GOTTFRIED WILHELM LEIBNIZ UNIVERSITÄT HANNOVER
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK

Bias Assessments of Benchmarks for Link Predictions over Knowledge Graphs

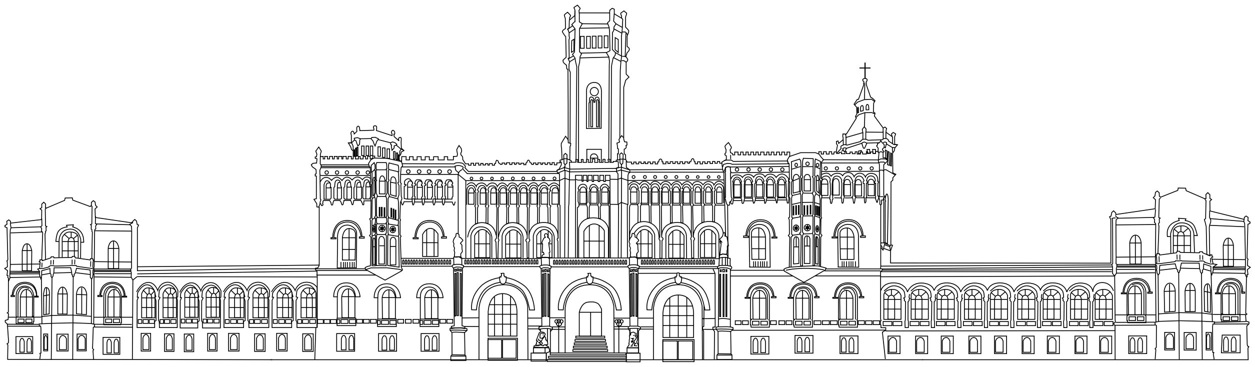
A thesis submitted in fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science

BY

Sammy Fabian Sawischa
Matriculation number: 10027805
E-mail: sawischa@stud.uni-hannover.de

First evaluator: Prof. Dr. Maria-Esther Vidal
Second evaluator: Jun.-Prof. Dr.-Ing. Alexander Dockhorn
Supervisor: M.Sc. Mayra Russo

April 26, 2023



Declaration of Authorship

I, Sammy Fabian Sawischa, declare that this thesis titled, 'Bias Assessments of Benchmarks for Link Predictions over Knowledge Graphs' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Sammy Fabian Sawischa

Signature: [illegible]

Date: 26th of April 2023

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts"

— Arthur Conan Doyle (Sherlock Holmes)

Acknowledgements

I would like to begin by expressing my appreciation to my research supervisor, Mayra Russo. This paper could not have been completed without her direction and unwavering participation at every stage of the process. I would like to express my gratitude for your relaxed attitude, patience and support throughout this process.

Prof. Dr. Maria-Esther Vidal is to be sincerely thanked for giving me the platform to write my bachelor's thesis in the Scientific Data Management research group at the German National Library of Science and Technology (TIB). The prior research experience in your group and your guidance largely contributed to the writing of this thesis.

My heartfelt gratitude goes to my family for making it possible for me to pursue this degree and for accompanying me throughout this journey.

Abstract

Link prediction (LP) aims to tackle the challenge of predicting new facts by reasoning over a knowledge graph (KG). Different machine learning architectures have been proposed to solve the task of LP, several of them competing for better performance on a few de-facto benchmarks. The problem of this thesis is the characterization of LP datasets regarding their structural bias properties and their effects on attained performance results. We provide a domain-agnostic framework that assesses the network topology, test leakage bias and sample selection bias in LP datasets. The framework includes SPARQL queries that can be reused in the explorative data analysis of KGs for uncovering unusual patterns. We finally apply our framework to characterize 7 common benchmarks used for assessing the task of LP. In conducted experiments, we use a trained TransE model to show how the two bias types affect prediction results. Our analysis shows problematic patterns in most of the benchmark datasets. Especially critical are the findings regarding the state-of-the-art benchmarks FB15k-237, WN18RR and YAGO3-10.

Keywords: Link Prediction, Benchmarks, Knowledge Graphs, Sample Selection Bias, Test Leakage

Zusammenfassung

Die Herausforderung in Link-Vorhersagen (LV) liegt darin, neue Fakten aus einem existierenden Wissensgraphen abzuleiten. Verschiedene Architekturen im Bereich des maschinellen Lernens wurden vorgeschlagen, um die Aufgabe von LV zu lösen. Viele der Herangehensweisen konkurrieren dabei bezüglich ihrer Leistungsfähigkeit auf wenigen De-facto-Benchmarks. Das Ziel dieser Arbeit besteht darin, LV-Datensätze zu charakterisieren bezüglich ihrer strukturellen Verzerrungseigenschaften, und deren Folgen auf erreichte Leistungswerte. Wir stellen ein domänen-agnostisches Framework zur Verfügung, das die Netzwerktopologie, Informationslecks und Stichprobenauswahlverzerrungen in LV Datensätzen bewertet. Das Framework umfasst SPARQL-Abfragen, die in der explorativen Datenanalyse von Wissensgraphen zur Aufdeckung ungewöhnlicher Muster wiederverwendet werden können. Schlussendlich nutzen wir unser Framework, um 7 häufig verwendete Benchmarks, die die Aufgabe von LV bewerten, zu charakterisieren. In durchgeführten Experimenten verwenden wir ein trainiertes TransE-Modell, um zu zeigen, wie die beiden Verzerrungsarten die Vorhersageergebnisse beeinflussen können. Unsere Analysen zeigen problematische Muster in den meisten der Benchmark-Datensätze. Die Ergebnisse bezüglich der aktuellsten Benchmarks FB15k-237, WN18RR und YAGO3-10 sind dabei besonders besorgniserregend.

Stichwörter: Link-Vorhersagen, Benchmarks, Wissensgraphen, Stichprobenverzerrung, Informationslecks

Contents

List of Figures

List of Tables

Acronyms

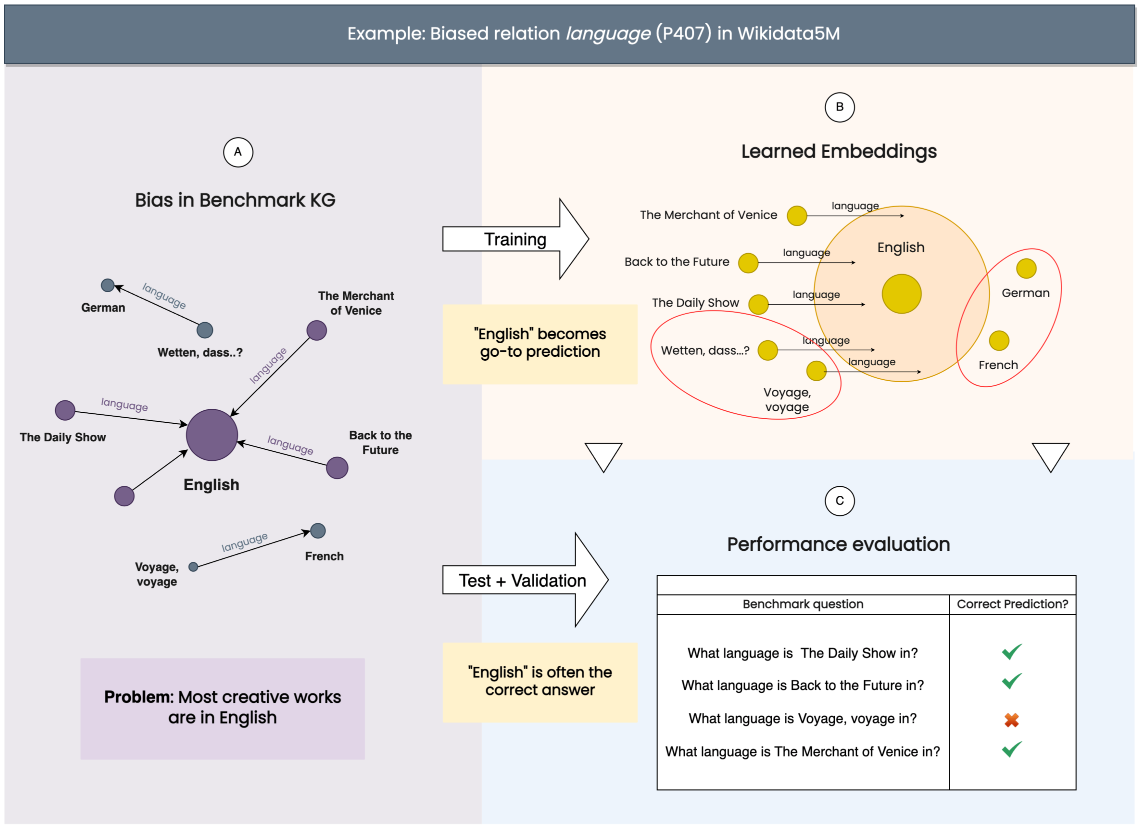
CSV	Comma-separated values
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
KG	Knowledge Graph
LP	Link Prediction
RDF	Resource Description Framework
RML	RDF Mapping Language
SPARQL	SPARQL Protocol And RDF Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Chapter 1

Introduction

In the field of data management and analytics, knowledge graphs (KGs) have emerged as a powerful asset for representing structured and semantically rich data, such as biomedical data [48], business intelligence [18] or general facts [62][3][9]. Due to their context-aware representations, KGs can improve explainability and transparency in machine learning (ML) systems by providing human-readable interpretations of ML models [44], which has been of particular research interest in recent years [59]. Prominent openly available KGs include DBpedia [3], Wikidata [62], Freebase [9], Wordnet [35] and YAGO [54]. Likewise, companies have been adopting KGs in domains like search, question-answering, or recommendation systems with notable KGs being Google KG [53] or Microsoft Satori [43]. Although KGs can contain billions of facts, they tend to suffer from incompleteness, a problem for which different refinement methods have been proposed [39]. A popular way to address this problem is through the task of link prediction (LP) by inferring new facts from given knowledge. In the past years, many architectures have been proposed to perform the task of LP [11][66][32][68][61], each competing for better performance on the same few *de-facto* benchmarks datasets like FB15k [11], FB15k-237 [60], WN18 [11], WN18RR [10] and YAGO3-10 [33]. Different studies, however, have been questioning the effectiveness of these benchmarks [1][46], thus casting doubt on obtained performance results. The variety of biases, i.e., unwanted factors that influence prediction results, is immense, starting from irregularities in the training data and ending in the way performance metrics are calculated. In this work, we focus on the assessment of biases in benchmark datasets used for LP tasks over KGs, and how they influence prediction results.

1.1 Motivating Example



Benchmark question	Correct Prediction?
What language is The Daily Show in?	✓
What language is Back to the Future in?	✓
What language is Voyage, voyage in?	✗
What language is The Merchant of Venice in?	✓

Figure 1.1: Motivating example. The relation *language* in the Wikidata5M benchmark dataset has the overrepresented entity *English* in the benchmark's KG (A). This imbalance propagates into the learned model in the embeddings (B) as well as the benchmark questions on which performance is assessed (C). The consequence is a good performance result on a benchmark that does not discourage overfitting behavior.

We motivate our work by illustrating how a structural imbalance found in the Wikidata5M dataset can lead to overfitting behavior and artificially inflate performance results (see Figure 1.1). A structural imbalance, here, refers to an unequal distribution of samples among entities or relations in a dataset. We, therefore, present a case of *Sample Selection Bias*.

First, the imbalance can be observed in the relation language (P407), which is used in Wikidata to describe the language of names or creative works, such as books, shows or songs [67]. The language relation often refers to the entity *English*. In fact, 68.9% of all existing names or creative works in Wikidata5M are in English (amongst potential other languages). Figure 1.1 A shows the overrepresented entity *English* in the KG, constituted by the benchmark dataset, amongst less frequently mentioned languages like German or French. The imbalance is then propagated into the training graph (see Figure 1.1 B). This can cause a situation in the embeddings where *English* is becoming the go-to prediction at the cost of neglecting other plausible options like German or French. Since the validation and test set also stem from the same biased KG, the benchmark questions used for evaluating the performance of the tested model also tend to have *English* as a correct answer (see Figure 1.1 C). This results in inflated performances in the evaluation process, where the benchmark is actually encouraging potential overfitting behavior. Even a simple model that adheres to such overrepresentation by only predicting *English* would reach good performance results in this particular instance.

1.2 Contributions

This thesis tries to shed light on current benchmark practices for the task of LP by deconstructing the factors that influence reported performance results. The main contributions can be summarized as follows:

- A domain-agnostic framework to characterize LP datasets.
- Reusable SPARQL queries that uncover imbalances in KGs.
- A thorough analysis of the influence of bias patterns on prediction results revealing insights on current benchmark practices.

1.3 Structure of the Book

The thesis is structured as follows. Chapter 2 explains preliminary background concepts regarding KGs and LP that will be taken up again in later chapters. Chapter 3 addresses related work and provides more context for this study. Chapter 4 deals with the approach of our study and introduces a framework for characterizing LP datasets. Chapter 5 presents the framework implementation and lists implementation details for conducted experiments. Chapter 6 evaluates the analysis of seven different benchmarks. Finally, concluding remarks are made in Chapter 7 with an outlook for future work.

Chapter 2

Background

This chapter introduces the main topics needed to understand the development of this thesis.

2.1 Knowledge Graphs and RDF

The idea behind the Semantic Web or Web 3.0, as first described by the inventor of the internet Berners-Lee, is to make the available data on the internet machine-readable [7]. Since then, many efforts have been made to represent knowledge in standardized ways to enable intercommunication between domains by both humans and machines [40]. A knowledge graph (KG) can be seen as a data graph that is used to gather and communicate knowledge about the real world [22]. For our use case, we define a KG as a labeled directed graph with $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{G})$: $\mathcal{E}$ is a set of node entities; $\mathcal{R}$ are the labels that describe a relation connecting two or more entities; $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times E$ is a set of facts. Each fact can be represented as a triple $\langle s, p, o \rangle$ consisting of a subject s, a predicate p and an object o. For instance, we can represent former chess world champion Magnus Carlsen's nationality with the triple $\langle MagnusCarlsen, hasNationality, Norwegian \rangle$. A typical framework to describe such statements inside a KG is the Resource Description Framework (RDF). RDF is a standardized data model recommended by W3C that consists of resources that can either be classes, instances, or properties. They are uniquely referenced by an URI or represented by a literal [29]. Incorporating web-based protocols like XML, HTTP, and JSON, RDF facilitates the representation of factual knowledge through triples consisting of a subject, object property (relation), and object [29].

2.2 RML and SPARQL

To produce a KG, the RDF mapping language RML [14] can be used to transform data sources from structured formats like CSV and JSON into a list of RDF triples. The generated RDF triples combined, then, constitute an RDF-conform knowledge graph. The mapping rules dictate how the source data is mapped to RDF data and is specified by one or more triples maps. Each triples map consists of a *Logical Source* (input data), a *Subject Map* linking the subject of generated triples to a column or section in the source data, and zero or more *Predicate-Object Maps*, a function that creates the predicate-object pairs for each record of a logical source [14]. Once triples are mapped and an RDF graph is correctly set up, the SPARQL [41] query language can be used for analyzing the graph and for extracting meaningful information.

2.3 The Link Prediction Task

In network theory, link prediction (LP) has traditionally been defined as the task of predicting edges between nodes in a graph [36]. Examples of such edges include friendships of users in a social network [31] or drug-target interactions in a pharmacological network [38]. Applied over knowledge graphs, LP addresses the challenge of incompleteness in a KG by inferring new facts from initial existing knowledge [47]. Other use cases of LP include question-answering [28] or recommendation systems [30]. In general, an LP model is trained with the aim of providing a plausibility score ϕ for a given fact. The plausibility score can be seen as a measure for displaying the truthiness of a triple. In the case of tail predictions, the model tries to complete the partially observed triple $\langle h, r, ?\rangle$ by returning the corresponding tail entity t which maximizes ϕ for the incomplete triple:

$$t = \underset{e \in \mathcal{E}}{\operatorname{argmax}}\, \phi(h, r, e)$$

For training models that are capable of predicting ϕ for unseen facts, most state-of-the-art LP methods first convert all entities and relations into low-dimensional vectors as an intermediate step [47], making it easier to apply optimization techniques that require mathematical operations like matrix factorization or other vector arithmetic. The learned vector representations of entities and relations are also known as *embeddings*. Idealistically, learned embeddings preserve the structure of information between entities connected through relations [11], enabling simple access to information. For instance, the capital of France could be retrieved using the following operation: $vector("France") - vector("isCapitalOf") \approx vector("Paris")$.

In recent years, however, a novel paradigm of LP called *inductive learning* has been under development [17]. It does not rely on learned entity representations, but instead, reasons over a training subgraph to learn meaningful structures. Learned features are then used to perform inference on a new inference graph with unseen entities [17], as opposed to the classical way of LP, *transductive learning*, where entities during inference must have been visible during training time for learning their representations (see Figure 2.1).

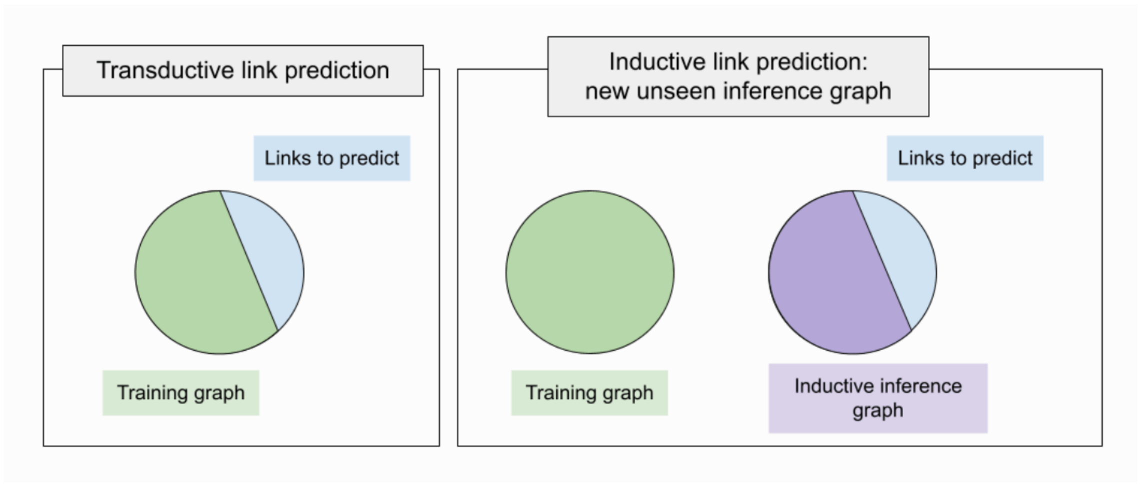

Transductive link prediction
Links to predict
Training graph
Inductive link prediction: new unseen inference graph
Links to predict
Training graph
Inductive inference graph

Figure 2.1: Inductive vs Transductive Learning. Visualization from [17]. In transductive LP, the training graph contains all entities which will be used for link predictions in the inference phase (test/validation predictions). In inductive LP, however, a new inductive inference graph containing new unseen entities is created. The LP model now needs to reason over the inference graph to make meaningful link predictions between entities that were not previously visible in the training data.

2.3.1 Transductive Learning

Transductive learning methods use different techniques to learn the right vector representations for entities and relations. However, they can be classified into three main families of approaches [47], all of which optimize embeddings with their own defined plausibility function ϕ to determine what good representations are: (1) Tensor decomposition, (2) Geometric approaches and (3) Neural network-based approaches.

Tensor Decomposition

A tensor can be seen as a multi-dimensional array of order n [24], that, in the context of machine learning, usually holds observed data like a user feedback matrix for movies in recommender systems [56]. The idea behind tensor decomposition, analogously to the technique of matrix factorization defined for the 2D space, is to approximate embeddings for the tensor's components such that the matrix product of all components results in the original tensor [50], where the embeddings can be consequently used to predict unseen compositions of the components. Link prediction approaches that use tensor decomposition define the tensor as a 3D adjacency "cube" containing entries A_{ijk}, where $i, j \in \{1, 2, ..., n\}$ denote the indices of entities in $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ and $k \in \{1, 2, ..., m\}$ the index of a relation in $\mathcal{R} = \{r_1, r_2, ..., r_m\}$, with $A_{ijk} = 1$ for observed facts. Due to the incompleteness of the KG, all non-existing fact triples can be either seen as both true or false (Open world assumption) or false (Closed world assumption), and it depends on the implementation which underlying assumption model to use for handling unobserved facts. The general goal is to optimize the embeddings in the component matrices E for entities and R for relations such that when combined with matrix multiplication they approximate A. For example, the popular LP framework RESCAL [64] in this family tries to optimize the embeddings as follows:

$$A \approx ERE^T \tag{1}$$

The transposed entity matrix E^T is used to describe tail objects, as head and tail entities have the same embeddings. Other models like SimpIE [27] take a different approach and incorporate two separate embeddings for every entity depending on its usage as a head or tail entity. Given head and tail embeddings $h, r \in \mathbb{R}^d$ and a relation embedding $r \in \mathbb{R}^{d \times d}$, models in this family of approaches that use a bilinear scoring function can be described with the plausibility formula [47]:

$$\phi(h, r, t) = h \times r \times t \tag{2}$$

where $\times$ symbolizes a matrix multiplication. Other models in this approach family include TuckER [4], DistMult [68] and ComplEx [61].

Geometric Approaches

Models in this category consider relations as geometric transformations on entities in the embedding space [47]. In an ideal scenario, a relation r is optimized such that it can be used to transform any of its head entity vectors h to the corresponding tail entity vectors t and vice versa for any true fact $\langle h, r, t\rangle$. For the cases where the transformation on h does not result in the exact vector for t, a distance function δ is able to quantify the resulting deviation and is used to determine the plausibility ϕ of a prediction. A popular geometric model that interprets relations as simple translations in the latent space is TransE [11], which aims to approximate the embeddings so that $h + r \approx t$, or e.g. using the previous example: $vector(\text{"}Berlin\text{"}) + vector(\text{"}isCapitalOf\text{"}) \approx vector(\text{"}Germany\text{"})$. Since experiments in the past suggest that TransE does not handle well one-to-many and many-to-many relationships [66], newer versions of the translational model have been developed. TransH [66], for instance, enables different entity representations in a hyperplane depending on the relational context. TransR [32] uses a similar idea and captures different aspects of a relation by distinguishing between entity space and relation spaces. Apart from linear transformations, some models like TorusE [15] or RotatE [55] integrate rotational-like transformations to optimize embeddings. All in all, the scoring function for approaches in this family can be generalized as [47]:

$$\phi(h, r, t) = \delta(\tau(h, r), t) \tag{3}$$

where τ is the transformation applied on the head h using relation r. For TransE, a negative distance scoring function is used:

$$\phi(h, r, t) = -\|h + r - t\|. \tag{4}$$

Neural Network-based Approaches

LP methods in this family of approaches are not confined to the transductive paradigm, as some architectures only use entity embeddings to complement learned patterns [57]. In neural network-based LP approaches the model learns parameters like weights and biases affecting the neuron connections between layers and the model's capability to learn meaningful patterns from the input data [47]. Calculating the plausibility function for a triple is more complex and costly compared to translation-based or tensor-based LP methods. ConvE [13], for example, first concatenates head entity and relation embeddings making up the input layer for 2D convolutional layers on which filters are applied. The result of that layer is a feature map that is reshaped and transformed. In the end, the output is matched it with the tail entity embeddings to generate a plausibility score for a given triple [13].

2.3.2 Inductive Learning

As the name suggests, LP architectures using the inductive paradigm need to be able to generalize rules and complex structural patterns of relations for making predictions about unseen entities - either explicitly or implicitly. In the explicit case, logical rules (horn clauses) are mined from the KG using a probabilistic approach. An example of such a rule would be:

$$parent(A, B) \wedge child(B, C) \implies grandparent(A, C) \tag{5}$$

To extract such horn clauses from KGs a rule-mining tool like AMIE [16] can be used. Implicit approaches can encode these logical rules as well [57] but are usually additionally supported by other features. For instance, the LP framework GraIL [57] uses a graph neural network-based approach by learning local paths between nodes to make deductions about the semantical structure of a relation.

2.4 LP Datasets

An LP dataset is usually a set of triples $\mathcal{G}$ that is split into subsets for the various steps in the LP pipeline. Every triple in the dataset is either assigned to a training set $\mathcal{G}_{train}$, a validation set $\mathcal{G}_{valid}$ or a test set $\mathcal{G}_{test}$. Triples in $\mathcal{G}_{train}$ are used to train a model by learning from structures and patterns found in the sampled KG. Hyperparameters, e.g. the embedding dimension, are optimized using facts in $\mathcal{G}_{valid}$. The validation facts are evaluated to determine the best-performing hyperparameter configuration. Finally, facts in $\mathcal{G}_{test}$ are used to estimate the prediction performance of the final learned model on unseen data. In general, there are two different approaches to setting up the split between training, validation, and test data: The transductive setup and the inductive setup. In the more common transductive setup, all entities that occur in the validation and test subset have already been seen in the training data. In the inductive setup, instead of using the same graph for training and inference, a separate inference graph is created with either unseen entities (fully-inductive) or both seen and unseen entities (semi-inductive).

2.5 LP Evaluation Metrics

The most common method to evaluate an LP model is the *entity ranking protocol*, which ranks the answers to specific questions derived from test triples [65]. For every test triple $\langle h, r, t \rangle \in \mathcal{G}_{test}$, the model to be evaluated is faced with two questions. First, what head answers fulfill the incomplete triple $\langle ?, r, t \rangle$. Second, what are possible tail answers for the triples of the form $\langle h, r, ? \rangle$. Based on the plausibility scores ϕ that the model generates for each possible answer, rankings are calculated for both the missing head and tail entity. The idea behind this approach is that good-performing models should be able to distinguish correct answers from wrong ones in these question scenarios. Sometimes a question has multiple correct answers and it depends on the setting at hand whether to consider them when calculating the corresponding ranking (*filtered setting*) or not (*raw setting*). In the *raw setting*, a valid head or tail answer that is not the one included in the test fact at hand is considered a mistake and therefore competes with the expected answer in the ranking computation. For example, if a person speaks the three languages English, German, and Spanish and the test triple sees Spanish as the correct answer, it might end up in the third rank simply because English and German had higher plausibility scores. Let $\langle h, r, t \rangle$ be a test fact, the raw ranking r_t for the expected tail prediction t is computed as [47]:

$$r_t = |\{e \in \mathcal{E} \setminus \{t\} : \phi(h, r, e) > \phi(h, r, t)\}| + 1 \tag{6}$$

In the *filtered setting*, a valid head or tail answer outside of the test fact is not considered a mistake and is skipped during ranking computation. Let $\langle h, r, t \rangle$ be a test fact, the filtered ranking r_t for the expected tail prediction t is computed as [47]:

$$r_t = |\{e \in \mathcal{E} \setminus \{t\} : \phi(h, r, e) > \phi(h, r, t) \wedge \langle h, r, e \rangle \notin \mathcal{G}\}| + 1 \tag{7}$$

The ranks r_h for head predictions can be computed analogously. It can also be noted that different tie policies exist to handle the edge case when two entities have the same plausibility score and a rank needs to be resolved [47]. Once the rankings Q have been computed on the basis of test triples, the following global metrics are obtained to give an estimate of the overall performance across all test predictions.

Mean Rank (MR). It is the average rank and can obtain values in the range of 1 to $|\mathcal{E}|$:

$$MR = \frac{1}{|\mathcal{Q}|} \sum_{q \in Q} q \tag{8}$$

The lower it is, the better the model results are. Since it is very prone to outliers, researchers are resorting to the Mean Reciprocal Rank instead [47].

Mean Reciprocal Rank (MRR). It is the average inverse rank and can obtain values in the range of 0 to 1:

$$MRR = \frac{1}{|\mathcal{Q}|} \sum_{q \in Q} \frac{1}{q} \tag{9}$$

The higher it is, the better the model result is.

Hits@K or H@K. It is the portion of test predictions that have an equal or smaller ranking than a defined threshold K:

$$H@K = \frac{|\{q \in Q : q \leq K\}|}{|Q|} \tag{10}$$

Commonly used values for K are 1, 3, 5, and 10. The higher the H@K metric is, the better the model result is.

Chapter 3

Related Work

The methodology by which LP models attain specific levels of performance still remains ambiguous to a certain extent, but previous work tried to shed some light on current LP benchmarking practices e.g. by questioning the use of common evaluation metrics [65] or analyzing the benchmark datasets for irregular patterns that may artificially affect performance results [45]. Studies that report problems in LP benchmark datasets, however, often only focus on a few irregularities in isolation [60] [1] and do not approach the investigation of LP benchmarks in a holistic manner, often either neglecting the data's network overall topology or important structural properties that could explain prediction results.

Toutanova and Chen [60] are the first to report test leakage issues caused by near-duplicate and inverse relations in the datasets FB15k and WN18, and are able to outrank embedding-based LP models with a simple observed feature model based on only direct links between entities. As a consequence, they constructed the more challenging dataset FB15k-237 (*original name*: FB15kSelected).

Dettmers follows up on this behavior when introducing the LP model ConvE [13] by creating the more difficult dataset WN18RR after observing increased performance results for FB15k and WN18 on a simple rule-based model, which capitalizes on inverse relations. He also tries to make the case that LP models learning embeddings with more than one feature layer like ConvE have an advantage over "shallow" models like DistMult. The hypothesis is supported by explaining the difference in model performance with the arbitrary occurrence of high in-degree entities. The average PageRank of a dataset's network is used as an additional explanatory factor. Making conclusions based on global evaluation measures like H@K or MRR is, although common practice, to be seen as critical, as prediction results depend on countless factors and even small modifications made in datasets can break existing graph structures [47] leading to unexpected behavior.

Another large contributing factor to performance outcome are hyperparameters. In their study, Kadlec et al. prove that DistMult can reach state-of-the-art performance on FB15k and WN18 simply by properly tuning hyperparameters [25], therefore additionally casting doubts on reported metrics. It also raises the question of whether new LP architectures are necessary or if a more extensive fine-tuning of existing models is sufficient. We are not aware of any study that reproduces such behavior on more challenging datasets like FB15k-237 and WN18RR.

Rossi and Matinata [46] question the effectiveness of formulated benchmark datasets to evaluate an LP model's capability to learn relations. They find out that the existing skew in the distribution of entities towards high-degree entities can lead to overfitting behavior, thus artificially boosting performance results, which is not highlighted by current global metrics in use for LP tasks [46].

Akrami et al. [1] study four types of data redundancy in LP benchmarks and identify Cartesian Product Relations as a redundancy pattern that occurs in both FB15k and FB15k-237 as well as near-duplicate relations in YAGO3-10. They use AMIE in their experiments and observe that a simple rule-based model can compete with other LP models, even on the more challenging datasets FB15k-237 and WN18RR. In their bias study, Rossi et al. define three types of sample selection bias, that is unrealistic patterns in the dataset, and notice that skipping these bias-affected triples from the test set leads to a drop in overall performance. Although not explicitly mentioned, the defined types of patterns can be attributed to already covered imbalances in previous work like entities with high relation-specific in-degree [13], Cartesian Product Relations [1] or near-duplicate relations [60].

Pujara et al. [42] provide a new perspective on LP results with their assessment of KG properties like sparsity and diversity. They conclude that KG embedding methods perform worse on sparse and unreliable data, especially on KGs extracted from text.

In their benchmark critique, Wang et al. [65] examine how the commonly used entity ranking protocol is rather suited for the task of question answering and fails to properly evaluate a model's performance to perform knowledge base completion tasks, i.e. discovering new facts with provided knowledge and avoiding adding nonsensical triples. The bias lies in the fact that the entity ranking protocol only considers "positive" facts, which are known to be true, therefore not evaluating a model's performance to penalize nonsensical triples or previously unknown facts. To mitigate such a biased evaluation procedure, they introduce a new pair ranking protocol, aiming to evaluate the task of knowledge base completion by ranking a test triple against all possible combinations of entity pairs, regardless of the test triple's relation.

Chapter 4

Approach

Over time several benchmark datasets for LP tasks have emerged, but only a portion of them coupled with a few standard LP evaluation metrics like H@K and MRR are considered *de facto* benchmarking standards in assessing the task of link prediction. However, it is still unclear what aspects of LP the benchmarks are actually testing for, making it hard to help explain why models achieve certain performances on these benchmarks. Previous studies have reported the problem of test leakage [60] and other unrealistic patterns [46] within LP benchmark datasets. By adhering to such observable irregularities (e.g. redundant information through duplicate relations) tested LP models are enabled to achieve artificially high performances [47] [13]. This leads to the question of how severe the problem of biased predictions is across different benchmark datasets. To investigate this question our approach is to define a new analytical framework with which we can characterize any LP dataset.

4.1 Problem Statement

Given an LP dataset $\mathcal{G}$, its splits $\mathcal{G}_{train}$, $\mathcal{G}_{valid}$, $\mathcal{G}_{test}$, and its KG representation $\mathcal{K}$ we tackle the problem of characterizing bias aspects of $\mathcal{G}$ regarding link prediction tasks over $\mathcal{K}$. We are interested in finding out the structural attributes of LP datasets that have the potential to influence prediction results.

4.2 Proposed Framework

We propose a new analytical framework for characterizing bias in LP datasets. It is domain-agnostic and can be used to improve the explainability of prediction results that are produced by the datasets.

Figure 4.1 depicts the architecture of our proposed approach. Assessing LP datasets for bias is a two-fold process consisting of a *Descriptive Analysis* and *Prediction Analysis*.

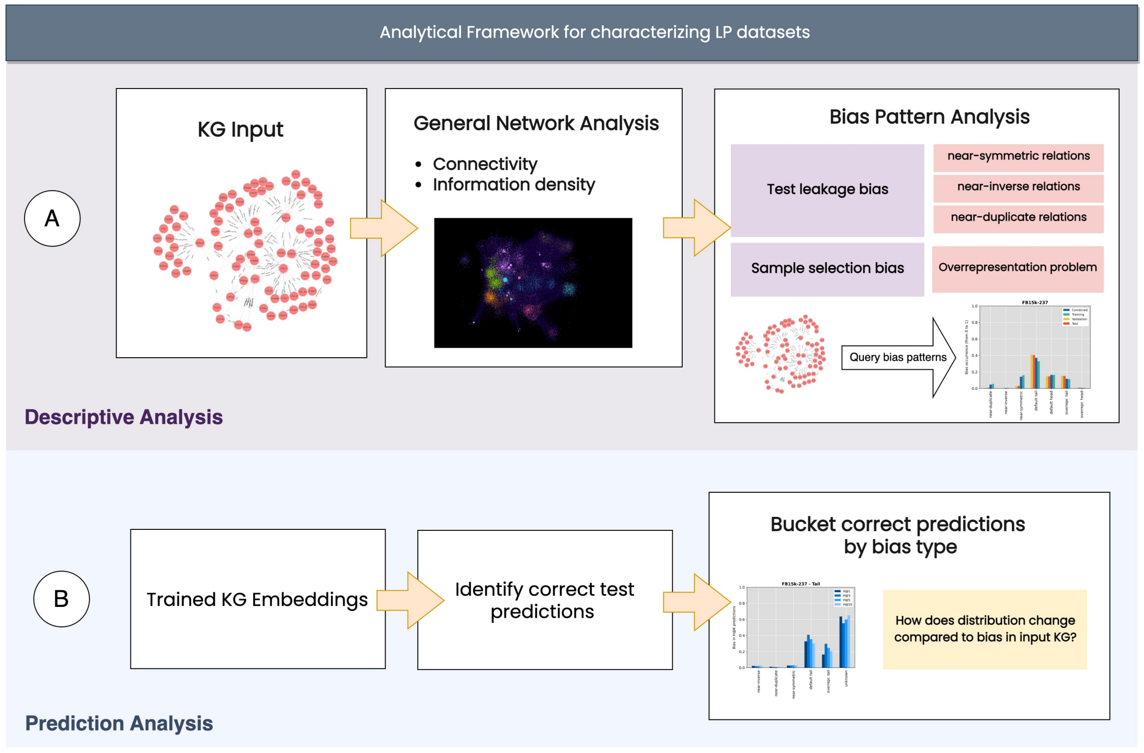


Figure 4.1: Approach. Our approach to characterizing LP datasets using our analytical framework consists of two steps (A) and (B). In the descriptive analysis (A), the pipeline first receives the KG which is implied by our LP dataset as an input. Then a general network analysis is performed regarding the measures of connectivity and information density. In the final step of the descriptive analysis, we analyze the KG for bias patterns regarding test leakage and sample selection. In the prediction analysis (B), we assess the influence of previously identified biases. First, correct predictions in the test set are identified. The correct predictions are then bucketed into different bias type categories to see how the distribution changes compared to our preliminary input analysis.

4.3 Descriptive Analysis

Figure 4.1A shows the components that are required to describe $\mathcal{G}$ on an input level. First, we perform a general network analysis. This provides us with an estimation of how well link predictions over $\mathcal{K}$ perform. A previous study has shown that KG embedding methods may not perform well on sparse graphs with low information density for either relations or entities [42]. Similarly, a training graph that is not well connected may make link prediction tasks more challenging. LP is even made impossible if we consider the extreme case of a completely unconnected graph without any edge information between entities. Since connectivity and information density contribute to the potential of KG embedding approaches to learning meaningful relations, we will compare these two properties for $\mathcal{K}$.

Once the overall network topology is established, we look into local network patterns and try to identify triples that are prone to defined bias types. In our work, we consider the two bias types *Test Leakage Bias* and *Sample Selection Bias*. For every bias type, different metrics with varying thresholds can be defined to determine if a triple is prone to a particular bias pattern. For example, to determine if a triple might be leaked in the test set due to a near-duplicate relation, the bias metric of *near-duplicate relations* is introduced for the bias type *test leakage*. Finally, each metric is transformed into a query to retrieve information on how affected $\mathcal{G}$ is concerning different defined bias patterns. It is important to recognize that our proposed framework does not operate within the limitations of the biases we have specified. This implies that the following measures proposed by the framework are flexible and capable of being augmented or extended beyond their current boundaries.

4.3.1 General Network Analysis

In this section, we explore commonly used measures and techniques used in network analysis that can act as indicators for the density and connectivity of the network graphs formed by triples in the datasets.

Connectivity

A graph is said to be connected if there is a path of edges between any pair of nodes in the graph, else we call the graph disconnected. In literature, the connectivity of a graph refers to the minimum number of nodes or edges that if removed disconnect the graph [58]. The following concepts can provide estimates of how well-connected a graph is.

Connected components. The connectedness of a graph is a necessary condition for connectivity. If a graph $G = (V, E)$ is not connected, we can partition its node set V into partitions $V_1, V_2, ..., V_n$ such that the nodes in each partition form a subgraph in which no node of V_i is connected to another node in V_j with $i \neq j$ [58]. These subgraphs are also known as *connected components*. In our analysis for connectivity, we consider every edge between nodes as undirected to determine which subset of nodes are completely disconnected and do not qualify as useful information for LP models. In other words, we consider weakly connected components in a directed graph for determining the property of connectedness. It is also worth noting that in a single connected component every node u is reachable to every other node v, meaning a path of edges exists between u and v.

Community structure. In graph theory, communities describe subsets of nodes that are more densely connected to each other than to nodes of another subset [19]. For extracting the community structure of a graph, we use the *Louvain Community Detection Algorithm*, which optimizes detected communities for modularity. Modularity is a scalar value between -1 and 1 that measures the density of connections within a community compared to the connections it shares with other communities [8]. In contrast to components, communities can share connections with other subsets of nodes, meaning that a connected component can consist of more than one community. As a consequence, a graph tends to have more communities than connected components, allowing for a more fine-grained perspective on the information-dense subgraphs that provide the playground for learning useful patterns for link prediction.

PageRank. The PageRank Algorithm was originally designed by Page et. al to rank web pages based on their importance in the internet's link structure [37]. The simplified PageRank PR for a given page u can be recursively expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}, \tag{11}$$

where the PageRank value for u is dependent on the PageRank value of every page v that it receives links from divided by the number of outgoing links $L(v)$ from page v. The idea is to give pages with more incoming links and better link quality higher PageRank values. Although PageRank was used in the context of search engines, its broader application in network analysis can shed light on the connectivity of a graph. For example, networks can show differences in the number of important nodes compared to insignificant ones with fewer links. In the task of LP, the PageRank values of two nodes can also be used intentionally or unintentionally as a heuristic to determine if a link exists between them or not, as more important nodes are more likely to connect with other nodes.

Density vs Sparsity

As opposed to a dense graph where large portions of nodes share edges with many other nodes, a graph is considered sparse if the number of edges it contains is low compared to the maximum number of edges (considering an unweighted graph where every node shares at most one edge with another node). We define the degree of a node as well as the frequency and diversity of edges to determine the information density for nodes and relations respectively. The metrics can indicate a lack of information, a structural bias potentially affecting the LP models that learn from such graphs.

Node Degree. The degree of a node refers to the number of edges it has and can be split up into in-degree (number of incoming links) and out-degree (number of outgoing links), depending on if an entity is seen in the tail of a triple or the head. A sparse graph will usually have a lot of low-degree nodes while more dense graphs will have a lot of high-degree nodes. Analyzing the degree distribution of the graph can provide insights into what entities might be overrepresented as they carry more information than others. We will use the average and median degrees of entities as one measure for node information density.

Relation Frequency and Diversity. Different relationship types are mentioned more or less frequently in triples. To make sense of what relationship types get the most mentions in the triples of a dataset, we consider a relation's frequency RF as the number of mentions in fact triples for a given relation. Likewise, we consider the diversity of relationship types RD for a given entity, that is, the number of distinct relationships where the entity either occurs in the head or the tail of a triple.

4.3.2 Test Leakage Bias

In the field of ML, leakage arises when the model at hand uses information in the training process that would not have been anticipated during prediction time, which often leads to overfitting behavior and an overestimation of the model's capability

[26]. Test leakage, in particular, occurs when information in the training set unexpectedly leaks into the test set, causing biased evaluations of a model's performance. The following connectivity patterns commonly found in KGs enable such test leakage bias due to their nature of propagating redundant information.

Near-duplicate Relations

Two relations are considered duplicates of one another if they connect the same pairs of head and tail entities. We consider a relation r to be a near-duplicate of another relation s based on their similarity calculated by the Jaccard index J of the two relations if:

$$J(r,s) = \frac{|R \cap S|}{|R \cup S|} = \frac{|R \cap S|}{|R| + |S| - |R \cap S|} > 0.5, \tag{12}$$

where $R = \{\langle h,r,t\rangle : \langle h,r,t\rangle \in \mathcal{G}_{train}\}$, $S = \{\langle h,s,t\rangle : \langle h,s,t\rangle \in \mathcal{G}_{train}\}$ and $r \neq s$.

Note that $J(r,s) = J(s,r)$.

Near-inverse Relations

Likewise, a relation r can be an inverse of another relation s if r sees the same entity pairs as s but with reversed head and tail placement, e.g. the relation *isParentOf* is an inverse of *isChildOf*. We consider r a near-inverse of a relation s analogously based on its inverse Jaccard index J' if:

$$J'(r,s) = \frac{|R \cap S'|}{|R \cup S'|} = \frac{|R \cap S'|}{|R| + |R'| - |R \cap S'|} > 0.5, \tag{13}$$

where $R = \{\langle h,r,t\rangle : \langle h,r,t\rangle \in \mathcal{G}_{train}\}$, $S' = \{\langle t,s,h\rangle : \langle h,s,t\rangle \in \mathcal{G}_{train}\}$ and $r \neq s$.

Near-symmetric Relations

A symmetric relation is a relation r between two entities where if entity a is related to entity b by r, then entity b is also connected to entity a by the same relation r. For example, in a KG representing people and their family relationships, the *isSiblingOf* relation is a symmetric relation. If Andrew is a sibling of Tristan, then Tristan is also a sibling of Andrew. Formally, we consider r a near-symmetric relation if:

$$\frac{|\{\langle h,r,t\rangle : \langle h,r,t\rangle \in \mathcal{G}_{train} \wedge \langle t,r,h\rangle \in \mathcal{G}_{train}\}|}{|\{\langle h,r,t\rangle : \langle h,r,t\rangle \in \mathcal{G}_{train}\}|} > 0.75. \tag{14}$$

4.3.3 Sample Selection Bias

Sample selection bias [21] occurs when the training data is imbalanced in the sense that its distribution of information cannot be assumed for future predictions. In case of erroneous assumptions due to a skew in the distribution of entities, relations or specific combinations between them, LP models may be prone to overfitting behavior [46] and, similar to test leakage behavior, cause an overestimation of its performance when trained on a benchmark dataset.

High-degree Entities

High-degree entities are entities that have a relatively high number of mentions in either the head or the tail of triples compared to other entities. It is naturally expected that not all entities have the same degree. However, the existence of high-degree entities can become problematic in the evaluation step. If a model optimizes plausibility scores towards a few high-degree entities and those entities are also overrepresented in the test set, then it becomes easy for the LP model to reach good performance scores, as high-degree entities become go-to answers to test questions and low-degree entities are neglected.

Relation-specific Imbalances

Rossi et. al define three types of *Sample Selection Bias* in their bias study of LP datasets [45], which we label and present in the following. Given a tail prediction $\langle h, r, t \rangle$, the different bias metrics, aim to highlight relation-specific overrepresentation in various forms, that may influence $\phi(\langle h, r, t \rangle)$. Although the presented biases are explained from the perspective of a tail prediction, they can be defined for head predictions analogously [45].

Overrepresented answers of a relation (Type 1 Bias). Tail predictions that are prone to *Type 1 Bias* feature a relation r where t is an overrepresented entity in the tail. For instance, if a dataset has disproportionally more instances of the form $\langle \cdot, hasGender, male \rangle$ than $\langle \cdot, hasGender, female \rangle$, the tail *male* would be considered overrepresented for the relation *hasGender*. In network theory terms, we are looking at nodes with a disproportionate high relation-specific in-degree (or out-degree for head predictions). The authors define a threshold for overrepresented answers of 0.75, which we adjust slightly to $\tau_1 = 0.5$ for our experiments. Formally, a triple

$\langle h, r, t\rangle$ in the training set $\mathcal{G}_{train}$ is considered prone to *Type 1 Bias* if:

$$\exists\ t_{over} : \frac{|\{\langle e, r, t_{over}\rangle : \langle e, r, t_{over}\rangle \in \mathcal{G}_{train}\}|}{|\{\langle e_h, r, e_t\rangle : \langle e_h, r, e_t\rangle \in \mathcal{G}_{train}\}|} > \tau_1 = 0.5 \tag{15}$$

Default answers in multi-relationships (Type 2 Bias). Tail predictions that are prone to *Type 2 Bias* feature a multi-relation r that tends to have the default answer t in the tail. A common example, given by Rossi et al., that can induce such bias is the one-to-many relationship *speaksLanguage* [45]. If different persons p speak English among other languages, the tail entity *English* is considered a default answer for the incomplete triple $\langle p, speaksLanguage, ?\rangle$, which can lead to the assumption that every newly seen person in the validation dataset speaks English too. The difference to *Type 1 Bias* lies in the subtlety that r can have several tail entities besides t that do not affect the decision process of classifying a tail prediction as *Type 2* bias-prone or not. The authors define a threshold for r having default tails across different head entities of $\tau_1 = 0.5$, which will be reused for our experiments. Formally, a triple $\langle h, r, t\rangle$ in the training set $\mathcal{G}_{train}$ containing entities $\mathcal{E}$ is considered prone to *Type 2 Bias* if:

$$\exists\ t_{def} : \frac{|\{e : e \in \mathcal{E}, \langle e, r, t_{def}\rangle \in \mathcal{G}_{train}\}|}{|\{e_h : e_h, e_t \in \mathcal{E}, \langle e_h, r, e_t\rangle \in \mathcal{G}_{train}\}|} > \tau_2 = 0.5 \tag{16}$$

Problem of false duplicates (Type 3 Bias). In the previous case, we have shown how duplicate and near-duplicate relations can contribute to test leakage if the relations are indeed duplicates of one another and thus contain redundant information that can be used to successfully infer test triples. We now look at near-duplicates from the perspective of potential false duplicate relations, meaning that similarity may be mistakenly assumed by the LP model. Tail predictions that are prone to *Type 3 Bias* feature a relation r that resembles a similar relation s in the sense that s connects the same entities as r. For example, if the training dataset $\mathcal{G}_{train}$ features persons that live in Germany and were also born there, the LP system might assume that the relation *bornIn* and *livesIn* are identical, and might not consider people living in Germany that were born outside of Germany. The authors define a threshold for r sharing the same head and tail entities as s of $\tau_3 = 0.5$. Formally, a triple $\langle h, r, t\rangle$ in the training set $\mathcal{G}_{train}$ is considered prone to *Type 3 Bias* by the authors if:

$$\exists\ s \neq r : \frac{|\{\langle e_h, s, e_t\rangle : \langle e_h, r, e_t\rangle \in \mathcal{G}_{train}\}|}{|\{\langle e_h, r, e_t\rangle : \langle e_h, r, e_t\rangle \in \mathcal{G}_{train}\}|} > \tau_3 = 0.5 \tag{17}$$

4.4 Prediction Analysis

Figure 4.1B shows the steps necessary to deconstruct if the bias in the input KG has been propagated into the trained KG embeddings of a model. First, correct predictions in $\mathcal{G}_{test}$ are identified. This could be achieved using by filtering test predictions for H@K settings or any other evaluation metric. Finally, the correct predictions are bucketed into the different bias pattern types or none - if the corresponding triple is not prone to any bias according to our defined metrics. A potential change in the distribution of bias types from the input to the output data can then be used as an indicator for the proneness of a model to certain bias types.

Chapter 5

Implementation

This chapter lists implementation details for our conducted experiments. The aim of the experiments is to follow along our framework defined in Chapter 4 and characterize LP benchmark datasets.

5.1 Framework Implementation

The underlying data architecture of our experiments for characterizing LP benchmarks is demonstrated in Figure 5.1. Following our framework for characterizing LP datasets, the implementation is split into two parts: *Descriptive Analysis* and *Prediction Analysis*. We first need to convert the original benchmark dataset files given as text files into a standardized CSV format. For the network analysis part in the descriptive analysis, visualizations are produced using the network exploration software Gephi [6], and network metrics are calculated with the Python package NetworkX [20]. To analyze how the benchmark datasets are affected by previously defined bias patterns, an RDF graph is created as an intermediate step to enable the use of the querying language SPARQL for our analysis. At first, RDF triples are generated using the RML interpreter SDM-RDFizer [23], taking our CSV dataset files and specified mapping files as input, and producing the resulting RDF graphs in form of N-Triples files. After uploading the RDF graphs to a semantic graph database like GraphDB, we can execute SPARQL queries against the KGs to retrieve bias-affected triples, relations and/or entities. Subsequently, the python library AmpliGraph [12] is utilized to learn the embeddings for the integrated datasets FB15k, FB15k-237, WN18 and WN18RR, and to observe what bias patterns could have caused test triples to get predicted correctly. This final step constitutes the prediction analysis according to our approach.

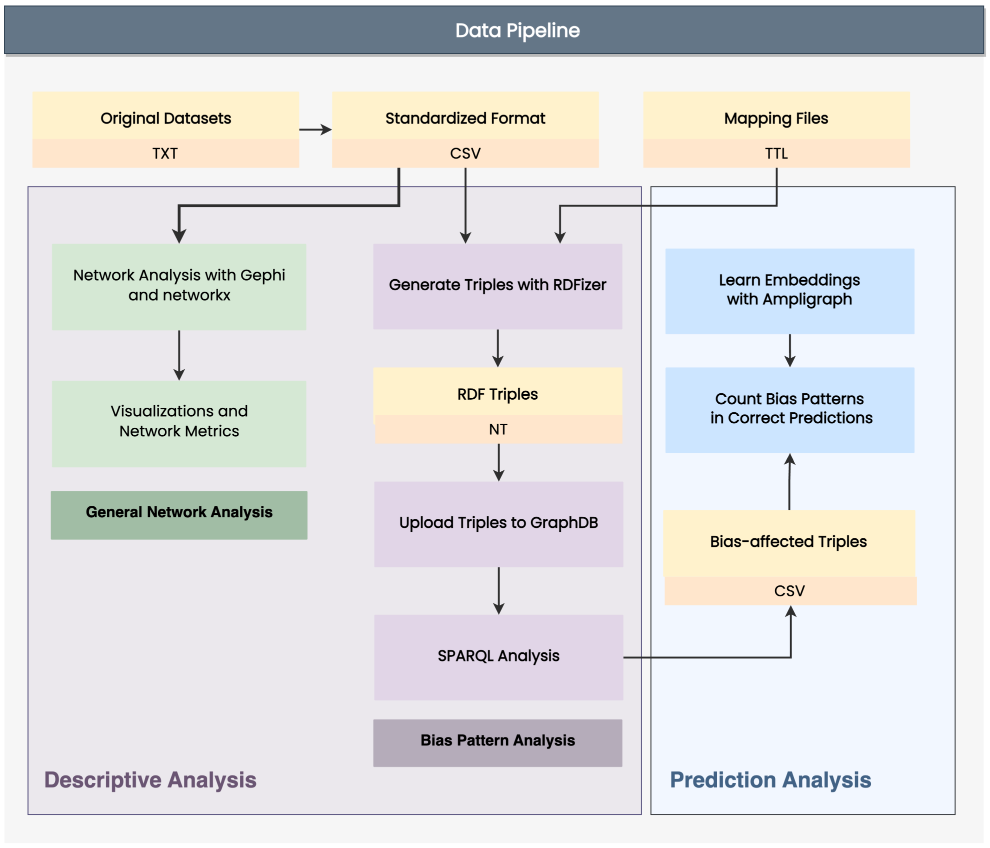


Figure 5.1: Framework Implementation. The illustration shows how the data in the original benchmark datasets is transformed (1) to produce network visualizations and network metrics, (2) to analyze the dataset for defined bias patterns using SPARQL queries and (3) to quantify the occurrence of bias patterns in correct predictions.

5.2 Network Analysis Procedure

To produce the visual representations of the graph networks derived from the benchmark datasets, a number of configuration steps were performed on Gephi. Firstly, the OpenOrd layout algorithm [34] was chosen due to its ability to efficiently handle large-scale networks and to distinguish clusters of entities in the datasets. Additionally, entity nodes were colored according to the communities they belong to, allowing for easier identification and analysis of patterns within the network. Finally, the size of nodes was adjusted based on their degree, thereby highlighting the nodes with the most connections and emphasizing their importance within the network.

To obtain further insights into the properties of the graph networks, network metrics were calculated using the in-built functions of NetworkX. To characterize the overall connectivity of a network, the number of communities and connected components as well as their average and median size is determined respectively. In addition to that, the PageRank and degree of every entity node are calculated using a directed version of the network graph.

5.3 Creating the RDF Graph

Once the benchmark datasets are given as CSV files with columns for each head entity, relation and object entity, creating the RDF graph is a straightforward process. Listing 5.1 shows the RML mapping file needed that transforms the triples in the CSV files to RDF triples in an N-Triples file. The triples map is named "DatasetMapping" and consists of the logical source (dataset as CSV), a subject map, taking the head column and mapping it to an entity URI, and a predicate object map, which is responsible for transforming the relation and tail entities to their URIs respectively. The mapping file is interpreted with version 4.0 of the SDM-RDFizer, as it is optimized to work for large-scale data and provides many configuration options, e.g. the option to remove duplicate triples. The produced N-Triples files are uploaded to an instance of GraphDB with version 9.8.1. All benchmark datasets are considered in each of their training, test and validation split as well as all of them combined, resulting in $7 * 4 = 28$ total graphs.

Listing 5.1: RML Mapping file

```
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .



<DatasetMapping>
  a rr:TriplesMap;

  rml:logicalSource [
    rml:source "<FILE_PATH_TO_DATA_FILE>";
    rml:referenceFormulation ql:CSV
  ];

  rr:subjectMap [
    rr:template "http://bias.org/entity/{head}";
  ];

  rr:predicateObjectMap [
    rr:predicateMap [
      rr:template "http://bias.org/vocab/{relation}";
    ];
    rr:objectMap [
        rr:template "http://bias.org/entity/{tail}";
      ]
   ].
```

5.4 SPARQL Analysis

In this section, we present questions addressing general statistics, network characteristics and structural bias patterns of a benchmark dataset and their corresponding answers as SPARQL queries.

How many triples are there across different splits?

```
SELECT ?tripleCountTotal ?tripleCountTraining ?tripleCountValidation ?tripleCountTest
WHERE {
     GRAPH <http://www.ontotext.com/explicit> {
       SELECT (COUNT(*) AS ?tripleCountTotal) WHERE { ?s ?p ?o}
     }
     GRAPH <http://bias.org/training-graph> {
       SELECT (COUNT(*) AS ?tripleCountTraining) WHERE { ?s ?p ?o}
     }
     GRAPH <http://bias.org/validation-graph> {
       SELECT (COUNT(*) AS ?tripleCountValidation) WHERE { ?s ?p ?o}
     }
     GRAPH <http://bias.org/test-graph> {
       SELECT (COUNT(*) AS ?tripleCountTest) WHERE { ?s ?p ?o}
     }
}
```

How many entities are there across different splits?

```
SELECT ?entitiesTotal ?entitiesTrain ?entitiesValidation ?entitiesTest WHERE {
  GRAPH <http://www.ontotext.com/explicit> {
    SELECT (COUNT(DISTINCT ?entity) AS ?entitiesTotal) WHERE {
      { ?entity ?p1 ?o1. }
      UNION
      { ?s2 ?p2 ?entity. }
    }
  }
  GRAPH <http://bias.org/training-graph> {
    SELECT (COUNT(DISTINCT ?entity) AS ?entitiesTrain) WHERE {
      { ?entity ?p1 ?o1. }
      UNION
      { ?s2 ?p2 ?entity. }
    }
  }
  GRAPH <http://bias.org/test-graph> {
    SELECT (COUNT(DISTINCT ?entity) AS ?entitiesTest) WHERE {
      { ?entity ?p1 ?o1. }
      UNION
      { ?s2 ?p2 ?entity. }
    }
  }
  GRAPH <http://bias.org/validation-graph> {
    SELECT (COUNT(DISTINCT ?entity) AS ?entitiesValidation) WHERE {
      { ?entity ?p1 ?o1. }
      UNION
      { ?s2 ?p2 ?entity. }
    }
  }
}
```

How many relations are there across different splits?

```
SELECT ?relationsTotal ?relationsTrain ?relationsValidation ?relationsTest WHERE {
  GRAPH <http://www.ontotext.com/explicit> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsTotal) WHERE { ?s ?p ?o. }
  }
  GRAPH <http://bias.org/training-graph> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsTrain) WHERE { ?s ?p ?o. }
  }
  GRAPH <http://bias.org/test-graph> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsValidation) WHERE { ?s ?p ?o. }
  }
  GRAPH <http://bias.org/validation-graph> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsTest) WHERE { ?s ?p ?o. }
  }
}
```

How many relations are there across different splits?

```
SELECT ?relationsTotal ?relationsTrain ?relationsValidation ?relationsTest WHERE {
  GRAPH <http://www.ontotext.com/explicit> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsTotal) WHERE { ?s ?p ?o. }
  }
  GRAPH <http://bias.org/training-graph> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsTrain) WHERE { ?s ?p ?o. }
  }
  GRAPH <http://bias.org/test-graph> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsValidation) WHERE { ?s ?p ?o. }
  }
  GRAPH <http://bias.org/validation-graph> {
    SELECT (COUNT(DISTINCT ?p) AS ?relationsTest) WHERE { ?s ?p ?o. }
  }
}
```

What is the in-degree of every tail entity?

```
SELECT ?entity (COUNT(*) AS ?inDegree) WHERE {
  GRAPH <http://www.ontotext.com/explicit> { { ?s1 ?p1 ?entity. } }
}
GROUP BY ?entity
ORDER BY DESC (?inDegree)
```

What is the out-degree of every head entity?

```
SELECT ?entity (COUNT(*) AS ?outDegree) WHERE {
  GRAPH <http://www.ontotext.com/explicit> { { ?entity ?p1 ?o1. } }
}
GROUP BY ?entity
ORDER BY DESC (?outDegree)
```

What is the combined degree of every entity? (in + out-degree)

```
SELECT ?entity (COUNT(*) AS ?degree) WHERE {
  GRAPH <http://www.ontotext.com/explicit> {
    { ?entity ?p1 ?o1. }
    UNION
    {
      ?s2 ?p2 ?entity.
      FILTER(?s2 != ?entity)
    }
  }
}
GROUP BY ?entity
ORDER BY DESC (?degree)
```

How many different relation types does each entity have? (Relation diversity)

```
SELECT ?entity (COUNT(*) AS ?relationDiversity) WHERE {
  GRAPH <http://www.ontotext.com/explicit> {
        { SELECT distinct ?entity ?p1 WHERE {?entity ?p1 ?o1.} }
    UNION
    {
      { SELECT distinct ?entity ?p1 WHERE {?s2 ?p1 ?entity.} }
      FILTER(?s2 != ?entity)
    }
  }
}
GROUP BY ?entity
ORDER BY DESC (?relationDiversity)
```

How many facts (edges) exist for every relation? (Relation frequency)

```
SELECT ?relation (COUNT(*) AS ?relationFrequency) WHERE {
  GRAPH <http://www.ontotext.com/explicit> {
        ?s ?relation ?o
  }
}
GROUP BY ?relation
ORDER BY DESC (?relationFrequency)
```

—

What are near-duplicate relations r, their duplicate counterpart relation s and their Jaccard index?

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?r (?s AS ?similarRelation) ?tripleCountR ?tripleCountS
((COUNT(*)) / (xsd:float(?tripleCountR + ?tripleCountS - COUNT(*))) AS ?jaccardIndex)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?r (COUNT(*) AS ?tripleCountR) WHERE { ?head ?r ?tail. }
      GROUP BY ?r
    }
    {
      SELECT ?s (COUNT(*) AS ?tripleCountS) WHERE { ?head ?s ?tail. }
      GROUP BY ?s
    }
        {
            SELECT ?h ?r ?s ?t WHERE {
                ?h ?r ?t;
                  ?s ?t.
                FILTER(?s != ?r)
            }
        }

  }
}
GROUP BY ?r ?s ?tripleCountR ?tripleCountS
HAVING((COUNT(*)) / (xsd:float(?tripleCountR + ?tripleCountS - COUNT(*))) > 0.5)
ORDER BY DESC(?jaccardIndex)
```

What are near-inverse relations r, their inverse counterpart relation s and their inverse Jaccard index?

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?r (?s AS ?inverseRelation) ?tripleCountR ?tripleCountS
((COUNT(*)) / (xsd:float(?tripleCountR + ?tripleCountS - COUNT(*))) AS ?inverseJaccardIndex)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?r (COUNT(*) AS ?tripleCountR) WHERE { ?head ?r ?tail. }
      GROUP BY ?r
    }
    {
      SELECT ?s (COUNT(*) AS ?tripleCountS) WHERE { ?head ?s ?tail. }
      GROUP BY ?s
    }
        {
            SELECT ?h ?r ?s ?t WHERE {
                ?h ?r ?t.
                ?t ?s ?h.
                FILTER(?s != ?r)
            }
        }

  }
}
GROUP BY ?r ?s ?tripleCountR ?tripleCountS
HAVING((COUNT(*)) / (xsd:float(?tripleCountR + ?tripleCountS - COUNT(*))) > 0.5)
ORDER BY DESC(?inverseJaccardIndex)
```

What are near-symmetric relations and their symmetry score?

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?relation ?tripleCount
((COUNT(*)) / (xsd:float(?tripleCount)) AS ?symmetryScore)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?relation (COUNT(*) AS ?tripleCount) WHERE { ?head ?relation ?tail. }
      GROUP BY ?relation
    }
    ?h ?relation ?t .
    ?t ?relation ?h .
  }
}
GROUP BY ?relation ?tripleCount
HAVING ((COUNT(*)) / (xsd:float(?tripleCount)) > 0.75)
ORDER BY DESC(?symmetryScore)
```

What are overrepresented tail answers of a relation? (> 50% of all tail mentions for a relation)

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?relation ?overrepresentedTail (COUNT(*) AS ?filteredTripleCount) ?tripleCount
(xsd:float((COUNT(*)) / (xsd:float(?tripleCount))) AS ?representation)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?relation (COUNT(*) AS ?tripleCount) WHERE { ?s ?relation ?o. }
      GROUP BY ?relation
    }
    ?head ?relation ?overrepresentedTail.
  }
}
GROUP BY ?relation ?overrepresentedTail ?tripleCount
HAVING (((COUNT(*)) / (xsd:float(?tripleCount))) > "0.5"^^xsd:decimal)
ORDER BY DESC (?filteredTripleCount)
```

What are overrepresented head answers of a relation? (> 50% of all head mentions for a relation)

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?overrepresentedHead ?relation
(COUNT(*) AS ?filteredTripleCount) ?tripleCount
(xsd:float((COUNT(*)) / (xsd:float(?tripleCount))) AS ?representation)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?relation (COUNT(*) AS ?tripleCount) WHERE { ?s ?relation ?o. }
      GROUP BY ?relation
    }
    ?overrepresentedHead ?relation ?tail.
  }
}
GROUP BY ?relation ?overrepresentedHead ?tripleCount
HAVING (((COUNT(*)) / (xsd:float(?tripleCount))) > "0.5"^^xsd:decimal)
ORDER BY DESC (?filteredTripleCount)
```

What are default tail answers of a relation?

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?relation ?option ?headEntities
(COUNT(DISTINCT ?h) AS ?headEntitiesForOption)
((COUNT(DISTINCT ?h)) / (xsd:float(?headEntities)) AS ?optionRepresented)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?relation (COUNT(DISTINCT ?head) AS ?headEntities)
      WHERE { ?head ?relation ?tail. }
      GROUP BY ?relation
    }
    ?h ?relation ?option.
  }
}
GROUP BY ?relation ?option ?headEntities
HAVING (((COUNT(DISTINCT ?h)) / (xsd:float(?headEntities))) > "0.5"^^xsd:decimal)
```

What are default head answers of a relation?

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?relation ?option ?tailEntities
(COUNT(DISTINCT ?t) AS ?tailEntitiesForOption)
((COUNT(DISTINCT ?t)) / (xsd:float(?tailEntities)) AS ?optionRepresented)
WHERE {
  GRAPH <http://bias.org/training-graph> {
    {
      SELECT ?relation (COUNT(DISTINCT ?tail) AS ?tailEntities)
      WHERE { ?head ?relation ?tail . }
      GROUP BY ?relation
    }
    ?option ?relation ?t.
  }
}
GROUP BY ?relation ?option ?tailEntities
HAVING (((COUNT(DISTINCT ?t)) / (xsd:float(?tailEntities))) > "0.5"^^xsd:decimal)
```

5.5 Learning the Embeddings

In this section, we describe the experimental set-up used to learn the TransE embeddings for calculating the rankings needed to explain prediction results. The datasets FB15k, FB15k-237, WN18 and WN18RR were trained using AmpliGraph version 2.0.0 on a Python 3.9 Google Compute Engine backend running on Ubuntu 20.04.5 LTS with an NVIDIA A100-SXM4-40GB. The sensitivity of benchmark dataset performance to tuned hyperparameters is a critical consideration in machine learning models learning LP [25], making the identification of optimal hyperparameters on the validation set a difficult endeavor. Hyperparameters require careful selection, as the configuration space is vast, encompassing numerous parameters like the dimension, learning rate, number of epochs, loss function, batch size and many more. Given the impracticality of determining the optimal parameters through grid search, we reuse the hyperparameters reported in the AmpliGraph documentation page [52] summarized in Table 5.1. The ranking evaluation is performed using the filtered setting (see Section 2.5).

	Dataset			
	FB15k	**FB15k-237**	**WN18**	**WN18RR**
Dimensions	150	400	150	350
Epochs	4000	4000	4000	4000
ETA*	10	30	10	30
Loss Function	multiclass_nll	multiclass_nll	multiclass_nll	multiclass_nll
Optimizer	ADAM	ADAM	ADAM	ADAM
Learning Rate	$5e-5$	0.0001	$5e-5$	0.0001
Regularizer	LP	LP	LP	LP
λ, p	0.0001, 3	0.0001, 2	0.0001, 3	0.0001, 2
Norm	1	1	1	1
Batch count	100	64	100	150

Table 5.1: Hyperparameter settings. (*) = Number of negatives to use during training per triple. λ and p are parameters for the regularizer [52].

Chapter 6

Evaluation

In this chapter, we evaluate our approach by applying our framework and assessing 7 benchmark datasets for LP tasks. The research questions addressed by this thesis are: **RQ1**) What does the topological structure of the benchmark datasets look like? **RQ2**) How can unwanted biases in the benchmarks affect performance results?

6.1 Benchmark Datasets for LP Tasks

The following benchmark datasets are used to assess the performance of an LP model. They are curated from common real-world KGs, such as DBPedia [3], Wikidata [62], Freebase [9], Wordnet [35] and YAGO [54]. For each dataset, different sampling techniques were used, e.g. to prevent test leakage or in general to prevent the phenomenon of predicting redundant facts. Since most benchmark datasets do not feature an inductive version yet, we only include transductive dataset setups in conducted experiments.

FB15k is a dataset created by [11] in 2013. It features facts from the Freebase knowledge base with entities that occur both in Freebase and in the Wikilink dataset [51], but also have at least 100 mentions in Freebase [11]. Further, all explicit inverse relations, annotated by an exclamation mark in Freebase, have been removed. To produce training, validation, and test set a random split has been performed [11]. FB15k contains in total 592,213 triples, 14,591 entities and 1,345 relations [11].

FB15k-237 is a benchmark dataset created by [60] at Microsoft Research in 2015 which was derived from the original FB15k dataset [11] after observing test leakage

			Triples			
	Entities	**Relations**	**Total**	**Training**	**Validation**	**Test**
FB15k	14,951	1,345	592,213	483,142	50,000	59,071
FB15k-237	14,541	237	310,116	272,115	17,535	20,466
WN18	40,943	18	151,442	141,442	5,000	5,000
WN18RR	40,943	11	93,003	86,835	3,034	3,134
Wikidata5M	4,594,485	822	20,624,575	20,614,279	5,163	5,133
YAGO3-10	123,182	37	1,089,040	1,079,040	5,000	5,000
DBpedia50k	30,449	365	43,490	32,203	397	10,890

Table 6.1: General overview of datasets

caused by inverse triples and near-duplicates [60] making it easy to achieve state-of-the-art performance on a simple model based on observed features [11]. First, the FB15k dataset was reduced to only feature the most frequent 401 relations. After that, relations that represent near-duplicates or near-inverses of other relations were removed by comparing how many head and tail entities match respectively. Additionally, all entity pairs that occur in the training set were removed from the test and validation set [11]. The dataset contains a total of 14,505 entities, 237 relations and 310,079 triples.

WN18 is a benchmark dataset referenced from the work of [10] and was first introduced as a benchmark for LP tasks by the same authors of FB15k in the introduction of the TransE model [11]. It was sampled from WordNet, a lexical database of the English language where words are grouped into sets of concepts, so-called "synsets" [35]. These concepts can then relate to other concepts. For instance, the hypernym relation denotes instances of another concept, resulting in a triple like $\langle Hawk, has\_hypernym, Raptor\rangle$. The authors of WN18 [11] did not specify how the dataset was constructed. The resulting dataset has 40,943 entities, 18 relations, and 151,442 triples.

WN18RR is a benchmark dataset created by [13] after observing that the predecessor dataset WN18 also suffers from inverse relations and test leakage [13]. Dettmers specifies seven predicates, e.g. the inverse relation *hyponym* or *holonym member*, to be filtered from WN18. The curation of WN18RR can be comprehended in detail in a Python script in the author's repository[1]. The resulting dataset has 40,943 entities, 11 relations, and 93,003 triples.

Wikidata5M was first introduced in 2021 by [63] and is a large-scale benchmark dataset that was built on top of Wikidata [62]. It was curated by filtering out all entities that do not have a Wikipedia article or if the article's description in the first section of a Wikipedia page has less than five words [63]. Further, the descriptions extracted from the first section have been also aligned with the entities to enrich the dataset [63]. The Wikidata5M dataset comes in both transductive and inductive settings and contains 4,594,485 entities, 822 relations, and 20,614,279 total triples.

YAGO3-10 is a benchmark dataset derived from YAGO3, an updated version of the YAGO KG [54] which is based on Wordnet and Wikidata. YAGO3 [33] was built to enable multilingual support but also to extend its predecessor by adding new facts through Wikipedia articles and infoboxes from other languages [54]. YAGO3-10 was first presented in 2018 and constructed by only including entities in the dataset that contain more than 10 relation types [13]. The resulting dataset contains 123,182 entities, 37 relations, and 1,179,040 triples.

DBpedia50k is a benchmark dataset created by [49] and published in 2017. It was randomly sampled from DBPedia [49], a community project first introduced in 2007 aiming to extract structured information from Wikipedia articles, as they can be found in Wikipedia's infoboxes [3]. DBPedia50k contains 30,449 entities and 37 relations distributed over 32,388 triples in the training split, 399 in the validation split, and 10,969 triples in the test split [49]. We further observed duplicate triples across the different data splits explaining why the total unique triple count is different from what the authors report.

6.2 Topological Analysis

The goal of the topological analysis is to characterize every benchmark dataset when looking at their network topology holistically and - if possible - to work out key differences between them.

[1] `https://github.com/TimDettmers/ConvE`

	Number of		**Component size**		**Community size**	
	Comp.	**Commun.**	**Average**	**Median**	**Average**	**Median**
FB15k	9	19	1,661.2	2	786.9	104
FB15k-237	6	16	2,423.5	2.5	908.8	710
WN18	13	64	3,149.5	2	639.7	518
WN18RR	13	64	3,149.5	2	639.7	508
Wikidata5M	1,491	1,633	3,081.5	2	2,813.5	2
YAGO3-10	11	43	11,198.4	3	2,864.7	1881
DBpedia50k	2,032	2,098	14.98	2	14.51	2

Table 6.2: Components and communities statistics. The first two columns show the number of components and communities. Columns 3-6 show their sizes.

6.2.1 Connectivity

Overall statistics concerning components and communities are reported in Table 6.2. Figure 6.1 provides a deeper insight into the distribution of component sizes. Figure 6.2 contains network visualizations for every network.

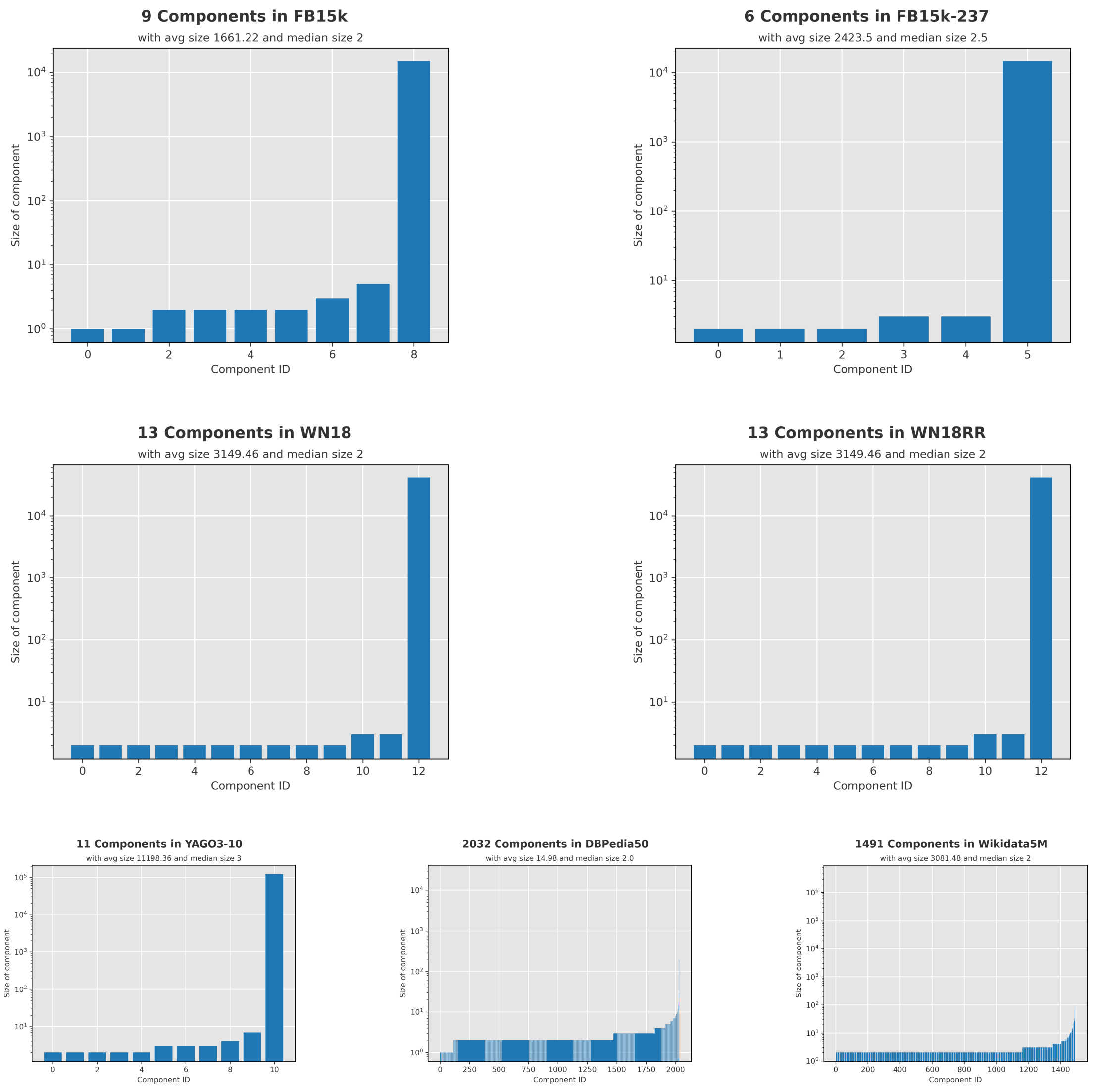
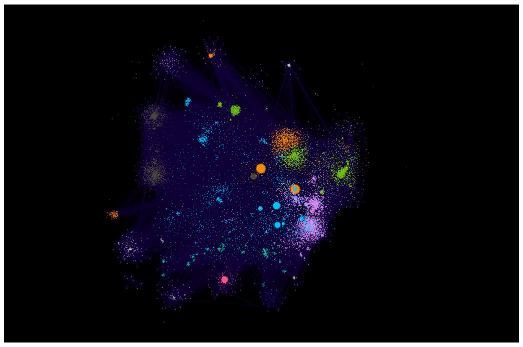
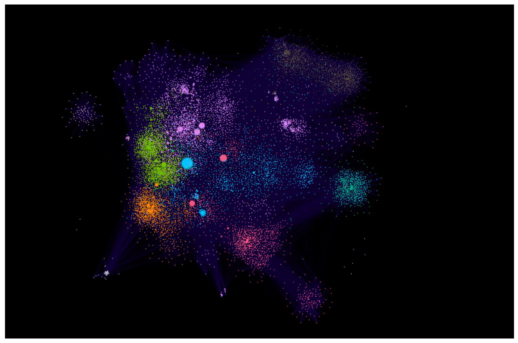
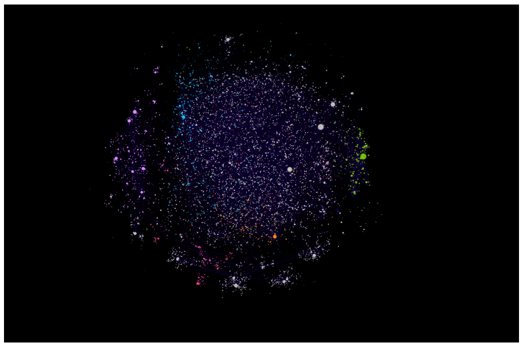


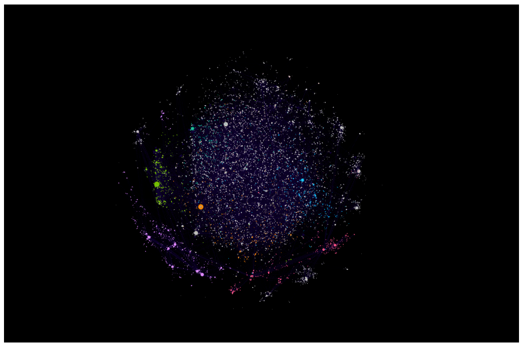
Figure 6.1: Component size distributions. Every plot shows all components in the network and their respective size. In all datasets - except for DBpedia50k - the majority of entities lie within a single main component.

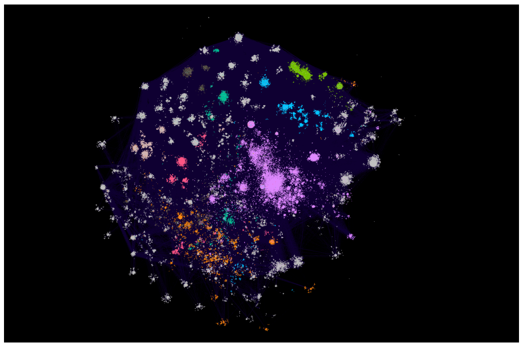
First, it can be observed that all underlying networks - with the exception of DBpedia50k - are high in terms of connectedness, in the sense that most entity nodes are contained within one component (see Figure 6.1), where they are all reachable from one another through a path of edges. Most components tend to feature only a few nodes, isolated from the rest of the network, while the biggest component tends to contain the majority of entities. This skew in the component size distributions explains the large mismatch between average and median component size. This behavior can also be observed in the network visualizations in Figure 6.2. Only DBpedia50k shows a large number of isolated node entities, whereas other networks seem more well-connected. In DBpedia50k the average component size is quite low with only 14.98 entities per component on average. The high connectivity of FB15k, FB15k-237 and YAGO3-10 is especially visible in Figure 6.2, as the purple-colored edges are arranged very densely. This pattern can be traced back to the formation of the datasets, as FB15k, FBk-237 and YAGO3-10 were filtered for entities with a high frequency of mentions, and therefore show more connections between entities.


(a) FB15k


(b) FB15k-237


(c) WN18


(d) WN18RR


(e) YAGO3-10

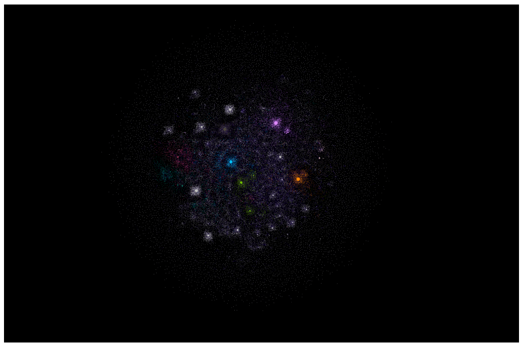

(f) DBpedia50k

Figure 6.2: Network visualizations. The network visualization of every benchmark dataset using the network analysis tool Gephi [6] using the OpenOrd layout [34]. The entity nodes are colored and positioned based on the community they belong to. We can see similar connectivity behavior between FB15k and FB15k-237 as well as between WN18 and WN18RR. DBpedia50k due to its many unconnected islands.

Second, a look into the communities' statistics reveals how well-connected the components are. Wikidata5M and DBpedia50k are amongst the datasets with the most well-connected components with 1.1 and 1.0 communities per component each respectively. This is to be expected since most components in these datasets consist of just two entities, which cannot be further divided into communities. It is interesting to see that there is almost no tangible difference in component and community structure between WN18 and its filtered version WN18RR. In the creation of FB15k-237, however, the number of components is reduced to six. This behavior, too, can be explained by the sampling technique that was used to construct it. By filtering the dataset for the most frequent 401 relations, some entities, especially those in small-sized components, might get eliminated. All the other datasets have around two to four communities per component, meaning that two to four clusters exist per component that are relatively well-connected on the inside and loosely connected to the other clusters.

6.2.2 Node and Relation Density

	Average network metric				
	Degree	**Clustering Coeff.**	**PageRank**	**RF**	**RD**
FB15k	62.5	0.200	0.067	440.3	786.9
FB15k-237	39.0	0.156	0.069	1,308.5	908.8
WN18	7.4	0.067	0.024	8,413.4	639.7
WN18RR	4.5	0.047	0.024	8,454.8	639.7
Wikidata5M	9.0	0.014	0.356e-3	25,090.7	2,813.5
YAGO3-10	13.0	0.073	0.008	29,433.5	2,864.7
DBpedia50k	2.6	0.049	0.033	119.2	14.51

Table 6.3: Network metrics. The columns display the average degree, clustering coefficient, PageRank, RF and RD. The PageRank value is given in 10^{-3}.

The network metrics provided in Table 6.3 give an estimation of the density (or sparsity) of nodes and relations in the network graphs. Figure 6.3 shows the distribution of entity nodes by degree while Figure 6.3 shows the percentage of mentions that each relation has. From the entity degree distribution plots, it can be inferred that all benchmark datasets show a distortion towards a few high-degree entities and a large number of low-degree entities. This dynamic is known as the power law, which is not confined to the KGs at hand and is a pattern that can be found in many large networks [5]. FB15k shows the highest node density with an average degree of 62.5, second is, FB15k-237 with 39.0 and third YAGO3-10 with 13.0. This matches again the previously mentioned sampling technique, where these three datasets are known to have a large number of entity mentions. WN18, WN18RR and Wikidata5M show quite sparse nodes with an average degree of each 7.4, 4.5 and 9.0. The sparsity in node information, however, is compensated by information-dense relations with average RFs for these datasets ranging from 8k to 25k per relation. DBPedia50k suffers from both node and relation sparsity. The highest relation density can be found in YAGO3-10 with an average RF of 29,433.5. It is also worth noting that the high relational information density in WN18, WN18RR and YAGO3-10 is caused by only up to three relationship types that make up more than 70% of all triples in each of the datasets.

6.2.3 Discussion of Topological Results

All entities in researched benchmark datasets of this thesis appear to be well-connected (with the exception of DBpedia50k). This high connectivity can be deemed as a critical requirement for successful link predictions. In addition to that, all datasets - again apart from DBpedia50k - show either high relation density or high entity density. We are able to explain the high node density for FB15k, FB15k-237 and YAGO3-10 with the technique that was used to construct them. The indirect filtering of low-information entities and relations can be seen as a biased approach in the construction process and is a different form of sample selection bias. RQ1 is therefore answered regarding the topological traits of connectivity and information density.

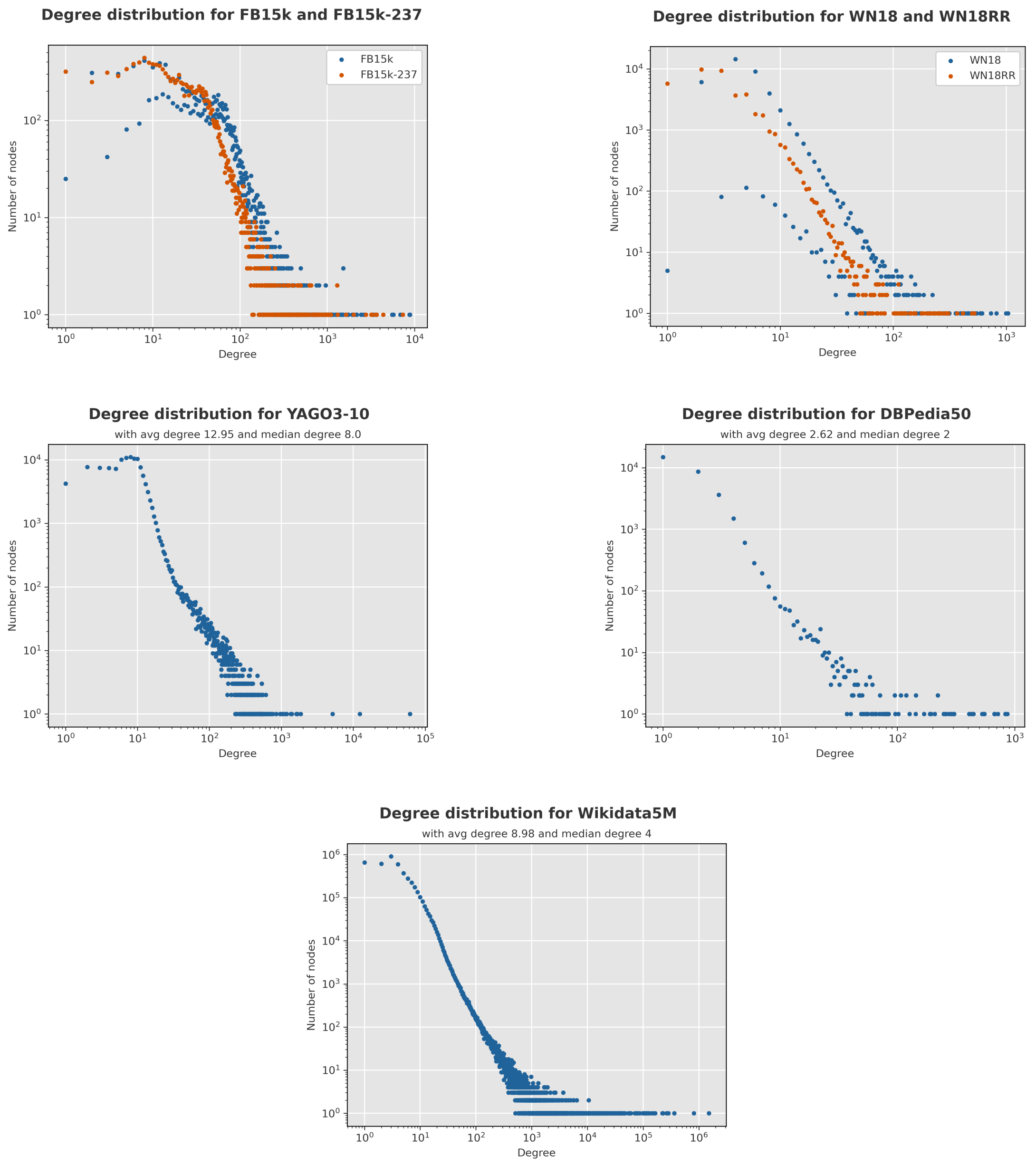


Figure 6.3: Distributions of entity degrees. All datasets are distorted towards fewer high-degree entities and a large number of low-degree entities.

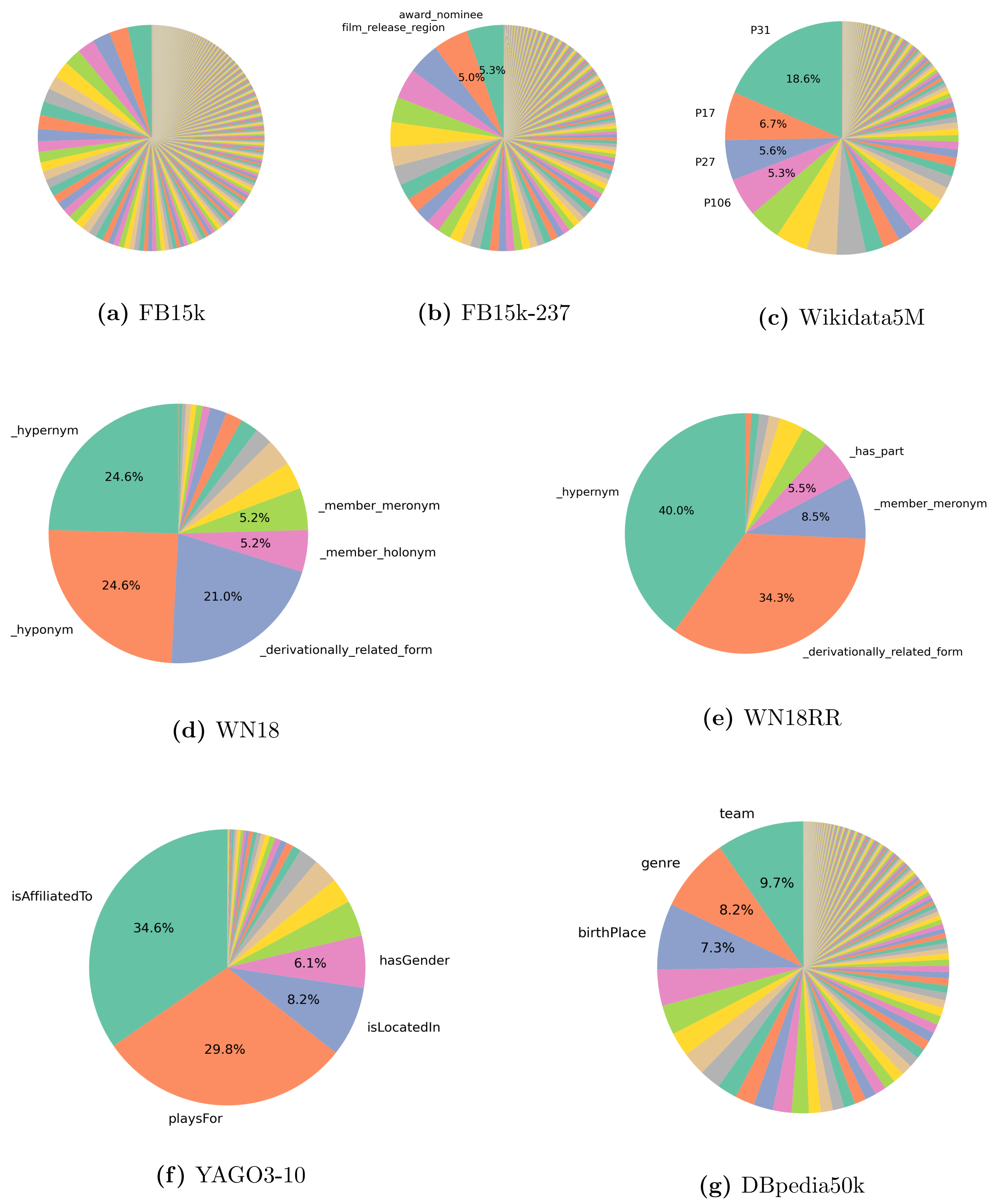


Figure 6.3: Distributions of relations and their frequency. We can identify dense relations with many mentions in the WN18 datasets and YAGO3-10.

6.3 Bias Analysis

In this section, we want to analyze how previously defined bias patterns concerning test leakage and sample selection are existent in the training data that affect triples in $\mathcal{G}_{test}, \mathcal{G}_{valid} or \mathcal{G}$. In the final step, we aim to use that evidence to (partially) explain prediction results.

6.3.1 Bias Patterns in Training Data

The results of the exploratory analysis are reported in Figure 6.4, showing what percentage of triples in the different dataset splits can be attributed to a particular bias type defined in Chapter 4. In line with reports in the literature, FB15k and WN18 both suffer from test leakage through near-inverse relations (83.9% of all test triples in FB15k and 76.6% in WN18). All other datasets have a rather low number of near-inverse or none at all. Another notable instance of test leakage can be seen in YAGO3-10, where over 63.3% of test triples are prone to near-duplicate relations. FB15k also contains near-duplicate relations but to a much lesser extent (15.0%). We can clearly see how near-inverse and near-duplicate relations have been almost completely removed from the test and validation set in FB15k to create FB15k-237, albeit still existent in the training data. A similar discrepancy between training and test/validation split can be seen for near-symmetric relations in FB15k-237. Another interesting change of dynamics is happening in WN18 when it comes to near-symmetric relations. In the original WN18 dataset, test triples are slightly prone to near-duplicate relations at a rate of 22.3%. The filtering process, however, caused a noticeable increase of near-duplicates in the filtered version WN18RR to now 37.0%. Other than near-symmetric relations, WN18RR does not show any proneness to one of the previously defined bias types. The variety of bias types is the biggest in FB15k with default answers being the most prevalent of sample selection biases. In both FB15k and FB15k-237 test triples prone to default tail answers are occurring at a higher rate with 28.8% in FB15k and even 40.3% FB15k-237, the highest occurrence in FB15k-237. The datasets DBpedia50k and Wikidata5M show also some variety in bias patterns but have always less than 10% triples that are affected by them except for default tail answers in Dbpedia50k, which occur at a slightly higher rate of 10.89%. Wikidata5M shows the lowest bias affection of all datasets. All in all, test leakage bias seems to be a more ubiquitous issue than sample selection bias in these datasets.

6.3.2 Explaining Prediction Results

Goal and Evaluation Approach With different bias metrics defined concerning test leakage bias and sample selection bias, we aim to investigate their impact on prediction results for the benchmark datasets FB15k, FB15k-237, WN18 and WN18RR trained on the model TransE. The attempt is to deconstruct all correct predictions in the test dataset to see what bias types - if any - can be attributed to the successful prediction of a test fact. Moreover, we scrutinize correct predictions in various tolerance settings depending on if the ranking of the prediction would contribute to the evaluation metrics H@1, H@3, H@5, and H@10 respectively. Since rankings and their derived metrics are calculated for head and tail entities separately, attributions are accounted for in both settings. For example, let $\langle BeachHouse, isHyponymOf, House\rangle$ be the test triple that we want to match bias types to, $r_h = 4$ be the ranking for *Beach House* and $r_t = 1$ the ranking for *House* in this triple configuration. In this case, the test triple would qualify for all correct tail predictions in H@K metrics but only for H@5 and H@10 in correct head predictions. A correct tail (or head) prediction $\langle h, r, t\rangle$ in the test set can be attributed to one or more of the following categories:

- Inverse bias-prone if r is near-inverse
- Duplicate bias-prone if r is near-duplicate
- Symmetry bias-prone if r is near-symmetric
- Overrepresented answer bias-prone if t (or h) is overrepresented for r
- Default answer bias-prone if t (or h) is a default answer for r
- Unknown bias if none of the above applies

Evaluation The experimental results for attributing bias patterns to correct H@K predictions can be found in Figure 6.5 - Figure 6.8. All bias types occur in buckets with a bias frequency scale from 0 to 1. If a correctly predicted test triple belongs to non of the defined bias types it is listed in the category "unknown". Table 6.4 shows the performances that the benchmarks reach when trained on TransE.

In FB15k and WN18, we can see how the severe issue of near-inverse relations is even more prominent in the prediction results with an increase of 5 to 10 percentage points. This indicates that the property of near-inverse relations are favoring the prediction of correct test triples in all H@K settings. Taking all bias pattern types into consideration, only $\approx 5\%$ of all correct predictions cannot be attributed to any

	H@1	**H@3**	**H@5**	**H@10**	**MRR**
FB15k	0.486	0.724	0.783	0.842	0.622
FB15k-237	0.107	0.293	0.366	0.465	0.232
WN18	0.440	0.875	0.923	0.950	0.661
WN18RR	0.038	0.366	0.470	0.551	0.224

Table 6.4: Performance results. Obtained performances in our experiments with TransE using the metrics H@K and MRR.

bias pattern in FB15k. For WN18, the case is even more extreme, as less than $\approx 1\%$ of correct predictions cannot be explained with our bias types. Almost all correctly predicted test triples in WN18 are either near-inverse or near-symmetric relations. The evidence here suggests that a model like TransE optimizes plausibility scores by first capitalizing on test leakage bias patterns, potentially disregarding any deeper pattern features that a dataset provides. In the case of FB15k and WN18, this potential overfitting behavior is sufficient enough to reach good performance results. This hypothesis is reinforced when looking at the prediction results of the more bias-prone dataset FB15k-237. Here, the absence of test leakage bias makes $\approx 65 - 70\%$ of correct predictions unexplainable with the sole occurrence of bias. In FB15k-237 the frequency of overrepresented tail entities appears to be amplified at a factor ≈ 2 with a rate of 29.7% appearing in correct H@3 predictions, but only 15.1% in the test dataset. The dataset WN18RR, which is only prone to near-symmetric relations, also sees a large amplification of that property in the correct test predictions, as they account for $\approx 70\%$ of prediction results, even though near-symmetric relations only appear at a rate of $\approx 35\%$ in the test split. It can be noted that the outlier in H@1 predictions for WN18RR cannot be deemed significant since they contain more than 3% of all test triples. However, it shows that the model learns some meaningful information outside the bias patterns to correctly predict these triples.

6.3.3 Discussion of Bias Results

It seems that the place of information concentration (entities vs relations) plays a smaller role than our defined bias patterns regarding test leakage and sample selection bias. To be exact, we found that test leakage is the most critical factor in explaining prediction results by LP models.

It is a known fact in the research community that both WN18 and FB15k suffer from test leakage. Python packages for training LP models are even actively

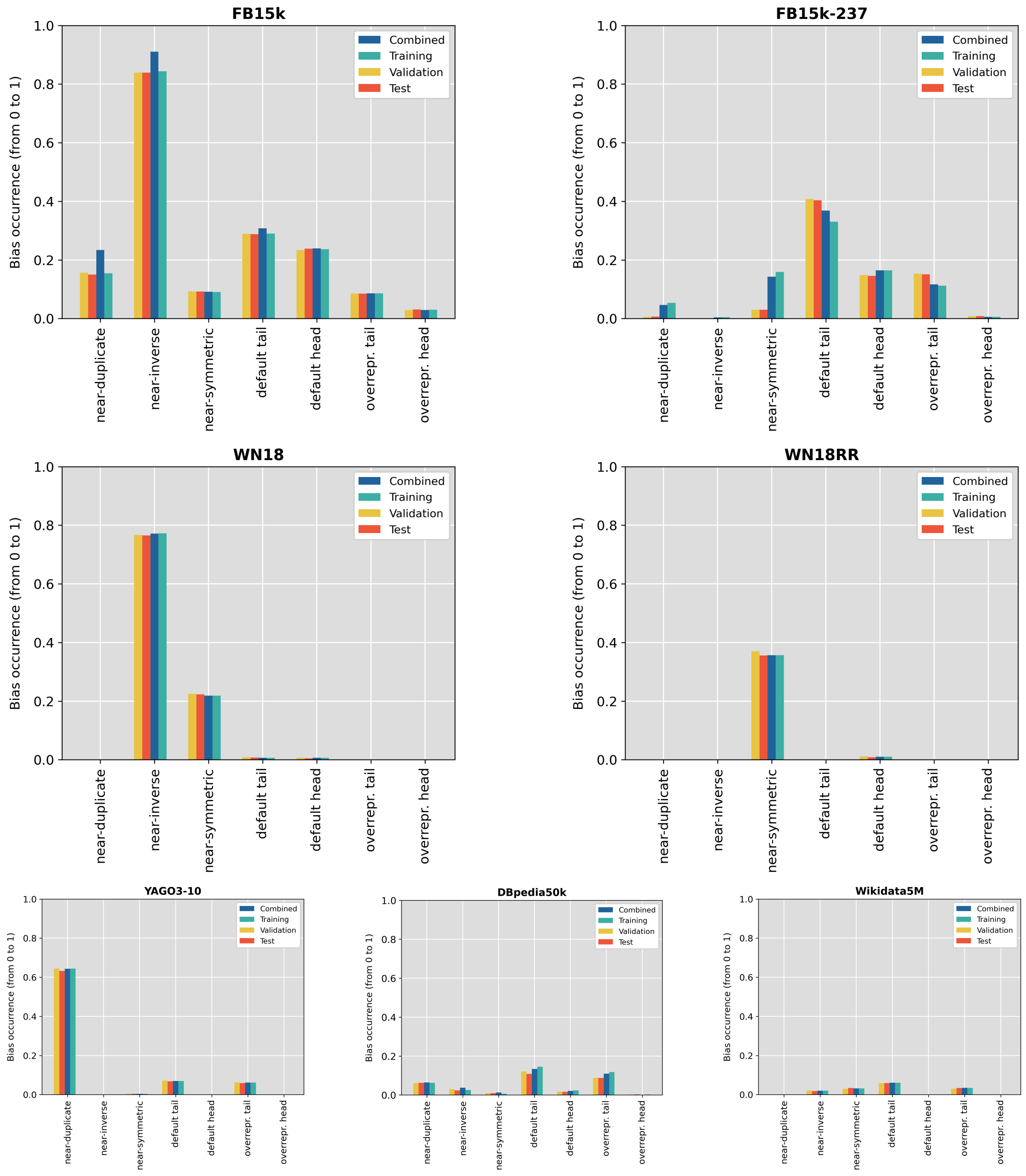


Figure 6.4: Bias in the input data. Every benchmark dataset is differently affected by our bias types. The datasets Dbpedia50k and Wikidata5M are the most-prone to it.

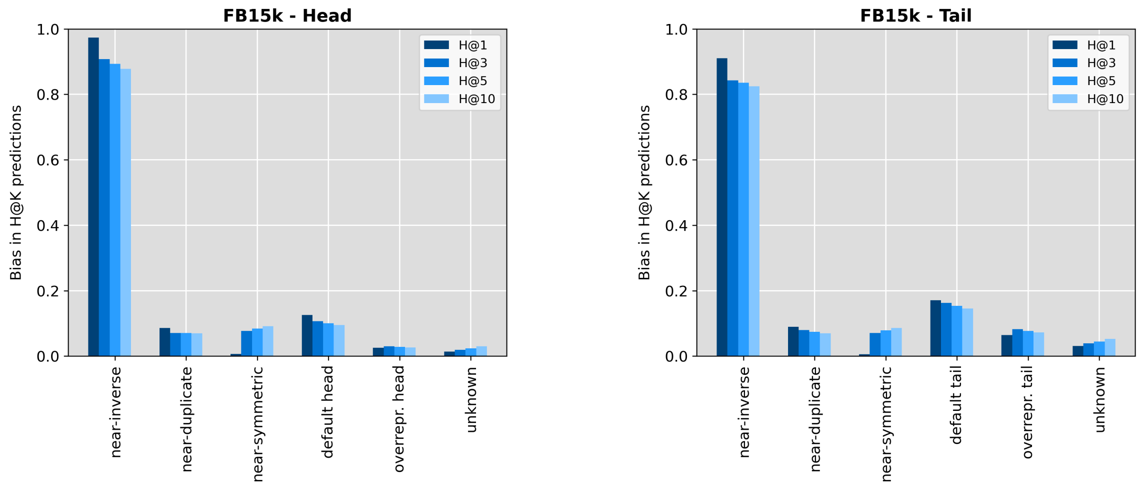


Figure 6.5: Bucketing correct predictions by bias type for FB15k

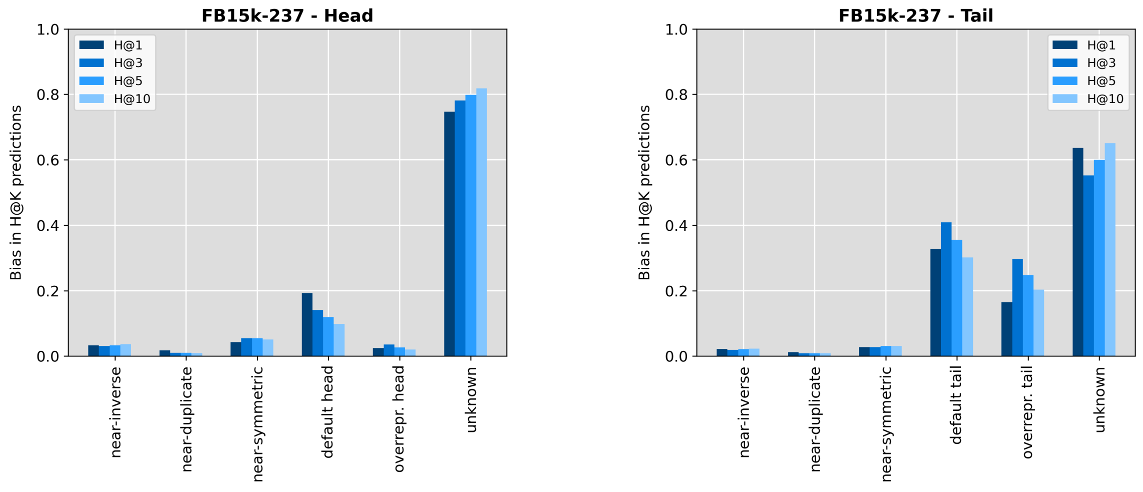


Figure 6.6: Bucketing correct predictions by bias type for FB15k-237

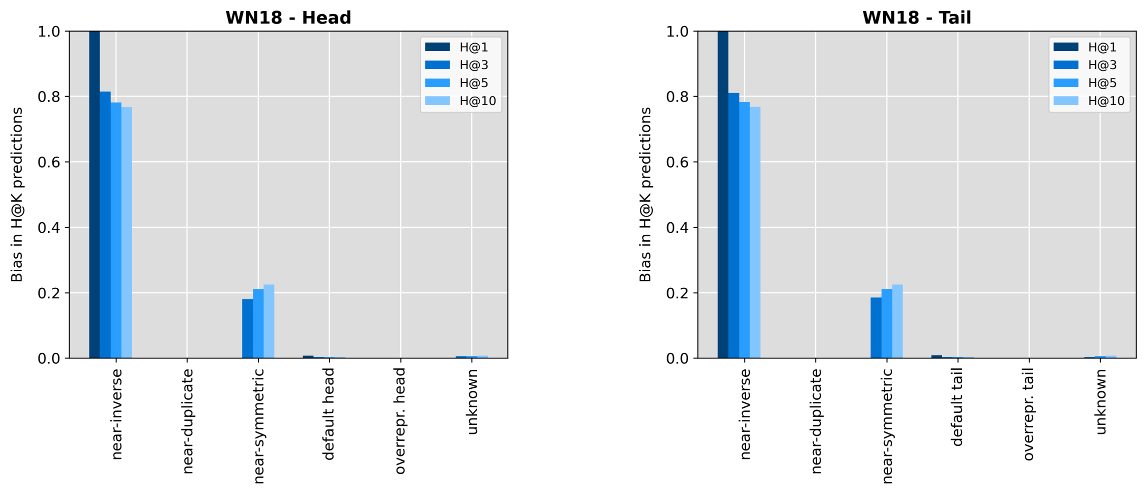


Figure 6.7: Bucketing correct predictions by bias type for WN18

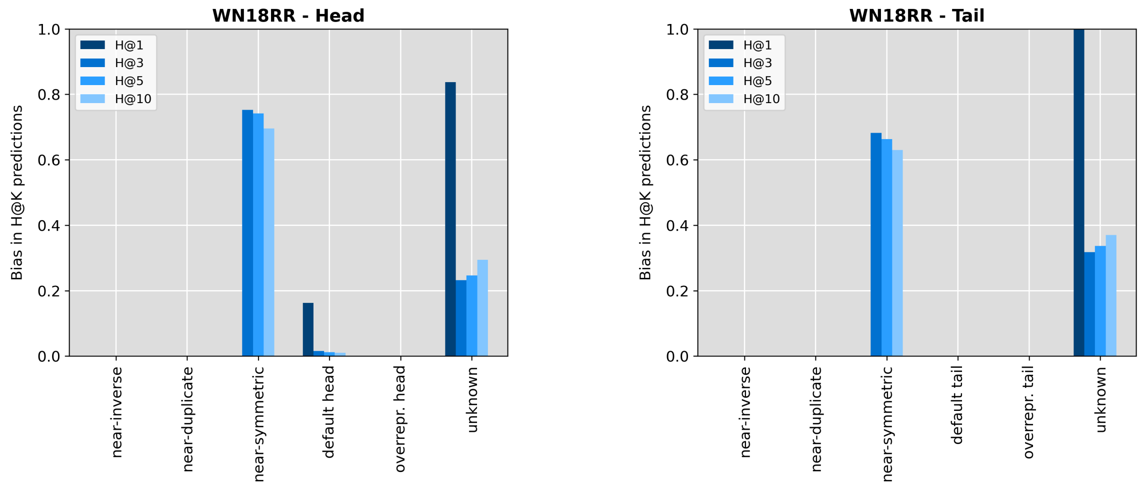


Figure 6.8: Bucketing correct predictions by bias type for WN18RR

discouraging their usage [12] [2]. However, it is interesting to see that even more de-facto ”stable” datasets like WN18RR and YAGO3-10 are prone to test leakage, an issue that has not been paid enough attention to in other studies. The majority of correct predictions in our experiments for WN18RR are triples that contain a near-symmetric relation. In YAGO3-10, more than 60% of all triples can be considered prone to near-duplicate relations. Even FB15k-237 shows signs of sample selection bias, albeit its effects on prediction results are not as clear as with test leakage. Wikidata5M and DBpedia50k appear to be most prone to such biases. We have therefore addressed RQ2 by showing that the bias types defined in our framework can explain major parts of correct test predictions.

6.4 Key Takeaways

We can summarize our findings with regard to analyzed LP benchmarks as follows:

1. All benchmarks feature a network with high connectivity - with the only exception being DBpedia50k.
2. The level of information density is usually high in either relations or entities.
3. Benchmarks with high relational information density are WN18, WN18RR, YAGO3-10 and Wikidata5M. The entity nodes in FB15k and FB15k-237 have a disproportionate high information density. DBpedia50k suffers from both information-sparse nodes and information-sparse relations.
4. All benchmarks are distorted towards a few over-represented entities and a large number of entities with only a few mentions - thereby following the power law.
5. FB15k and WN18 suffer mainly from test leakage due to near-inverse relations.
6. YAGO3-10 suffers from test leakage in form of near-duplicate relations.
7. WN18RR has a notable number of triples prone to near-symmetric relations and account for $\sim 70\%$ of correct predictions.
8. Our experimental results suggest that the model TransE picks up the pattern of overrepresented tail answers in the benchmark FB15k-237. The dataset also has a lot of relations with default tail answers, although its influence on prediction results is not entirely clear.

Chapter 7

Conclusions and Future Work

This chapter summarizes the content of the thesis, addresses the limitations of our work and provides directions for future research in the area.

7.1 Conclusions

In this thesis, we characterized seven commonly used benchmarks for LP regarding their structural biases. We assessed the benchmark datasets' networks as a whole and unraveled structural imbalances within entities and relations. In doing so, we produced 15 SPARQL queries that can be reused in any exploratory analysis of KGs for detecting biases and unusual patterns.

Further, we defined six bias types for test leakage and sample selection bias. In combination with our conducted experiments using the model TransE, they were able to highlight alarming issues in considered state-of-the-art datasets. In WN18RR, we found a concerning number of near-symmetric relations, which explain most of the correct predictions for TransE. YAGO3-10 features a lot of near-duplicate relations and FB15k-237 suffers from a large number of default answer-prone triples.

It is important to acknowledge why certain performance results are reached to increase the credibility of statements regarding the capability of a model. We, thus, call for a more cautious usage of LP benchmarks in general and ask authors of new models to include the different benchmark characteristics when reporting benchmark performance. While this thesis establishes important groundwork for explaining why LP models attain certain performances on benchmark datasets, we also encourage the research community to investigate further factors that can improve the explainability and transparency of LP models.

7.2 Limitations and Future Work

Explaining performance results in an LP pipeline is a very difficult task, as the reasoning process of LP models tends to be quite complex and cannot be easily traced. Our findings are limited in that regard because our experiments are mere indications of the correlation between our defined bias metrics and prediction results. They do not constitute definitive proof of cause for any reported predictions. However, the amplification of bias in the correctly predicted triples makes a correlation more likely. In addition to that, there exist correct predictions that we cannot explain with any of our defined biases. This could either mean that the LP model actually infers a portion of new facts without the influence of bias or that there are other bias types that we have not considered in this study. It is also not clear if these predictions can be made based on observed features like simple logical rules, or, more complex patterns. Newer studies [57] [17] argue that transductive LP approaches are shallow by nature, and therefore not able to learn deeper structures like inductive graph network-based approaches would. While we have shown that TransE picks up the bias patterns in the dataset, it would be interesting to see if the behavior is similar for other LP models as well. Improving explainability when learned patterns become more sophisticated then turns into an even bigger challenge. Further, our work only assesses bias within the network structure of LP benchmark datasets. Other bias factors, e.g. the method with which evaluation metrics are calculated, are out of scope in this thesis. Another critical aspect to consider is that our defined biases actually make up for the learning of useful patterns such as the inverse rule: $\langle X, parentOf, Y\rangle => \langle Y, isChildOf, X\rangle$. While a good benchmark should not only consist of such easily learnable patterns, neglecting such useful test statements is also not ideal. It is more important that a benchmark establishes awareness of the capability traits it is testing for.

For future work, we, therefore, encourage the reporting of more detailed characteristics of benchmarks like their networks' sparsity and bias affection to make acquired performance results more meaningful. We also recommend publishers of new models to investigate their model's performance in a critical manner and to try to explain how performance results are reached. Creating awareness for the types of test triples that get predicted more accurately than others, as we did in this study, can provide deeper insights into the strengths and weaknesses of an LP model. It is also our hope that our presented framework for characterizing bias in LP datasets will continue to be refined and developed by future researchers, by adding new bias types and metrics as explanatory factors for prediction results.

Bibliography

[1] Farahnaz Akrami et al. "Realistic Re-Evaluation of Knowledge Graph Completion Methods: An Experimental Study". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 1995–2010. ISBN: 9781450367356. DOI: 10.1145/3318464.3380599. URL: https://doi.org/10.1145/3318464.3380599.

[2] Mehdi Ali et al. "PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings". In: *Journal of Machine Learning Research* 22.82 (2021), pp. 1–6. URL: http://jmlr.org/papers/v22/20-825.html.

[3] Sören Auer et al. "DBpedia: A Nucleus for a Web of Open Data". In: *The Semantic Web*. Springer Berlin Heidelberg, 2007, pp. 722–735. DOI: 10.1007/978-3-540-76298-0_52. URL: https://doi.org/10.1007/978-3-540-76298-0_52.

[4] Ivana Balazevic, Carl Allen, and Timothy Hospedales. "TuckER: Tensor Factorization for Knowledge Graph Completion". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5185–5194. DOI: 10.18653/v1/D19-1522. URL: https://aclanthology.org/D19-1522.

[5] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pp. 509–512. DOI: 10.1126/science.286.5439.509. eprint: https://www.science.org/doi/pdf/10.1126/science.286.5439.509. URL: https://www.science.org/doi/abs/10.1126/science.286.5439.509.

[6] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. *Gephi: An Open Source Software for Exploring and Manipulating Networks*. 2009. URL: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284.5 (May 2001), pp. 34–43. URL: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[8] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 2008), P10008. DOI: 10.1088/1742-5468/2008/10/P10008. URL: https://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

[9] Kurt Bollacker et al. “Freebase”. In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, June 2008. DOI: 10.1145/1376616.1376746. URL: https://doi.org/10.1145/1376616.1376746.

[10] Antoine Bordes et al. “A semantic matching energy function for learning with multi-relational data”. In: *Machine Learning* 94.2 (May 2013), pp. 233–259. DOI: 10.1007/s10994-013-5363-6. URL: https://doi.org/10.1007/s10994-013-5363-6.

[11] Antoine Bordes et al. “Translating Embeddings for Modeling Multi-relational Data”. In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[12] Luca Costabello et al. *AmpliGraph: a Library for Representation Learning on Knowledge Graphs*. Mar. 2019. DOI: 10.5281/zenodo.2595043. URL: https://doi.org/10.5281/zenodo.2595043.

[13] Tim Dettmers et al. “Convolutional 2D Knowledge Graph Embeddings”. In: New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[14] Anastasia Dimou et al. “RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data”. In: *Proceedings of the 7th Workshop on Linked Data on the Web*. Apr. 2014.

[15] Takuma Ebisu and Ryutaro Ichise. “TorusE: Knowledge Graph Embedding on a Lie Group”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11538. URL: https://doi.org/10.1609/aaai.v32i1.11538.

[16] Luis Antonio Galárraga et al. “AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases”. In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW ’13. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 413–422. ISBN: 9781450320351. DOI: 10.1145/2488388.2488425. URL: https://doi.org/10.1145/2488388.2488425.

[17] Mikhail Galkin, Max Berrendorf, and Charles Tapley Hoyt. “An Open Challenge for Inductive Link Prediction on Knowledge Graphs”. In: *CoRR* abs/2203.01520 (2022). URL: https://doi.org/10.48550/arXiv.2203.01520.

[18] Mikhail Galkin et al. “Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems”. In: (Apr. 2017).

[19] M. Girvan and M. E. J. Newman. “Community structure in social and biological networks”. In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826. DOI: 10.1073/pnas.122653799. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.122653799. URL: https://www.pnas.org/doi/abs/10.1073/pnas.122653799.

[20] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[21] James Heckman. “Sample Selection Bias As a Specification Error (with an Application to the Estimation of Labor Supply Functions)”. In: *National Bureau of Economic Research, Inc, NBER Working Papers* (Jan. 1977).

[22] Aidan Hogan et al. “Knowledge Graphs”. In: *ACM Computing Surveys* 54.4 (July 2021), pp. 1–37. DOI: 10.1145/3447772. URL: https://doi.org/10.1145/3447772.

[23] Enrique Iglesias et al. “SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20. Virtual Event, Ireland: Association for Computing Machinery, 2020, pp. 3039–3046. ISBN: 9781450368599. DOI: 10.1145/3340531.3412881. URL: https://doi.org/10.1145/3340531.3412881.

[24] Yuwang Ji et al. “A Survey on Tensor Techniques and Applications in Machine Learning”. In: *IEEE Access* 7 (2019), pp. 162950–162990. DOI: 10.1109/access.2019.2949814. URL: https://doi.org/10.1109/access.2019.2949814.

[25] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. “Knowledge Base Completion: Baselines Strike Back”. In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 69–74. DOI: 10.18653/v1/W17-2609. URL: https://aclanthology.org/W17-2609.

[26] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. “Leakage in Data Mining: Formulation, Detection, and Avoidance”. In: vol. 6. Jan. 2011, pp. 556–563. DOI: 10.1145/2020408.2020496.

[27] Seyed Mehran Kazemi and David Poole. “SimplE Embedding for Link Prediction in Knowledge Graphs”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., 2018, pp. 4289–4300.

[28] Maria Khvalchik, Artem Revenko, and Christian Blaschke. “Question Answering for Link Prediction and Verification”. In: *The Semantic Web: ESWC 2019 Satellite Events*. Ed. by Pascal Hitzler et al. Cham: Springer International Publishing, 2019, pp. 116–120. ISBN: 978-3-030-32327-1.

[29] Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. Feb. 1999. URL: https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[30] Jing Li et al. “Recommendation Algorithm based on Link Prediction and Domain Knowledge in Retail Transactions”. In: *Procedia Computer Science* 31 (2014). 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014, pp. 875–881. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2014.05.339. URL: https://www.sciencedirect.com/science/article/pii/S187705091400516X.

[31] David Liben-Nowell and Jon Kleinberg. “The Link Prediction Problem for Social Networks”. In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. CIKM ’03. New Orleans, LA, USA: Association for Computing Machinery, 2003, pp. 556–559. ISBN: 1581137230. DOI: 10.1145/956863.956972. URL: https://doi.org/10.1145/956863.956972.

[32] Yankai Lin et al. “Learning Entity and Relation Embeddings for Knowledge Graph Completion”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI’15. Austin, Texas: AAAI Press, 2015, pp. 2181–2187. ISBN: 0262511290.

[33] Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. "YAGO3: A Knowledge Base from Multilingual Wikipedias". In: *Conference on Innovative Data Systems Research*. 2015.

[34] Shawn Martin et al. "OpenOrd: An Open-Source Toolbox for Large Graph Layout". In: *Proc SPIE* 7868 (Jan. 2011), p. 786806. DOI: 10.1117/12.871402.

[35] George A. Miller. "WordNet". In: *Communications of the ACM* 38.11 (Nov. 1995), pp. 39–41. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

[36] Galileo Namata and Lise Getoor. "Link Prediction". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 609–612. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_481. URL: https://doi.org/10.1007/978-0-387-30164-8_481.

[37] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[38] Guillermo Palma, Maria-Esther Vidal, and Louiqa Raschid. "Drug-target interaction prediction using semantic similarity and edge partitioning". In: *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*. Springer. 2014, pp. 131–146.

[39] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic Web* 8.3 (Dec. 2016). Ed. by Philipp Cimiano, pp. 489–508. DOI: 10.3233/sw-160218. URL: https://doi.org/10.3233/sw-160218.

[40] Ciyuan Peng et al. "Knowledge Graphs: Opportunities and Challenges". In: *Artificial Intelligence Review* (2023), pp. 1–32.

[41] Eric Prud'hommeaux and Andy Seaborne. *SPARQL Query Language for RDF*. W3C Recommendation. Jan. 2008. URL: https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[42] Jay Pujara, Eriq Augustine, and Lise Getoor. "Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1751–1756. DOI: 10.18653/v1/D17-1184. URL: https://aclanthology.org/D17-1184.

[43] Richard Qian. *Understand Your World with Bing*. Last accessed 25th of March 2023. URL: https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing.

[44] Enayat Rajabi and Kobra Etminani. "Knowledge-graph-based explainable AI: A systematic review". In: *Journal of Information Science* 0.0 (0), p. 01655515221112844. DOI: 10.1177/01655515221112844. eprint: https://doi.org/10.1177/01655515221112844. URL: https://doi.org/10.1177/01655515221112844.

[45] Andrea Rossi, Donatella Firmani, and Paolo Merialdo. *Knowledge Graph Embeddings or Bias Graph Embeddings? A Study of Bias in Link Prediction Models*. 2021. URL: https://alammehwish.github.io/dl4kg2021/papers/knowledge_graph_embeddings_or_.pdf.

[46] Andrea Rossi and Antonio Matinata. "Knowledge graph embeddings: Are relation-learning models learning relations?" In: *EDBT/ICDT Workshops*. 2020.

[47] Andrea Rossi et al. "Knowledge Graph Embedding for Link Prediction". In: *ACM Transactions on Knowledge Discovery from Data* 15.2 (Apr. 2021), pp. 1–49. DOI: 10.1145/3424672. URL: https://doi.org/10.1145/3424672.

[48] Manuel Scurti et al. "A Data-Driven Approach for Analyzing Healthcare Services Extracted from Clinical Records". In: *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*. 2020, pp. 193–196. DOI: 10.1109/CBMS49503.2020.00044.

[49] Baoxu Shi and Tim Weninger. "Open-World Knowledge Graph Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (Nov. 2017). DOI: 10.1609/aaai.v32i1.11535.

[50] Nicholas D. Sidiropoulos et al. "Tensor Decomposition for Signal Processing and Machine Learning". In: *IEEE Transactions on Signal Processing* 65.13 (2017), pp. 3551–3582. DOI: 10.1109/TSP.2017.2690524.

[51] Sameer Singh et al. "Wikilinks: A Large-scale Cross-Document Coreference Corpus Labeled via Links to Wikipedia". In: 2012.

[52] Amit Singhal. *AmpliGraph - Predictive Performance*. Last accessed 30th of March 2023. URL: https://docs.ampligraph.org/en/latest/experiments.html.

[53] Amit Singhal. *Introducing the Knowledge Graph: things, not strings*. Last accessed 30th of March 2023. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[54] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago". In: *Proceedings of the 16th international conference on World Wide Web*. ACM, May 2007. DOI: 10.1145/1242572.1242667. URL: https://doi.org/10.1145/1242572.1242667.

[55] Zhiqing Sun et al. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HkgEQnRqYQ.

[56] Panagiotis Symeonidis and Andreas Zioupos. "Introduction". In: *Matrix and Tensor Factorization Techniques for Recommender Systems*. Springer International Publishing, 2016, pp. 3–17. DOI: 10.1007/978-3-319-41357-0_1. URL: https://doi.org/10.1007/978-3-319-41357-0_1.

[57] Komal Teru, Etienne Denis, and Will Hamilton. "Inductive Relation Prediction by Subgraph Reasoning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 9448–9457. URL: https://proceedings.mlr.press/v119/teru20a.html.

[58] K. Thulasiraman and M.N.S. Swamy. *Graphs: Theory and Algorithms*. A Wiley interscience publication. Wiley, 1992. ISBN: 9780471513568. URL: https://books.google.de/books?id=4kHN6uSinQoC.

[59] Ilaria Tiddi and Stefan Schlobach. "Knowledge graphs as tools for explainable machine learning: A survey". In: *Artificial Intelligence* 302 (2022), p. 103627. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103627. URL: https://www.sciencedirect.com/science/article/pii/S0004370221001788.

[60] Kristina Toutanova and Danqi Chen. "Observed versus latent features for knowledge base and text inference". In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, 2015. DOI: 10.18653/v1/w15-4007. URL: https://doi.org/10.18653/v1/w15-4007.

[61] Théo Trouillon et al. "Complex Embeddings for Simple Link Prediction". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2071–2080. URL: https://proceedings.mlr.press/v48/trouillon16.html.

[62] Denny Vrandečić and Markus Krötzsch. "Wikidata". In: *Communications of the ACM* 57.10 (Sept. 2014), pp. 78–85. DOI: 10.1145/2629489. URL: https://doi.org/10.1145/2629489.

[63] Xiaozhi Wang et al. "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation". In: *Transactions of the Association for Computational Linguistics* 9 (Mar. 2021), pp. 176–194. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00360. URL: https://arxiv.org/abs/1911.06136.

[64] Xu Wang, Frank van Harmelen, and Zhisheng Huang. "Recommending scientific datasets using author networks in ensemble methods". In: *Data Science* 5.2 (July 2022). Ed. by Stephen Pettifer, pp. 167–193. DOI: 10.3233/ds-220056. URL: https://doi.org/10.3233/ds-220056.

[65] Yanjie Wang et al. "On Evaluating Embedding Models for Knowledge Base Completion". In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 104–112. DOI: 10.18653/v1/W19-4313. URL: https://aclanthology.org/W19-4313.

[66] Zhen Wang et al. "Knowledge Graph Embedding by Translating on Hyperplanes". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 28.1 (June 2014). DOI: 10.1609/aaai.v28i1.8870. URL: https://ojs.aaai.org/index.php/AAAI/article/view/8870.

[67] Wikidata. *Property:P407 — Wikidata*. Last accessed 10th of April 2023. 2023. URL: %5Curl%7Bhttps://www.wikidata.org/w/index.php?title=Property:P407&oldid=1870199776%7D.

[68] Bishan Yang et al. *Embedding Entities and Relations for Learning and Inference in Knowledge Bases*. 2014. DOI: 10.48550/ARXIV.1412.6575. URL: https://arxiv.org/abs/1412.6575.