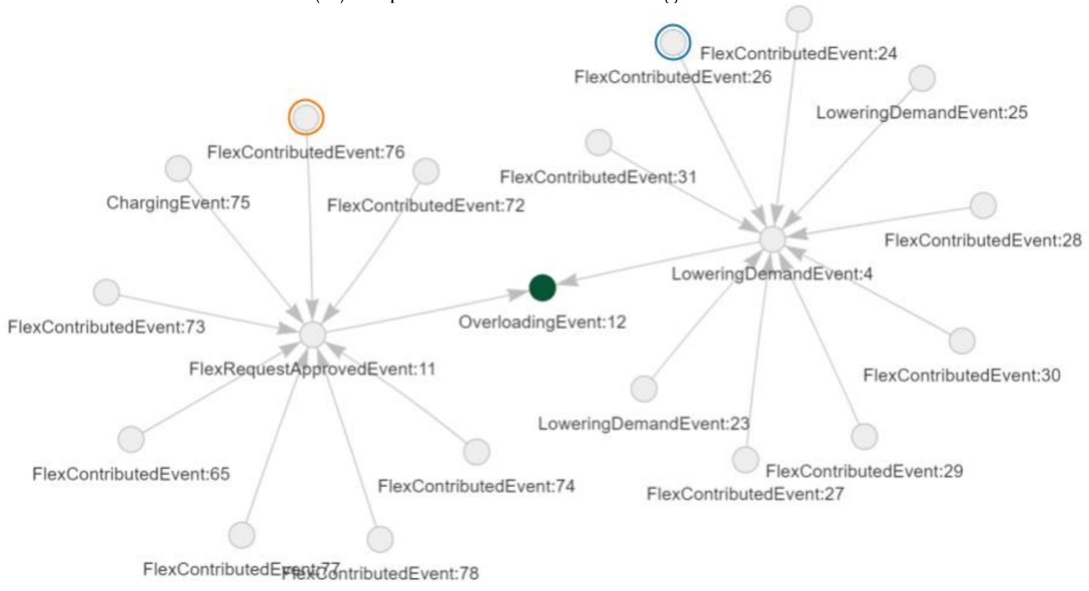
ComplEx does not seem to be a good model to predict any causal events which are not immediate. Over all three models, only a single level2 cause was correctly predicted by the model. Generally, the good performance of TransE and TransH can be seen in this visualisation as well, as is already visible in other performance graphs, such as figure 5.4. Additionally, worse performance for later events in the simulation is also visible in this graph. OverloadEvent 12 (depicted in 5.8(c)), which occured at 18:00 contains a lot less correctly predicted causal events than the other two events, which happened at 04:00 (Overload2)(Normalizing3) and 05:00.



(a) Explanation of Overload Event 2



(b) Explanation of Normalizing Event 3



(c) Explanation of Overload Event 12

Figure 5.8: Visualization of level2 predictions for three events. Colored circles show the predicitons of the respective models (● TransE, ● TransH, ● ComplEx, ● TTransE)

CHAPTER 6
Summary

In this final chapter, the main findings of the research project will be summarised. In section 6.1, the research questions will be revisited and the contribution of the thesis to answer these questions is analysed. In section 6.2, the findings will be discussed and drawbacks of the results and the experiment are considered. In the final section of this chapter, the contributions of this master thesis to the research community will be explained and evaluated.

In conclusion, the performance results show that TransE generally performs best on a causality link prediction task in the ExpCPS Knowledge Graph. Some general analysis on peroformance differences on different event types and timeframes was conducted, but additional analyses on reasons for perfomance differences between models is subject to further research. Additionally, the small size of test data in this experiment does not allow for valid statistical analysis of the results.

6.1 Conclusion

In this thesis, the application of Knowledge Graph Embedding Methods for causality link prediction was tested. Four KGE models were trained on a simulated Smart Energy Grid to test their ability to understand the semantics of a Cyber-Physical System and to use this knowledge to find causalities between events happening during the simulation. At the start of the thesis, three research questions were posed, which I tried to answer in the course of the thesis.

Q1 Which Knowledge Graph Embedding Methods are most suitable for causality detection in a Cyber Physical System?

A thorough literature research was conducted to find different KGE methods and to understand their strengths and weaknesses. While a lot of research on meta-analyses and studies to compare the most common KGE methods was conducted, these papers usually focussed on model performances on benchmark datasets, such as DBpedia. These benchmark datasets are very helpful for general comparison of performances, but these Knowledge Graphs contain very different information compared to a Knowledge Grpah which is used to model a Cyber-Physical System. For now, no research on the specific case of causality prediction in a cyber-physical system using KGEs was found. Therefore, the ability of different KGE models on this task is not clear yet.

To determine suitable embedding methods for causality detection in the ExpCPS Knowledge Graph, the ability of modelling different types of relations (Symmetry, Antisymmetry, Inversion, Composition, 1-N Relations) was investigated for a set of commonly used KGE methods. This analysis showed that there is no single embedding method, which is able to intrinsically cover all types of relations relevant in the ExpCPS Knowledge Graph. As mere analysis of the theoretical capabilities of the embedding methods did not yield a satisfying result, a set of four embedding models was chosen to be trained on the ExpCPS Knowledge Graph for further evaluation.

Out of a set of eight embeddings methods, which have been researched in detail, TransE is the only model which is capable of explicitly representing composite relations in the embedding. However, it is not able to model symmetry. As a development of TransE, TransH manages to model symmetry at the cost of composition. ComplEx has the same capabilities of TransH, but it follows a very different approach as it is based on semantic matching. ComplEx was included in the analysis as this model is recommended in various papers to be used for any type of first KGE analysis of a Knowledge Graph as it generally yields very good results. As a new development of KGE models, temporal KGEs use time as an additional variable to correctly embed Knowledge Graphs. As the ExpCPS KG does evolve over time and time is an important factor for event analysis, TTransE was included in the analysis to check if this type of embedding improves prediction performance.

In conclusion, **Research Question 1** could not be fully answered in this thesis, but the exploration of KGE models for the task of causality link prediction was started by applying a set of promising KGE models, which can model relation types that are important for ExpCPS and which show promising results in benchmark analyses.

Q2 How well do the chosen Knowledge Graph Embedding Methods from Q1 perform in representing the knowledge base captured by the ExpCPS Knowledge Graph?

The four embedding methods were trained on ExpCPS data and evaluated on different subgraphs of the Knowledge Graph (explanation, events, inferred graph). Generally, there was a big difference in terms of performance between the subgraphs. Overall, performance was between 0.7 and 0.8 *hits@10* for any of the four embedding models. ComplEx performed best overall in terms of *hits@10*, while TransE outperformed the other models on Mean Rank. Moreover, performance differences were small when comparing performance on the whole Knowledge Graph, but bigger differences occured when looking at subgraphs of the data. Overall, ComplEx performed best on time-invariant data, such as topology of the network and potential causes between sensors. TransE and TransH did not perform very well on general data, but outperformed the other models on explanation data. Generally, all of the four models managed to perform decently on representing the ExpCPS knowledge base. However, there is still room for improvement. No single embedding model outperformed the others over all categories, but different types of data could be predicted better by different models.

Q3 How well can the embedding space be used to uncover causality relations in a system?

For the last research question, the model performances on the task of link prediction in the explanation subgraph was tested. TransE performed best on this task, achieving 0.24 on the hits@5 metric on ground truth data. TransH was not far behind, while TTransE and ComplEx both achieved less than half of the peroformance of TransE. These performance results are not perticularly high and these models are not ready to be deployed for any real-world usage yet. However, the tested KGE models managed to understand the semantics of the KG and derived unknown causality from the trained data. Additionally, TransE and TransH predicted indirect causal relations (causes of a cause), which were not present in the ground truth data.

Similarly to research question 2, performances vary a lot between event types. TransE achieved a hits@5 metric of 0.6 for Normalizing events, while no model could predict any correct causes for FlexUnavailableState events.

Additionally, a set of events could be predicted that did not occur in the ground truth, but an expert evaluation showed that they might be true causes missing from the current data set. These events should be investigated further as they could be used to complete the knowledge graph by adding these missing links.

6.2 Discussion

In conclusion, this master thesis showed that there is potential in the application of Knowledge Graph Emeddings for causality link prediction in Cyber-Physical Energy Systems. However, there are still a lot of potential improvements and further research to be conducted on this topic. Firstly, the analysis of causality link prediction was only conducted on simulated data from the BIFROST application, while the applicability on other energy systems, real world data or other cyber-physical systems is still to be explored. Additionally, the data was currently measured in 1h timesteps, while events occuring between two timestamps are not registered by the system. Working towards continuous time data for causality prediction could uncover interesting relations that may be missed in the current setting.

On another perspective, the improvement of prediction performance by knowledge graph embeddings could also be an interesting issue to work on. In this thesis, I tried to test a set of embedding models which seemed fitting to the use case. However, I was constrained by the number of models to be tested as well as the complexity of the models as I was restricted by limited computational resources. In future research, more advanced and complex embedding models might yield better results. Moreover, other machine learning approaches, such as Graph Neural Networks could even be better suited to predict causality in this setting. There are many topics which could potentially still be explored in this area.

6.3 Contributions of the Work

In this thesis, I introduced a new framework to test the ability of a model to find causality relations in a (simulated) energy village based on knowledge derived from another village. As the setup of a village is complete in its definition and causality relations to be trained are rare, a new training architecture was introduced to be able to test the performance of a machine learning model on causality prediction in an energy village.

Based on this setup, the application of Knowledge Graph Embeddings for the task of causality link prediction in a Cyber-Physical Energy System was explored. This thesis evaluated the potential of using machine learning approaches on this use case and could show that the right models can find not only ground truth data, but also indirect causalities as well as possible causality links which are missing in the existing data. Therefore, applying machine learning approaches could help in verifying existing causality links as well as finding new links which may have been missed by existing approaches, such as rule-based causality prediction.

Concerning the performances of different Knowledge Graph Embeddings, this work showed that TransE seems to be the most capable embedding model to predict causality links in the system. Even though it has one of the most simple scoring functions, it is one of few models which can inherently represent transitive relations. As causality is usually represented in causality paths, this ability is very helpful in finding causal relation.

In conclusion, this thesis started the exploration of hybrid AI, leveraging the abilities of symbolic and sub-symbolic AI in the use case of causality prediction in a Cyber-Physical Energy System. Some initial performance evaluation of KGE models showed the potential of this approach to yield new insights into a system that is not inherently explainable.

List of Figures

List of Tables

Bibliography

[ABH$^+$21] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, 22(82):1–6, 2021.

[ADES21] Peb R. Aryan, Matthias Deimel, Fajar J. Ekaputra, and Marta Sabou. Using SPARQL to express Causality in Explainable Cyber-Physical Systems. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pages 1–5, September 2021.

[AES$^+$20] Peb R. Aryan, Fajar J. Ekaputra, Marta Sabou, Daniel Hauer, Ralf Mosshammer, Alfred Einfalt, Tomasz Miksa, and Andreas Rauber. Simulation Support for Explainable Cyber-Physical Energy Systems. In *2020 8th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, pages 1–6, April 2020.

[AES$^+$21] Peb Ruswono Aryan, Fajar Juang Ekaputra, Marta Sabou, Daniel Hauer, Ralf Mosshammer, Alfred Einfalt, Tomasz Miksa, and Andreas Rauber. Explainable cyber-physical energy systems based on knowledge graph. In *Proceedings of the 9th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, pages 1–6, Virtual Event, May 2021. ACM.

[Ary21] Peb Ruswono Aryan. Knowledge Graph for Explainable Cyber Physical Systems: A Case study in Smart Energy Grids. *CEUR Workshop Proceedings*, 3005:8, 2021.

[BG11] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.

[BGCG$^+$19] Mathias Blumreiter, Joel Greenyer, Francisco Javier Chiyah Garcia, Verena Klos, Maike Schwammberger, Christoph Sommer, Andreas Vogelsang, and Andreas Wortmann. Towards Self-Explainable Cyber-Physical Systems. In *2019 ACM/IEEE 22nd International Conference on Model*

Driven Engineering Languages and Systems Companion (MODELS-C), pages 543–548, Munich, Germany, September 2019. IEEE.

[BGWB14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, February 2014.

[BRC+20] Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge Graph Embeddings and Explainable AI, April 2020. arXiv:2004.14843 [cs].

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[CBB+12] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, Vincent Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin Page, Alexandre Passant, Amit Sheth, and Kerry Taylor. The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, December 2012.

[CPMJ20] Luca Costabello, Sumit Pai, Nicholas McCarthy, and Adrianna Janik. Knowledge graph embeddings tutorial: From theory to practice, September 2020. https://kge-tutorial-ecai2020.github.io/.

[DWXG20] Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. A Survey on Knowledge Graph Embedding: Approaches, Applications and Benchmarks. *Electronics*, 9(5):750, May 2020. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[EW16] Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. *Citeseer*, 48:4, September 2016.

[FEM+18] Michael Färber, Basil Ell, Carsten Menne, Achim Rettinger, and Frederic Bartscherer. Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *IOS Press*, 9(1):77–129, 2018.

[FSA+20] Dieter Fensel, Umutcan Simsek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. Introduction: What Is a Knowledge Graph? In *Knowledge Graphs*, pages 1–10. Springer International Publishing, Cham, 2020.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth*

International Conference on Artificial Intelligence and Statistics, pages 249–256. JMLR Workshop and Conference Proceedings, March 2010. ISSN: 1938-7228.

[GLV19] Joel Greenyer, Malte Lochau, and Thomas Vogel. Explainable software for cyber-physical systems (es4cps): Report from the gi dagstuhl seminar 19023, january 06-11 2019, schloss dagstuhl. *arXiv preprint arXiv:1904.11851*, 2019.

[GSC+19] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. XAI—Explainable artificial intelligence. *Science Robotics*, 4(37):eaay7120, December 2019. Publisher: American Association for the Advancement of Science.

[GSL+21] Nabila Guennouni, Christian Sallaberry, Sébastien Laborie, Richard Chbeir, and Elio Mansour. ISEE: A heterogeneous information system for event explainability in smart connected environments. *Internet of Things*, 16:100457, December 2021.

[HLJZ15] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to Represent Knowledge Graphs with Gaussian Embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 623–632, New York, NY, USA, 2015. Association for Computing Machinery.

[HS17] Katsuhiko Hayashi and Masashi Shimbo. On the Equivalence of Holographic and Complex Embeddings for Link Prediction, September 2017. arXiv:1702.05563 [cs].

[HZMT21] Zhen Han, Gengyuan Zhang, Yunpu Ma, and Volker Tresp. Time-dependent Entity Embedding is not All You Need: A Re-evaluation of Temporal Knowledge Graph Completion Models under a Unified Framework. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8104–8118, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[IK20] Eleni Ilkou and Maria Koutraki. *Symbolic Vs Sub-symbolic AI Methods: Friends or Enemies?* November 2020.

[JGC+21] Zhuochen Jin, Shunan Guo, Nan Chen, Daniel Weiskopf, David Gotz, and Nan Cao. Visual Causality Analysis of Event Sequence Data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1343–1352, February 2021. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[Jha22] Sanjiv Subodhnarayan Jha. An Overview on the Explainability of Cyber-Physical Systems. *The International FLAIRS Conference Proceedings*, 35, May 2022.

[JPC$^+$22] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, February 2022. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[Kar11] Stamatis Karnouskos. Cyber-Physical Systems in the SmartGrid. In *2011 9th IEEE International Conference on Industrial Informatics*, pages 20–23, July 2011. ISSN: 2378-363X.

[KP18] Seyed Mehran Kazemi and David Poole. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[LLL$^+$15] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling Relation Paths for Representation Learning of Knowledge Bases, August 2015. arXiv:1506.00379 [cs].

[LLS$^+$15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), February 2015.

[Man19] Elio Mansour. *Event detection in connected environments.* phdthesis, Université de Pau et des Pays de l'Adour, November 2019.

[MDE$^+$19] Ralf Mosshammer, Konrad Diwold, Alfred Einfalt, Julian Schwarz, and Benjamin Zehrfeldt. BIFROST: A Smart City Planning and Simulation Tool. In Waldemar Karwowski and Tareq Ahram, editors, *Intelligent Human Systems Integration 2019*, Advances in Intelligent Systems and Computing, pages 217–222, Cham, 2019. Springer International Publishing.

[NNNP18] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333, 2018. arXiv:1712.02121 [cs].

[NRP16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic Embeddings of Knowledge Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), March 2016. Number: 1.

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 809–816, Madison, WI, USA, June 2011. Omnipress.

[PSL14] Joern Ploennigs, Anika Schumann, and Freddy Lécué. Adapting Semantic Sensor Networks for Smart Building Diagnosis. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014*, Lecture Notes in Computer Science, pages 308–323, Cham, 2014. Springer International Publishing.

[PY10] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[SCMN13] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[Set07] Anil Seth. Granger causality. *Scholarpedia*, 2(7):1667, July 2007.

[Sin12] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[Ste18] Bram Steenwinckel. Adaptive Anomaly Detection and Root Cause Analysis by Fusing Semantics and Machine Learning. In Aldo Gangemi, Anna Lisa Gentile, Andrea Giovanni Nuzzolese, Sebastian Rudolph, Maria Maleshkova, Heiko Paulheim, Jeff Z Pan, and Mehwish Alam, editors, *The Semantic Web: ESWC 2018 Satellite Events*, Lecture Notes in Computer Science, pages 272–282, Cham, 2018. Springer International Publishing.

[Tri19] Mayank Tripathi. Initialization Techniques for Neural Networks, March 2019.

[TWR$^+$16] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2071–2080. PMLR, June 2016. ISSN: 1938-7228.

[WHCMS18] Marcin Wylot, Manfred Hauswirth, Philippe Cudré-Mauroux, and Sherif Sakr. RDF Data Storage and Query Processing Schemes: A Survey. *ACM Computing Surveys*, 51(4):84:1–84:36, September 2018.

[WMT+07] Xiang-jun Wang, Swathi Mamadgi, Atit Thekdi, Aisling Kelliher, and Hari Sundaram. Eventory – An Event Based Media Repository. In *International Conference on Semantic Computing (ICSC 2007)*, pages 95–104, September 2007.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, December 2017. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[WWG15] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules, 2015.

[WWMK20] Jian Wang, Xi Wang, Chaoqun Ma, and Lei Kou. A survey on the development status and application prospects of knowledge graph in smart grids. *IET Generation Transmission & Distribution*, 15, December 2020.

[WZFC14a] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph and Text Jointly Embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, Doha, Qatar, October 2014. Association for Computational Linguistics.

[WZFC14b] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph Embedding by Translating on Hyperplanes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), June 2014. Number: 1.

[WZL06] Kamin Whitehouse, Feng Zhao, and Jie Liu. Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Kay Römer, Holger Karl, and Friedemann Mattern, editors, *Wireless Sensor Networks*, volume 3868, pages 5–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. Series Title: Lecture Notes in Computer Science.

[XHHZ17] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. TransG : A Generative Mixture Model for Knowledge Graph Embedding, September 2017. arXiv:1509.05488 [cs].

[XLL+22] Chongchong Xu, Zhicheng Liao, Chaojie Li, Xiaojun Zhou, and Renyou Xie. Review on Interpretable Machine Learning in Smart Grid. *Energies*, 15(12), 2022.

[XLLS17] Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Image-embodied Knowledge Representation Learning, May 2017. arXiv:1609.07028 [cs].

[YYH+15] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases, August 2015. arXiv:1412.6575 [cs].