

Structural Alignment in Link Prediction

Jeffrey Seathrún Sardina

Supervised by Prof. Declan O'Sullivan and Prof. John D. Kelleher

Trinity College Dublin, the University of Dublin

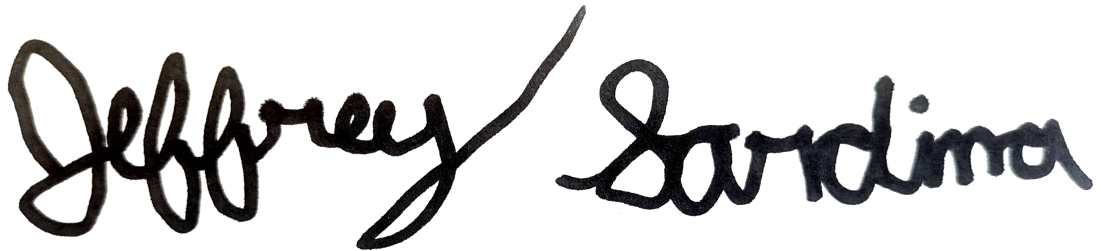
12 April 2025

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

A handwritten signature in black ink, reading "Jeffrey Sandidina". The signature is written in a cursive style with a large, sweeping initial 'J' and a long, horizontal stroke extending from the end of the name.

Summary

While Knowledge Graphs (KGs) have become increasingly popular across various scientific disciplines for their ability to model and interlink huge quantities of data, essentially all real-world KGs are known to be incomplete. As such, with the growth of KG use has been a concurrent development of machine learning tools designed to predict missing information in KGs, which is referred to as the Link Prediction Task. The majority of state-of-the-art link predictors to date have followed an embedding-based paradigm. In this paradigm, it is assumed that the information content of a KG is best represented by the (individual) vector representations of its nodes and edges, and that therefore node and edge embeddings are particularly well-suited to performing link prediction.

This thesis proposes an alternative perspective on the field’s approach to link prediction and KG data modelling. Specifically, this work re-analyses KGs and state-of-the-art link predictors from a graph-structure-first perspective that models the information content of a KG in terms of whole triples, rather than individual nodes and edges. After building up a theoretical foundation for this structure-first approach from the state-of-the-art literature, it is evaluated in two contexts.

The first evaluation asks if link predictors’ outputs are aligned to aspects of KG structure. Results indicate that, not only are link predictors heavily influenced by structure, but that their patterns of hyperparameter preference, and their overall performance, can be explained and simulated in terms of the structure of the graph they were trained to learn.

The second evaluation builds upon this observation and asks if graph structural features of triples in a KG are sufficient to enable link prediction. The results of this second round of experiments indicate that structure-based link prediction is not only possible, but highly effective compared to state-of-the-art approaches. Finally, it has been found that, by representing the information content of a KG in terms of triple-level structure, cross-KG (including cross-domain) transfer learning becomes viable for the link prediction task.

The thesis concludes that a structure-first perspective on KGs and link prediction is both viable and useful for understanding KG learning. This observation is used to create and propose the Structural Alignment Hypothesis, which postulates that link prediction can be understood and modelled as a structural task.

All code and data used for this thesis, including the link prediction simulator (TWIG) and the structure-based link predictor (TWIG-I) are open-sourced to encourage further work in this area. Finally, this thesis was written bilingually, with the main document in English and an informal extended summary in Irish. An Irish-language translation dictionary of machine learning terms (the *Foclóir Tráchtais*) created for this work is open-sourced as well.

Ethics Statement

The author believes that the use of generative AI to write or edit publications or code is a form of plagiarism, on account of how generative AI models are trained on copyrighted data without consent of the data creators. The author confirms that no part of this thesis, nor any of the code used within it, was written using any generative AI tools. Further, the author confirms that no aspect of this thesis or its related materials come from any use of generative AI. The author refuses permission for any part of this thesis or its related materials to be used as training data for generative AI models without prior written permission of the author.

Finally, the author asserts that no application of AI / machine learning tools should ever be applied in such a way that harms the natural world or threatens human jobs or livelihoods. It is the sincere belief of the author that the content of this work does not represent such a risk.

Dedication

For Callie, the best sister in the world.

Acknowledgements

To start off, I would like to thank my family – especially my mother, Suzanne Sardina, my father, Jason Sardina, and my sister, Callie Sardina for their love and support over the entire course of my PhD. Your support means the world to me. I have also been very fortunate in having amazing friends and family who have helped me along my journey – both in Ireland and at home in the United States. Thank you all for helping me in this journey – and for always being there for me, even when I am a few thousand miles away.

During my PhD, I was blessed to be able to work with many great collaborators, including my supervisors, Prof. Declan O’Sullivan and Prof. John D. Kelleher, fellow PhD candidates Alok Debnath and Matt Murtagh, my internship supervisors and colleagues Dr. Luca Costabello, Dr. Christophe Guéret and Dr. Alberto Bernardi from Accenture, and even for one paper, my sister Callie! Collaborations and co-authored papers have honestly been among the most fun and most rewarding aspects of my PhD, and I am extremely grateful to all who have worked with me throughout my degree.

I would further like to thank my supervisors, Prof. Declan O’Sullivan and Prof. John D. Kelleher, for their constant support and advice over the course of my PhD. I could not have asked to work with a better, more helpful, or more caring team. Both Declan and John constantly motivated me to pursue even my most far-fetched ideas – many of which later turned into core components of this thesis. Their insight, and foresight, has been foundational to this thesis and to all of my research.

The Irish language has been a huge part of my time in Ireland. Its community (especially Club Chonradh na Gaeilge and the many Pop-Up Gaeltacht groups), were a huge source of motivation and support for me throughout the last three-and-a-half years. Ongoing Irish-language research, as well the general Irish-language community, were a major inspiration for my efforts to make the results of this thesis available bilingually. Míle búíochas libh go léir!

My gratitude goes out the entire ADAPT Research Centre and D-REAL community for providing space for open collaboration and communication of research among researchers from diverse fields. I would finally also like to thank Fergal Marrinan from Sonas Innovation, the RI CRT D-REAL, and Taighde Éireann | Research Ireland for the funding and support they provided for my research.

Contents

Declaration	i
Summary	ii
Ethics Statement	iii
Dedication	iv
Acknowledgements	v
Contents	vi
Definitions	x
Code and Data	xiv
Structural Alignment in Link Prediction	1
1 Introduction	2
1.1 Motivation and Overview	2
1.2 Research Question	7
1.3 The Structural Alignment Framework	8
1.4 Contributions and Impact	10
1.5 Overview of the Chapters	11
2 Background and State of the Art	13
2.1 Knowledge Graphs	13
2.1.1 Data Modelling in Knowledge Graphs	13
2.1.2 Application Domains of Knowledge Graphs	15
2.2 Link Prediction	18
2.2.1 The Link Prediction Task	18
2.2.2 Rank-Based Evaluation of Link Predictors	20
2.2.3 Applications of Link Prediction	27
2.3 Knowledge Graph Embedding Models	28
2.3.1 Components of KGEMs	28
2.3.2 The KGEM Training Loop	36
2.3.3 Hyperparameter Selection for KGEMs	38
2.3.4 A Note on non-KGEM Link Predictors	40

2.4	Graph Structure and Link Prediction	43
2.4.1	Measures of Knowledge Graph Structure	43
2.4.2	Hyperparameter Preference and Link Prediction	52
2.4.3	Towards Structural Alignment	60
3	Structural Alignment	63
3.1	Formal Definition	63
3.1.1	Claim 1: Of Hyperparameters and Link Prediction	63
3.1.2	Claim 2: Of Structure-based Link Prediction	65
3.2	Instantiating Structural Alignment	66
3.2.1	Selection of Structural Features	66
3.2.2	Examples of Structural Feature Calculation	68
3.2.3	Calculating Structural Features for LP Queries	73
3.3	Summary of Structural Alignment	74
4	TWIG	75
4.1	Data and Task Model	76
4.1.1	Modelling KG Structure	76
4.1.2	Modelling KGEM Hyperparameters and LP Performance	76
4.1.3	Bringing Structure and Hyperparameters Together	78
4.2	Case Study: ComplEx and UMLS	79
4.2.1	Determination of Signal	80
4.2.2	Designing TWIG’s Neural Architecture	82
4.2.3	Designing a Training and Evaluation Protocol	84
4.2.4	Choosing KGs for Use	88
4.3	Evaluating TWIG	89
4.3.1	Cross-KGEM Evaluation of TWIG	94
4.3.2	Cross-KG Evaluation of TWIG	95
4.3.3	Training TWIG on More Hyperparameter Experiments	96
4.3.4	TWIG Ablation Studies	98
4.3.5	A Final Note on TWIG	103
4.4	Structure and Hyperparameter Analysis	104
4.4.1	ComplEx on UMLS	105
4.4.2	Bringing it all Together	112
5	TWIG-I	117
5.1	General Methodology of TWIG-I	117

5.2	Evaluating Structural Alignment with TWIG-I	120
5.2.1	Experiments and Results	120
5.2.2	Discussion of TWIG-I as a Link Predictor	121
5.3	TWIG-I Structural Feature Ablation Studies	124
5.4	Evaluating Transfer Learning with TWIG-I	127
6	Discussion and Conclusions	132
6.1	Implications of Structural Alignment	133
6.1.1	Structure and Semantics	133
6.1.2	Hyperparameters and Model Creation	136
6.1.3	Ontologies and Learning	136
6.1.4	Implications for Knowledge Graph Creation	138
6.1.5	Other Future Directions	141
6.2	Research Publications, Outputs and Awards	143
6.2.1	Code Contributions	144
6.2.2	Listing of Research Outputs	145
6.2.3	Listing of Patents	147
6.2.4	Work as a Reviewer	147
6.3	Final Remarks	147
	Appendix	149
	Appendix Contents	150
A	Further TWIG Ablations	151
A.1	TWIG on ComplEx	152
A.2	TWIG on DistMult	154
A.3	TWIG on TransE	156
B	Structural Feature Distributions	158
B.1	CoDExSmall	159
B.2	DBpedia50	160
B.3	Kinships	161
B.4	OpenEA	162
B.5	UMLS	163
C	Further KG-KGEM Data	164
C.1	ComplEx on CoDExSmall	165
C.2	ComplEx on DBpedia50	168

C.3	ComplEx on Kinships	171
C.4	ComplEx on OpenEA	174
C.5	ComplEx on UMLS	177
C.6	DistMult on CoDExSmall	180
C.7	DistMult on DBpedia50	183
C.8	DistMult on Kinships	186
C.9	DistMult on OpenEA	189
C.10	DistMult on UMLS	192
C.11	TransE on CoDExSmall	195
C.12	TransE on DBpedia50	198
C.13	TransE on Kinships	201
C.14	TransE on OpenEA	204
C.15	TransE on UMLS	207
D	TWIG-I vs. the Intersection Features Model	210
D.1	Summary of the Intersection Features Model	210
D.2	Comparison with TWIG-I	211
D.3	Comparison of Empirical Performance	211
	References	213

Definitions

All Definitions in this section will draw on a single running example knowledge graph, which is given in Figure 1.1 (in graphical format) and in Table 1.1 (in tabular format as a list of triples).

Hyperparameter. A hyperparameter is any non-learnable parameter or component that is used in the creation or training of a machine learning model. This definition is intentionally broad as the author adopts the perspective that all components of knowledge graph embedding models (including the choice of scoring function, negative sampler, and loss function) are hyperparameters of that knowledge graph embedding model.

Knowledge Graph. A Knowledge Graph (KG) is a graph-based data store in which all data is represented as a series of nodes (representing concepts) and edges (representing relationships) [33]. The atomic unit of information in a KG is a statement called a triple, with the form (s, p, o) where:

- s is the subject node, also called the head of the relationship;
- o is the object node, also called the tail of the relationship;
- p is the directed, labelled predicate (or “relationship”) that describes how the subject node relates to the object node.

An example knowledge Graph is given in Figure 1.1. Example triples in this graph include $(Sauron, Enemy-Of, Frodo)$ and $(Rohan, has-Alliance-With, Gondor)$. While the literature uses the terms subject / tail, predicate / relationship, and object / tail interchangeably, in this work only the terms subject, predicate, and object will be used.

Knowledge Graph Embedding. Knowledge Graph Embedding (KGE) is the process of projecting a knowledge graph into vector space such that every node and edge in the KG is represented by a vector in that vector space [2, 33, 62, 70, 97]. Individual embeddings are called Knowledge Graph Embeddings (KGEs) [2, 33, 62, 70, 97].

It is technically correct to refer to any system of assigning embedding vectors to nodes and edges, regardless of the manner in which it is done, a knowledge graph embedding system. However, in this thesis, focus is placed particularly on knowledge graph embedding models as a means of achieving knowledge graph embedding.

Knowledge Graph Embedding Model. A Knowledge Graph Embedding Model (KGEM) is a machine learning model that learns to create knowledge graph embeddings for a given knowledge graph [2, 33, 62, 70, 97]. KGEMs contain three core components [2, 62, 97] that determine their overall function:

- a scoring function $f(e_s, e_p, e_o) \rightarrow R^1$ that takes as input the embeddings of a triple (s, p, o) , denoted (e_s, e_p, e_o) , and uses those embeddings to calculate a scalar-valued plausibility score S for the triple. Higher scores indicate greater confidence that the input triple is true.
- a negative sampler $NS(s, p, o, side) \rightarrow (s', p, o)$ or (s, p, o') that takes as input a triple (s, p, o) and a side (either subject or object) and returns a “negative” triple in which the subject or object is corrupted and replaced with another (typically randomly chosen) node from the graph. Negative examples are typically used as counter-examples during KGEM learning as a contrast to known-true (or positive) triples observed in the KG.
- a loss function (which has several different possible forms) that scores a KGEM based on its ability to distinguish true triples from generated negatives during link prediction; lower values indicate better performance. The loss function is minimised during KGEM training.

In addition to these, all KGEMs have a variety of other hyperparameters, such as the learning rate and embedding dimension, that must be defined in order for the KGEM to be run. In addition, each of these components may have their own hyperparameters, such as the number of negatives samples output from the negative sampler.

While many applications of KGEMs are possible, they are typically trained on link prediction even if their application is for another task such as node clustering. In this thesis, the term KGEM is used specifically to describe the sort of knowledge graph embedding model that is used for, and evaluated in the context of, link prediction.

Link Prediction Task. The Link Prediction (LP) Task is the task of answering link prediction queries on a knowledge graph so as to predict new triples in the graph based on the triples observed in it [2, 70]. A link prediction query necessarily takes one of two forms:

- subject prediction, denoted $(?, p, o)$, in which a predicate and an object are provided and the task is to predict which subject(s) can complete the triple such that it would be true, and
- object prediction, denoted $(s, p, ?)$, in which a subject and a predicate are provided and the task is to predict which object(s) can complete the triple such that it would be true.

Link prediction is a machine learning task often (but not exclusively) performed by knowledge graph embedding models [2, 33, 62, 70, 97]. The output of link prediction is a 1-indexed ranked list of all possible subject or object completions such that items at the start of the list (closer to index 1) are considered to be more plausible completions than completions that come at the end of the list.

As an example, the link prediction query (*Gandalf*, *Ally-Of*, ?) could be posed to the example knowledge graph in Figure 1.1. A link predictor would be expected to return a list such as the following:

1. Frodo
2. Aragorn
3. ...
4. Sauron

where the items higher in the list (at lower ranks) are more plausible, and the items lower in the list (at higher ranks) are less plausible.

Structure. Definitions of graph structure in the literature typically refer to measures of how nodes and edges are connected, such as measures of node degree [9, 22, 25, 33, 45, 58, 70, 71, 73, 74, 98, 110] or the frequency of an edge [9, 45, 58, 70, 71, 73]. While there is no single universal definition of graph structure, this work follows the examples set in existing literature and defines structure as so: the structure of a graph is a quantitative description of the local connectivity of individual nodes and edges in a graph, such as node degree or edge frequency. Any one such description of structure is called a “structural feature” (which is defined fully later in this section).

As an example, take the node *Aragorn* and the edge *Ally-Of* in the example knowledge graph given in Figure 1.1. It could be calculated that *Aragorn* has a degree of 2, since exactly two edges connect to it, and that *Ally-Of* has a frequency of 5, since it occurs exactly 5 times in the graph.

Structural Alignment. Structural Alignment is the hypothesis presented in this work that claims that knowledge graph embedding and link prediction can be modelled and explained in terms of graph structure. The Structural Alignment Framework is a framework, arising from this hypothesis, for directly modelling KG learning and link prediction as a function of specific KG structural features identified in this work.

Structural Feature. A (Graph) Structural Feature is defined in this work as a quantitative measure of an aspect of a graph that:

1. does not refer to individual node or edge labels,
2. can be calculated for any knowledge graph in a graph-agnostic manner, and
3. describes the frequency or pattern of a graph element, such as nodes, edges, or triples.

In other words, graph structural features describe the structure of a graph without describing labels or underlying knowledge / information content. Note that all structural features in this thesis are calculated for knowledge graphs; as such, the terms “graph structural feature” and “structural feature” are identical in the context of this thesis.

Code and Data

All code and data produced and used in this work are made openly available with research-compatible open-source licences. The code for TWIG and TWIG-I, the two core contributions of this thesis, can be found on GitHub:

- TWIG : <https://github.com/Jeffrey-Sardina/TWIG-TWM-dev>
- TWIG-I: <https://github.com/Jeffrey-Sardina/TWIG-I>

Both were also made into publicly-available PyPi projects that can be easily installed, run, and extended by future researchers and developers. These projects are available at:

- TWIG : <https://pypi.org/project/twig-twm>
- TWIG-I: <https://pypi.org/project/twigi>

Furthermore, they can also be directly installed into a Python 3.7+ environment using the following commands:

- `pip install twig-twm`
- `pip install twigi`

Standard knowledge graph benchmark datasets and implementations of state-of-the-art knowledge graph embedding models are taken from the PyKEEN project at: <https://github.com/pykeen/pykeen> [3].

All other data used in these experiments can be generated directly from the code above. Moreover, data from large-scale hyperparameter studies, which is very expensive to compute, is made available in long-term data storage for download at the following link: <https://figshare.com/s/7b2da136e05f3548399f>. It is provided in a form that can be directly loaded and used by TWIG.

This thesis was written bilingually with the main text in English and an informal extended summary in Irish. As a part of translating this work into Irish, an Irish-language dictionary of computer science terms was curated. This dictionary can be found in the Irish-language version of the thesis and online at <https://focloir-riomheolaiochta.github.io>.

Finally, QR codes linking to all major outputs of this thesis can be found on the following page.



TWIG GitHub Repository.



TWIG PyPi Repository.



TWIG-I GitHub Repository.



TWIG-I PyPi Repository.



TWIG Data Repository.

*Foclóir Tráchtais* Website.

QR codes linking to all major outputs of this work.

Structural Alignment in Link Prediction

1. Introduction

1.1 Motivation and Overview

Knowledge Graphs (KGs) are large databases that represent data in a graphical format [33]. In KGs, all data can be represented as statements called triples consisting of a subject node, an object node, and a predicate edge that describes how the subject relates to the object [33]. An example KG is given in Figure 1.1, and a tabular list of all of its triples in (s, p, o) (subject, predicate, object) form is given in Table 1.1.

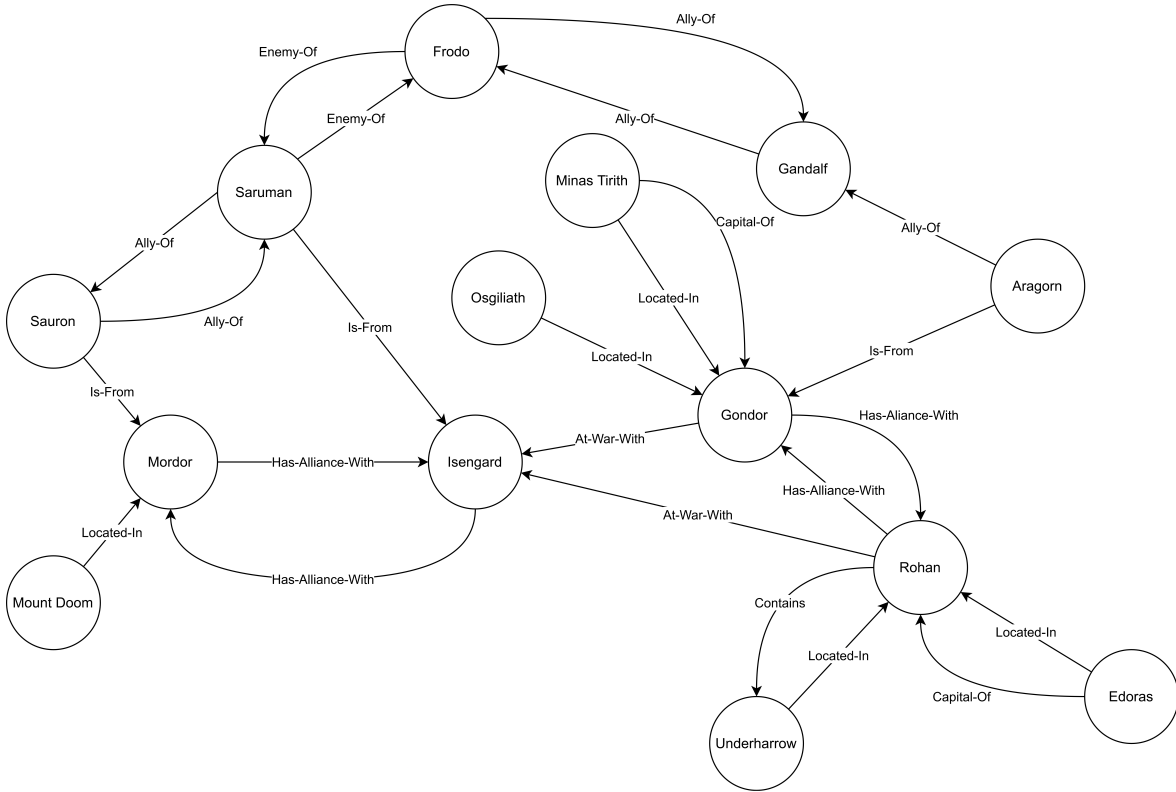


Figure 1.1: An example knowledge graph featuring information about *The Lord of the Rings* [92], expressed in graphical format.

The graphical format of knowledge graphs has allowed them to naturally represent a large variety of real-world data, including social networks, computer networks, biological networks, linguistic data, climate data, general knowledge, and much more [33, 42, 51, 56, 93, 100]. Recently, larger and larger KGs have been published, leading to substantial growth in big graph data and KGs that can contain millions or even hundreds of millions of triples [6, 18, 54, 90].

In response, several approaches to automatic processing and machine learning of KG data have been adopted. These approaches, collectively called Knowledge Graph Embedding Models (KGEMs), aim to represent the information content of a KG in vector space. This embedding representation then allows machine learning models to make predictions about,

Subject	Predicate	Object
Aragorn	Ally-Of	Gandalf
Aragorn	Is-From	Gondor
Edoras	Capital-Of	Rohan
Edoras	Located-In	Rohan
Frodo	Ally-Of	Gandalf
Frodo	Enemy-Of	Saruman
Gandalf	Ally-Of	Frodo
Gondor	At-War-With	Isengard
Gondor	Has-Alliance-With	Rohan
Isengard	Has-Alliance-With	Mordor
Minas Tirith	Capital-Of	Gondor
Minas Tirith	Located-In	Gondor
Mordor	Has-Alliance-With	Isengard
Mount Doom	Located-In	Mordor
Osgiliath	Located-In	Gondor
Rohan	At-War-With	Isengard
Rohan	Contains	Underharrow
Rohan	Has-Alliance-With	Gondor
Saruman	Ally-Of	Sauron
Saruman	Enemy-Of	Frodo
Saruman	Is-From	Isengard
Sauron	Ally-Of	Saruman
Sauron	Is-From	Mordor
Underharrow	Located-In	Rohan

Table 1.1: An example knowledge graph featuring information about *The Lord of the Rings* [92], expressed in tabular format as triples.

and reason on, data in the graph [62, 97]. While embeddings have multiple potential applications, KGEMs are conventionally built to perform what is called the Link Prediction (LP) Task [2, 62, 70, 97]. In link prediction, the goal is to predict either:

1. the object of a triple given the subject and the predicate, or
2. the subject of a triple given the object and the predicate.

For example, a KGE model could attempt to predict the links (*Sauron*, *Ally-Of*, ?) or (?, *Enemy-Of*, *Frodo*) in keeping with the example in Figure 1.1.

In practice, this formulation of link prediction is highly relevant to applied domains, particularly in the realm of biomedicine and bioinformatics. Many biomedical questions, such as drug-disease and drug-protein interaction prediction, are natively modelled using the aforementioned formulation of link prediction in the recent state-of-the-art [9, 60, 105]. It is of similar note that KGEMs are the most commonly used tool for this, and related, knowledge graph-based tasks in biomedical literature [9, 14, 16, 30, 32, 59, 60, 89, 105, 108].

This formulation of link prediction as a scientifically-relevant task is not unique to biology, but can be found in any domain in which:

- knowledge is modelled as a knowledge graph, and
- scientifically-relevant information can be expressed as predicting a missing link in that knowledge graph.

Many such domains already model their information in knowledge graphs (for example, computer networking [51], linguistics [10, 93], general knowledge [11, 54, 90, 93], climate science [100], and biological / biomedical sciences [6, 14, 18, 23, 32, 39, 56, 99, 108]). This opens up many possible applications of link prediction, both in the biomedical domain and outside of it.

From a machine learning point of view, training knowledge graph embedding models to perform link prediction, regardless of the domain in which link prediction is being applied, requires several different components [2, 62, 97]:

1. **the scoring function**, which defines how the KGEM assigns a plausibility score to every triple,
2. **the negative sampler**, which defines how counterexamples are created during KGEM training to help it learn,
3. **the loss function**, which defines the KGEM’s metric for error that it should minimise over the process of learning, and
4. **the other hyperparameters**, such as the dimensionality of embeddings and the number of counterexamples to sample.

In this work, all of these items are collectively referred to as “KGE hyperparameters”. While this definition departs from the traditional literature definition of hyperparameters (which would exclude the scoring function, negative sampler, and loss function) [2, 62, 97], those are included here to emphasise that they are, in effect, all configurable parts of a broader model formulation.

The plurality of hyperparameter choices means that the KGE-based approach is not without its own limitations: correctly choosing an effective and efficient hyperparameters requires a large-scale, time-consuming hyperparameter search for each individual knowledge graph that a KGEM will be trained on [2]. Such hyperparameter search is based on the implicit assumption that pre-hoc determination of optimal hyperparameters cannot be reliably done for diverse KGs.

In response, several research groups have attempted to give a more structured characterisation of how these various hyperparameter choices affect KGE performance as a function of various properties of the KG being learned [2, 45, 61, 70, 73]. Some of these studies have

been specific in their approach, looking at loss functions [61], negative samplers [45] or scoring functions [70]; others are more general, attempting to understand KGE systems at a broader level [2, 73].

However, despite the substantial progress made in this area, none of these approaches have been able to produce a unifying hypothesis for KG learning. Neither has existing literature addressed hyperparameter selection for hyperparameters other than the scoring function, negative sampler, and loss function [2, 45, 61, 70, 73]. While existing works have attempted to map elements of KG structure to KGEM performance and link prediction [9, 25, 33, 45, 58, 70, 71, 73, 74, 110], to the extent of the knowledge of the author no work has done so in a systematic, generalisable level.

This work attempts to fill this gap in the state-of-the-art with a graph-structure-first perspective. For the purposes of this work, graph structure is defined as a measure of local graph connectivity patterns, in keeping with conventional literature usage [9, 25, 45].

For example, take Table 1.2, which describes the degrees of all nodes and the frequencies of all edges in the example knowledge graph given in Figure 1.1. Any one of these degree or frequency values would be referred to as a graph structural feature. Taken together, we refer to the set of all of these features as a representation of graph structure. A more detailed description of all the graph structural features considered in this work is given in Chapter 3.

This work uses a graph-structure-first perspective to formulate the Structural Alignment Hypothesis, which is the crux of this thesis. The Structural Alignment Hypothesis states that KG learning and link prediction can be modelled in terms of graph structure. The Structural Alignment Hypothesis is then applied in the context of KGEMs to create the Structural Alignment Framework, which directly models KG learning and link prediction as a function of graph structure, and gives a clear method for mapping from specific structural elements to hyperparameter preference, KGEM performance, and link prediction results.

The Structural Alignment Framework is evaluated based on its ability to produce actionable results in:

1. structure-based simulation of KGEMs; i.e. being able to summarise and predict the output of KGEMs on unseen hyperparameters and on unseen KGs from graph structural features of the training KG, and
2. structure-based link prediction on diverse KGs; i.e. being able to perform link prediction using structural features describing the triples being predicted.

With regards to point 1) above, this work presents a system called “Topologically-Weighted Intelligence Generation” (or TWIG) as an instantiation of the Structural Alignment Framework that takes as input:

1. hyperparameters used to run a KGEM, and
2. the structure of the KG on which the KGEM was run,

and outputs the expected performance of the knowledge graph embedding model on each KG and set of hyperparameters, both locally (at the level of individual link-prediction queries) and globally (at the level of overall KGEM performance). In other words, TWIG is created to simulate the output of KGEMs in terms of graph structure, without actually using node or edge embeddings. The crux of this design choice is based on the idea that if structure is sufficient to simulate KGEM output, then inversely KGEM output can necessarily be phrased in terms of KG structure, as posited by the Structural Alignment Hypothesis. The author makes no claim that this is the *only* or *necessary* way to describe KGEM output. However, since the Structural Alignment Hypothesis posits *sufficiency*, not *necessity*, this simulation-based perspective allows direct testing of the sufficiency of structure to model KGEMs, and therefore enables empirical analysis of the Structural Alignment Hypothesis.

The aforementioned perspective, while it does provide some evidence for Structural Alignment, is not enough on its own to make the full Structural Alignment claim posited in this thesis. As a result, an evaluation of point 2) above is provided. This work presents a second instantiation of the Structural Alignment Framework, based on TWIG, called “Topologically-Weighted Intelligence Generation for Inference” (or TWIG-I). TWIG-I proposes to replace learned embeddings of KGEMs with structural features of a graph, using those features directly for link prediction rather than for simulation of KGEMs. As such, TWIG-I directly maps graph structure to the truth value of link prediction queries.

Similar to TWIG, evaluation of TWIG-I is undertaken to allow claims about the *sufficiency* of structure to describe, and perform, link prediction. If link prediction can be performed as a structural task, then graph structure is sufficient to model link prediction even without learned embeddings.

The results of both of these studies, on TWIG and on TWIG-I, support the Structural Alignment Hypothesis by showing that structure is sufficient to model KGEMs, KGEM hyperparameter preference, and the link prediction task itself. It is further shown that this structure-first perspective allows transfer learning between diverse KGs, something that KGEMs cannot achieve because their embeddings are necessarily KG-specific. Towards this end, it is shown that TWIG is able to use structural knowledge to model hyperparameter preference on unseen hyperparameter combinations and on unseen KGs, and that TWIG-I can achieve more accurate results in link prediction when pre-trained on KGs from other domains.

Based on these results, the author proposes that the Structural Alignment approach is

sufficient to explain the KGEM learning and link prediction, and to present a unified view of KG learning in terms of KG structure, in the general case. Finally, the unified perspective of Structural Alignment, as well as the mapping from KG structure to link prediction performance, is presented and analysed in terms of its broader impact and implications.

1.2 Research Question

This work is guided by one central research question:

To what extent can KG learning and link prediction be modelled as a function
of graph structure?

This guiding research question, in essence, asks whether the Structural Alignment Hypothesis holds. However, while this question clearly defines the domain, it does not provide clear direction as to how it can be empirically answered. As such, it is split into two sub-questions for clarity:

1. To what extent can KGEM hyperparameter preference and KGEM performance be modelled as a deterministic function of knowledge graph structure?
2. To what extent can the link prediction task be performed using a deterministic function of that graph’s structure?

This work proposes evaluation of the TWIG system (which simulates KGEMs to model hyperparameter preference and link prediction performance as a function of graph structure) to address the first of these sub-questions, and evaluation of TWIG-I (which learns to construct a deterministic function mapping graph structure to link prediction answers) to address the second. Note that in answering the first sub-question above, TWIG necessarily creates a structure-to-performance mapping that provides insights into what KG structures are learned and, therefore, how to construct KGs that contain a maximal number of learnable substructures.

Finally, it is important to note that all studies in this thesis aim to show feasibility, not optimality. While the results of this work do in many cases exceed state-of-the-art performance, they are presented primarily as evidence that structure-based modelling and learning of knowledge graphs is possible. The author expects that further research in the domains of TWIG and TWIG-I will lead to significant opportunities for further optimisation of their model formulations and overall performance.

1.3 The Structural Alignment Framework

The Structural Alignment Hypothesis is the hypothesis that learning on knowledge graphs can be modelled as a function of KG structure. Importantly, the need to explicitly or implicitly use the concept of node labels or edge labels in the graph in order to model KG learning is explicitly left out of this hypothesis.

To understand the Structural Alignment Hypothesis, start with the example graph given in Figure 1.1. The process of knowledge graph embedding involves assigning, to each node and each edge, a unique learnable embedding vector that represents that node / edge. An example of how embeddings represent, and map to, nodes and edges is shown in Figure 1.2.

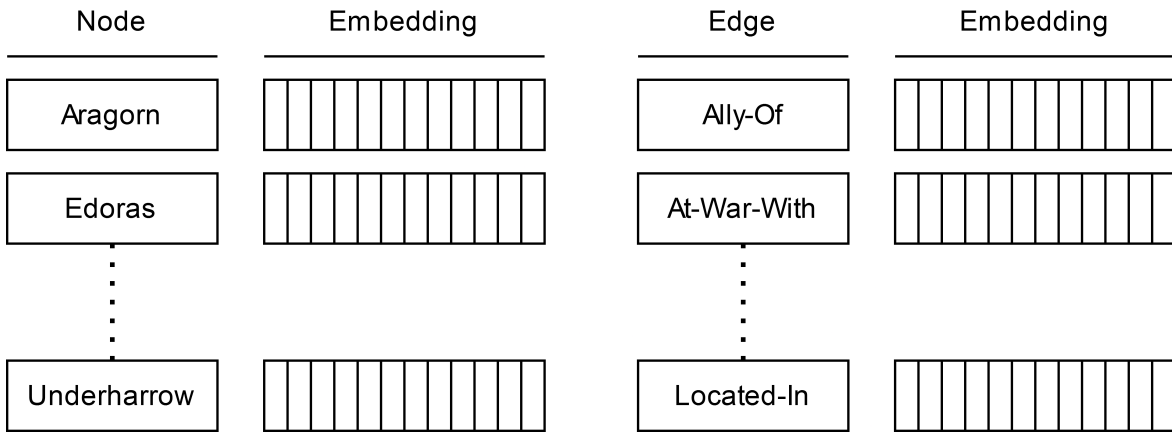


Figure 1.2: An example of knowledge graph embedding based on the KG in Figure 1.1. In KGEMs, each node and edge are mapped to a unique embedding vector that describes them.

As a result of the 1:1 label:embedding mapping, every node and edge embedding is in effect a numeric code that represents that node or edge’s label. The value of these embeddings are then used to perform link prediction, following the standard KGEM pipeline [62, 97].

When referring to Structural Alignment as describing learning on KGs as a function of structure, this sort of 1:1 mapping is specifically excluded from use. In other words, non-unique graph structural features are used to describe parts of a KG such that recovery of the original labels of the graph is not possible. A simple example of this is given in Table 1.2.

Node degree, being the number of predicates incident on a node, and predicate frequency, being the number of times a predicate occurs in the KG, are both graph structural features in that they describe elements of the connectivity patterns of the graph. Also note that, as a direct result of using graph structural features rather than embeddings, there is no longer a 1:1 label:value mapping. Most values describing either the degree of a node, or the frequency of a predicate, are non-unique such that the original label of each cannot be obtained from the structural values.

Therefore, the Structural Alignment Hypothesis can be rephrased as so: “learning on

Node	Degree
Aragorn	2
Edoras	2
Frodo	4
Gandalf	3
Gondor	7
Isengard	5
Minas Tirith	2
Mordor	4
Mount Doom	1
Osgiliath	1
Rohan	7
Saruman	5
Sauron	3
Underharrow	2

(a) The degree of all nodes in the example knowledge graph.

Predicate	Frequency
Ally-Of	5
At-War-With	2
Capital-Of	2
Contains	1
Enemy-Of	2
Has-Alliance-With	4
Is-From	3
Located-In	5

(b) The frequency of all edges in the example knowledge graph.

Table 1.2: An example of structure-based annotation of a knowledge graph, using the example knowledge graph given in Figure 1.1. Nodes and edges are mapped one-way to non-unique structural values.

knowledge graphs can be modelled in terms of statistics describing graph structure (and without the use of learned embeddings).”

The Structural Alignment Framework is the result of applying the Structural Alignment Hypothesis to the analysis of KGs and link predictors. Specifically, it consists of two phases:

1. selection of a set of graph structural features to use from the universe of all possible graph structural features, and
2. use of those graph structural features to model an aspect of KG learning, such as KGEM simulation or performing structure-based link prediction.

A diagrammatic overview of the Structural Alignment Framework is given in Figure 1.3.

From this formulation of the Structural Alignment Framework, it is clear that many different instantiations of it are possible based on the selection of a different set of structural features. In essence, any Structural Alignment Framework is principally defined by what structural features it uses to model KG structure.

In order to examine Structural Alignment, therefore, a representative set of graph structural features must be chosen. In this work, a single Structural Alignment Framework is instantiated based on a set of structural features noted as most relevant to link prediction in the literature. This framework is then used to test the Structural Alignment Hypothesis. A full description of all structural features used, and the Structural Alignment Framework built from them, can be found in Chapter 3. This instantiation is then tested in two settings: TWIG (in Chapter 4), for predicting KGEM output and hyperparameter preference

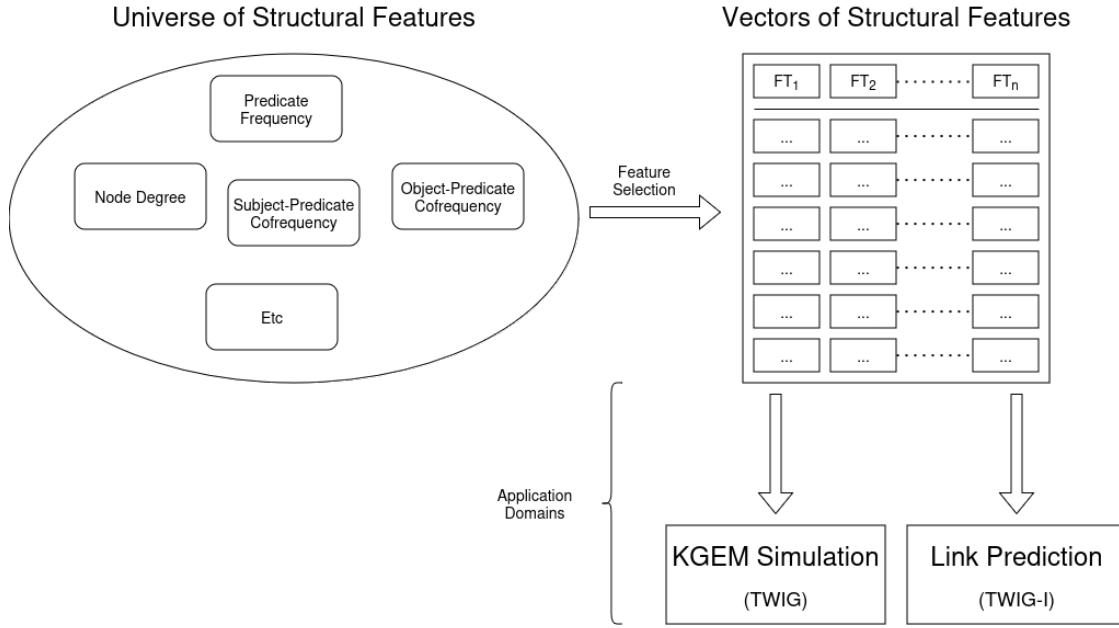


Figure 1.3: An outline of the Structural Alignment Framework. The Framework begins by taking a set of graph structural features out of the set of all possible graph structural features. It then uses those structural features to directly model KG learning, such as through KGEM simulation or structural link prediction.

as a function of KG structural features, and TWIG-I (in Chapter 5), for performing link prediction as a function of KG structure.

1.4 Contributions and Impact

The major contribution of this work is the Structural Alignment Framework, which provides a way to map specific elements of KG structure to the results of learning on that KG – such as KGEM performance, hyperparameter preference, and link prediction results. This contribution is both theoretic, in helping to build a deeper understanding of KG learning and KG structure, and practical, in building tools that exploit understanding of Structural Alignment to assist in KGEM construction and performing the link prediction task.

Specifically, the author anticipates that Structural Alignment will have three key impacts on the state of the art:

1. on theory, increased understanding of KGEM learning and link prediction as structural tasks,
2. on application, the ability to apply TWIG to perform pre-hoc hyperparameter selection and performance prediction for KGEM models, and
3. on application, the creation of a TWIG-I as a transfer-learning compatible, structure-based link predictor and as a baseline for future link prediction experiments.

It is further anticipated that Structural Alignment will impact two key domains:

1. on researchers, by contributing to the theory of KG learning and explaining the Structural Alignment hypothesis, and
2. on industry and in applied settings, by providing a framework that can be used to re-analyse existing models and build new KGs that account for structural learnability.

Finally, it is notable that Graph Foundation Models (GFMs), whose goal is to enable cross-graph transfer learning, have recently begun to attract interest in the general domain of graph learning [52, 101]. Seeing as Structural Alignment provides an explicit, extensible framework for cross-graph transfer learning based on graph structure, it is anticipated that Structural Alignment’s influence on the field will also occur through impact on the development of graph foundation models.

All code and data produced as a part of this thesis, notably TWIG and TWIG-I, are released as open-source. Code and data availability is described in the Code and Data section.

1.5 Overview of the Chapters

Chapter 2 will give a detailed description of the literature and the current state of the art in knowledge graphs, knowledge graph embedding models, and link prediction, including examples and the major results of past experiments. Chapter 3 will present, in detail, Structural Alignment and the Structural Alignment Framework, including its theory and how it is instantiated.

The following two chapters will instantiate and evaluate the Structural Alignment Framework. Chapter 4 will examine Structural Alignment by presenting structure-based simulation of KGEM-based link predictors and KGEM hyperparameter preference using TWIG, thereby showing that they can be modelled as structural learners. Chapter 5 will examine Structural Alignment through the lens of TWIG-I, a structure-based link predictor that replaces learned embeddings with fixed graph structural features for learning.

The final chapter, Chapter 6, will summarise and conclude the thesis, and provide future directions for research in this area. Several appendices are included following the conclusion to provide further data, experiments, and explanations that are not necessary for the main text, but nonetheless remain relevant for further research.

Finally, this thesis was written bilingually, with the main document in English and an informal extended summary in Irish. As a part of the work of translating this thesis into Irish, the author compiled an Irish-language dictionary of computer science terms. This dictionary,

called *An Foclóir Tráchtais*, can be found in the Irish-language extended summary, as well as online at <https://focloir-riomheolaiochta.github.io>.

2. Background and State of the Art

This chapter describes the state-of-the-art in modern Knowledge Graphs (KGs), Link Prediction (LP), and Knowledge Graph Embedding Models (KGEMs). Its purpose is both to provide a survey of the general field of link prediction on knowledge graphs, and to define each relevant sub-domain in detail to provide a foundation for further discussion and analysis later in this thesis. It is organised as follows:

1. Section 2.1 describes knowledge graphs in general, including the most common benchmark KGs, KGs used in various application domains, and the properties of KGs;
2. Section 2.2 describes the link prediction task in detail – how it is formulated and how it is evaluated, as well as the literature-standard methods and metrics for evaluation;
3. Section 2.3 describes state-of-the-art knowledge graph embedding models and how they are used to perform link prediction; and finally
4. Section 2.4 synthesises the state-of-the-art knowledge in the field of how KGs, KG structure, KGEM hyperparameters, and KGEMs relate to each other. This is meant to provide a clear basis for formulation of the Structural Alignment Hypothesis, which will be described in Chapter 3.

2.1 Knowledge Graphs

This section provides an overview of knowledge graphs and their position in the state of the art. Section 2.1.1 begins with a definition of knowledge graphs and their ontologies, including how they model and represent information. Section 2.1.2 provides an overview of how KGs are used in state-of-the-art research and application domains, including for link prediction, as well as which KGs are considered benchmarks for the link prediction task.

2.1.1 Data Modelling in Knowledge Graphs

Knowledge graphs are directed, labelled graphs that model data as statements called triples [33]. Triples take the form (s, p, o) , where s is the subject (or head) node, o is the object (or tail) node, and p is the directed predicate (or edge) that describes how s relates to o [33]. Every triple in a KG represents an atomic unit of knowledge [33]. The direct result of this is that the information content of a KG is present principally in its individual triples, and in how they are connected to each other.

An example KG is depicted in Figure 1.1, and its triples are shown in tabular form in Table 1.1.

In addition to the triples of a KG, many (but not all) knowledge graphs also have ontologies, which are descriptions of the logical semantics of a KG [33, 41]. An ontology may define, for example:

- **The domain and range of a predicate.** Predicates can be labelled by their domain (what set of nodes can act as subjects) and range (what set of nodes can act as their objects).
- **Logical relations among predicates.** Predicates can be annotated as symmetric or transitive, or as inverses of other predicates, among other possible logical annotations.
- **Type hierarchies of nodes.** Nodes can be defined as representing or belonging to various types in a type hierarchy. For example, nodes could be labelled as type “Person” with possible sub-types such as “Hobbit” or “Elf”.

Many other possible ontological annotations exist [33]. Taken together, this means that knowledge representation in KGs is managed by two constructs:

- **Triples**, which represent atomic units of knowledge in the KG, and
- **An ontology**, which describes the logical relationships of nodes and predicates in a KG.

This data model has a direct impact on link prediction. Link prediction is defined at the level of triples and only asks for the link prediction model to complete triples [2, 28, 57, 62, 96, 97, 109]. Because triples are so extensible, but also so consistent in structure, link predictors have very well-defined, simple, and modular ways they can deal with learning at the triple level [2, 28, 57, 62, 96, 97, 109].

On the other hand, ontologies are not specifically required for the link prediction task. The breadth of different logical constructs modelled by each ontology results in heterogeneous methods by which different aspects of an ontology are or are not taken into account in link prediction in the state-of-the-art [29, 35, 46, 53].

Overall, this means that the triples stream of data in KGs is very well modelled and understood by the state-of-the-art in link prediction, but that ontological modelling of KG information content is not, yet, as heavily developed or researched in the context of link prediction. While the literature shows that ontological information can be effectively applied to help solve link prediction problems [29, 35, 46, 53], the majority of work in the state-of-the-art of link prediction research and application remains focused on triples-only methods [2, 62, 96, 97, 109]. For this reason, in this work focus is placed on triple-based models of information content in KGs, and on triple-based link prediction models.

2.1.2 Application Domains of Knowledge Graphs

Knowledge graphs' graphical format allows KGs to naturally represent a large variety of real-world data, including social networks, computer networks, biological networks, linguistic data, climate data, general knowledge, and much more [6, 13, 14, 18, 23, 32, 33, 39, 40, 42, 51, 54, 56, 75, 88, 90, 91, 93, 99, 100, 108]. A general overview of knowledge graphs and their domains can be found on the Linked Open Data (LOD) Cloud (<https://lod-cloud.net/>), which serves to collect references to KGs across various domains and annotate how they relate to each other [20]. The LOD Cloud contains a total of 1,650 KGs as of the time of writing from 9 different domains: Cross-Domain, Geography, Government, Life Sciences, Linguistics, Media, Publication, Social Networking, and User-Generated KGs [20]. A visualisation of the full LOD Cloud is given in Figure 2.1.

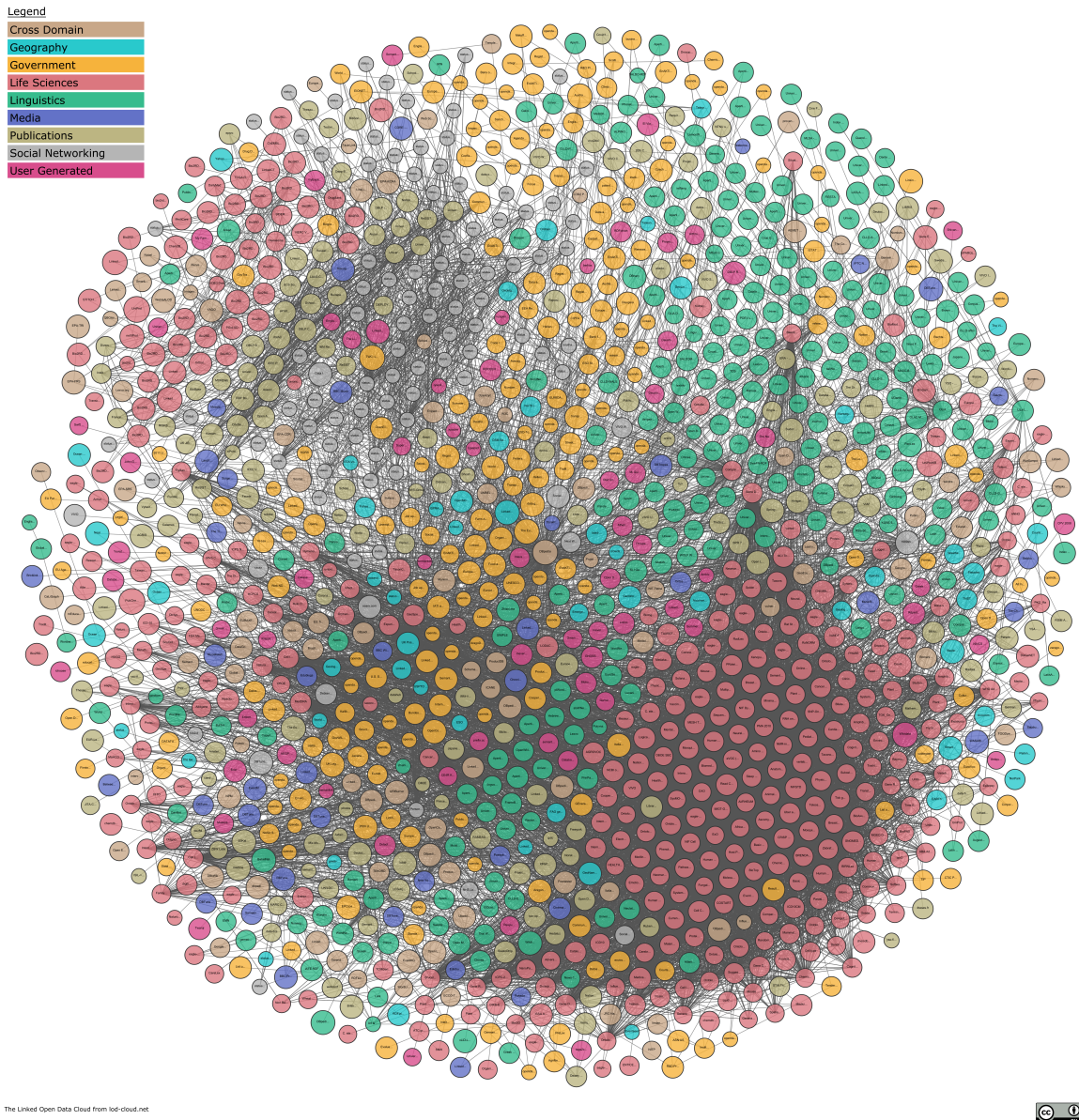


Figure 2.1: A visualisation of KGs in the LOD Cloud and the sources from which they were extracted, reproduced with permission from the Linked Open Data Cloud project [20].

A summary of KGs commonly used in the literature, and their general information content domain, is given in Table 2.1 for general KGs, and in Table 2.2 for biological KGs in particular.

Knowledge Graph	Domain
CoDEX [75]	General Knowledge. CoDEX contains data from Wikipedia and Wikidata, and was constructed to serve as a link prediction benchmark. It has three versions: CoDEXSmall, CoDEXMedium, and CoDEXLarge, distinguished by size.
ConceptNet [90]	General Knowledge. ConceptNet contains general-knowledge data modelled in multiple languages.
Countries [13]	Maps. Countries contains statements describing which real-world countries border each other, and which continents they are in.
DBpedia50 [88]	General Knowledge. DBpedia50 contains general-knowledge data extracted from DBpedia and Wikipedia.
FB15k [11]	General Knowledge. FB15k models a wide range of general facts about the world, and previously was a standard benchmark for link prediction.
FB15k-237 [93]	General Knowledge. FB15k-237 is a subset of FB15k designed to be a stronger benchmark for link prediction. Along with WN18RR, it is now one of the standard link prediction benchmark KGs.
Kinships [42]	Anthropology. Kinships models familial relations among members of the Alyawarra tribe in Australia.
Nations [3]	Politics. Nations models political and economic relations among the world’s nations.
OpenEA [91]	General Knowledge. OpenEA is a general knowledge KG constructed from DBpedia, Wikidata, and YAGO3, designed as a benchmark for entity alignment (predicting which entities in a KG are identical).
WN18 [10]	Linguistics. WN18 models words and their relationships to each other, and previously was a standard benchmark for link prediction.
WN18RR [93]	Linguistics. WN18RR is a subset of WN18 built to be a stronger benchmark for link prediction. Along with FB15k-237, it now is a standard benchmark KG for link prediction.
YAGO3-10 [54, 55]	General Knowledge. YAGO3-10 models general knowledge from Wikipedia, integrating multilingual data into a single KG.

Table 2.1: A summary of the domains of knowledge modelled by various common KGs.

Despite this preponderance of biological KGs in literature and application, other (typically non-biological) KGs have become standard benchmarks for the evaluation of link prediction models. Principally, FB15k-237 and WN18RR [93] are the go-to benchmark datasets for link prediction, with results on them reported in essentially every link prediction study [2, 38, 61, 73]. Both of these benchmarks are revised versions of the original FB15k [11] and WN18 [10] datasets, which were previously used as link prediction benchmarks in KGE studies [2, 38, 45].

Many other KG benchmarks exist, but no others have the universal status that FB15k-237 and WN18RR do. Among these others are UMLS [56], Countries [13], CoDEX [75], DBpedia50 [88], OpenEA [91], Kinships [42], and YAGO3-10 [54]; all of which are contained in the PyKEEN library for KGs and KGEMs [3]. Within the biological domain, KEGG [39], and DrugBank [99] are particularly commonly used as benchmark knowledge graphs for link prediction. This is in addition to OpenBioLink [14], PharmKG [108], and HetioNet [32],

Knowledge Graph	Domain
Bio2RDF [6]	Biology. Bio2RDF is a mash-up of many different biological KGs containing data from a wide range of biological and biomedical domains.
BioPortal [63]	Biology. BioPortal contains a large variety of biomedical ontologies that can be used to annotate biological KGs.
CTD [23]	Biology. CTD is a biomedical KG that models relationships between genes, diseases, chemicals, and environmental exposure to chemicals.
DrugBank [99]	Biology. DrugBank models information about pharmaceutical drugs, including how they work, what they interact with, and what they target.
HetioNet [32]	Biology. HetioNet is a large KG built by integrating data from millions of biomedical studies on diseases, cells, molecules, and drugs.
KEGG [39]	Biology. KEGG is a KG that describes genetics, genes, and genomics.
OpenBioLink [14]	Biology. OpenBioLink is a large-scale biomedical KG designed as a benchmark for link prediction.
PharmKG [108]	Biology. PharmKG is a biomedical KG describing the relations between genes, drugs and diseases.
PrimeKG [18]	Biology. PrimeKG is a large biological KG containing information on diseases, proteins, biological pathways / processes, anatomy, and medical drugs, among other biomedical data.
UMLS [56]	Biology. UMLS describes medical terms and their groupings.

Table 2.2: A summary of the domains of knowledge modelled by various common biological KGs.

which were explicitly created to be used for link prediction in biology.

This leaves a huge amount of space for KG-based machine learning in general, and link prediction in particular, to fill. The following section will discuss link prediction on KGs, making note specifically to the use of various benchmark KGs in the literature and how various KGs (and especially biological KGs) have defined the state-of-the-art in link prediction for various applications.

2.2 Link Prediction

Link prediction refers to the task of predicting new knowledge (triples) in a KG based on other (observed) information in the KG. This section gives an introduction to the link prediction task, followed by a detailed explanation of how link predictors are trained and evaluated under literature-standard settings.

2.2.1 The Link Prediction Task

The link prediction task is to predict new triples in a knowledge graph based on those already seen in it. Specifically, it is defined as answering a link prediction query, which can take one of two forms [2, 70]:

- $(s, p, ?)$, where the subject and predicate of a triple are given and the link predictor

must predict the correct object node to complete the triple, and

- $(?, p, o)$, where the object and predicate of a triple are given and the link predictor must predict the correct subject node to complete the triple.

For example, take the graph in Figure 1.1. A possible link prediction query that could be posed on this graph is $(?, \textit{Enemy-Of}, \textit{Saruman})$; in essence, this query asks a link predictor for which nodes in the graph could be described as enemies of Saruman. A pictorial overview of for this example query link prediction is given in Figure 2.3.

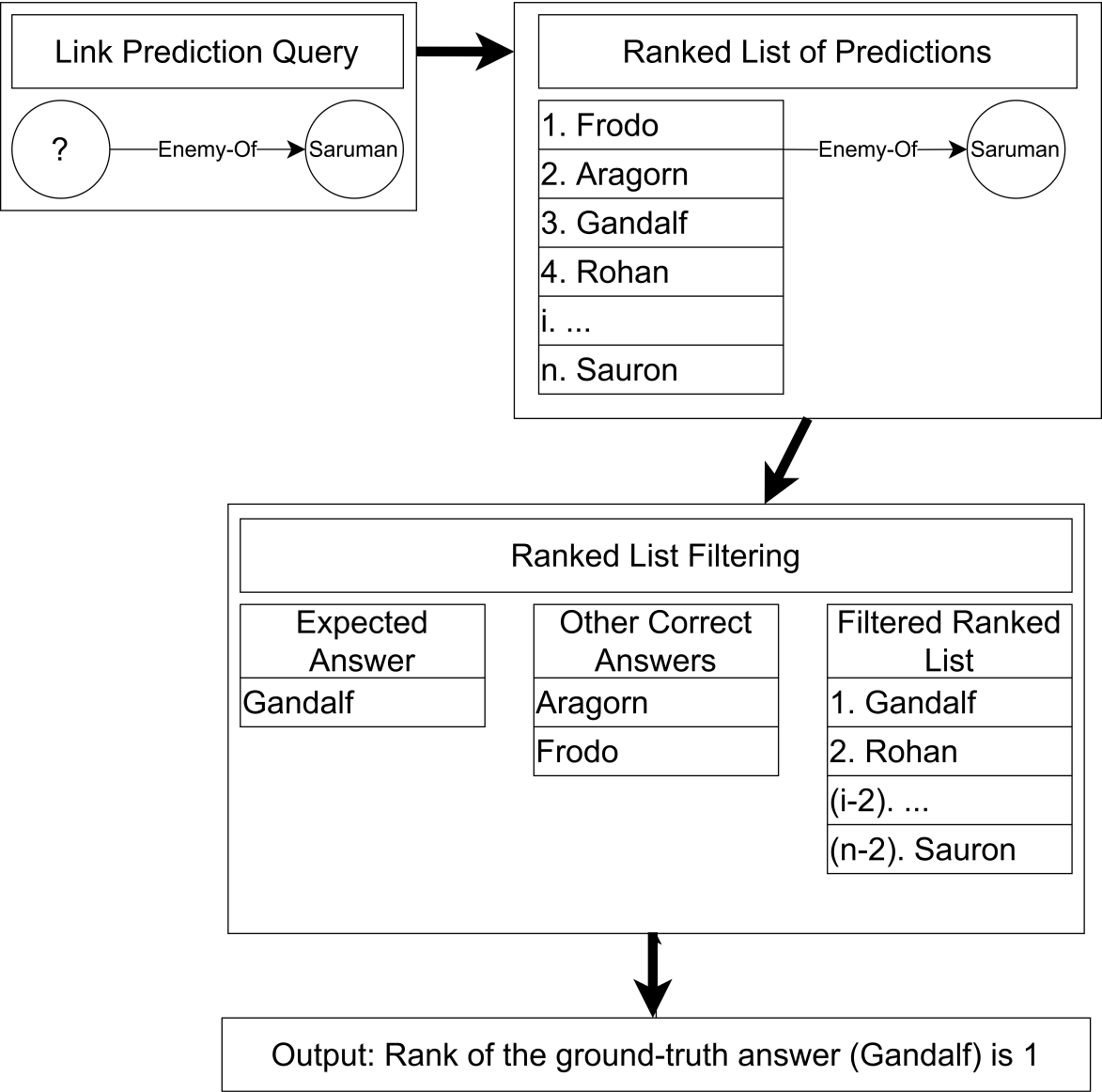


Figure 2.3: An overview of the process of link prediction in knowledge graphs.

In essentially all scenarios in which link prediction is applied, there are a plurality of possible answers, not just one. In the current example, multiple completions, including $(\textit{Aragorn}, \textit{Enemy-Of}, \textit{Saruman})$ and $(\textit{Frodo}, \textit{Enemy-Of}, \textit{Saruman})$, are equally true. Other completions such as $(\textit{Sauron}, \textit{Enemy-Of}, \textit{Saruman})$ and $(\textit{MountDoom}, \textit{Enemy-Of},$

Saruman) are incorrect. Multiple correct answers and multiple incorrect answers can be present.

This effect arises because knowledge graphs are highly heterogeneous and can allow the same relation to occur among many different nodes [33]. KGs are also assumed in most cases to contain only a fraction of all possibly true statements that could be made in their domain, an assumption called the Open World Assumption [2]. The Open World Assumption is the standard assumption for knowledge graphs, since its inverse (assuming that the KG contains all possible true statements) is considered unreasonable in most contexts [2]. In other words, the example of multiple correct and incorrect answers above is merely a logical application of the understanding that KGs are diverse in the knowledge they can represent, and necessarily incomplete.

As a result of this, link prediction is typically formulated as a task of “learning to rank” [2, 62, 70, 97]. What this means is that link predictors are built to score all possible answers to a link prediction query, such that higher scores indicate higher confidence in the plausibility of the statement [2, 62, 70, 97]. All possible completions are then sorted by score and ranked, resulting in a ranked list where elements at the start of the list (closer to index 1) are considered to be more likely true completions, and elements near the end of the list are considered more likely incorrect completions [2, 62, 70, 97]. While it is possible to formulate the link prediction task in other terms without a concept of ranking (such as reconstructing the graph’s adjacency matrix all at once as done in VGAE [44]), formulating link prediction as learning to rank has become a standard for link prediction in the literature for both KGEM methods [2, 62, 70, 97] and non-KGEM methods [49]. It is further supported by all major link prediction libraries for knowledge graphs [3, 12, 21, 50].

2.2.2 Rank-Based Evaluation of Link Predictors

Since link prediction is performed as a learning-to-rank task, its evaluation is necessarily rank-based as well [2, 49, 62, 70, 97]. This is achieved by asking a link predictor to rank possible link predictions on a set of triples that it has not seen before to test its generality. An overview of this process is given in Figure 2.4, and it is described in detail in this section.

In order to do this, all knowledge graphs used to evaluate link predictors are first split into three distinct sets: the training set T_{train} , the testing set T_{test} , and the validation set T_{valid} [2, 73]. All of these sets contain disjoint triples, which means that a triple which is present in one split is not present in any of the others [2, 73].

All training is undertaken on the training set T_{train} . In essence, the training phase is what teaches a link predictor to be able to actually perform link prediction on a given KG. Note that, since the training set is used to train a link predictor, the triples it contains are

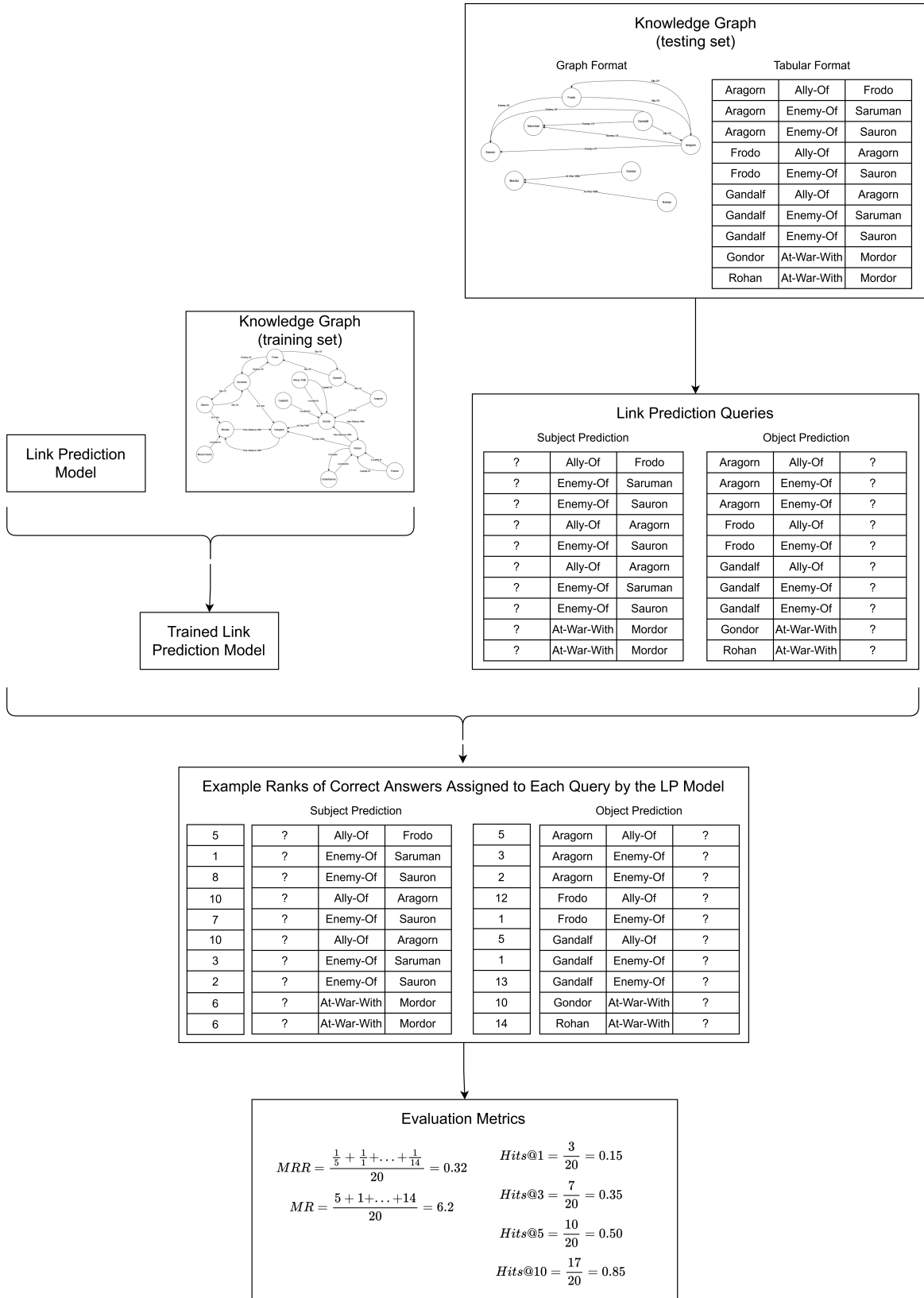


Figure 2.4: An overview of the process of training and evaluating a link predictor using rank-based evaluation.

never used in any downstream evaluation phase to avoid data leakage and unfair evaluation. The training process is described in detail in Section 2.3.2.

Evaluation of trained link predictors is done on either the validation set T_{valid} or the testing set T_{test} . [2, 73]. The validation set is used during the creation of a link predictor, when various model components and hyperparameters are evaluated and compared to see which result in the best performance [2, 73], a process described in Section 2.3.3. The testing set is used for final evaluation of the link predictor after all hyperparameters are selected, and must have never been seen before (either in training or during hyperparameter selection) in order to ensure fair evaluation.

Notwithstanding the different uses of these sets, the mathematical process of evaluating a link predictor with any of them is identical. This section will refer to the testing set always as the set being evaluated on, but the processes it describes apply without loss of generality to evaluation using the validation set as well.

An example of such a test set that could be constructed for the running example KG in Figure 1.1 is given in graph format in Figure 2.5 and in tabular format in Table 2.3. Note that none of the triples in this test set were present in the original graph. Finally, a pictorial overview of the general process of link prediction training and evaluation on this example knowledge graph is given in Figure 2.4.

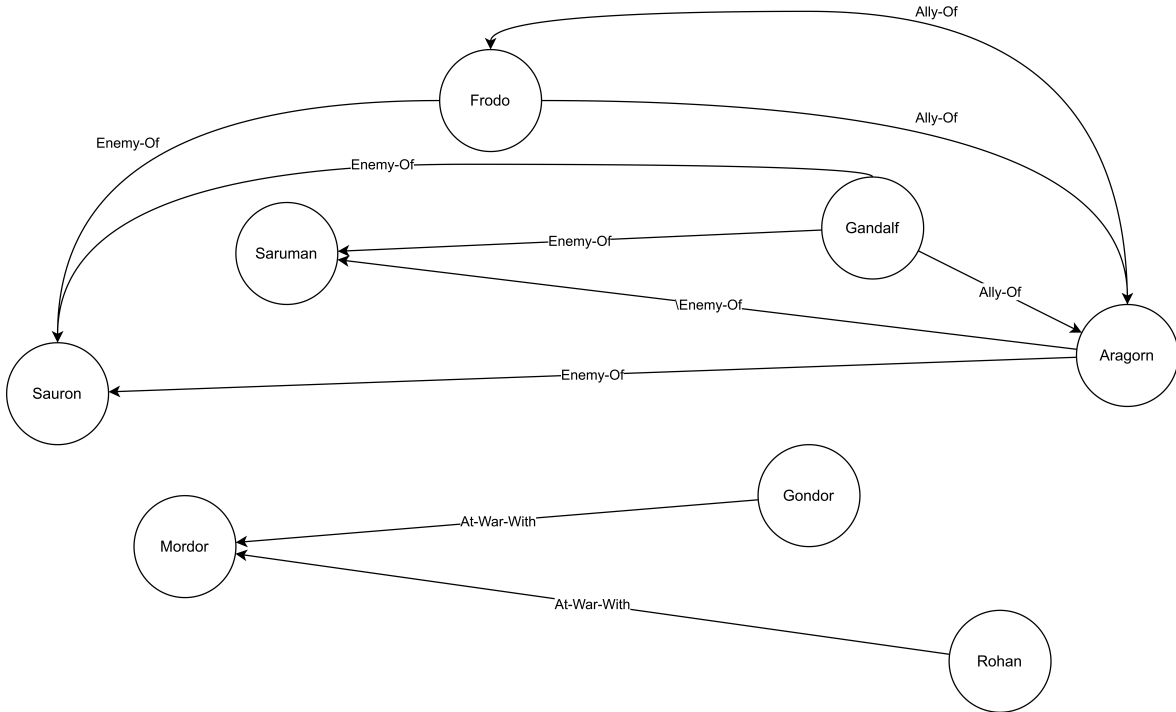


Figure 2.5: An example test set, shown as a graph that could be constructed for the example KG in Figure 1.1.

For every individual triple $(s_i, p_i, o_i) \in T_{test}$ in the testing set, two link prediction queries are posed. These are referred to as object prediction and subject prediction queries; these

Subject	Predicate	Object
Aragorn	Ally-Of	Frodo
Aragorn	Enemy-Of	Saruman
Aragorn	Enemy-Of	Sauron
Frodo	Ally-Of	Aragorn
Frodo	Enemy-Of	Sauron
Gandalf	Ally-Of	Aragorn
Gandalf	Enemy-Of	Saruman
Gandalf	Enemy-Of	Sauron
Gondor	At-War-With	Mordor
Rohan	At-War-With	Mordor

Table 2.3: An example test set, shown as a table, that could be constructed for the example KG in Figure 1.1.

queries take the following forms:

- $(s_i, p_i, ?)$, where the ground-truth object o_i is replaced with all possible nodes in the knowledge graph (including the node o_i and all other nodes)
- $(?, p_i, o_i)$, where the ground-truth subject s_i is replaced with all possible nodes in the knowledge graphs (including the node s_i and all other nodes)

An example of the link prediction queries that can be formed for the example set set in Table 2.3 is given in Table 2.4 (for object prediction) and Table 2.5 (for subject prediction).

Subject	Predicate	Object
Aragorn	Ally-Of	?
Aragorn	Enemy-Of	?
Aragorn	Enemy-Of	?
Frodo	Ally-Of	?
Frodo	Enemy-Of	?
Gandalf	Ally-Of	?
Gandalf	Enemy-Of	?
Gandalf	Enemy-Of	?
Gondor	At-War-With	?
Rohan	At-War-With	?

Table 2.4: All object link prediction queries generated for the triples in the example test set given in Table 2.3.

For both the subject and object link prediction queries generated from any one triple, the link prediction model is used to assign a plausibility score to all possible completions of the query [2]. These scores are then sorted into a ranked list such that higher-scored completions are placed closer to the start of the list at index 1. For a knowledge graph with n nodes, this means that there are n possible predictions which all must be scored and ranked.

Mathematically, the process of scoring a possible triple completion $(s_i, p_i, \hat{o}_i)$ to the query $(s_i, p_i, ?)$ is given in Equation 2.1. The process of producing a ranked list from a link prediction

Subject	Predicate	Object
?	Ally-Of	Frodo
?	Enemy-Of	Saruman
?	Enemy-Of	Sauron
?	Ally-Of	Aragorn
?	Enemy-Of	Sauron
?	Ally-Of	Aragorn
?	Enemy-Of	Saruman
?	Enemy-Of	Sauron
?	At-War-With	Mordor
?	At-War-With	Mordor

Table 2.5: All subject link prediction queries generated for the triples in the example test set given in Table 2.3.

query is given in Equation 2.2. Note that these equations also apply, without loss of generality, to link prediction queries in the form $(?, p_i, o_i)$.

$$Score_i = LP(s_i, p_i, \hat{o}_i) \quad (2.1)$$

$$R(s_i, p_i, ?) = Rank(LP(s_i, p_i, \hat{o}_i) \forall \hat{o}_i \in N) \quad (2.2)$$

In these equations, LP is the link prediction model, and $(s_i, p_i, \hat{o}_i)$ is the triple to be scored, where s_i and p_i are given and $\hat{o}_i$ is one of n -many possible completions. $Score_i$ is the score assigned to the triple by the link prediction model. $Rank$ is a function that sorts an input list to products ranks for each element, $?$ represents the element to be predicted to complete the triple, N is the set of all nodes in the KG, and $R(s_i, p_i, ?)$ is the ranked list of all possible completions of the given link prediction query.

Once a full ranked list is produced, the rank of the ground truth answer o_i (if the object is being predicted) or s_i (if the subject is being predicted) is extracted. Again, lower ranks indicate that the model considers the true answer more plausible than the alternatives. This results in, for every link prediction query, a single number reflecting how well that query was predicted.

At this point, there is one more thing to note. In some cases the same link prediction query (i.e. $(Gandalf, Enemy-Of, ?)$) can be formed from multiple different triples (i.e. $(Gandalf, Enemy-Of, Saruman)$ and $(Gandalf, Enemy-Of, Sauron)$) – in this case, both queries remain *and map to different ground truth answers*. This means that if a link prediction model predicts *Sauron* for $(Gandalf, Enemy-Of, ?)$, this will result in a ranks of both 1 and 2 being obtained, depending on which triple the link prediction query came from. Similarly, it is possible that a link predictor predicts a correct answer (from the training set,

perhaps) that is not what was expected for a given link prediction query in the test set.

In order to address this, it is standard practice in link prediction to do a filtering step [2]. In this step, all true answers that rank higher than the given answer are removed. To understand this, take an example. Suppose that the query $(Gandalf, Enemy-Of, ?)$ is generated from the triple $(Gandalf, Enemy-Of, Saruman)$. Now suppose that the link predictor predicts the most likely completions to be, in order, $Sauron, Saruman, \dots$. In the filtered setting, $Sauron$ will be removed from the ranked list since $(Gandalf, Enemy-Of, Sauron)$ has been observed to be true. The resulting list, $Saruman, \dots$ has $Saruman$ first, which means that this completion will be assigned a rank of 1. In this manner, link predictors are not ever penalised for predicting correct statements, even if a different statement was “expected”.

The final list, consisting of the ranks of all correct answers to all link prediction queries, is the basis for evaluation of link predictors. There are many ways to quantify overall performance from this list, two of which have become literature standards: Mean Reciprocal Rank (MRR) and Hits@K [2, 49, 62, 70, 97]. A third, Mean Rank, was commonly used (and still sometimes is [70]) but is no longer a literature standard, with many studies preferentially reporting Mean Reciprocal Rank and Hits@k without Mean Rank [2, 45, 61, 73].

These metrics are described in the following subsections. A pictorial overview of rank-based evaluation, including how Mean Rank, Mean Reciprocal Rank, and Hits@K are applied, is given in Figure 2.4.

Mean Rank

Mean Rank (MR) calculates, for all triples in the testing set T_{test} , the arithmetic mean of the ranks assigned to all of their link prediction queries. This can be written as shown in Equation 2.3:

$$MR = \frac{1}{n} \left(\sum_{i=1}^n rank_i \right) \quad (2.3)$$

where n is the total number of link prediction queries posed and $rank_i$ is the rank assigned to the correct answer of the i^{th} link prediction query. MR is bounded on $[1, \infty)$, where lower values indicate better performance of the link predictor.

MR gives an intuitive view of how well the link predictor is able to correctly predict answers to link prediction queries. However, since it uses the arithmetic mean, it is highly sensitive to outliers that have high ranks [2]. This conversely means it is not sensitive to low ranks where the link predictor did particularly well. This bias towards high ranks is especially true in large graphs with many nodes, where the maximum possible rank value is quite large and even a few poor predictions can significantly skew MR towards higher values.

Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) calculates, for all triples in the testing set T_{test} , the arithmetic mean of the reciprocal of the ranks assigned to all of their link prediction queries. This can be written as shown in Equation 2.4:

$$MRR = \frac{1}{n} \left(\sum_{i=1}^n \frac{1}{rank_i} \right) \quad (2.4)$$

where n is the total number of link prediction queries posed and $rank_i$ is the rank assigned to the correct answer of the i^{th} link prediction query. MRR is bounded on the interval $(0, 1]$, and higher MRR values indicate better performance of the link predictor.

MRR is the *de facto* measure of link predictor performance [2, 45, 61, 70, 73]. It is preferred because it is able to take into account the entire list of ranks, while also not being overly sensitive to outlier high-rank values in the way that Mean Rank is [2].

Hits@K

Hits@K (or H@K) calculates, for all link prediction queries created from the testing set T_{test} , the fraction of those that have a rank equal to, or less than, a given value K . For example, Hits@1 calculates the fraction of link prediction queries in which the top-ranked answer (rank 1) was the correct answer. Similarly, Hits@3 calculates the fraction of link prediction queries in which the correct answer was given a rank of 3 or below. Hits@K is defined mathematically in Equation 2.5:

$$Hits@K = \frac{1}{n} \left(\sum_{i=1}^n I(rank_i \leq K) \right) \quad (2.5)$$

where n is the total number of link prediction queries posed, I is an indicator function that returns 1 if the expression it contains is true and 0 otherwise, and $rank_i$ is the rank assigned to the correct answer of the i^{th} link prediction query. Hits@K is bounded on $[0, 1]$, where higher values indicate better performance of the link predictor.

While Hits@K remains a commonly reported link prediction metric, MRR is typically preferred because Hits@K disregards all ranks higher than K from its score, whereas MRR uses all ranks assigned to all link prediction queries in its calculation.

When Hits@K is reported, the most common values of K are 1, 3, 5, and 10 [2, 58, 70, 73].

2.2.3 Applications of Link Prediction

Link prediction can be applied to any domain whose data is modelled by a knowledge graph – which leaves a massive range of possible applications from computer networking [51] and linguistics [10, 93] to general knowledge [11, 54, 90, 93], climate science [100], and biological / biomedical sciences [6, 14, 18, 23, 32, 39, 56, 99, 108].

Recent literature has shown increasing focus on link prediction for biological and biomedical applications in particular [9, 14, 32, 89, 108]. Specifically, KGEM-based link prediction approaches have become increasingly common in the biomedical domain. A summary of the applications KGEMs and link prediction in published studies, and the dates of those studies, are given in Table 2.6.

Authors	Year	Methods	Task
Bonner et al. [9]	2022	KGEMs	Drug re-purposing to discover new medical uses for existing drugs
Celebi et al.* [16]	2019	KGEMs, Walks	Drug re-purposing to discover new medical uses for existing drugs
Gualdi et al.* [30]	2024	KGEMs, GNNs, Walks, Other	Predicting gene-disease associations (GDAs)
Mohamed et al. [60]	2020	KGEMs	Predicting which proteins were targeted by various medical drugs
Mohamed et al.* [59]	2020	KGEMs, Walks, Other	Predicting drug targets, and predicting the effects of combining multiple drugs in treatment (“polypharmacy”)
Zheng et al. [108]	2021	KGEMs, GNNs	Benchmarking of KGEMs and GNNs for biomedical link prediction
Zhang et al. [105]	2021	KGEMs, Other	Drug re-purposing to discover potential drugs to treat COVID-19

Table 2.6: A summary of applications domains and uses cases of link prediction models in the state of the art. KGEMs = Knowledge Graph Embedding Model-based methods, GNNs = Graph Neural Network-based methods, Walks = Random Walk-based methods, Other = other methods. Details on these methods are given in Section 2.3 and Section 2.3.4.

Articles marked with an asterisk (*) used KGEMs or related methods to create embeddings for a KG, but performed the actual task as a classification problem using those embeddings as input feature vectors, rather than for direct link prediction. However, the task being addressed in all cases was still to predict if relationships exist between distinct entities, meaning that the task still remains one of predicting a (single) unknown link.

Overall, Table 2.6 shows that the link prediction task is highly relevant to recent biomedical literature. Link prediction is particularly relevant in biomedicine because many biomedical questions (such as predicting which drugs can treat which diseases [9, 105], and predicting which proteins are targeted by which drugs [60]) can be directly and natively modelled as link

prediction tasks using the standard link prediction pipeline outlined in this thesis. It is for this reason that emphasis is given in this work to KGEMs for link prediction as state-of-the-art link predictors both in theory and in practical application.

Note that studies on link predictors in the general case (on standard link prediction benchmarks, i.e. FB15k-237 and WN18RR) that were performed from a comparative perspective rather than for application to an end-use domain are not included here. The details of such studies will be given in Section 2.4. The following section gives a detailed account of KGEMs, including their components and how they are trained.

2.3 Knowledge Graph Embedding Models

Knowledge graph embedding models are a class of machine learning models built to perform link prediction by learning to embed all nodes and edges in a KG as vectors. They use these vectors to score individual triples and perform link prediction [62, 97]. While other link prediction methods exist, KGEMs generally hold state-of-the-art status for link prediction [49, 96, 97], and are often go-to models for applied link prediction [9, 17, 59]. This section provides a detailed introduction to KGEMs. It begins with an overview of KGEM structure and their various components. It then introduces the three most common KGE baselines in the state-of-the-art [2, 36, 38]: ComplEx [47, 94], DistMult [102], and TransE [11]. Finally, a brief overview of non-KGEM methods for link prediction is given as a reference and to justify the focus on specifically KGEMs for link prediction in this work.

2.3.1 Components of KGEMs

Knowledge graph embedding models consist of several key components [2, 62, 97]:

- the scoring function, which assigns a plausibility score to triples;
- the negative sampler, which generates counterexample triples during training;
- the loss function, which is used to generate loss scores for propagation during training; and
- the optimiser and other hyperparameters, which specify all other configurations (such as the learning rate and the embedding dimension of embedding vectors).

Each of these components defines a different aspect of the model and how it learns, and a description of a KGEM is only complete once all components have been specified [2]. That notwithstanding, the defining feature of a KGEM is its scoring function; all other components are typically taken as being secondary to it [62, 97]. As such, in this work, the name of a

scoring function (such as ComplEx or DistMult) is taken to also refer to the KGEM using that scoring function as a whole, in keeping with the precedent in the literature. When only the scoring function specifically is meant, that will be specified fully (i.e. “the ComplEx scoring function”).

Similarly, it is traditional in the literature to refer to the negative sampler, the loss function, and the optimiser as model components and all other configuration values (such as the optimiser’s learning rate, or the dimension of embedding vectors) as hyperparameters [2]. However, since all of these model components are, functionally, alternative choices for how to construct a KGEM, **in this work all model components are referred to as hyperparameters to the KGE model**. This is done to allow brevity of expression, as the treatment of each component and (traditional) hyperparameter in this work is pragmatically identical.

An overview of all KGEM hyperparameters, and the most common options in KGEM literature and application, is given in Figure 2.6. The following sections give a mathematical description of the functions of each of these components and their role in training KGEMs. Following that, a final section goes into detail on the KGEM training loop, with end-to-end description of how each component works and interacts with all others.

Core Components	{	Scoring Function	Negative Sampler	Loss Function
		TransE	Full Random	Binary Cross Entropy
		DistMult	Bernouilli	Cross Entropy
		ComplEx	Pseudotyped	Margin Ranking
			Ontological	
Other Hyperparameters	{	Regulariser	Optimiser	Other
		L_p regulariser	Adam	Batch Size
		No regulariser	Adagrad	Embedding Dimension
				Learning Rate
				Negatives per Positive
				Regulariser Coefficient

Figure 2.6: An overview of the hyperparameters used in KGEMs.

The Scoring Function

The scoring function is the crux of a KGE model in that it determines how the KGEs themselves model data in a graph [2, 62, 96, 97]. In general, the literature splits these into three main categories [2, 62, 96, 97]:

- **Additive Models** (also called Translational Models), which use vector translations (addition and subtraction of node and edge vectors) to score triples,
- **Multiplicative Models** (also called Semantic Matching Models or Tensor Decomposition Models), which use the multiplicative properties of embeddings (such as the inner product or matrix products) to score triples, and
- **Other Models**, which may include a wide variety of other techniques – neural networks, convolutions, inductive reasoning, neural-symbolic methods, etc.

Regardless of their implementation, all scoring functions necessarily perform the same computation, defined in Equation 2.6:

$$Score(s, p, o) = f(e_s, e_p, e_o) \quad (2.6)$$

where (s, p, o) represents the subject, predicate, and object of a triple, f is the scoring function itself, and (e_s, e_p, e_o) are the embedding vector values for the subject, predicate, and object respectively. Note also that $Score$ is a scalar-valued number and all embeddings (e_s, e_p, e_o) are d -dimensional vectors. As such, the scoring function can be considered a sort of dimensional reduction – from three high-dimension vectors to a single plausibility score for the given triple.

The state-of-the-art literature describes many KGEMs, and posits many different advantages and disadvantages of each [2, 62, 70, 73, 96, 97]. Notwithstanding, three main KGEMs have surfaced as being most representative of the state-of-the-art, as well as generally high-performing, on standard link prediction benchmark datasets [2, 36, 38, 73]: TransE [11], DistMult [102], and ComplEx [47, 94]. Specifically, The ComplEx scoring function is generally considered to be among the strongest in the state-of-the-art [2, 36, 38, 47, 73]. The DistMult and (especially) TransE scoring functions are less powerful, but have been shown repeatedly to be strong baselines and are considered standard KGEMs to compare to in the state-of-the-art [2, 38, 73]. TransE is an additive scoring function, and both DistMult and ComplEx are multiplicative scoring functions [2, 11, 94, 102]. The scoring functions defined by TransE, DistMult, and ComplEx are as follows.

TransE. TransE is an additive scoring function that models relations as translations between nodes, which are modelled in turn as points in d -dimensional space. It is based

on the idea that, if a triple (s, p, o) is true, then the property $e_s + e_p = e_o$ should hold in embedding space. This idea is transformed into the scoring function given in Equation 2.7:

$$\text{TransE}(e_s, e_p, e_o) = -\|e_s + e_p - e_o\|_p \quad (2.7)$$

where e_s is the embedding vector of the subject node in a triple, e_p is the embedding of the predicate edge, e_o is the embedding of the object node, and p is a hyperparameters defining what p -norm should be used to reduce the score to a single value [2, 11]. The most common p -norms used for TransE are 1 (which represents the Manhattan distance between $e_s + e_p$ and the target object embedding e_o) and 2 (which represents the Euclidean distance between $e_s + e_p$ and the target object embedding e_o) [2]. As such, TransE’s scoring function can be interpreted as a distance-based score, where lower distances represent higher plausibility of the given triple. Note that the TransE score is negated to ensure that higher-valued scores represent more plausible, rather than less-plausible, answers.

DistMult. DistMult is a multiplicative scoring function that models nodes as vectors and predicates as diagonal matrices. It computes triple scores as the matrix product of subject, predicate, and object embeddings [2, 102]. The scoring function used by DistMult is given in Equation 2.8:

$$\text{DistMult}(e_s, e_p, e_o) = e_s \cdot e_p \cdot e_o \quad (2.8)$$

where e_s is the embedding vector of the subject node in the triple, e_p is the embedding of the predicate edge, and e_o is the embedding of the object node [2, 102]. It is critical to note that, due to the properties of matrix multiplication, this equation is necessarily symmetrical; the identity $e_s \cdot e_p \cdot e_o = e_o \cdot e_p \cdot e_s$ holds in all cases [2]. The result of this is that DistMult models all relations as symmetric and cannot distinguish the triples (s, p, o) and (o, p, s) [2]. Despite this apparent weakness, however, it has been shown to have strong empirical performance even on KGs with non-symmetric relations [2, 38, 73].

Complex. ComplEx is a multiplicative scoring function that models relations nearly identically to DistMult, with one critical exception: it uses complex-valued embeddings to create an asymmetric scoring function [2, 94]. The result of this is that it is explicitly able to distinguish asymmetric relations, and scores the triples (s, p, o) and (o, p, s) differently, rather than identically. The scoring function defined by ComplEx is given in Equation 2.9:

$$\text{ComplEx}(e_s, e_p, e_o) = \text{Re}(e_s \cdot e_p \cdot e_o) \quad (2.9)$$

where Re is a function that extracts the real component of a complex number, e_s is the

embedding vector of the subject node in a triple, e_p is the embedding of the predicate edge, and e_o is the embedding of the object node [2, 94]. By introducing this asymmetry, ComplEx has become generally considered to be among the strongest KGEM scoring functions [2, 36, 38, 47, 73].

For all scoring functions, as outlined in the previous section on link prediction, higher scores indicate greater plausibility that a triple is true. Therefore, it is expected that training will result in a KGEM assigning higher scores to triples it has seen (or triples that resemble those it has seen) and lower scores to triples it has not. However, this task in itself presents a problem – since a KG contains only positive facts for a given domain, there are no pre-made counter-examples it can use to train the scoring function to distinguish true and false triples. Instead, the negative sampler is responsible for creating counter-examples for learning, as described in the next section.

The Negative Sampler

The negative sampler is responsible for generating false triples as counterexamples during training [2, 62, 96, 97]. For a single input triple (s, p, o) , the job of a negative sampler is to produce n -many negative counterexamples in the form (s', p, o) (subject corruption) or (s, p, o') (object corruption) in which either the subject or object node is corrupted to some other node in the graph.

While there are many ways in which this can be done, most methods use random corruptions of existing, known-true triples from the training set. The simplest possible form of negative sampling is full-random (or basic) negative sampling, in which a node is chosen, completely at random, from the KG to serve as a corruption for the correct node [2, 45, 73].

As an example of negative sampling, the ground truth triple $(Frodo, Ally-Of, Gandalf)$ could result in the negative triples $(Sauron, Ally-Of, Gandalf)$ (by subject corruption) and $(Frodo, Ally-Of, Saruman)$ (by object corruption). However, “negatives” that are actually true (such as $(Frodo, Ally-Of, Aragorn)$), as well as nonsensical negative triples (such as $(Frodo, Ally-Of, MountDoom)$), are entirely possible under full-random negative sampling (and under some other negative sampling paradigms) [2, 45, 73].

Many other negative sampling paradigms exist, most of which generally outperform full-random sampling particularly by avoiding generating (as many) true statements or nonsensical statements by accident [2, 45, 73]. The most common of negative sampling methods are as follows:

- **Full Random (or Basic):** A triple (s, p, o) is corrupted into (s', p, o) or (s, p, o') by choosing another replacement node with uniform random sampling of all possible nodes

in the KG [45].

- **Bernoulli Random:** Akin to full random but using weighted probabilities to determine how many subject or object corruptions to generate [98]. Specifically, the subject (or object) node is replaced preferentially if it is seen in fewer other relationships [98]. This means that if n -many negatives are wanted for a triple, and if the predicate in that triple maps one-to-many subjects to objects, then many more subject corruptions will be samples than object corruptions [98].
- **Pseudotyped Random:** Akin to full random, but using a set of filters for what nodes can be used to corrupt any given triple [45]. Specifically, only nodes that have been observed as the subject of a given predicate (for subject corruption) or as the object (for object corruption) are considered candidates that can be chosen [45]. This prevents most nonsensical statements (for example, $(Frodo, Ally-Of, MountDoom)$) from being generated.
- **Logic or Ontology-based:** Uses a given set of logical rules (such as an ontology) for a knowledge graph to generate negatives that logically must be false [35]. This necessarily avoids the possibility of nonsensical negative triples (for example, $(Frodo, Ally-Of, MountDoom)$) from being generated [35].

The output of negative sampling is n -many negatives for each positive. Once a KGEM has a positive triple and all its corresponding negatives, it feeds the scores of both positive and negative triples into its loss function to penalise cases where the KGEM cannot distinguish the true triples from negatives. Link prediction loss functions are described in detail in the following section.

The Loss Function

The purpose of a loss function in KGEMs (and in link predictors generally) is to compute a penalty (called *loss*) for the model based on its predictions during training. This penalty is then backpropagated to update KGEM parameters and allow for learning.

Loss functions in link predictors are categorised into three distinct types: pointwise losses [2, 61], pairwise losses [2, 61], and setwise losses [2]. Each of these is described below.

- **Pointwise losses** are calculated based on the score assigned to each (true and negative) triple individually and independently [2, 61]. They take the form $loss_i = L(Score_i, flag_i)$, where $Score_i$ is the score of triple i and $flag_i$ is the expected score of triple i [2, 61]. The expected score assigned to each triple is a binary flag representing whether the triple is true (typically using a flag value of 1), or a generated negative

(typically using a flag value of 0 or -1) [2]. This means that higher scores are more plausible, and that scores above a certain threshold (typically 0.5 or 0) all indicate that the triple for which they were computed should be considered true [2].

- **Pairwise losses** are calculated based on pairs consisting of the true triple and one of the negative triples created for it at a time [2, 61]. Pairwise loss functions have the general form $loss_i = L(Score_{i,true}, Score_{i,negative_k})$, where $Score_{i,true}$ is the score of the ground-truth triple i and $Score_{i,negative_k}$ is the score of the k^{th} negative generated for triple i [2, 61]. This is computed for every true triple paired with each of negatives generated for that true triple [2, 61].
- **Setwise losses** are calculated based on the score of a true triple and the score of all negatives generated for the triple at once (rather than in pairs) [2]. These loss functions take the form $loss_i = L(Score_i, Score_{i,negatives})$ where $Score_i$ is the score of the ground-truth triple i and $Score_{i,negatives}$ is the scores of all negatives generated for triple i [2].

While there are many loss functions in each of these categories [2], focus will be placed on the most commonly used loss functions from each of these categories in the recent state-of-the-art [2, 61, 73]: Binary Cross Entropy Loss (a pointwise loss), Margin Ranking Loss (also called Pairwise Hinge Loss, a pairwise loss), and Cross Entropy Loss (a setwise loss). While other loss functions exist, these generally result in state-of-the-art performance across various KGE models and are among the most common losses attested in the state-of-the-art literature [2, 11, 24, 38, 47, 61, 73, 94, 102]. These loss functions are described mathematically below.

Binary Cross Entropy Loss. Binary Cross Entropy Loss (BCEL) is calculated in a pointwise manner as shown in Equation 2.10:

$$BCEL(Score_i, flag_i) = -(flag_i \cdot \log(\sigma(Score_i))) + (1 - flag_i) \cdot (\log(1 - \sigma(Score_i))) \quad (2.10)$$

where $flag_i$ is 1 for true triples and 0 for negatives, $Score_i$ is the score of the input triple (which may be either true or a negative), and σ is the sigmoid function [2].

Margin Ranking Loss. Margin Ranking Loss (MRL) is calculated in a pairwise manner as shown in Equation 2.11:

$$MRL(Score_{i,true}, Score_{i,negative_k}) = \lambda + \max(0, Score_{i,negative_k} - Score_{i,true}) \quad (2.11)$$

where $Score_i$ is the score of the i^{th} input triple, $Score_{i,negative_k}$ is the score of the k^{th}

negative generated for that input triple, and λ is a user-defined hyperparameter representing the margin the link predictor should attempt to maintain between the scores of true and negatives triples [2].

Cross Entropy Loss. Cross Entropy Loss (CEL) is calculated in a setwise manner as shown in Equation 2.12:

$$CEL(Score_i, Score_{i,negatives}) = \frac{e^{Score_i}}{\sum_{k=0}^n e^{Score_{i,negatives_k}}} \quad (2.12)$$

where $Score_i$ is the score of the i^{th} input triple and $Score_{i,negatives}$ is the scores of all n -many negatives generates for triple i .

The loss function is the last major component of KGEMs. However, other hyperparameters remain that define how KGEMs are optimised and regularised, as well as how various aspects of learning (such as embedding and negative generation) are performed. All remaining hyperparameters to KGEMs are outlined in the following section.

Other Hyperparameters

KGEMs have several other hyperparameters [2, 38, 73]. These largely fall into two categories: functional components (namely, the optimiser and the regulariser) and scalar-valued settings (such as the embedding dimension and the learning rate). The two critical functional component hyperparameters are [2, 73]:

- **Optimiser.** The optimiser is the component responsible for backpropagating loss and updating the KGEM’s learnable parameters (i.e. the embedding values). Adagrad [103] and Adam [43] are the most commonly used, and best-performing, optimisers in the state-of-the-art of KGEMs [2, 24, 38, 47, 73, 94, 102]. The optimiser Adadelata [103] has also been used, but a mass benchmarking study conducted by Ali et al. showed that Adam generally resulting in better performance relative to Adadelata [2].
- **Regulariser.** The regulariser is the component responsible for regularising embedding values to prevent over-fitting. In state-of-the-art KGEMs, this is done by penalising larger embedding values based on the L_p norm of embedding vectors, such that lower-magnitude embeddings are preferred over higher-magnitude embeddings [73]. The value returned by the regulariser is added to the loss, allowing for the regulariser’s penalty to be directly incorporated into learning. The L_2 and L_3 norms are most common in the state-of-the-art [38, 47, 73, 94, 102]. State-of-the-art regularisation also typically weights the L_p norm by the frequency of the node / edge each embedding vector describes [38, 47, 73, 94, 102]. The choice of p -norm, and whether the regulariser is weighted by node / edge frequency, are hyperparameters to regulariser construction.

The scalar-valued hyperparameters settings are [2, 62, 97]:

- **Embedding Dimension** (d). The dimension into which node and edge embeddings will be placed.
- **Number of Negatives per Positive** (npp). The number of negative triples to generate for each true triple in the training set.
- **Learning Rate** (lr). The learning rate to be used by the optimiser.
- **Regulariser Coefficient** (r). The coefficient that the regulariser’s penalty value is multiplied by before it is added to model loss. If set to 0, it is mathematically equivalent to not using a regulariser.
- **Batch size** (b). In KGEMs, batches are sets of triples from the training set that are used as training examples. The batch size determines how many positive triples are contained in any one batch – typically, the batch size is small compared to the total size of a knowledge graph, which results in many batches being run per epoch [2, 38, 73]. The optimiser is run at the end of every batch, not the end of every epoch, which means that more batches necessarily result in more calls to the optimiser.

These hyperparameters constitute all of the remaining hyperparameters to KGEMs; once all are defined, a KGEM is fully defined as a link prediction model. The following Section 2.3.2 gives a description of how all of these elements are used together in the KGEM training loop, and gives a mathematical description of end-to-end KGEM training with explicit reference to the role of each of these components. Finally, Section 2.3.3 provides details on how optimal hyperparameters are selected for state-of-the-art KGEMs.

2.3.2 The KGEM Training Loop

Once all core elements of a KGEM – its scoring function, negative sampler, loss function, and other hyperparameters – have all been defined, it is possible to begin discussing the general form of the KGEM training loop. The manner in which KGEMs are trained is largely independent of any of the specific components chosen. That is, changing any component or hyperparameter value does not change the format of the training loop [2, 73].

The KGEM training loop describes how KGEMs are trained to perform link prediction on a knowledge graph. The training loop is run on the training set of a knowledge graph, which contains all triples that are not meant to be used for evaluation. Importantly, the test set T_{test} , which is used for final evaluation of a KGEM, is not visible to a KGEM during training. The KGEM training loop consists of, in order:

1. **Batch extraction.** In this step, a batch of triples is extracted from the training set of the knowledge graph.
2. **Negative Sampling.** Negative triples are generated based on the selected ground-truth triple to serve as counter-examples.
3. **Triple scoring.** All triples, true and negative, are scored, and their scores are recorded.
4. **Loss calculation.** A loss value is calculated based on the scores of the true and negative triples. If a regulariser is in use, a regularisation penalty is added to the loss.
5. **Backpropagation.** The loss value is backpropagated using the optimiser to update the values of all embeddings. This is the step in which learning is done to make the KGEM better at predicting links in the KG it is being trained on.
6. **Repeat.** A new batch is sampled from the KG, and the process begins again.

A schematic overview of the KGEM training loop is given in Figure 2.7. Note that in this overview, it is assumed that all triples are extracted individually, rather than in batches of several triples, for simplicity of representation. In practice, it should be noted that batches of more than one triple are typically used [2, 38, 73].

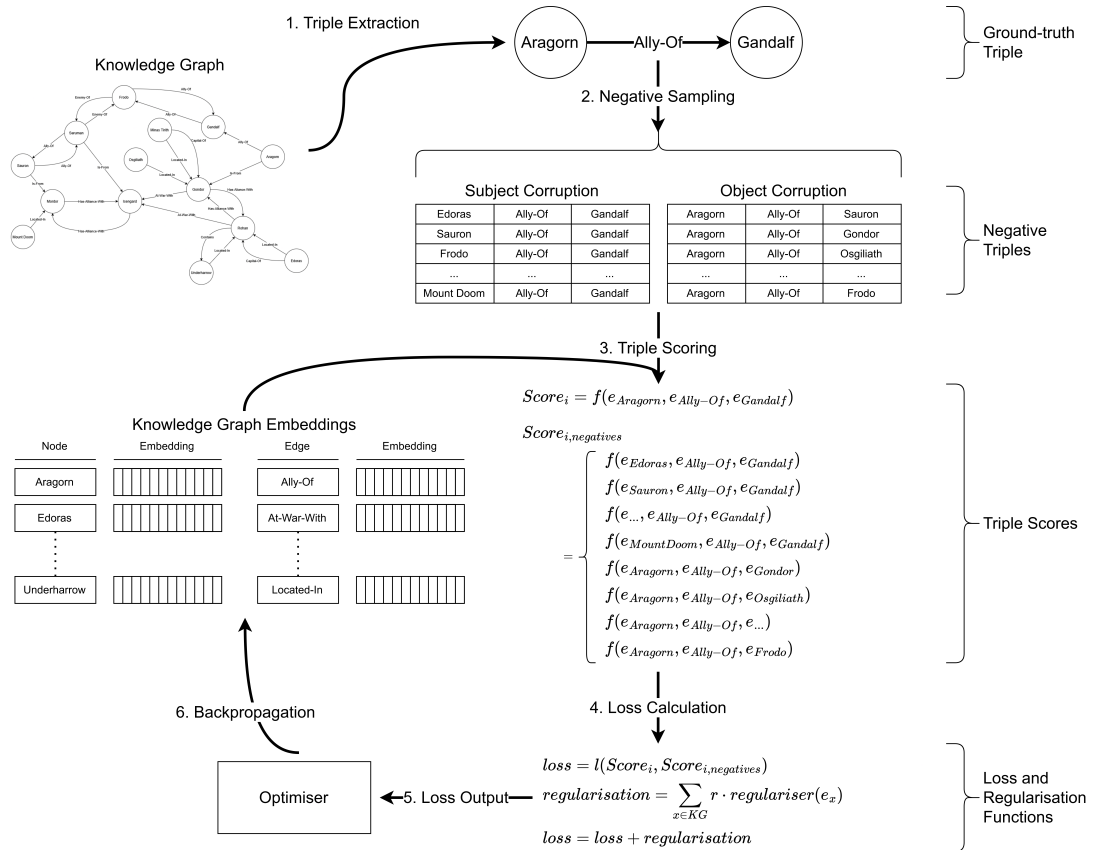


Figure 2.7: An overview of the KGEM training loop.

Each full pass of the training loop through all triples in the training set of the knowledge graph is referred to as one epoch [2, 73]. KGEMs are typically run for many epochs (hundreds or even thousands) [2, 73]. Once all epochs are finished, training is complete and the resultant KGEM can be evaluated on its test set.

For reference, a fully detailed analysis of how KGEMs (and link predictors in general) are evaluated is given in the previous Section 2.2.2. An analysis of state-of-the-art KGEMs, hyperparameter choices, and link prediction performance is given in Section 2.4. Finally, the following section gives details on how hyperparameter values to be used for KGEM training are selected.

2.3.3 Hyperparameter Selection for KGEMs

As outlined in the previous sections, there are 8 key hyperparameters that must be defined in order to run a KGEM. Specifically, for a given scoring function, the following must all be selected from a pool of several options each:

- Negative Sampler
- Loss Function
- Optimiser
- Regulariser
- Embedding Dimension
- Number of Negatives per Positive
- Learning Rate
- Regulariser Coefficient
- Batch Size

Each of these has at least several options, and some (such as the embedding dimension) have an infinite number of possible values they can take. In order to determine which combinations do (or do not) result in effective KGEM learning for the link prediction task, the conventional method is to perform a hyperparameter search [2, 36, 38, 45, 61, 73]. While there are many methods by which such a search can be performed, all of them necessarily follow the same series of steps:

1. **Definition of a hyperparameter search space.** This set represents not the set of all possible hyperparameter values (which is infinite), but the subset of values that

are considered candidates to be searched [2, 45]. For example, for the hyperparameter negatives-per-positive, the set 1, 2, 5, 10, 20, 50, 100 could be used (as done in [45]). The idea of such a set is to be representative – after all, there is little reason to expect that there would be a substantial difference between using 99 vs 100 negatives per positive [45].

2. **Selection of a hyperparameter combination to evaluate.** Once all values that should be searched have been defined for all hyperparameters, one of the (many) possible combinations of hyperparameters is queried and used to train a KGEM. This KGEM is then evaluated on its validation set T_{valid} , and the performance of the model (typically in terms of MRR) is recorded [2].
3. **Repeat.** A new hyperparameter combination is queried and evaluated, and the performance it achieves is recorded. This is repeated until some end condition is reached (such as a maximum number of iterations or a timeout), or until all hyperparameter combinations have been searched [2].

An overview of the hyperparameters of KGEMs, as well as the general procedure of hyperparameter optimisation, is given in Figure 2.8.

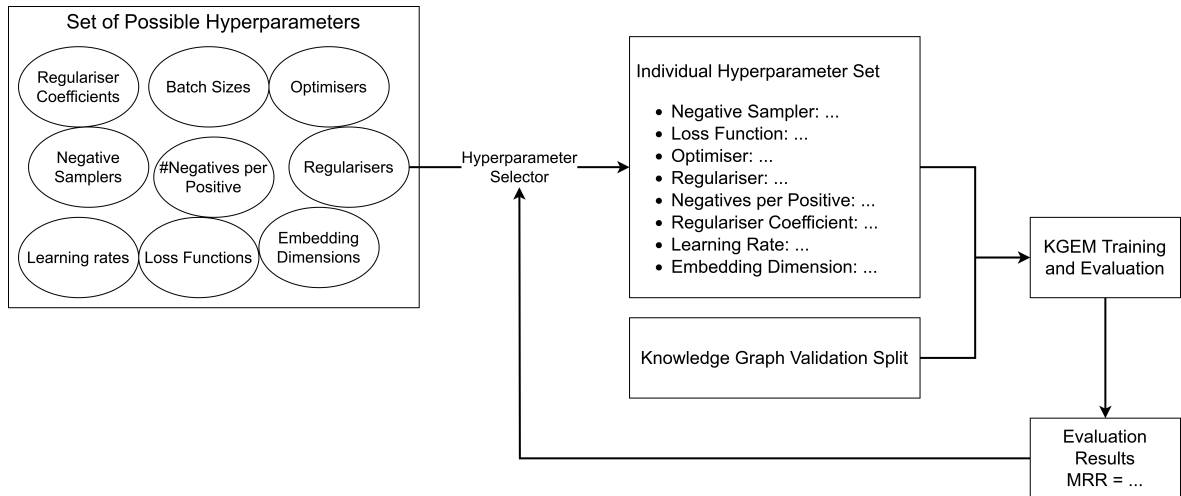


Figure 2.8: An overview of the process of hyperparameter selection for KGEMs.

In KGEMs, as in machine learning in general, there are several methods by which hyperparameter search can be performed:

- **a grid search**, in which all combinations of all hyperparameter values are searched sequentially until all have been searched [19, 73],
- **a random search**, in which random combinations of hyperparameters are sampled (without replacement) from the set of all possible hyperparameter combinations until

an end condition (typically a number of samples or a fixed amount of time) is reached [1, 2, 19, 73], or

- **a weighted random search.** in which new hyperparameter combinations to use are sampled with bias, such as using Bayesian methods or genetic algorithms, in attempt to sample higher-quality configurations that are similar to those that have been observed to work well [1, 19, 26, 27, 73]. As with random search, a weighted search is typically performed until an end condition is reached, and it does not search all possible hyperparameter combinations.

In general, grid search is considered sub-optimal as the amount of time required to run it on a large grid is prohibitive in practice for KGEMs [2, 73]. For this reason, use of random search or more advanced weighted random search methods is most common, especially in large-scale studies of KGEMs [2, 73].

There are two critical points to highlight from the current state-of-the-art in hyperparameter optimisation:

- that there is a growing consensus that effective and ineffective hyperparameter combinations can be predicted with some level of accuracy based on how well similar hyperparameter combinations perform [1, 19, 26, 27, 73], and
- despite this, that there is also a general assumption in the state-of-the-art that hyperparameters must be searched for and cannot be determined in a pre-hoc manner based on the dataset at hand [1, 2, 19, 26, 27, 73].

The latter assumption is questioned in this work, particularly in that Structural Alignment posits that hyperparameter preference (i.e. which KGEM / hyperparameters are optimal for a given KG) can be modelled as a function of graph structure. The details of this claim, and its implications, will be discussed in full in Chapter 3.

2.3.4 A Note on non-KGEM Link Predictors

As KGEM models generally hold state-of-the-art performance, they are central to this thesis. However, it is important to note that other link predictors exist that do not fall under the literature definition of the term “knowledge graph embedding model” (even though they still do produce knowledge graph embeddings in some cases). The major categories of such models are:

- Graph Neural Networks (GNNs),
- Random Walk-based models,

- Logic-based methods (also called rule miners),
- Neuro-Symbolic methods, and
- Graph Intersection-based methods.

Graph Neural Networks

GNNs for link prediction use neural networks to process and propagate node and edge features via a message-passing architecture [109]. These feature vectors are then interpreted as embeddings and are used to predict whether unseen triples are plausible or not [109]. Unlike KGEMs, GNN models are typically focused on modelling nodes / edges as sources of information with (given or randomly initialised) feature vectors [109]. Therefore, GNN models are defined primarily by their neighbourhood size and neighbourhood aggregation function, rather than by a triple scoring function as done in KGEMs [62, 97, 109]. Examples of GNNs for link prediction on knowledge graphs include R-GCN [85], CompGCN [95], CensNet [37], and GVAE [44].

See Zhou et al. (2020) for an overview of GNN-based graph learning [109].

Random Walk-based models

Random-Walk-based models use random graph walks in which the learner follows random directed paths from node to node in a KG to extract series of triples as sequences [65, 68]. These sequences are then input into a language model to produce embeddings [65, 68]. Most random walk-based methods derive from RDF2Vec, forming the so-called “RDF2Vec family” of models [65]. Note that while the express goal of these models is to produce KG embeddings, these models are conventionally not referred to as knowledge graph embedding models because their embeddings are based on graph walks, rather than on triple scoring functions [2, 73, 97].

See Portisch et al. (2024) [65] for a detailed overview of random walk-based link prediction and KG embedding methods.

Logic-based methods

Logic-based methods mine rules directly from the triples in a KG, and then attempt to generalise these rules to new predict the truth value of unseen triples [15, 28]. As such, they do not use learnable weights or create embeddings for nodes or edges. For example, using our example KG shown in Figure 1.1, the rule “If A is allied with B, and B is enemies with C, then A is enemies with C” could be mined. This rule can then be instantiated based on specific triples in the KG; for example, that rule would allow the conclusion that Sauron is

an enemy of Frodo by plugging in A for Sauron, B for Saruman, and C for Frodo to get “If Sauron is allied with Saruman, and Saruman is enemies with Frodo, then Sauron is enemies with Frodo”.

The state-of-the-art in logic-based link prediction is held by AMIE+ [28], though other models such as AnyBURL [57] and DL-Learner [15] do exist.

Neuro-Symbolic methods

Neuro-Symbolic methods are a recent direction taken for the link prediction task. They use some learnable parameters well as logical rules in an attempt to combine the benefits of rule-based and embedding-based / neural-based approaches [4, 29, 35, 53, 66, 69, 86, 87, 106, 107]. The neuro-symbolic category is necessarily very variable, and includes logical models augmented by GNNs, KGEMs, or other forms of machine learning. The defining property of these models is that all link prediction queries are answered by integrating knowledge from (explicit or implicit) logical rules, as well as machine learning-based (typically embedding-based) methods [4, 29, 35, 53, 66, 69, 86, 87, 106, 107]. They can largely be divided into two broad categories:

- Methods that require user-defined logical rules (often in the form of an ontology) [29, 35, 53, 106]
- Methods that infer logical rules during the training process [4, 66, 69, 86, 87, 106, 107]

See Zhang et al. (2022) [106] for a detailed survey of neuro-symbolic methods for link prediction.

Graph Intersection-based methods

Graph Intersection-based methods are used in only one paper to the knowledge of the author – Le et al. (2024) [48]. This methods models link prediction in terms of intersections of the neighbourhoods of graph elements, and assigns higher plausibility to triples whose component elements have larger intersections at a k -hop range [48]. While it is a new method, it has shown an ability to significantly outperform state-of-the-art KGEMs on common link prediction benchmarks datasets [48].

Non-KGEM Methods in the State of the Art

It remains important to highlight that while non-KGEM methods are powerful, KGEMs are generally considered to be at the fore of the state-of-the-art for link prediction [49, 96, 97] and have become established as the go-to models for a variety of link prediction use cases

[9, 59] despite evidence that some non-KGEM models can outperform KGEMs [48, 87, 96]. These claims, however, must be taken into context: many previous algorithms and models that claimed to beat state-of-the-art KGEMs have been shown, upon re-evaluation, to not outperform them, or to outperform them by a lesser margin [2, 36, 38, 73]. This typically occurs when better selecting hyperparameters (such as negative samplers, optimisers, and loss functions) for all models (novel and baselines) to allow for a more fair comparison of their relative performance [2, 36, 38, 73].

As non-KGEMs are so widely varied, less robustly evaluated and re-evaluated in the literature for link prediction [2, 36, 38, 73], and typically used less in downstream applications of link prediction [9, 17, 59], focus in this thesis is instead given to KGEMs. Further discussion of non-KGEMs is left aside to allow focus to be given to the elements of the state-of-the-art most directly relevant to the work at hand.

2.4 Graph Structure and Link Prediction

This section gives an overview of state-of-the-art studies that have attempted to find relationships between KG structure, KGEM hyperparameters, and link prediction performance. It is intended as direct background for the formulation of the Structural Alignment Hypothesis presented in the next chapter. Specifically, this section describes:

1. in Section 2.4.1, a discussion of how knowledge graph structure is measured in the literature; and
2. in Section 2.4.2, a discussion of published works looking at patterns of hyperparameter preference and overall link prediction performance.

2.4.1 Measures of Knowledge Graph Structure

In this work, knowledge graph structure is defined as a quantitative description of the local connectivity of individual nodes and edges in a graph. For example, (local) structural features such as node degree and edge frequency are directly included in this definition. While there are many global graph structural features (for example, the mean degree of all nodes, or the median frequency of all edges), in this work structure is always used to refer to localised structure at the level of specific nodes and edges unless otherwise noted.

The purpose of this section is not to describe every possible measure of graph (or knowledge graph) structure. Instead, the purpose of this section is to discuss specifically those measures of knowledge graph structure which have been documented in the state-of-the-art either for KG structural analysis or for relating KG structure to link prediction performance.

To do this, a survey of structure-based literature is provided specifically in the domain of knowledge graphs and link prediction. Table 2.7 gives an overview of the structural metrics discussed in the literature and the manner in which they were referenced, ranging from * (mentioned but not analysed), ** (used to characterise KGs), and *** (used to characterise KGs and LP).

Article	Node Degree	Rel Freq	Node-Rel Co-Freq	Node-Node Co-Freq	Node: Other	Rel: Other
Rossi (2020) [71]	***	**	.	.	.	.
Sadeghi (2021) [74]	**	.	.	.	***	.
Bonner (2022) [9]	***	**	.	.	.	.
Zietz (2024) [110]	***	.	**	***	***	***
Mohamed (2020) [58]	***	***	**	**	.	.
Kotnis (2017) [45]	***	***	***	.	.	.
Hogan (2021) [33]	**	.	.	.	**	**
Ruffinelli. (2020) [73]	*	*	.	.	.	.
Dörpinghaus (2022) [25]	**	.	.	.	**	.
Rossi (2021) [70]	**	**	***	.	.	***
Dave (2024) [22]	***	.	.	.	.	.
Wang (2014) [98]	***	.	.	.	.	.

Table 2.7: An overview of papers describing KG structure in the state-of-the-art, what structural metrics they used, and how they used them. Key: *** = used to characterise both KGs and link prediction on those KGs; ** = used to characterise KGs; * = mentioned but not analysed; . = not mentioned. Abbreviations: Rel = relation; Freq = frequency. Note the “et al.” is omitted above due to space restrictions.

Most literature that describes knowledge graph structure focuses on frequency-based metrics [9, 22, 25, 33, 45, 58, 70, 71, 73, 74, 98, 110]. Specifically, the four metrics that appear consistently in KG / link prediction literature and that are highlighted as relevant to structural characterisation of knowledge graphs and link prediction are:

- **Degree:** The degree of a node is the number of relationships that connect to it. In other words, it is the number of times that node appears as either a subject or an object in a triple in the KG.
- **Relationship Frequency:** The frequency of a relationship is the number of times that relationship is present in a triple in the KG.
- **Node-Relationship Co-Frequency:** Node-relationship co-frequency is the number of times that a given node and relationship co-occur in the same triples in the KG.
- **Node-Node Co-Frequency:** Node-node co-frequency is the number of times that two nodes co-occur in the same triples in the KG.

All of these metrics have been discussed in link prediction literature as relevant to the link prediction task in at least one publication, and noted as relevant to characterising KG structure in at least two publications, as outlined in Table 2.7.

The majority of existing publications in the area of KG structure show that node-based characterisation of KGs is more common than relationship-based characterisation in the literature, and is taken as generally representative of KG connectivity [9, 25, 33, 45, 58, 70, 71, 73, 74, 110]. Relationship-based metrics, while still highly relevant, tend to be considered less.

Similarly, node degree and relationship frequency dominate in use over all other metrics due to their simplicity and their ability to represent the most relevant aspects of KG structure [9, 25, 33, 45, 58, 70, 71, 73, 74, 110]. Co-frequency based metrics, while shown to be relevant in the literature on graph structure [45, 58, 70, 110], are less-heavily represented there.

Many other structural features have been considered in the literature. Dörpinghaus et al. (2022) [25], Hogan et al. (2021) [33], and Zietz et al. (2024) [110] are particularly notable in giving a detailed descriptions of more complex measures of KG structure. However, they tend to focus on *different* metrics, resulting in little consensus in the literature for use of these other metrics for characterising KGs and LP performance. As can be seen in Table 2.7, a majority of the KG / link prediction literature tends to converge on the four frequency-based metrics outlined previously in practice.

Of all the studies outlined in Table 2.7, focus is placed on those that provide the most insight into the commonly-used frequency-based metrics outlined above. These are: Rossi et al. (2020) [71], Sadeghi et al. (2021) [74], Bonner et al. (2022) [9], Zietz et al. (2024) [110], Mohamed et al. (2020) [58], Kotnis et al. (2017) [45], and Rossi et al. (2021) [70]. A summary of each of these studies follows.

Rossi et al. (2020)

Rossi et al. (2020) perform a KG-structure-based evaluation of the KGEMs TransE and DistMult on FB15k, FB15k-237, WN18, and WN18RR [71]. Specifically, they examine whether the degree of a node influences how well that node is learned and can be used in link prediction [71]. Their results indicate that, on both TransE and DistMult on all datasets tested, higher degree nodes are learned substantially better than lower-degree nodes [71]. They further highlight that existing KGEM benchmark datasets (FB15k, FB15k-237, WN18, and WN18RR) can exaggerate KGEM performance because high-degree entities are over-represented in their testing sets, which means that KGEMs that learn to predict only a few nodes well can still appear to have high performance [71].

While their analysis is primarily focused on node degree and its resulting impacts on link

prediction, they use both node degree and relationship frequency to profile KG structure [71]. Importantly, they show that these common KG benchmark datasets exhibit extreme skew in node degree and relationship frequency in both the training and the test sets, something they show biases link prediction evaluation. They further highlight that low-degree nodes and low-frequency relations have less information about them that can be learned by link predictors, an effect they characterise both through re-analysis of KGEM evaluation and through an analysis of embedding space [71].

Specifically, through an analysis of the position of nodes in embedding space, they highlight that higher-degree nodes tend to be more isolated from other nodes because there is more information about them in the graph to allow for high-quality representation and distinction from other nodes in embedding space [71]. They further note that lower-degree nodes tend to be very close to each other in embedding space, meaning that they cannot be as readily distinguished from each other [71]. The result of this is that link prediction queries asked to predict a high-degree node will be more successful than those that are asked to predict a low-degree node [71].

The overall result of their analysis is a very clear conclusion that node degree is, at least in part, a determinant of link prediction performance and thereby highly relevant to characterising KG structure and link prediction [71].

Sadeghi et al. (2021)

Sadeghi et al. (2021) build a GNN-based link predictor called GFA-NN that explicitly models node centrality and relative node position [74]. They model node centrality specifically using Katz Centrality, which is a generalised version of node degree that accounts for the degrees of nodes nearby every node as well [74]. They show that their model can match or exceed the performance of KGEMs on link prediction, and attribute this increased performance to its ability to model KG structure.

While the Sadeghi et al. (2021) paper is focused mostly on GNN-based link prediction, it is included in this analysis because of how they annotate the KGEM baselines they use. They suggest that GFA-NN is able to beat KGEM baselines on WN18RR but not on FB15k-237 because WN18RR has a wider distribution of node centrality values [74]. They suggest that on FB15k-237, where degrees are more consistent and less spread out, that KGEMs are better able to learn – thus providing some evidence that node centrality is key to understanding how KGEMs learn [74].

Finally, they call out KGEMs specifically for learning based on (only) a “1-hop neighbourhood” around each node – suggesting that they are only able to learn from very localised graph features [74]. Considering their findings on FB15k-237 and WN18RR [74], as well as

Rossi et al. (2020)’s findings that FB15k-237 and WN18RR have a massive skew in degree values [71], this is consistent with the idea that KGEMs learn better in more dense, connected regions of a graph.

Bonner et al. (2022)

Bonner et al. (2022) take a similar approach to Rossi et al. (2020) [71] in that they both identify significant skew of node degrees in common KGs, and examine how this can lead to degree-related biases in link prediction using KGEMs [9]. They perform their analysis specifically in the biomedical context, and show that degree imbalance, and degree bias in link prediction, remains very common and very problematic in that domain.

Specifically, Bonner et al. (2022) calls out node degree as the most commonly used topological measure (although they do cite that other works mentioned that relationship frequency is relevant as well) [9]. They extract information from the biomedical KG HetioNet and learn the graph using the KGEMs TransE, TransH, ComplEx, RotatE and DistMult [9].

They then examine the ranked list for predicting which genes were associated with given diseases [9]. Taking the ranked list outputs of these predictions, they find that higher-degree gene nodes tend to be preferentially predicted across all 137 diseases tested [9]. They additionally found that high-degree nodes that were not directly observed to be connected to the disease of interest in the KG would be preferred over lower-degree nodes that had been observed in the KG training set to be linked to the disease of interest [9]. This last point particularly is of note, suggesting that KGEMs overfit based on degree even to the point of disregarding other observed connections in the dataset [9].

Their results agree with Rossi et al. (2020) [71] in showing that it was the degree of the node being predicted, not the degree of the node given in the link prediction query, that was influential on link prediction results [9].

Finally, they show that deleting edges incident on a node (so as to lower its degree) results in it being considered less plausible as an answer in link prediction and that adding edges similarly could make a node be considered more plausible as an answer [9]. Taken together, this suggests that node degree is largely influential on link prediction outputs in KGEMs.

Zietz et al. (2024)

Zietz et al. (2024), like Rossi et al. (2020) [71] and Bonner et al. (2022) [9] highlight that KGs tend to have very skewed degree distributions and perform an analysis of degree-based bias in KGs [110]. They do this by asking if degree (as well as some other structural metrics) are sufficient to allow for link prediction on their own [110]. The system they create uses one of several structural features – typically based on degree or node-node co-frequency – to

estimate the probability that an edge should exist [110].

To do this, they take the KG HetioNet and split it into distinct sub-graphs in which only one edge type is present (meaning that each individual sub-graph is effectively an unlabelled graph, not a true KG) [110]. They then take each of these unlabelled networks and construct what they call an “edge prior” that calculates edge probability as a function of various frequency-based properties of the graph [110]. Their evaluation shows that they are able to reconstruct each of these unlabelled graphs with very high accuracy using this method, suggesting that simple graph structural features are sufficient to predict links in unlabelled graphs [110].

It is important to reiterate that, while the unlabelled graphs they used are extracted from a knowledge graph, they are not multi-relational KGs in the form present in standard KG literature [110]. As such, these results must be interpreted with care in the context of KGs and KGEMs in the general case.

Mohamed et al. (2020)

Mohamed et al. (2020) establish that both node degree and edge frequency are subject to heavy skew in knowledge graphs, with there being many nodes / edges with low frequency, and very few that have very high frequencies [58]. They are particularly notable for showing that this skew follows a power law, which they mathematically annotate and describe in the context of the benchmark KGs FB15k, WN18, and YAGO3-10 [58].

The bulk of the paper then focuses on how to re-define evaluation metrics to assign lower weight to higher frequency nodes / relations as a method of re-balancing the testing set to stratify evaluation equally across all nodes and relations [58]. They first provide evidence that the degree of subject, predicate, and object items in a triple are not correlated – i.e. the presence of a high-degree subject does not imply the presence of a high (or low) frequency predicate, nor the presence of a high (or low) degree object [58]. Because of this, they note that re-weighting of the test set cannot be done directly at the triple level – there is no way to label a triple as “over-represented” or “under-represented”, since such effects exist only at the sub-triple level of nodes and relations [58].

As such, they define a stratification procedure that first calculates link prediction performance in the context of all relationships individually [58]. They then re-balance all of these results based on subject and object degree, and finally combine all relationship-specific performance metrics into a single performance score by re-weighting based on relationship frequency [58]. They allow the degree of re-weighting to be configurable, meaning that they can choose to fully re-balance (i.e. removing all frequency bias in the test set), to counter-balance (inserting a bias inversely proportional to node / relation frequency) or not re-balance

at all. They perform this operation on both the Hits@K and MRR metrics, resulting in new metrics called “strat-Hits@k” and “strat-MRR” [58].

Mohamed et al. (2020) then use their stratified link prediction metrics to evaluate 4 KGEMs (TransE, DistMult, ComplEx, and HolE) trained on FB15k and YAGO3-10 [58]. Their results indicate that re-balancing to remove degree-related and relation-related biases results in a drop in reported KGEM performance [58]. In other words, their re-evaluation exposes that low-degree nodes and low-frequency relations are learned substantially less reliably than those with higher degree / frequency [58].

While they call out node-relationship and node-node co-frequencies as being relevant structural qualities, they do not specifically re-balance for these or empirically test for their impact on link prediction performance [58]. Regardless, their results indicate very strongly that frequency-related structural metrics are of high relevance to the characterisation of KGs and, particularly, to the link prediction task.

Kotnis et al. (2017)

Kotnis et al. (2017) primarily focus on the impact of negative sampler choice on link prediction using KGEMs, rather than on characterising link prediction in terms of KG structure [45]. Notwithstanding, in their analysis of which negative samplers work for different KGEMs and KGs, they find several important results regarding which elements of KG structure directly impact how well various negative sampling protocols work and, therefore, how well various KGEMs using them can learn [45].

They perform an analysis of 6 negative sampling protocols on 4 KGEMs (TransE, DistMult, ComplEx, and RESCAL) trained and evaluated on 2 different KGs (FB15k-237 and WN18RR) [45]. Their results show that how effective various negative samplers are for the purpose of training KGEMs is based on [45]:

- **Relationship Frequency** – They show that learning low-frequency relations is much less reliable than learning high-frequency relations, which leads to lower link prediction performance on low-frequency relations [45]. They further show that this effect persists generally regardless of the negative sampling strategy used [45].
- **Node-Relationship Co-Frequency** – They show that using pseudo-typed negative sampling is sensitive to node-relationship co-frequency. As the co-frequency decreases, the number of possible pseudo-typed corruptions necessarily decreases as well – meaning that fewer negatives can be generated [45]. They show that this lack of negatives can lead to reduced LP performance [45].

They further highlight that node degree and relationship frequency are critical to characterising KGs, and that node degree is expected to be partially determinant of link prediction from a theoretical perspective [45].

While these results are presented in terms of negative samplers, negative sampling (necessarily) has a direct impact on link prediction as a core part of its training – thereby providing yet another lens through which to understand how the influence of frequency-based structural statistics affect link prediction on KGs.

Rossi et al. (2021)

Rossi et al. (2021) present a general comparative overview of KGEMs for the link prediction task [70]. Specifically, they evaluate 16 different KGEMs (including TransE, DistMult, and ComplEx) on 5 different benchmark KGs (FB15k, FB15k-237, WN18, WN18RR, and YAGO3-10) [70]. They then define a few main structural and non-structural features and explore how each feature correlates to the ranks assigned to link prediction queries [70]. The 2 KG structural features they chose are:

- **Number of peers.** Number of peers is what this paper refers to as “node-relation co-frequency”, except that it is defined in terms of how frequently a node and relation connect to a *single* given other node [70].
- **Relation path support** – an estimate of how various paths (multi-hop) from the subject to the object in a triple contribute to its information content [70].

Overall, their results indicate that triples with more possible alternatives for the node being predicted result in lower performance, as KGEMs struggle to distinguish between a larger set of possible nodes. The opposite effect also holds – when there are many triples connecting to the same object (or subject), and that object (or subject) is being predicted, it is generally predicted with much higher accuracy. Finally, they show that higher relation path support of a triple leads to better predictions in almost all cases.

While they provide an analysis of their other KG features, that analysis is omitted from this thesis as their other features are non-structural. For information on those, the reader is directed to their article [70].

While they do reference node degree and relation frequency as structural statistics that have been used to guide KG construction for some of their benchmark datasets, Rossi et al. (2021) do not include these features in their analysis of link prediction [70].

Dave et al. (2024)

Dave et al. (2024) explore how iteratively adding ontology-derived relations into the KG FB15k-237 affect how well it can be learned by different KGEMs [22]. This study is unique in that it uses a *structure-controlled* protocol, since iteratively adding sets of edges changes KG structure in a controlled manner [22]. They then explored how well 6 different KGEMs (ComplEx, TransE, DistMult, RotatE, RESCAL, and TransR) were able to learn each structure-controlled variant of FB15k-237 [22].

Their results show that adding in extra ontology-based relations to FB15k-237 generally results in decreased performance of the KGEMs tested [22]. While they do not provide a detailed numerical description of how adding in various relations affects the distribution of node degrees [22], adding relations can only result in *increasing* the degree of at least some nodes. As such, their results suggest that increasing degree of some nodes can have detrimental affects on learning – something that echoes the results of Bonner et al. [9].

That said, the study has some limitations. In particular, they did not perform a hyperparameter search, and instead used a constant set of hyperparameters for all KGEMs and KG-structure variants tested [22]. In light of other works noting that hyperparameters are KG and KGEM dependent [2, 73], this could have resulted in biased results of the relative performance of each model on each dataset.

Wang et al. (2014)

Wang et al. (2014) primarily develop a new KGEM, TransH, that extends TransE to be able to better model many-to-one and one-to-many relations [98]. As a part of this, they created the Bernoulli negative sampling protocol (described in Section 2.3.1), under which the negative sampler preferentially corrupts the subject (or object) in a triple that has been seen the least [98]. At a broad level, this paradigm results in selecting nodes to corrupt in a way that is inversely proportional to node degrees [98].

They then run experiments on their proposed TransH model under both full-random and Bernoulli negative sampling on FB15k and WN18 [98]. Their results indicate that, on FB15k, Bernoulli negative sampling leads to improved link prediction performance relative both to various baseline KGEMs and to TransH trained without Bernoulli negative sampling [98]. However, they show the opposite for WN18, where Bernoulli sampling actually underperforms full random negative sampling – an effect they attribute to FB15k having a much more diverse structure than WN18 [98]. Overall, their results on link prediction can be summarised as so: on KGs with a high diversity of connectivity patterns, accounting for node degree during negative sampling leads to increased performance on link prediction for the TransH model

[98].

While their results can be seen as a characterisation of the effect of node degree on KGEM learning, they do not test the Bernoulli negative sampler on any KGEMs other than their new TransH model [98], which limits the generalisability of their study.

Graph Structural Features Considered in this Work

The established state-of-the-art focuses specifically on node degree, relationship frequency, node-relationship co-frequency, and node-node co-frequency as core measures of KG structure and as a key tool for understanding how link prediction works on KGs. As a result, these four frequency-based metrics are adopted as the core metrics for annotation of KG structure in this work.

The following section continues the discussion on link performance begun in this section by exploring studies that characterise hyperparameter preference and the impact of hyperparameters on overall link prediction performance.

2.4.2 Hyperparameter Preference and Link Prediction

This section gives an overview of hyperparameter preference and how choices for various hyperparameter values have been documented in the literature to affect link prediction performed via KGEMs. It further describes what elements of graph structure have been documented to affect hyperparameter preference in KGEMs so that a view of hyperparameters, KG structure, and link prediction performance can be presented as a unified construct based on the state-of-the-art literature.

Based on the literature review conducted in this thesis, an overview of hyperparameters, KG structure, and link prediction performance is given in a graphical format in Figure 2.9. In this figure, a directed link between two nodes indicates that the subject node is dependent upon the object node. Edges connecting a node to “Link Prediction Performance” indicate that that node has specifically been described in terms of its effects on link prediction. All edges are annotated by references to the study or studies in the literature providing evidence for such a dependency relationship. Only those elements that have been documented in this review as most relevant to understanding state-of-the-art KGEMs and link prediction are included in the graph.

Further to this, Table 2.8 presents an overview of state-of-the-art literature that has explored hyperparameter preference in KGEMs. A detailed description of each of the studies, and their key findings, follows.

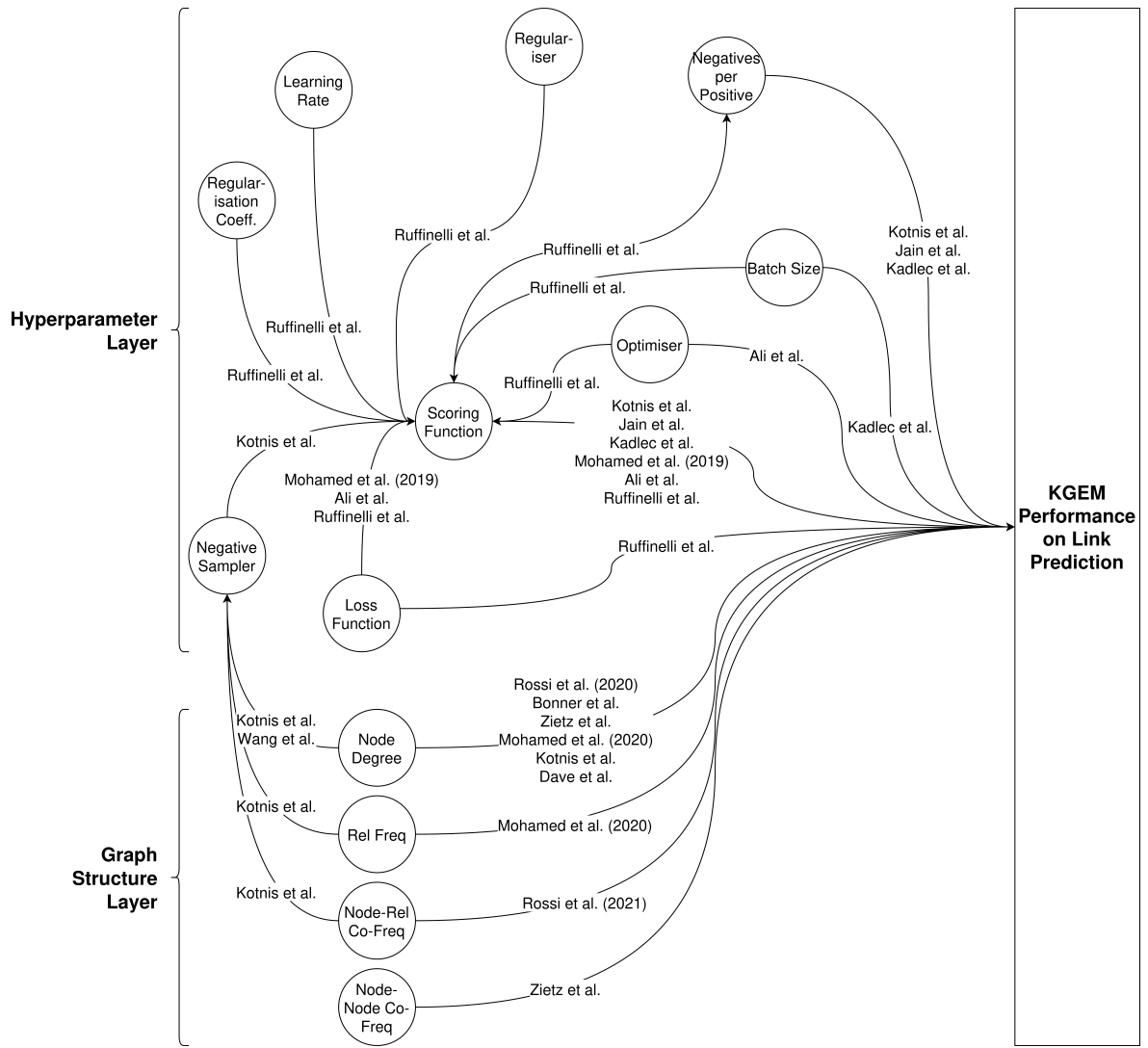


Figure 2.9: An overview of hyperparameter and graph structure influence on KGEM performance on the link prediction task, presented as a dependency graph annotated by evidence in the literature. A directed edge from a subject node to an object node indicates that the item subject is dependent upon the object. Nodes are connected to “Link Prediction Performance” if they have been shown to have concrete influence on overall link prediction performance. Note that as the literature has shown that hyperparameters are generally dependent on both the KGEM and KG used [2, 73], only hyperparameters that have been *specifically* studied are included.

Kotnis et al. (2017)

Kotnis et al. (2017) provide an in-depth study of negative sampler preference in KGEMs – specifically, how choice of negative sampler interacts with KGEM scoring functions and KG structure [45]. Specifically, they examine 4 different KGEM scoring functions (TransE, DistMult, ComplEx, and RESCAL) paired with 6 negative samplers on 2 KGs (FB15k and WN18) [45]. They vary the number of negatives per positive on a grid of values, and report results for all negatives-per-positive values tested [45]. All other hyperparameters were determined either arbitrarily by the authors, or through a random hyperparameter search [45].

Article	Scoring Fn	Negative Sampler	Loss	Optimiser	Other
Kotnis (2017) [45]	**	**	.	.	.
Mohamed (2019) [61]	**	.	**	.	.
Jain (2017) [38]	**	.	.	.	**
Kadlec (2020) [36]	**	.	.	.	**
Ali (2022) [2]	**	.	**	**	*
Ruffinelli (2020) [73]	**	*	**	**	**

Table 2.8: An overview of studies in the literature documenting the effects of various hyperparameters on KGEM performance. Key: ** = the given hyperparameter was directly evaluated in the context of link prediction; * = limited evidence for the hyperparameter’s impact in LP was given; . = the given hyperparameter was not the subject of evaluation in that study. Abbreviations: Fn = Function. Note that “et al.” has been omitted due to spacing restrictions.

A brief description of the negative sampling protocols Kotnis et al. (2017) investigated follows [45]:

- **Full Random:** randomly replacing the subject or object of a triple with any other node.
- **Corrupting Positive Instances:** randomly replacing the subject or object of a triple with any other node that has been observed as the subject or object of the triple’s predicate before. Note that this is equivalent to “Pseudo-Typed Sampling” in the terminology of this thesis.
- **Typed Sampling:** randomly corrupting the subject or object of a given training triple subject to type constraints to restrict what corruptions are valid.
- **Relational Sampling:** randomly corrupting the predicate, rather than the subject or object,
- **Nearest Neighbour Sampling:** using a pre-trained KGEM to suggest corruptions that are close to the correct answer in embedding space.
- **Near Miss Sampling:** using a pre-trained KGEM to suggest negatives that it estimates to be the hardest to learn.

Their experiments resulted in several conclusions about the nature of negative samplers in KGEMs:

- **Preference for sampling more negatives.** They found that increasing the number of negatives generated per true triple almost always increases KGEM performance, regardless of the negative sampler, scoring function, or KG being used [45]. However,

after a certain point, further increases to the number of negatives generated has little to no impact on performance [45].

- **Dependency on KG connectivity.** They find that the primary determinant of negative sampler preference is KG connectivity – how densely connected the graph is [45]. Node degree and node-relation co-frequency are shown to be the most relevant measures of connectivity in this case [45]. They specifically highlight that low node-relation co-frequency in FB15k results in poor performance of pseudo-typed sampling, as low node-relation co-frequency means that there are not enough options for negative triple generation, which requires (lower-quality) pure random negatives to be generated instead [45]. On WN18, where this is not an issue, pseudo-typed sampling is typically optimal [45].
- **Minor dependency on the scoring function.** The optimal negative sampler depends weakly on what KGEM scoring function is being used [45]. They find that more realistic negatives (i.e. those that are not pure random) tend to be better for most scoring functions, but that TransE, being so simplistic, does better on full random negative sampling [45].
- **Minor dependency on relationship frequency.** They find that the negative sampling strategy that is globally optimal for a KG is not always the one that learns low-frequency relations the best, and that switching to a different negative sampler can result in learning lower-frequency relations more reliably. However, this effect is inconsistent, and generally the globally-optimal negative sampler for a given KG-KGEM combination is also optimal for most low-frequency relations in the graph [45].

Overall, the major result of the Kotnis et al. (2017) study is to conclude that negative sampler preference is firstly a function of KG structure, and secondarily a function of KGEM scoring function [45]. By specifically referencing KG structure in terms of density and node-relation co-frequency, their results can be readily framed in the framework of the frequency-based structural characteristics described in the previous section.

Mohamed et al. (2019)

Mohamed et al. (2019) provide an in-depth analysis of loss functions used in KGEMs and what factors determine loss function preference [61]. They examine how three KGEMs (TransE, DistMult, and ComplEx) and 5 KGs (FB15k-237, NELL50k, NELL50k239, WN18, and WN18RR) interact with 8 choices of loss functions. They categorise these loss functions into three categories:

- **Pointwise losses:** In this category they consider Pointwise Hinge Loss, Pointwise Logistic Loss, Pointwise Square Error Loss, and Pointwise Square Loss [61].
- **Pointwise losses:** In this category they consider Pairwise Hinge Loss and Pairwise Logistic Loss [61].
- **Multi-class losses:** Note that this category is what this thesis refers to as “setwise losses”. They include Binary Cross Entropy Loss (defined in a setwise manner, rather than in a pointwise manner) and Negative Log Softmax Loss [61].

They train every combination of scoring function, loss function, and KG on their optimal hyperparameters (determined with a grid search) [61]. Following this, they analyse how each of their loss functions (on its own and as a part of its category) influences KGEM performance on link prediction. The main results that they report are as follows:

- **Loss function preference depends on the KGEM.** They show that TransE (as an additive model) does better when trained with pairwise losses than with pointwise losses [61]. For DistMult and ComplEx (multiplicative models), this trend is reversed – they do better when trained with pointwise losses compared to pairwise losses [61]. This also necessarily implies that optimal choice of loss function is dependent on the scoring function being used.
- **Some loss functions dominate others.** Some loss functions were found to be universally better than others. For example, Negative Log Softmax Loss was found to outperform Binary Cross Entropy Loss in all cases, and in most cases Pointwise Square Loss outperformed all other pointwise losses [61].

Mohamed et al. (2019) did not perform any analysis on how loss functions interact with elements of KG structure, so claims about how hyperparameters interact with KG structure cannot be directly drawn from their work.

Kadlec et al. (2020) and Jain et al. (2017)

Kadlec et al. (2020) and Jain et al. (2017) both perform very similar studies on KGEM benchmarking. They take various KGEMs and show that when re-training the common baseline KGEMs (such as DistMult) on better hyperparameter combinations, that they can outperform the best reported results of more recent KGEMs in the literature [36, 38]. While the goal of their studies is not to compare different hyperparameter combinations, they do call out some trends in hyperparameter preference that became clear during their evaluation.

Specifically, Jain et al. (2017) and Kadlec et al. (2020) both highlight that increasing the number of negatives per positive used during negative sampling leads to better results in all cases [36, 38] – therefore agreeing with the results presenting in Kotnis et al. (2017) [45].

On top of this, Kadlec et al. (2020) finds that increasing batch size always leads to better performance of KGEMs, and propose that higher batch size is generally ideal for achieving higher link prediction performance [38].

Finally, both studies show that the choice of scoring function is, as entirely expected, highly relevant to link prediction performance [36, 38]. However, in doing their re-evaluation on new hyperparameter sets, they show that the influence of hyperparameters is also quite strong, even to the extent of outweighing the scoring function and allowing older scoring functions to perform better than newer ones once trained on the correct hyperparameter sets [36, 38].

Ali et al. (2022)

Ali et al. (2022) perform a large-scale evaluation of 21 KGEMs (including TransE, DistMult, and ComplEx) on 4 KGs (FB15k-237, WN18RR, Kinships, and YAGO3-10), running a large hyperparameter search on each KG-KGEM combination to determine optimal performance in all cases [2]. They then examine the overall performance of each KGEM and use their array of results to analyse relative efficacy of different KGEMs and hyperparameters [2]. Finally, they examine how well various logical relations in a KG (such as symmetry, transitivity, etc.) can be modelled by each KGEM tested [2]. However, they do not analyse performance in terms of frequency-based structural metrics of KGs [2]. Overall, the results they obtain on hyperparameter preference and link prediction performance are as follows:

- **Hyperparameters are dependent both on the KG and KGEM in general.** The optimal hyperparameters they found for each KG-KGEM pair indicate that optimal hyperparameters are a function of the KG being learned and the KGEM being used [2]. As such, knowing only the KG being learned, or only the KGEM being used, is not enough to determine the full set of optimal hyperparameters.
- **KGEM scoring functions strongly influence performance.** They provide evidence that the choice of scoring function has a huge impact on link prediction performance [2]. While they show that many models (when trained on their optimal hyperparameters) achieve similar results across different KGs, others are more variable and generally less-well performing [2]. Notably, they show that ComplEx and DistMult generally do quite well, and that TransE typically lags behind them [2].

- **Different KGEMs are differently sensitive to different hyperparameter configurations.** They show that different KGEMs are more or less sensitive to the hyperparameter configuration chosen [2]. For example, ComplEx tends to achieve high performance on a wide range of hyperparameter combinations on all KGs [2]. DistMult and TransE trained on Kinships is highly sensitive to hyperparameter choice, but when trained on WN18RR and FB15k-237 their performance is less sensitive to hyperparameter choice [2]. Similar trends hold for other KGs / KGEMs tested [2].
- **Loss function preference depends on the KGEM and KG.** They show that different loss functions result in better or worse link prediction performance depending both on the KGEM scoring functions and on the KG being learned [2]. However, they do not provide a detailed analysis of this, or explore what elements of KG structure might be responsible for loss function preference being dependent on the KG being learned [2].
- **The Adam optimiser generally outperforms Adadelta.** They find that the Adam optimiser tends to outperform Adadelta on the Kinships dataset across all KGEMs and hyperparameter combinations tested, and posit that this holds across other KGs as well [2].

They do not explore the effect of different negative samplers, or that of other hyperparameters, in detail [2]. Further, while most of their results clearly indicate that hyperparameter preference is a function of the KG as well as of the KGEM scoring function, they do not explore the structural characteristics of the KG that could result in this preference [2].

Ruffinelli et al. (2020)

Ruffinelli et al. (2020) perform a large-scale evaluation of 7 KGEMs (including TransE, DistMult, and ComplEx) on 2 KGs (FB15k-237 and WN18RR) using a Bayesian (weighted) hyperparameter search for each KG-KGEM combination [73]. In doing so, they show that common KGEM baselines (such as DistMult and ComplEx) can outperform more recent KGEMs when trained on a better set of hyperparameters [73]. This leads them to question if the state-of-the-art has been producing better models, or just using better hyperparameters for newer models [73].

Their overall results on hyperparameter preference and link prediction performance are outlined below:

- **Cross entropy loss dominates other loss functions:** They show that, across all KGEMs tested on both FB15k-237 and WN18RR, that cross entropy loss outperforms

all other loss functions [73]. Note that this finding does not necessarily contradict Mohamed et al. (2019) [61] as they do not examine cross entropy loss; however, it does contradict Ali et al. (2022) [2], whose results indicate that loss function preference is more nuanced.

- **Hyperparameters are dependent both on the KG and the KGEM:** The optimal hyperparameters they determined for every KG-KGEM pair was distinct from that of other KG-KGEM pairs in general [73]. However, they do not analyse what aspects of a KG (structural or otherwise) might lead to different KGs having different influences on hyperparameter preference [73]. They show this effect for almost all hyperparameters studied, including the optimiser, regulariser, and other hyperparameters [73].
- **Some hyperparameters are more important than others.** For each hyperparameter value found for each KG-KGEM pair, they report also the best competing configuration that assigns a different value to that hyperparameter [73]. While this is an imperfect comparison, they note that it can serve as a general proxy for the importance of each hyperparameter [73]. By applying this, they show that some hyperparameters (such as loss function) are typically more influential on overall MRR than others (such as the embedding dimension) [73]. They finally show that the relative importance of hyperparameters does sometimes vary based on the KG and KGEM being used [73].
- **Hyperparameter sensitivity depends on both the KG and the KGEM.** They show that how sensitive link prediction results are to hyperparameters (i.e. how much performance changes between different hyperparameter configurations) is dependent both on the KG and KGEM being used [73]. However, in general sensitivity to hyperparameter values is more influenced by the KGEM than by the KG it is trained on [73].
- **KGEM scoring functions influence performance.** They show that the KGEM scoring function used (i.e. ComplEx, DistMult, TransE, etc) is strongly influential on overall model performance [73]. Specifically, they highlight that many models achieve very similar results when trained on their optimal hyperparameters, but that some KGEMs (such as TransE) generally lag behind the others in performance [73].

Despite the quite strong and far-reaching results of their study, two limitations remain. First off, their experiments on negative samplers did not include most common negative sampling strategies [73]. Instead, they compared full-random negative sampling against all possible negatives [73] – which means that their analysis of negative sampling preference

cannot necessarily be expected to apply to how different negative sampling protocols interact with each other.

Second, their comparison of hyperparameter sensitivity is potentially biased. Since they use a weighted hyperparameter search for all values, rather than a full random or grid search, the hyperparameter values they sample are potentially biased [73]. This means that assessing their variance (to determine hyperparameter sensitivity) or treating them as ablations (to assess the relative importance of each hyperparameter) is also potentially biased [73]. As such, while their results on hyperparameter sensitivity and importance are powerful, they must be checked by acknowledgement that they come from a weighted Bayesian optimisation protocol, rather than an unweighted (full random or grid search) protocol.

2.4.3 Towards Structural Alignment

At a broad level, the state-of-the-art understanding of structural impacts on knowledge graphs and link prediction, outlined in Section 2.4.1, can be summarised as so:

- **Simple frequency-based structural characteristics dominate structural influence.** A consistent, cross-study trend is that node degree and relationship frequency are highly influential on how KGEMs learn KGs for the link prediction task. Node-node co-frequency and node-edge co-frequency, while less widely used, remain similarly important to link prediction. It is further relevant that all of these four features are conceptually and mathematically simple, making them easy to use and calculate on diverse knowledge graphs of various sizes.
- **Learnability aligns to frequency.** In almost all cases, state-of-the-art results show that the most well-learned aspects of a knowledge graph are those with the highest frequency – such as high-degree nodes and high-frequency relationships. Similarly, the worst-learned aspects of a knowledge graph have been consistently shown to be those with the lowest degree / frequency. This effect persists across a wide variety of KGEMs.
- **Knowledge graphs tend to have significant structural variation.** Essentially all knowledge-graphs used in the state-of-the-art for link prediction, even those from different knowledge domains, have been shown to have very diverse (and skewed) structure. This typically manifests as having many low-frequency elements (nodes and relationships) and few high-frequency elements. While this has been repeatedly shown to be a significant source of bias for link predictors, it also implies to the author that structure could be a significant source of signal for structure-based learning of knowledge graphs.

Similarly, the overall consensus in the state-of-the-art regarding hyperparameter preference, outlined in Section 2.4.2, can be summarised as follows:

- **Hyperparameter preference depends on the KG and on the KGEM.** State-of-the-art studies on hyperparameter preference show that the optimal hyperparameters for link prediction depend both on the KGEM being used, and on the KG being learned. Note that this does not exclude structure as being a determinant of hyperparameter preference; state-of-the-art results are compatible with the hypothesis that changes in KG structure drive different hyperparameter preferences across different KGs.
- **Hyperparameter sensitivity varies depending both on the KG and on the KGEM.** Similar to the aforementioned, changing the KG or the KGEM also can change how sensitive link prediction results are to changes in any single hyperparameter value. This means that, in different learning settings (on different KGs or using different KGEMs), relative hyperparameter importance is variable, not uniform.
- **Hyperparameter preference has complex inter-dependencies with other hyperparameters and with structure.** Hyperparameter preference has been repeatedly shown in the state-of-the-art to have inter-dependencies as well – that is, different hyperparameter values work better or worse depending on other hyperparameter values. This effect is shown schematically as a dependency graph in Figure 2.9. Similarly, negative sampler preference has been shown to directly relate to some elements of knowledge graph structure. In particular, it is dependent upon the four core frequency-based structural features highlighted as most relevant to link prediction in Section 2.4.1.

Overall, the established state-of-the-art shows a clear impact of both hyperparameters and graph structural features on link prediction performance. This dependency, as described in the state-of-the-art, is summarised in Figure 2.9. It is very notable, however, that only one study (Kotnis et al.) explicitly connects elements of graph structure to hyperparameter preference [45]. Despite this, the other studies outlined here that examined hyperparameter preference note that hyperparameter preference is dependent both on the KGEM being used and on the KG being learned [2, 36, 38, 61, 73].

This dependency on the KG being learned suggests a very real possibility that it is actually the structure of the KG being learned that drives the difference in hyperparameter preference. While only Kotnis et al. (2017) directly supports this [45], none contradict it. Further, the wide evidence outlined in Section 2.4.1 about the importance of structure in KGEM-based learning provides clear motivation for the hypothesis that KG structure is at least partially determinant of hyperparameter preference.

Elucidating the patterns of structural influence on link prediction and hyperparameter preference in KGEMs forms the basis for the central hypothesis of this work: the Structural Alignment Hypothesis. The Structural Alignment Hypothesis claims, in essence, that KG structure drives KG learning and link prediction – including hyperparameter preference. This hypothesis, and its potential implications for understanding link prediction, will be formally defined and explored in the next chapter.

3. Structural Alignment

This chapter is split into 2 main parts, which are as follows:

- Section 3.1 gives a formal definition of Structural Alignment based on the existing state-of-the-art, and outlines the general methodologies by which it can be evaluated.
- Section 3.2 describes how a Structural Alignment Framework is instantiated in this work – specifically, how it models structure and what structural features it uses.

The following two chapters, Chapter 4 and Chapter 5, outline how the instantiated Structural Alignment Framework is evaluated and what conclusions can be drawn from those evaluations.

3.1 Formal Definition

Structural Alignment is the hypothesis that structural features of the training set of a knowledge graph (outlined in Section 2.4.1) can be used to explicitly model hyperparameter preference and link prediction performance. This hypothesis encompasses two specific claims:

1. that hyperparameter preference across various KGEMs, and a KGEM’s final performance on the link prediction task, are functions of graph structure, and
2. that link prediction itself can be performed as a function of graph structural features (and without using learned embeddings).

Note that, as outlined in Section 1.2, each of these claims regards feasibility, not optimality. As such, focus in this chapter is not placed on how to formulate these problems to find the optimal solution to them, but rather to show that structure-based understanding of KG learning is a possible approach. Each of these claims is expanded on and mathematically defined in the following two sections.

3.1.1 Claim 1: Of Hyperparameters and Link Prediction

To understand how Structural Alignment applies to understanding hyperparameter preference and link prediction performance, it is important to begin at the basics: the current, most common formulation of link prediction that does not consider Structural Alignment. As outlined in Section 2.3, essentially all KGEM-based link predictors in the literature model link prediction performance as given in Equation 3.1.

$$Performance = trainAndEval(KG, KGEM, Hyp) \tag{3.1}$$

where *trainAndEval* is a function that represents training and evaluating a KGEM on the given KG and hyperparameters, *KG* is the specific knowledge graph being learned, *KGEM* is the KGEM scoring function selected to perform link prediction, and *Hyp* is the set of hyperparameters being used. The term *Performance* is used in lieu of a specific performance metric (such as MRR) to maintain generality, but note that any of the standard link prediction evaluation metrics could be substituted for it in the equation.

In other words, the state-of-the-art models link prediction performance in terms of three distinct elements: the KG being learned, the scoring function being used, and all other hyperparameters used.

Structural Alignment goes one step further and hypothesises that it is primarily the *structure* of that KG, denoted S_{KG} , that matters for learning. In other words, the Structural Alignment Hypothesis posits that hyperparameter preference and link prediction performance can be expressed in terms of KG structure as shown in Equation 3.2:

$$Performance = predict(S_{KG}, KGEM, Hyp) \quad (3.2)$$

where *predict* is a function that uses structural features of the KG, as well as the hyperparameter choices, to predict ultimate link prediction performance. The structure-based formulation here is motivated by the aggregate set of structural and hyperparameter dependencies described in Section 2.4.1 and Section 2.4.2, which generally suggest that a deterministic function with the properties given in Equation 3.2 should exist.

Instead of having to compute link prediction performance on a given set of hyperparameters by training a KGEM on an entire KG, the formulation in Equation 3.2 allows direct computation of KGEM performance for a given set of hyperparameters directly from KG structure. To do this, the exact same hyperparameter search protocol defined in Section 2.3.3 is followed, with one deviation: instead of training a KGEM to measure its performance, the *predict* function is queried with KG structure and the given hyperparameters. This allows for hyperparameter optimisation to be phrased as an optimisation problem over the domain of *predict* – thereby allowing *predict* to model both hyperparameter preference and link prediction performance.

As a result, the first claim of Structural Alignment given in Equation 3.2 can be validated or refuted very simply by determining if a function with the properties of *predict* indeed does exist. Defining such a function, and evaluating its ability to accurately model link prediction performance, is outlined in Chapter 4.

3.1.2 Claim 2: Of Structure-based Link Prediction

The second claim of the Structural Alignment Hypothesis is that KG structure is enough not only to predict link prediction performance, but also to directly perform link prediction itself. This formulation of link prediction follows the exact same conceptual flow as outlined in Figure 2.4 in Section 2.2.2 – the only difference is the inner workings of the link predictor.

To understand this, first note that KGEMs are trained to assign scores to triples based on learnable embeddings. They represent a triple (s, p, o) with the embedding of its subject, predicate, and object in the form (e_s, e_p, e_o) . In fact, (e_s, e_p, e_o) can be considered to be a representation of the triple as a whole – composed of three atomic units. These embeddings are what KGEMs learn and use to score triples to perform link prediction. This process is shown mathematically in Equation 3.3:

$$KGEM(e_s, e_p, e_o) \rightarrow Score \quad (3.3)$$

Structural Alignment posits that it is possible instead to assign scores to triples based on their structure. This means that in a structure-based approach, the numeric representation of a triple is fixed (i.e. its values are not learnable parameters) and atomic (i.e. the representation of a triple cannot be broken down into smaller units representing the subject, predicate, and object). These vectors of structural features representing each triple serve as the input to a machine learning model that outputs a score for each triple. The brunt of learning in a structural learner is therefore on the scoring model and its learnable parameters, not on learning the representation of a triple. This process is shown in Equation 3.4:

$$StructLP(S_{triple}) \rightarrow Score \quad (3.4)$$

where S_{triple} is the structural encoding of a triple using a given set of KG structural features. This equation is left in a general form intentionally; specific instances of what structural features are used will be given in Section 3.2.

Despite the difference in formulation between KGEM-based and structure-based link prediction, the input (triple representations) and output (plausibility scores) of structure-based link prediction is identical to that of KGEM-based link prediction (and of almost all other link predictors in general). In other words, both are equally encompassed by the general form shown in Equation 3.5:

$$LP(triple) \rightarrow Score \quad (3.5)$$

where *triple* is some undisclosed representation of a triple in a KG.

The Structural Alignment Hypothesis claims that structure-based link predictors given in the form shown in Equation 3.4 should be able to perform link prediction. However, it does not posit whether such approaches should exceed, match, or lag behind those in the established state-of-the-art literature on link prediction.

This second claim of Structural Alignment can be validated or refuted, as such, by testing if a structure-based link predictor in the form of Equation 3.4 can learn to predict links. An examination of whether this is possible, and how effectively it is able to perform link prediction compared to the state-of-the-art, is outlined in Chapter 5.

3.2 Instantiating Structural Alignment

Application of the Structural Alignment Hypothesis requires a clear definition of what exactly is meant by “structure” – which structural features it includes and how they are calculated for a given KG. This section explains in full the instantiation of Structural Alignment used in this work. Specifically, it outlines:

1. in Section 3.2.1, what structural features were chosen, the reasoning for choosing them, and how they are calculated,
2. in Section 3.2.2, examples of how structural features are calculated for triples in the train, test, and validation splits of a knowledge graph, and
3. in Section 3.2.3, a note of clarification on how link prediction queries are represented using structural features.

3.2.1 Selection of Structural Features

Existing literature has shown the prevalence of frequency-based structural features. As detailed in Section 2.4.1 and Section 2.4.2, four main frequency-based structural features have been repeatedly described in the literature as being relevant to characterising KGs and link prediction:

- **Node Degree.** Node degree is the total number of edges that connect to a node.
- **Relation Frequency.** Relation Frequency is the total number of times a given relationship is used in the triples of a KG.
- **Node-Relationship Co-Frequency.** Node-Relationship Co-Frequency is the number of times a given node and relationship co-occur in the same triples in a KG.
- **Node-Node Co-Frequency.** Node-Node Co-Frequency is the number of times a given pair of nodes co-occur in the same triples in a KG.

Based on the established ability of these features to characterise some elements of link prediction performance and hyperparameter preference, as outlined in Section 2.4.1, they are the structural features used in this work.

Existing literature has further shown that KGEMs tend to learn not only from the subject, predicate, and object in a triple, but also from adjacent triples directly connected to them [7, 8, 64, 104]. Importantly, this effect has been documented across various different KGs (including the standard benchmarks FB15k-237 and WN18RR, among others) and various different KGEMs (including the standard models ComplEx, DistMult, and TransE, among others) [7, 8, 64, 104]. These studies show that alterations in immediately adjacent triples can lead to notable changes in link prediction results at the level of individual link prediction queries [7, 8, 64, 104] – meaning that calculating the structural features listed above for a single triple, while disregarding those adjacent to it, could potentially discard a relevant structural data.

As such, when annotating a triple, this work considers not only the degrees, frequencies, and co-frequencies of those triple elements, but also the structure of the features directly adjacent to it. A visualisation of this is given in Figure 3.1. The nodes and edges of the core triple whose structural representation is wanted are labelled in blue. The triples adjacent to that triple, from which structural features are also drawn, are labelled in red. All other triples, which are not part of structural calculations for the core triple, are labelled in black.

It is critical to highlight that, while the number of structural features that can be calculated for the core triple is always the same, the large (and variable) number of adjacent triples means that a large and widely variable number of structural features could be calculated about them. A constant number of input structural features was desired for both simplicity, as well as to prevent the (potentially very large) number of adjacent structural features from overshadowing the 6 structural features of the core triple. As a result, a fixed number of aggregate statistics of the structural features of neighbour triples are calculated, rather than all possible structural features for them.

The outcome of this is a 2-tier method for calculation of structural features – the first tier includes “fine-grained features” of the core triple, and the second tier includes “coarse-grained features” that summarise the structure of all triples adjacent to the core triple. For these coarse grained features, focus is given particularly to node degree and relation frequency, as those are documented as being the most generally important and influential structural features for characterising KGs and link prediction (as demonstrated Table 2.7).

Finally, since KGs are directed graphs, all structural features are calculated in a position-aware manner. That is, when calculating node-relation co-frequency, both subject-relation and object-relation co-frequency are used so as to maintain the directed nature of KGs.

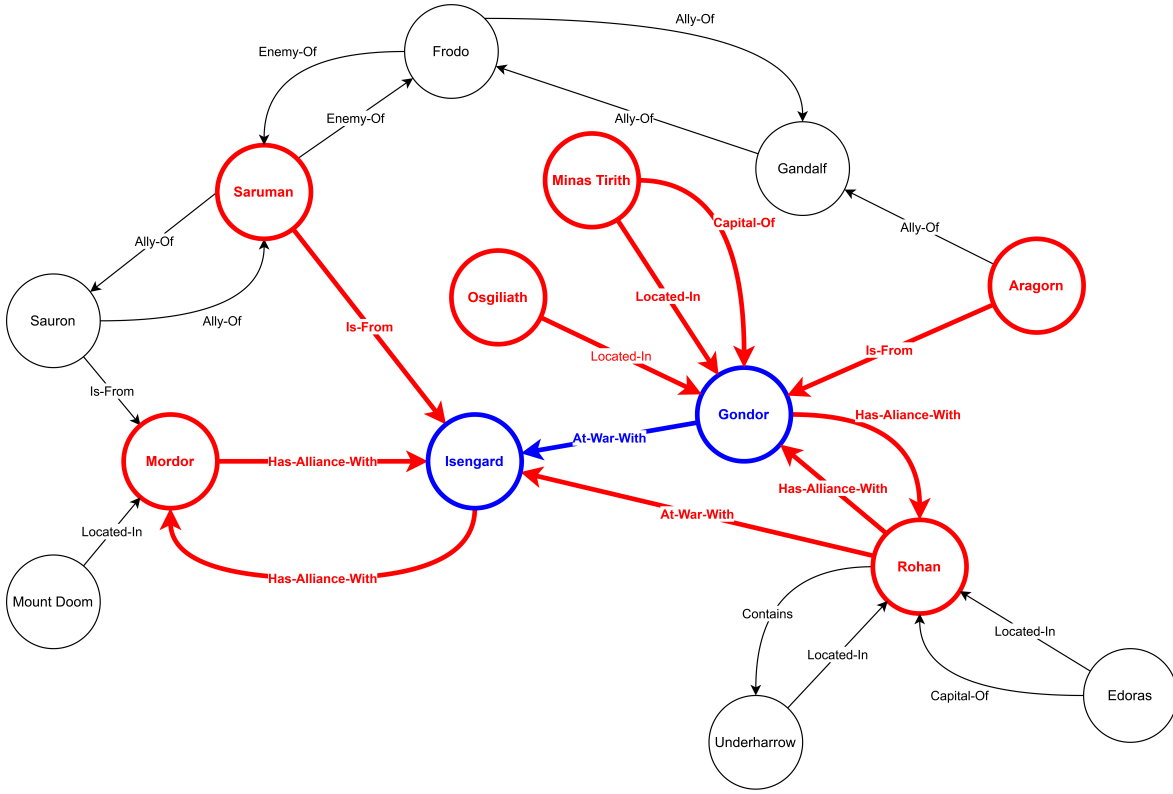


Figure 3.1: The regions from which frequency-based structural features are calculated. The core triple is shown in blue; neighbouring triples whose structure also contributes to annotating the core triple are in red; all other triples (whose structures are not used to annotate the core triple) are shown in black.

A full listing of all structural features used, as well as their meanings and how they are calculated, is given in Table 3.1. Note that *s deg* and *s num neighbours* (as well as *o deg* and *o num neighbours*) are distinct measures: while *s deg* measures the number of links to the subject node, *s num neighbours* measures the number of links from distinct nodes, such that in the case that one node connects to the subject multiple times, it is only counted once.

In all cases that structural features are calculated, they are calculated from the training set of the KG only to avoid data leakage from the testing or validation sets. Examples of how all of these structural features are calculated, both in training and in testing / validation, are given in the following section.

3.2.2 Examples of Structural Feature Calculation

For Triples in the Training Set

Calculation of triple structure involves querying the training set for structural features of the subject, predicate, and object in a triple. Using the knowledge graph as given in Figure 1.1 as an example training set and $(Gondor, At-War-With, Isengard)$ as an example triple from the training set, structural features are calculated from the neighbourhood around that triple as shown in Figure 3.1.

Feature	Meaning
Fine-Grained Fts	
s deg	The degree of the subject node
o deg	The degree of the object node
p freq	The frequency of the predicate
s-p cofreq	The number of times the given subject and predicate co-occur
o-p cofreq	The number of times the given object and predicate co-occur
s-o cofreq	The number of times the given subject and object co-occur
Coarse-Grained Fts	
s min deg neighbour	The degree of the lowest-degree neighbour of the subject
s max deg neighbour	The degree of the highest-degree neighbour of the subject
s mean deg neighbour	The mean of the degrees of the subject's neighbours
o min deg neighbour	The degree of the lowest-degree neighbour of the object
o max deg neighbour	The degree of the highest-degree neighbour of the object
o mean deg neighbour	The mean of the degrees of the object's neighbours
s num neighbours	The total number of distinct neighbours the subject has
o num neighbours	The total number of distinct neighbours the object has
s min freq edge	The frequency of the least-frequent edge linked to the subject
s max freq edge	The frequency of the most-frequent edge linked to the subject
s mean freq edge	The mean frequency of edges linked to the subject
o min freq edge	The frequency of the least-frequent edge linked to the object
o max freq edge	The frequency of the most-frequent edge linked to the object
o mean freq edge	The mean frequency of edges linked to the object
s num edges	The total number of distinct edges incident on the subject
o num edges	The total number of distinct edges incident on the object

Table 3.1: A summary of all graph structural features used, and their definitions. Abbreviations: ft = feature; s = subject; p = predicate; o = object; deg = degree; freq = frequency; cofreq = co-frequency.

There are a total of 6 fine-grained features (i.e. structural features of the core triple (*Gondor*, *At-War-With*, *Isengard*) itself), which are calculated as so:

- **Subject Degree:** a total of 7 triples in the training set involve the subject *Gondor*, so its degree is 7.
- **Object Degree:** a total of 5 triples in the training set involve the subject *Isengard*, so its degree is 5.
- **Predicate Frequency:** a total of 2 triples in the training set involve the predicate *At-War-With*, so its frequency is 2.
- **Subject-Predicate Co-Frequency:** The triple (*Gondor*, *At-War-With*, *Isengard*) is the only triple in the training set in which *Gondor* occurs as a subject and *At-War-With* occurs as a predicate, so the subject-predicate co-frequency of these is 1.
- **Object-Predicate Co-Frequency:** Two triples (*Gondor*, *At-War-With*, *Isengard*) and (*Rohan*, *At-War-With*, *Isengard*) have *Isengard* as an object and *At-War-With*

as a predicate, so the object-predicate co-frequency of these is 2.

- **Subject-Object Co-Frequency:** The triple (*Gondor*, *At-War-With*, *Isengard*) is the only triple in the training set in which *Gondor* occurs as a subject and *Isengard* occurs as an object, so the subject-object co-frequency of these is 1.

Appended to this set of structural features are a set of “coarse-grained” features describing the neighbourhood around the core triple (*Gondor*, *At-War-With*, *Isengard*). These features are divided into two sides: one representing the structure in the neighbourhood around the subject, and the other representing the structure in the neighbourhood around the object. Both of these calculations begin by providing a list of node degrees and predicate frequencies, and then computing aggregate statistics about those lists. There are a total of 16 coarse-grained features, which are calculated as so:

- **Subject-Side Node Features:** A total of 5 distinct nodes connect to the subject *Gondor*; these nodes have the degrees 1 (*Osgiliath*), 2 (*MinasTirith*), 2 (*Aragorn*), 7 (*Rohan*), and 5 (*Isengard*). From these, the minimum degree (2), the maximum degree (7), the mean degree (3.4), and the number of nodes (5) are extracted.
- **Object-Side Node Features:** A total of 4 distinct nodes connect to the object *Isengard*; these nodes have the degrees 5 (*Saruman*), 7 (*Gondor*), 7 (*Rohan*), and 4 (*Mordor*). From these, the minimum degree (4), the maximum degree (7), the mean degree (5.75), and the number of nodes (4) are extracted.
- **Subject-Side Edge Features:** A total of 5 distinct predicates connect to the subject *Gondor*; these have frequencies 5 (*Located-In*), 2 (*Capital-Of*), 3 (*Is-From*), 4 (*Has-Alliance-With*), and 2 (*At – War – With*). From these, the minimum frequency (2), the maximum frequency (5), the mean frequency (3.2), and the total number of distinct predicates (5) are extracted.
- **Object-Side Edge Features:** A total of 3 distinct predicates connect to the object *Isengard*; these have frequencies 3 (*Is-From*), 4 (*Has-Alliance-With*), and 2 (*At – War – With*). From these, the minimum frequency (2), the maximum frequency (4), the mean frequency (3), and the total number of distinct predicates (3) are extracted.

A summary of all 22 of these graph structural features in a single feature vector, as would be used when this instantiation of Structural Alignment is applied, is given in Table 3.2.

Feature	Value
s deg	7
o deg	5
p freq	2
s-p cofreq	1
o-p cofreq	2
s-o cofreq	1
s min deg neighbour	1
s max deg neighbour	7
s mean deg neighbour	3.4
o min deg neighbour	4
o max deg neighbour	7
o mean deg neighbour	5.75
s num neighbours	5
o num neighbours	4
s min freq edge	2
s max freq edge	5
s mean freq edge	3.2
o min freq edge	2
o max freq edge	4
o mean freq edge	3
s num edges	5
o num edges	3

Table 3.2: All structural features that would be calculated for the triple (*Gondor*, *At-War-With*, *Isengard*) in the training set of the example KG given in Figure 1.1.

For Triples in the Testing / Validation Sets

Calculating structural features for triples in the testing / validation sets is done with the exact same procedure as those in the training set. However, triples in the testing / validation sets are (by definition) not in the training set. To avoid data leakage, only the structural features of the training set are used to annotate the structure of triples in the testing / validation sets.

For example, suppose that the structural features of the triple (*Aragorn*, *Enemy-Of*, *Sauron*) from the testing set of the example KG given in Figure 2.5 are to be calculated. Note that *Aragorn* and *Sauron* are not connected by any predicate in the training set. The calculation of their structural features is drawn, as before, from the predicate *Enemy-Of*, from the nodes *Aragorn* and *Sauron*, and from the localised neighbourhoods around those nodes. A visualisation of the regions of the training set used for this calculation is given in Figure 3.2.

The calculation of these features is done exactly as for the training set. When certain features (such as node-node co-frequency) are not observed in the training set (as evidenced by *Aragorn* and *Sauron* not being connected), they are assigned a (co-)frequency value of 0. Cal-

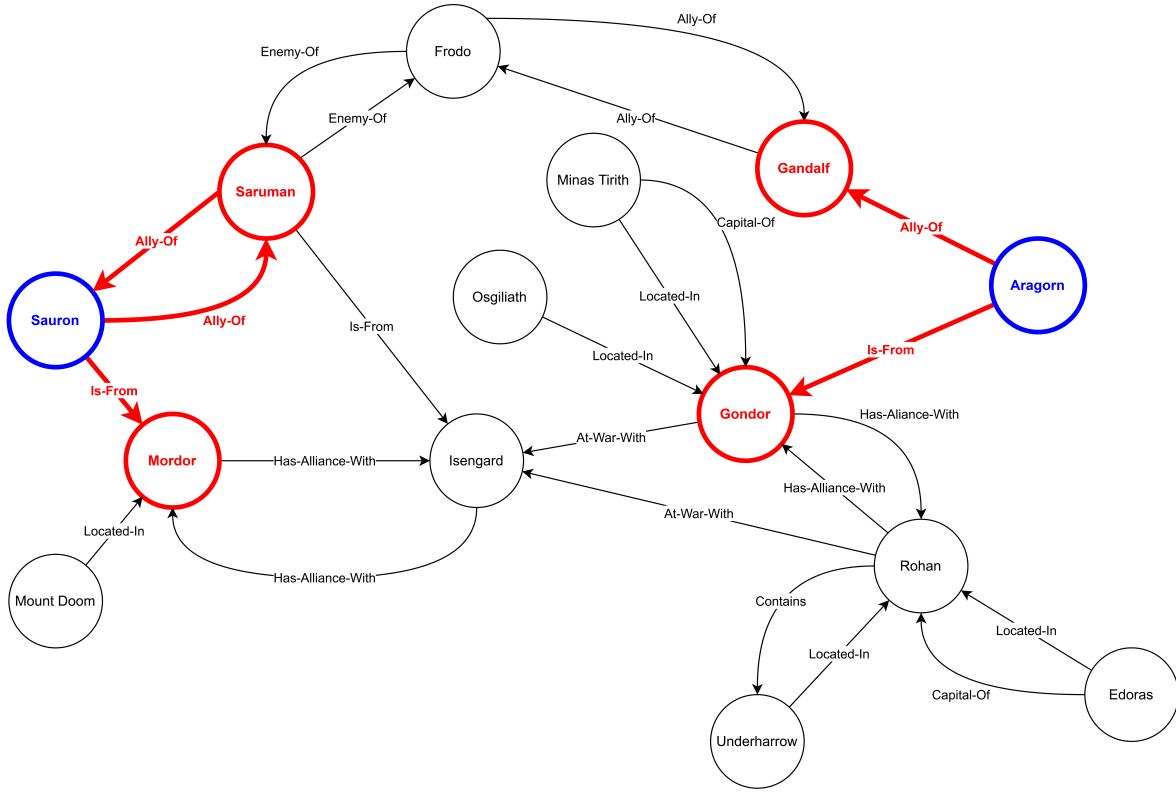


Figure 3.2: The regions of the *training set* from which the structural features for the triple $(Aragorn, Enemy-Of, Sauron)$ in the *testing set* are calculated. Note that as *Aragorn* and *Sauron* are not connected in the training set, these features are necessarily calculated from distinct regions of the graph.

culating features involving the predicate *Enemy-Of* (such as predicate frequency or subject- / object- predicate co-frequency) is just as simple – the frequencies and co-frequencies required are simply queried from the training set again. Critically, note that the potential existence of the triple $(Aragorn, Enemy-Of, Sauron)$ is never taken into account in this process. Since it is in the test set, is not part of the training set and cannot be added to structural feature calculation based on the training set without (necessarily) creating data leakage and biasing structural calculation. The full set of features calculated for $(Aragorn, Enemy-Of, Sauron)$ are shown in Table 3.3.

Note that, as lower-degree nodes were chosen in this case, the structural features produced for it in Table 3.3 are more uniform. This highlights both the power and a potential drawback of this approach – while it can very readily model diversity of structure (as shown in Table 3.2), it also produces more similar, potentially less distinguishable, features when little or no data is available from the neighbourhood of the triple. This effect will be referenced later, particularly in discussion of how it impacts the ability of Structural Alignment to perform link prediction in Chapter 5.

Feature	Value
s deg	2
o deg	3
p freq	2
s-p cofreq	0
o-p cofreq	0
s-o cofreq	0
s min deg neighbour	1
s max deg neighbour	1
s mean deg neighbour	1
o min deg neighbour	1
o max deg neighbour	2
o mean deg neighbour	1.5
s num neighbours	2
o num neighbours	2
s min freq edge	3
s max freq edge	5
s mean freq edge	4
o min freq edge	3
o max freq edge	5
o mean freq edge	4
s num edges	2
o num edges	2

Table 3.3: All structural features of the training set that would be calculated for the example testing-set triple (*Aragorn*, *Enemy-Of*, *Sauron*).

3.2.3 Calculating Structural Features for LP Queries

Now that the calculation of structural features for triples in the training and testing sets has been established, it is important to note how link prediction queries are represented. When a link prediction query in the form $(s, p, ?)$ or $(?, p, o)$ is given, structural features cannot be directly calculated for it. Instead, structural features are only calculated when an object $\hat{o}$ or a subject $\hat{s}$ is proposed to complete the link.

Following the standard link prediction protocol, all possible completions are inserted to result in complete triples in the form $(s, p, \hat{o}_i)$ and $(\hat{s}_i, p, o)$. These triples, having all required elements, can then have their structural characteristics calculated exactly as outlined in Section 3.2.2.

Finally, note that this property of only representing complete (potential) triples is also equally present in KGEMs: the link prediction query $(s, p, ?)$ (or $(?, p, o)$) cannot be scored since scoring functions in KGEMs require embeddings of all three triple elements. KGEMs also can only score triples and produce link prediction results once potential completions in the form $(s, p, \hat{o}_i)$ or $(\hat{s}_i, p, o)$ are provided. However, as link prediction queries are such a core part of link prediction, it is worth highlighting the exact manner by which they can be

resolved in a structural context.

3.3 Summary of Structural Alignment

The Structural Alignment Hypothesis posits that KG learning and link prediction can be modelled in terms of graph structure. The Structural Alignment Framework, presented in this chapter, presents one way of concretely annotating structure such that the Structural Alignment Hypothesis can be directly tested. A summary of the Structural Alignment Framework is given in Figure 3.1 (at a general level) and in Table 3.1 (at the level of specific structural features used).

The following two chapters, Chapter 4 and Chapter 5, both draw upon this Structural Alignment Framework to evaluate how it can be used to simulate the output of KGEMs and to perform link prediction, respectively.

4. TWIG

This chapter describes TWIG (Topologically-Weighted Intelligence Generation), an instantiation of the Structural Alignment Framework that aims to directly model KGEM performance and hyperparameter preference as a function of KG structure. The purpose is to specifically address Claim 1 of the Structural Alignment Hypothesis given in Section 3.1.1; i.e. that hyperparameter preference in KGEMs, and their final performance on the link prediction task, is a function of graph structure. To do this, the TWIG model is constructed to serve as this function that relates structural and hyperparameter information to KGEM performance. This chapter then explores to what extent TWIG can use structure to model KGEM performance and hyperparameter preference at both the global (whole-graph) and local (individual-triple) levels. Finally, it is important to note that the purpose of this study is not to determine the best-possible hyperparameter combinations for KGEMs for benchmark KGs – this has already been done by Ali et al. [2] and by Ruffinelli et al. [73]. Instead, this chapter aims only to evaluate Claim 1 of the Structural Alignment Hypothesis.

The sections contained in this chapter are:

1. Section 4.1, which presents how TWIG, and the data needed for it, were modelled. It also describes how TWIG data was obtained.
2. Section 4.2, which provides an in-depth explanation of structural and hyperparameter data in a specific case-study: the KGEM ComplEx with the KG UMLS. This data is analysed heavily, and results in a variety of key insights into KG structure and hyperparameter preference in general. This analysis is then used to explain how the TWIG neural network was constructed.
3. Section 4.3, which describes how TWIG was extended to be used on other KGs and on other KGEMs, including how it can be used to predict hyperparameter preference and link prediction performance in the few-shot and zero-shot settings.
4. Section 4.4, which extends the evaluation results of TWIG, and the case-study on ComplEx and UMLS, with a manual analysis of KG structure, KGEM hyperparameters, and link prediction performance to allow further general conclusions about how graph structure, hyperparameters, and link prediction performance relate.

Note that some of the data and methods contained in this chapter have been published in peer-reviewed venues by the author [80, 82]. Finally, note that all data used for TWIG can be found at <https://figshare.com/s/7b2da136e05f3548399f>, and that

all the code for TWIG can be found under an open-source licence at <https://github.com/Jeffrey-Sardina/TWIG-TWM-dev>.

4.1 Data and Task Model

As outlined in Section 3.1.1, the aim of TWIG is to predict KGEM performance on link prediction given KG structure and KGEM hyperparameters. This means that TWIG must, in essence, take the form of the function shown in Equation 4.1:

$$Performance = TWIG(S_{KG}, KGEM, Hyp) \quad (4.1)$$

where S_{KG} is some set of structural features for a given KG, $KGEM$ is the scoring function, and Hyp is all other hyperparameters.

This general form is also given as a diagram in Figure 4.1. In this figure, the process of using KGEMs for link prediction, and for using TWIG to simulate KGEM output, are shown side-by-side to highlight the differences in both. Note, in particular, that TWIG uses KG structure and KGEM hyperparameters to entirely skip the embedding step, going directly to predicted rank values.

TWIG’s general form ultimately requires a data model for two distinct data streams: KG structure and KGEM hyperparameters, which are outlined in the following sections.

4.1.1 Modelling KG Structure

Considering that the instantiation of Structural Alignment given in Section 3.2.1 defines KG structure at the level of individual triples, the general form of TWIG shown in Equation 4.1 is specified even further to use the structure around each individual triple as a metric of structure. This yields the general form given in Equation 4.2:

$$Performance = TWIG(S_{triple}, KGEM, Hyp) \quad (4.2)$$

where S_{triple} is the localised structure around a triple calculated as described in Section 3.2.1. Note that all structural values are only calculated from the training split of the KG. This is done because KGEMs are evaluated on the testing / validation splits, and as such using data from either of those splits would result in data leakage and biased evaluation.

4.1.2 Modelling KGEM Hyperparameters and LP Performance

In order to actually create a model such as TWIG, it is necessary to also create a mapping of a wide range of hyperparameters to their ultimate performance on link prediction for a KG.

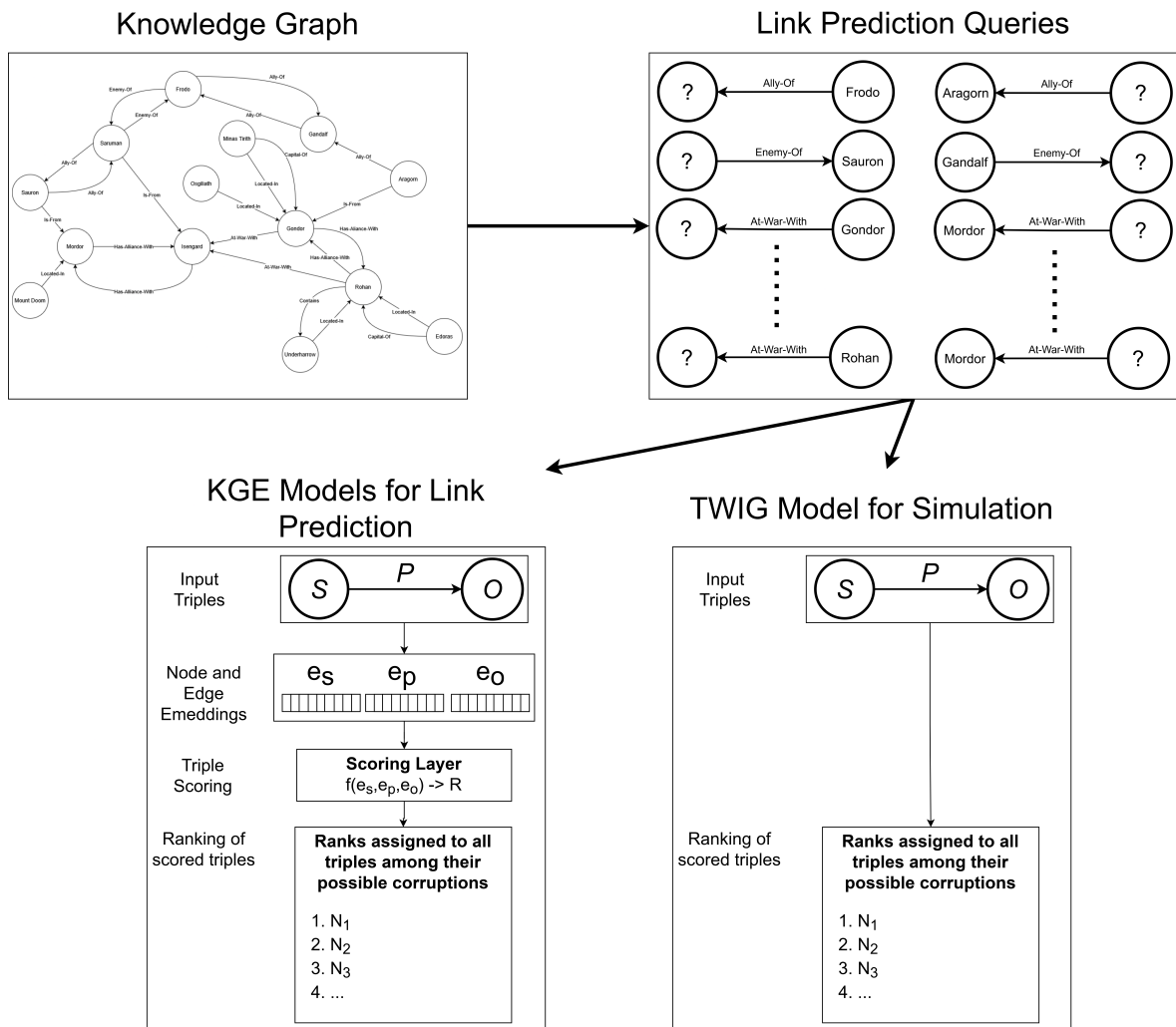


Figure 4.1: An overview of how TWIG learns to simulate KGEMs vs how KGEMs learn to perform link prediction.

To do this, the hyperparameters identified in Section 2.3.1 as critical to link prediction were used to construct a hyperparameter grid containing a total of 1215 possible combinations. This grid is given in Table 4.1.

Note that the batch size hyperparameter is intentionally not varied on the above grid. This was done for two reasons:

1. To allow focusing on hyperparameters more specific to, and more evidenced as relevant to, KGEM-based link prediction.
2. To reduce the size of the (already very large) grid for computational tractability.

Training a given KGEM on a given KG using every possible combination of hyperparameters in this hyperparameter grid would therefore provide all data needed to model both hyperparameter variation and link prediction performance. Note that the whole grid is used, rather than a random subset, to ensure that all results are consistent and reproducible.

Given this model, as well as the model of KG structure in the previous section, it is

Hyperparameter	Values Searched
Embedding dimension	50, 100, 250
Loss Function	Margin Ranking, Binary Cross Entropy (with Logits), Cross Entropy
Margin (if applicable)	0.5, 1, 2
Negative Sampler	Basic, Bernoulli, Pseudo-Typed
Negatives per Positive	5, 25, 125
Optimiser	Adam (constant)
Learning Rate	1e-2, 1e-4, 1e-6
Regulariser	L_3 (constant)
Regularisation Coefficient	1e-2, 1e-4, 1e-6

Table 4.1: The grid of hyperparameters used to train the KGEMs that TWIG simulates.

possible to construct a full data model for TWIG. This unified data model is described in the following section.

4.1.3 Bringing Structure and Hyperparameters Together

While the above data models for KG structure and KGEM hyperparameters ultimately work, they are not immediately compatible. This is because the most natural way to measure KGEM performance is via global performance on its testing / validation set in terms of MRR. However, structure in this work is annotated at the local level of individual triples – and no global structural features are considered.

In order to address this conflict and create a joint data model, analysis of link prediction performance is done at a local level. As outlined in Section 2.2, for every triple (s, p, o) that a link predictor is evaluated on, two link prediction queries are constructed $(s, p, ?)$ and $(?, p, o)$. All possible completions of the triple $((s, p, \hat{o})$ and $(\hat{s}, p, o))$ are found, scored, and ranked. Finally, the rank of the correct completion (s, p, o) is found among all possible alternatives. This results in two rank values for each triple: an object rank value for when the link prediction query $(s, p, ?)$ is posed, and a subject rank value for when the link prediction query $(?, p, o)$ is posed. These ranks describe how well the link predictor was able to correctly answer each link prediction query. Importantly, they are also defined at the level of a triple – meaning that ranks and triple structures can be directly connected in a data model.

The resultant data model takes the form shown in Figure 4.2. For every link prediction query, the rank it was assigned by its KGEM learner is recorded. The triple from which the link prediction query was constructed is annotated with the structural features described in Table 3.1. The link prediction query is finally labelled by the set of all hyperparameters used by the KGEM when learning it.

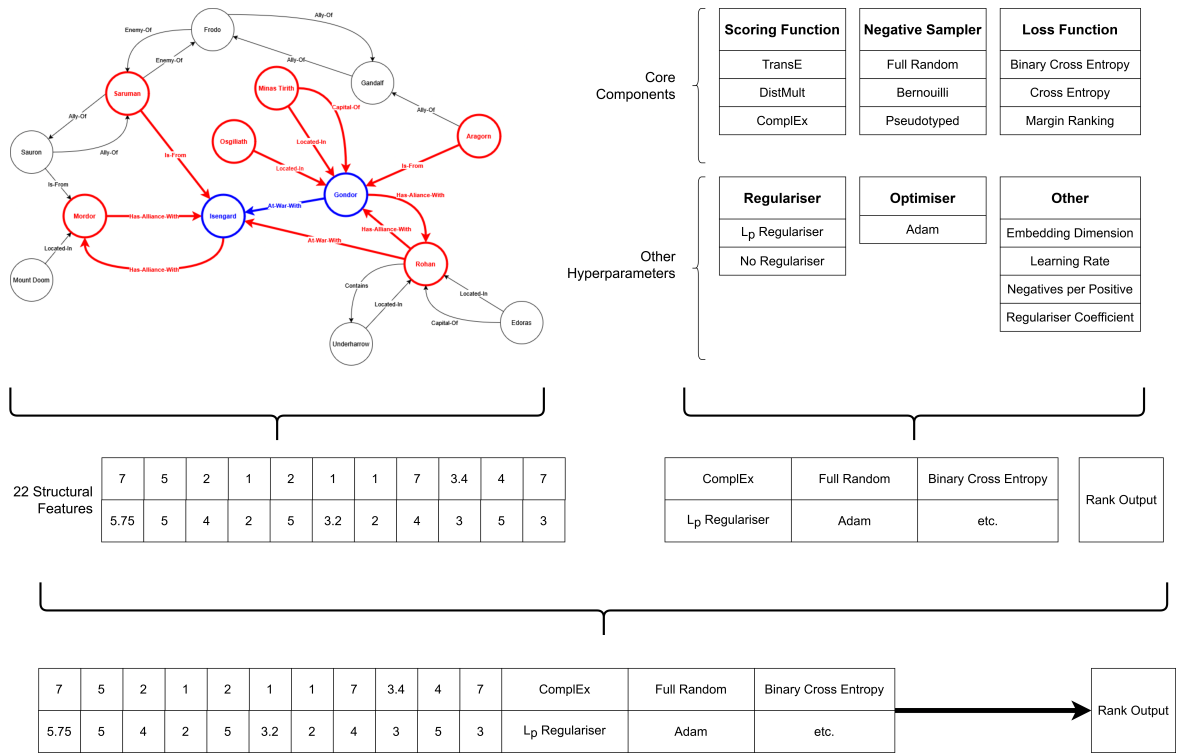


Figure 4.2: A pictorial overview of the TWIG data model.

4.2 Case Study: ComplEx and UMLS

In order to assess the viability of a model such as TWIG, a feasibility study was first done on the specific case of the KGEM ComplEx with the KG UMLS. The KGEM ComplEx was used for this initial study because it is generally considered a state-of-the-art KGEM for practical application [2, 36, 38, 47, 73]. UMLS was used as a KG both because it is a standard biomedical KG [3, 56] and because its small size (6,529 triples across all training, testing, and validation sets), allowed quicker learning and use for iteration. Notwithstanding that this case-study is limited in scope (to one KG and one KGEM), Section 4.4 shows that these results remain relevant across various KGs and KGEMs more generally.

Note that all experiments were run across two different computers with different compute performance, which are listed below:

1. RTX 1070 with 8 GB vRAM and 16 GB RAM
2. RTX 3070-TI with 12 GB vRAM and 32 GB RAM

The following sections explore the three core parts of the case study: determining what sources of signal could be used for learning, determining how to design TWIG’s neural architecture, and defining a training and evaluation protocol for TWIG. Each of these elements are treated in turn in the following sections.

	Exp run 1	Exp run 2	Exp run 3	Exp run 4
Exp run 1	1			
Exp run 2	0.994	1		
Exp run 3	0.994	0.995	1	
Exp run 4	0.9939	0.9943	0.9951	1

Table 4.2: Pairwise correlations of MRR scores for all 1215 hyperparameter combinations across 4 runs using different random seeds for each run and hyperparameter combination.

4.2.1 Determination of Signal

The first step necessary to create a TWIG model for ComplEx and UMLS was to determine what sources of signal in the aforementioned data model could be reliably used for learning. For this case study, the link performance outputs (both globally, in terms of MRR and locally, in terms of ranked list results) of ComplEx on UMLS were recorded for each hyperparameter combination as outlined in Table 4.1. This was repeated a total of 4 times using different random initialisation, resulting in 4 replicates of 1215 hyperparameter experiments. The reference implementation of ComplEx in PyKEEN [3] was used for all experiments. In all cases, ComplEx was trained on the training set of UMLS for 100 epochs and then evaluated on the validation set. The validation set was used, rather than the testing set, because use of the validation set is standard when evaluating hyperparameter grids.

An analysis of ranked lists and MRR values across all hyperparameter sets was performed in order to determine if there was a sufficient source of learnable signal, the results of which are described below.

With respect to global link prediction performance in terms of MRR, it was observed that the correlation of MRR values between the four rounds of hyperparameter experiments were greater than 0.99 in all cases. This is shown in Table 4.2.

These results indicate that, even under different random initialisation, global performance of KGEMs is extremely reliable when hyperparameter values are constant. This suggests that learning to predict MRR could be a viable source of signal, as MRR values map so directly to individual hyperparameter combinations.

As mentioned before, local link prediction performance is measured by the rank of the correct answer to each link prediction query among all possible (incorrect) answers. However, it was found that these ranked lists are highly sensitive to random initialisation. In KGEMs, changing the random seed changes the initialisations of all embedding vectors, as well as which negative triples are randomly sampled for each ground-truth triple during training. Interestingly, when the random seed used to initialise the random number generator varied, the correlation of ranked lists trained with different random initialisations, but identical hyperparameters, was near 0. Figure 4.3 shows the distribution of correlation values for all

combinations of ranked lists that used the same hyperparameters but different random seeds.

The effect of random initialisation is not particularly surprising – since ranked lists will (necessarily) contain many elements, even small changes in scores assigned to individual link prediction queries could cause the ranks assigned to them to change. This is especially true for low-scoring elements (ranked at the end of the ranked list), which (as outlined in Section 2.4) tend to contain low-degree nodes that are very difficult to distinguish using KGEMs. However, this does mean that across different random initialisations, which link prediction queries drive link prediction performance in KGEMs will necessarily vary. As such, annotating individual triples as more learnable or less learnable would have little reliable meaning.

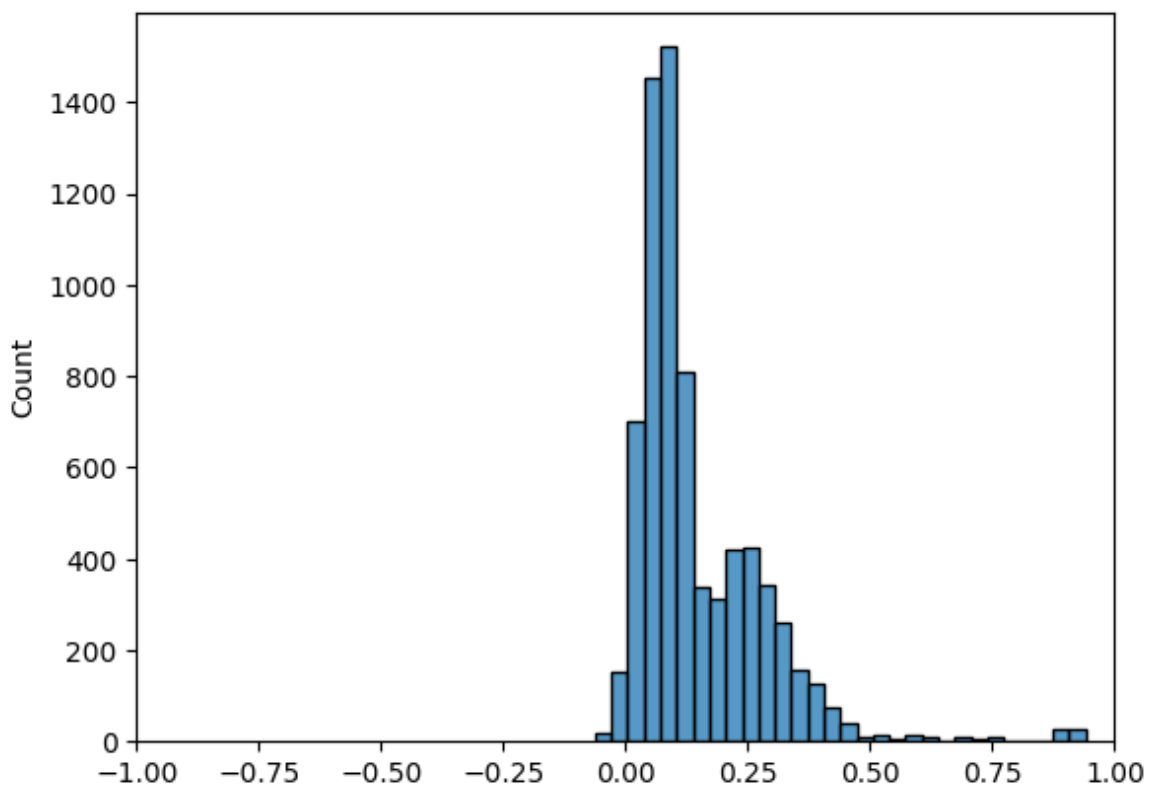


Figure 4.3: Distribution of the correlation values between all ranked lists using the same hyperparameter configurations.

This lack of clear correlation suggests that ranked values assigned to individual link prediction queries cannot be used directly as a source for signal in learning. As TWIG is intended to model both local and global performance, however, an alternate source of signal for learning local KGEM performance was needed. In particular, it was found that while the correlation of ranked lists was low, that the distributions of ranks for each link prediction query tended to match, even across different random initialisations. That makes intuitive sense, because only similar distributions of ranked predictions would result in the high stability of the global MRR value that was already observed for KGEMs across different random initialisations.

	Exp run 1	Exp run 2	Exp run 3	Exp run 4
Exp run 1	0			
Exp run 2	0.0250	0		
Exp run 3	0.0250	0.0245	0	
Exp run 4	0.0248	0.0248	0.0247	0

Table 4.3: Average KL Divergence values of the distributions of values in output ranked lists for all 1215 hyperparameter combinations across 4 runs using different random seeds for each run.

The degree to which two distributions matched was measured using KL divergence. Average KL Divergence values between experiments with all matching hyperparameters is given in Table 4.3. Note that KL divergence values were calculated after creating histograms of ranks with a total of 30 bins to represent rank distributions. Finally, note that lower KL divergence values indicate greater similarity of the distributions being compared.

In summary, two sources of signal were identified that could be very readily used to model hyperparameter preference and link prediction performance:

1. near-1 correlation of MRRs from KGE models run on the same hyperparameters but with different random seeds, and
2. near-0 KL Divergence of the distribution of values in output ranked lists from KGEMs run on the same hyperparameters but with different random seeds.

These two sources of signal also immediately suggest in part how to mathematically define TWIG’s learning procedure. Based on these sources of signal, two loss functions for use in TWIG were defined:

1. **Mean Squared Error (MSE) Loss** between predicted MRR values and actual MRR values, to take advantage of how reliably the same MRRs were observed for the same hyperparameter combinations, and
2. **KL Divergence Loss** between the distributions of predicted rank outputs and observed rank outputs.

Now having established sources of signal for TWIG’s learning, as well as its loss functions, the following section will turn to the last remaining element of TWIG: its neural architecture.

4.2.2 Designing TWIG’s Neural Architecture

The design chosen for TWIG’s neural architecture was inspired by the existing dependencies upon hyperparameter choice and KG structure observed in the literature, as outlined in

Section 2.4. Specifically, looking at Figure 2.9, it was observed that the literature has documented two (largely distinct) dependencies of link prediction performance: hyperparameter choice and KG structure. As such, a similar approach was adopted in this work: TWIG’s neural network first learns latent representations of hyperparameter data and of KG structure individually, and only later aggregates them into a single, shared latent representation in the final layers of the network.

In other words, TWIG’s general neural architecture was constructed as shown in Figure 4.4. On the left the documented evidence in the literature for TWIG’s neural architecture is shown; on the right, TWIG’s neural architecture is shown. Note that colour is used to show which evidence in the literature contributed to the creation of which subset of TWIG’s neural architecture. Note that the graph in the left-hand side image is also reproduced in a larger format in Figure 2.9.

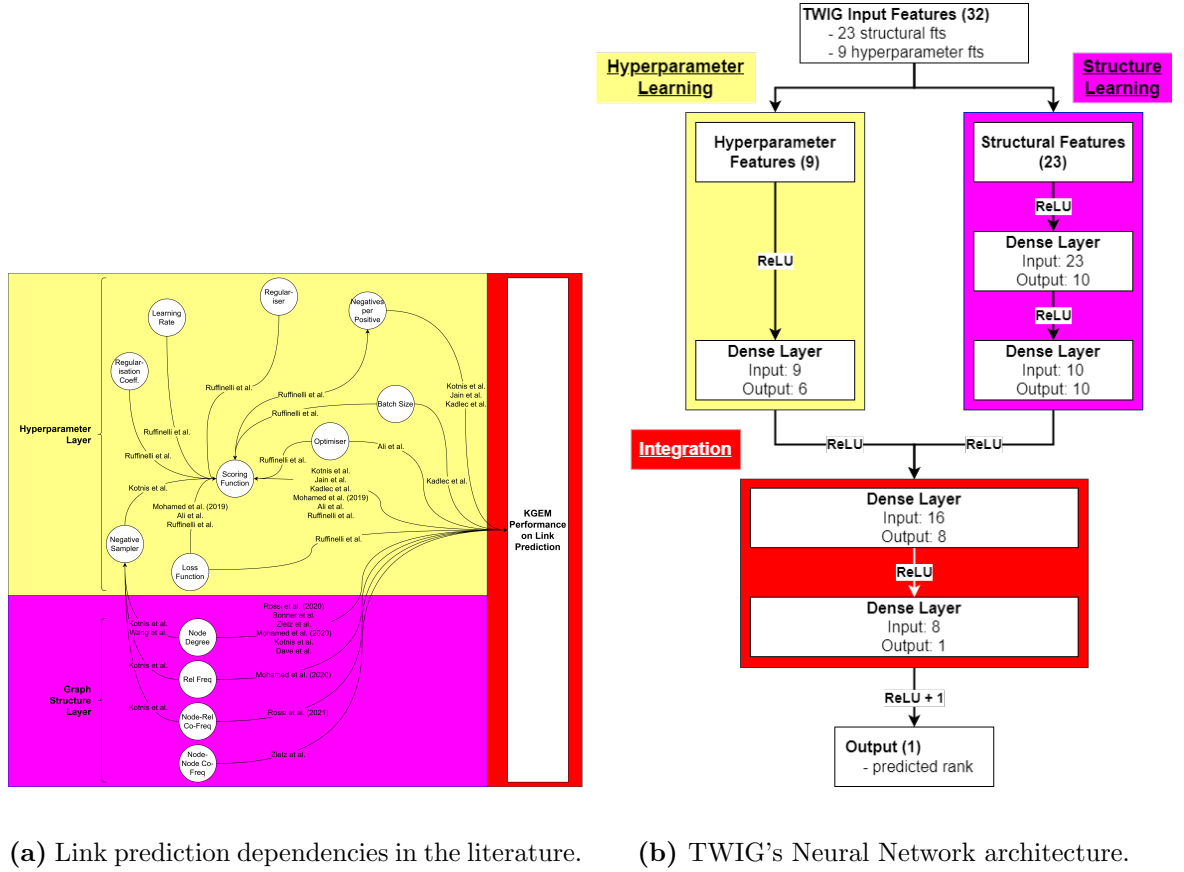


Figure 4.4: The literature describing hyperparameter and structural dependencies in link prediction (left) and how this corresponds to TWIG’s neural architecture (right). The same colour is used to annotate both the neural network component in TWIG and the literature evidence providing motivation for the inclusion of that component.

This neural architecture, as described above, takes as input a vector containing structural and hyperparameter features describing a single link prediction query, and outputs the predicted rank of that link prediction query.

However, this formulation of the neural network has one (critical) issue: from a single predicted value, neither TWIG’s ability to match MRR scores (at the level of global link prediction performance) nor its ability to predict the distribution of ranks (at the local level) can be assessed because both of those measures require having ranks for all link prediction queries in the validation set of a KG at once, not just one of them. In other words, for any given hyperparameter combination used by ComplEx to learn UMLS, TWIG needs to predict all ranks that ComplEx will achieve on all link prediction queries in UMLS’s validation set at once in a setwise manner in order to use the sources of signal determined in the previous section. The manner in which this is done, as well as other critical aspects of the TWIG training and evaluation protocol, are given in the following section.

4.2.3 Designing a Training and Evaluation Protocol

As outlined in the previous section, TWIG’s output cannot be understood in isolation. For example, there is very strong evidence to support using MRR and the distribution of rank outputs as sources of signal for learning (see Table 4.2 and Table 4.3). However, there is also very clear evidence that TWIG’s predictions *cannot* be done at the level of a single link prediction query – as outlined in the previous section, the correlation of individual link prediction query ranks is essentially 0 across identical KGEMs differing only in random initialisation. Link prediction queries, as such, can only be modelled in aggregate (as a distribution) and not at the level of singular link prediction queries.

Instead, the two sources of signal for TWIG and their associated loss functions are defined at the level of all link prediction queries. Since ComplEx was evaluated on the validation set of UMLS for all 1215 all hyperparameter combinations, this means that TWIG must be evaluated on all link prediction queries at once for each hyperparameter configuration in turn. As such, it predicts entire sets of ranks.

This set of ranks can then be used to calculate predicted MRR to be compared to the ground-truth MRR value from ComplEx. Similarly, the distributions of the output ranks from TWIG can be compared to the ground-truth distribution using KL divergence. This set-wise training of TWIG is outlined in Figure 4.5.

TWIG was evaluated in all cases on an $X\%$ hold-out set of hyperparameter combinations. In most cases $X = 10\%$ was adopted, but in some cases even more hyperparameter combinations were held out to assess the impact of different training ratios on TWIG. For each $X\%$ hold-out test set uses, the same hyperparameters were held out each time.

Since its input data consists of 1215 hyperparameter combinations, this means that a random 121 hyperparameter combinations were held-out, and the remaining 1094 were used for training. As such, in evaluation, TWIG was asked to predict the MRR results of hyper-

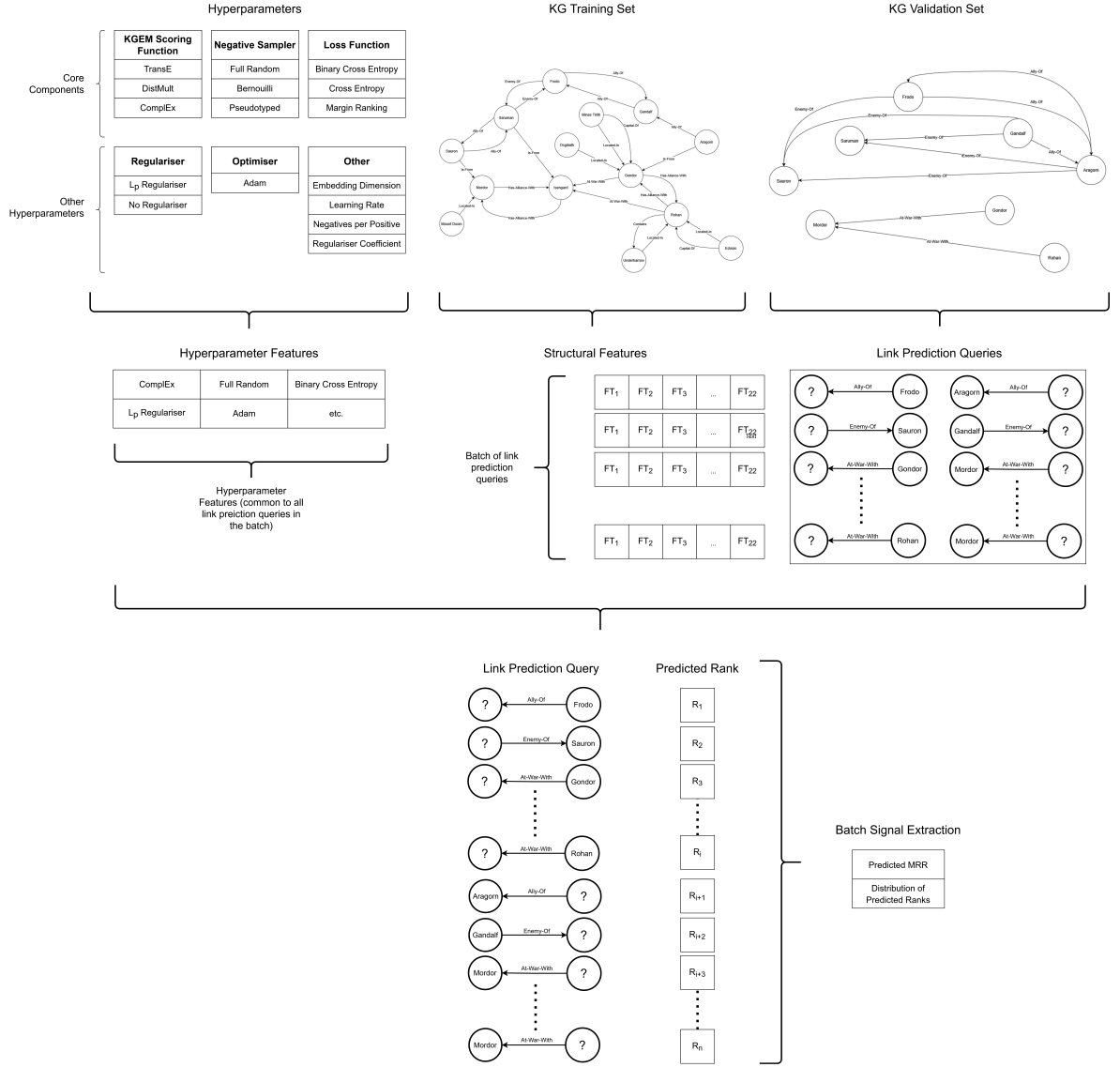


Figure 4.5: A pictorial overview of TWIG’s batch training and evaluation, in which it is trained and evaluated in the context of an entire set of link prediction queries rather than in terms of isolated (single) link prediction queries.

parameter combinations it had never seen before. It was evaluated based on how well its predicted MRR scores matched ground-truth MRR scores from the KGEM it was trained to simulate. The coefficient of determination (R^2) was used as an evaluation metric specifically because it captures what percent of the variation on the value being predicted (i.e. KGEM performance in MRR) is captured by the learner (i.e. TWIG). R^2 is bounded on the interval $(-\infty, 1]$, where

- Values below 0 indicate that the model’s predictions are worse than a baseline that only predicts the average value (and that, as such, the model is not effective).
- Values at 0 indicate that the model’s predictions are just as good as a model that predicts only the average value (and that, as such, the model is not effective).

- Values above 0 indicate performing above the level of a constant baseline, with values nearer to 1 indicating better performance overall.

Finally, previous results on TWIG have shown that it tends to work better using a 2-phase training protocol [80, 82], which is shown diagrammatically in Figure 4.6. In such a protocol, TWIG is trained first only to match the distribution of ranks. After this, all but the final two layers of TWIG’s neural network are frozen, and it is then trained to match both the distribution of ranks and the MRR for each hyperparameter combination. This is in contrast to a 1-phase approach, in which TWIG learns to match the distribution of ranks and MRR values all at once, and in which none of its weights are ever frozen.

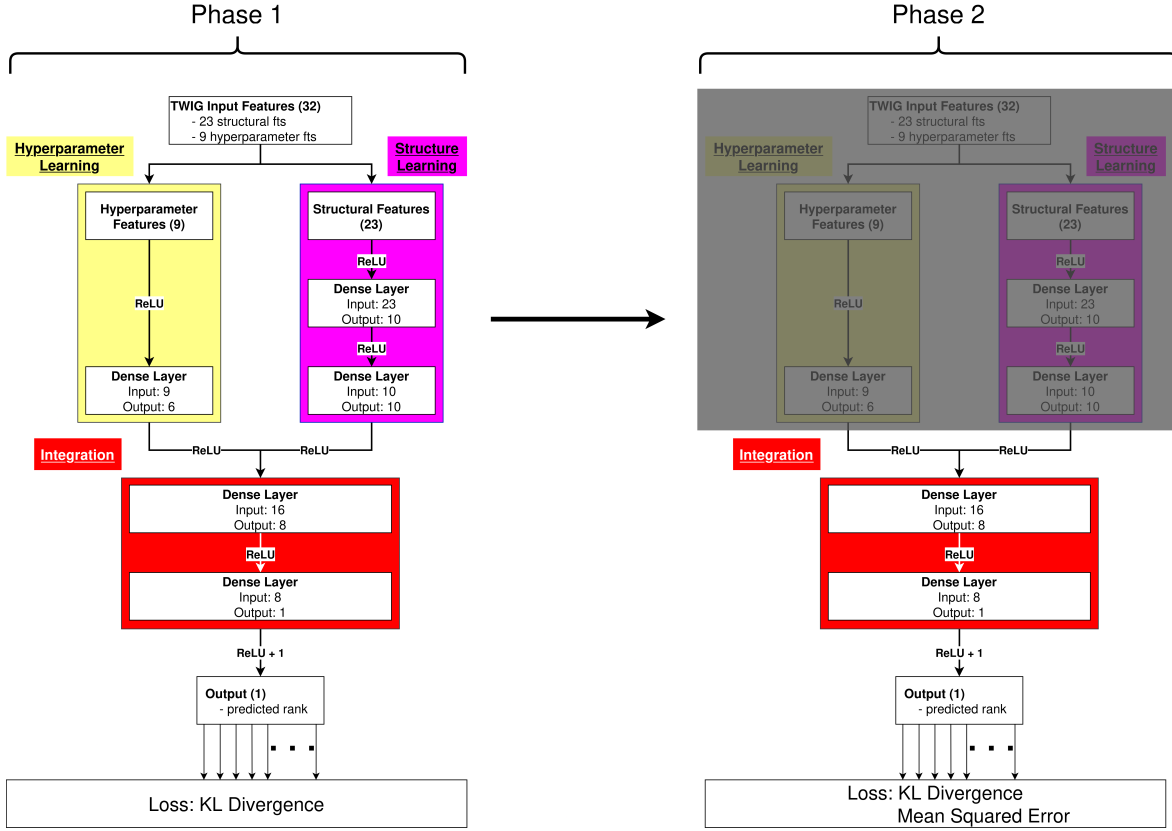


Figure 4.6: The 2-phase training method used for TWIG.

In order to test TWIG’s ability to simulate the output of ComplEx on UMLS, a simple run of 15 total epochs was performed on the ComplEx-UMLS data using both 1-phase and 2-phase training. The choice of 15 epochs was made because 15 epochs alone allowed TWIG to achieve stable results, and because (as seen in Table 4.4 below) more epochs were not needed in order to achieve a significant degree of learning with TWIG.

TWIG’s results are shown in Table 4.4 – overall, it achieved an average R2 score of 0.91 when 1-phase training was used, and 0.94 when 2-phase training was used. As such, and in line with previous TWIG publications [80, 82], in this work 2-phase training is used.

Overall, TWIG’s ability to successfully simulate the output of ComplEx on UMLS sug-

Training Method	KGEM	KG	TWIG's Performance (R2)
1-phase	ComplEx	UMLS	0.86
1-phase	ComplEx	UMLS	0.90
1-phase	ComplEx	UMLS	0.96
2-phase	ComplEx	UMLS	0.94
2-phase	ComplEx	UMLS	0.93
2-phase	ComplEx	UMLS	0.94

Table 4.4: The results obtained by TWIG when trained and evaluated using 1-phase and 2-phase training. In all cases, a 10% hold-out test-set was used. Experiments are performed in triplicate, and all performance values are reported as R2 scores.

gests that it is a viable learning protocol, and that it may indeed be possible to model KGEM performance as a function of KG structure and KGEM hyperparameters. Having now established the TWIG's core methodology, Section 4.3 will detail evaluation results of TWIG across various KGs and KGEMs. The following section provides implementation details of TWIG (such as its hyperparameters and data normalisation) that have not yet been described.

Implementation Details

TWIG has several implementation details that must be noted for completeness. First, for all runs of TWIG, a fixed set of hyperparameters are used. Once again a total of 15 epochs is used. The Adam optimiser was chosen because it is generally considered to be state-of-the-art for optimising neural networks, and 2-phase optimisation was used in line with the findings from the previous experiment on TWIG's training (see Table 4.4). The remaining values were initially chosen arbitrarily and subsequently adopted because of their strong general performance in all cases TWIG was used, as outlined in the following sections. These hyperparameter values are:

- **Epochs:** 5 (phase 1), 10 (phase 2)
- **Optimiser:** Adam
- **Learning Rate:** 5e-3
- **Number of bins:** 30
- **Regulariser:** None
- **Training Approach:** 2-phase
- **MSE Loss Coefficients:** 0 (phase 1), 10 (phase 2)
- **KL Divergence Loss Coefficients:** 1 (phase 1), 1 (phase 2)

Note that “number of bins” refers to the number of bins used to construct histograms of all ranks output by TWIG (or in the ground-truth output of ComplEx). These histograms are constructed so that they can be directly input to the KL divergence function for loss calculation, as KL divergence requires histogram input. As the purpose of this study was not to make TWIG perfectly optimal, but rather to show that it can work effectively, we intentionally omit larger hyperparameter searches for the TWIG model itself as this set of hyperparameters was entirely sufficient.

All structural / hyperparameter input data to TWIG was z-score normalised. In the case of non-numeric hyperparameter values (such as the negative sampler), one-hot coding was used to represent it was numeric attributes.

On top of this, note that TWIG’s output is always passed through a sigmoid layer, so predicted “ranks” are always on the range (0,1). In order to correct for this before MRR or loss values are calculated, the predicted rank is transformed to the distribution $(1, Rank_{max})$, where $Rank_{max}$ is the maximum possible rank for the given KG.

Finally, previous versions of TWIG have used a 2-phase learning protocol to force TWIG to first learn to match the distribution of ranks and then to match MRR [80, 82]. As the results in Table 4.4 show, doing so results in on average better (and more consistent) results than using only one-phase training, so in this work 2-phase training is used. For hyperparameters that vary by phase, two values are given in the listing above.

4.2.4 Choosing KGs for Use

A total of 5 KGs were selected to evaluate TWIG: CoDExSmall [75], DBpedia50 [88], Kinships [42], OpenEA [91], and UMLS [56]. These datasets were chosen for both their relatively small size, which allowed all hyperparameter combinations to be run on them in a feasible amount of time, as well as for their very wide diversity of structure. A structural characterisation of all of these datasets is given in Table 4.5.

In this table, it can be seen that most of the KGs have very skewed node degrees and relationship frequencies. The one exception to this (on both accounts) is Kinships, which has a much higher-valued and more uniform distribution of node degrees and relationship frequencies than is observed in any other KG. DBpedia50 and OpenEA are notable for being on the other end of the extreme – very sparse in most of the graph, but with a very small number of nodes and relations with extremely high degrees and frequencies. CoDExSmall and UMLS both lie between these two extremes, being neither as sparse and skewed as DBpedia50 / OpenEA, nor as dense and uniform as Kinships.

In terms of size, these KGs vary from 6,529 to 38,265 triples. CoDExSmall, DBpedia50, and OpenEA are larger, while Kinships and UMLS are on the smaller side of that range.

	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
Node Stats					
min degree	10	1	148	1	3
25% degree	15	1	161	2	31.5
50% degree	17	1	164.5	3	58
75% degree	25	2	168	4	82.5
max degree	1008	781	174	285	306
Relation Stats					
min freq	1	1	2	1	1
25% freq	28.25	3	183	2	16.25
50% freq	143.5	10	367	8	45
75% freq	370.5	46	404	42.25	156
max freq	10197	3006	1004	4788	803
Other Stats					
#triples (all splits)	36543	34421	10686	38265	6529
#nodes	2034	24624	104	15000	135
#relations	42	351	25	248	46

Table 4.5: Structural characterisation of all five KGs used in the TWIG experiments. Percents in the table refer to percentiles; i.e. “25%” refers to the 25th percentile of degrees or relationship frequencies, respectively. Stats on the number of triples, nodes, and relations were taken from PyKEEN [3].

4.3 Evaluating TWIG

To evaluate TWIG, several different approaches are taken. The first, and simplest, is to train TWIG on each KG-KGEM pair individually, and evaluate it on its ability to predict MRR values in a 10% hold-out test set of unseen hyperparameter combinations. This results in a simple matrix of values shown in Table 4.6 in which TWIG clearly is able to predict MRR output with generally high reliability. In particular, while TWIG’s minimum R2 score observed is 0.40, it achieves an R2 score of at least 0.70 in 10 out of 15 experiments. TWIG further achieves an R2 score of at least 0.90 in 5 out of 15 experiments. Since R2 represents the percent of variation in ground-truth MRR that TWIG can model, this means that TWIG is able to model 70% of all variation in MRR results of KGEMs in a large majority (two-thirds) of all cases, and 90% of all such variation in a third of all cases tested. These results show clearly that, for a given KG-KGEM pair, TWIG can predict the performance of various hyperparameter combinations on that KG-KGEM pair with generally decent accuracy.

	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
Complex	0.76	0.71	0.92	0.40	0.94
DistMult	0.49	0.65	0.74	0.87	0.98
TransE	0.43	0.43	0.98	0.73	0.97

Table 4.6: Results of TWIG when trained and evaluated on every KG-KGEM pair individually. All results are R2 values between predicted and ground-truth MRR values on a 10% hold-out test set.

To the extent of the knowledge of the author, no other research has attempted to simulate link prediction output in the way that TWIG does. However, the use of the R2 metric implies a simplistic baseline. The average-baseline, in which only the average MRR value is predicted, by definition has an R2 score of 0. As such, TWIG’s R2 values represent increase in learning capacity relative to a baseline that always predicts the average MRR.

While these values provide some insight into TWIG’s performance, they do not fully explain where TWIG’s performance comes from. To help explain this, Figure 4.7, Figure 4.8, and Figure 4.9 provide details on TWIG’s patterns of prediction on every KG and KGEM.

In each figure, the first column shows a scatter-plot of the ground-truth MRR and the MRR value predicted by TWIG for all hyperparameter combinations in the hold-out test set. The second column shows the distribution of all ground-truth MRR values in the hold-out test set. Finally, the third column shows the distribution of all MRR values predicted by TWIG for the hold-out test set. From these figures, several general trends can be seen:

- **Homoscedasticity.** TWIG’s predictions tend to be similarly accurate regardless of whether the ground-truth MRR value is low, moderate, or high. Where there is variation in TWIG’s prediction, it tends to be stronger at lower ground-truth MRR values.
- **Clear correlation of predictions.** Even in cases where TWIG achieves relatively low R2 values (such as on DistMult / CoDExSmall), there is a clear trend of TWIG being able to correctly match the variation in true MRR values over their full range. TWIG experiments with higher R2 values, of course, show this trend much more clearly.
- **Hyperparameter clusters.** In many cases, there is a clear trend that various hyperparameter combinations fall into visually distinct clusters (for example, as seen on all three KGEMs on UMLS). There is typically a very large collection of low-performing hyperparameters (seen as a large left-hand spike in the histograms) and a smaller, but still clear, set of high-performing hyperparameters in the distribution tail. This effect holds in general for both ground-truth MRR values and for those predicted by TWIG.
- **Distributional and pointwise accuracy.** In almost all of the 15 cases, TWIG shows high accuracy on individual MRR predictions (as seen in the scatter-plots) and on reconstruction of the general distribution of MRR values (as seen in the histograms).
- **Non-linearity of predictions.** While a perfect predictor would be linear (on the line $y=x$), in many cases TWIG actually produces predictions on a non-linear curve (TransE / Kinships shows this effect very clearly). While this indicates a reduction in the accuracy of individual predictions, it also shows that TWIG, even when inaccurate, tends to correctly assign higher MRR predictions to higher ground-truth MRR values.

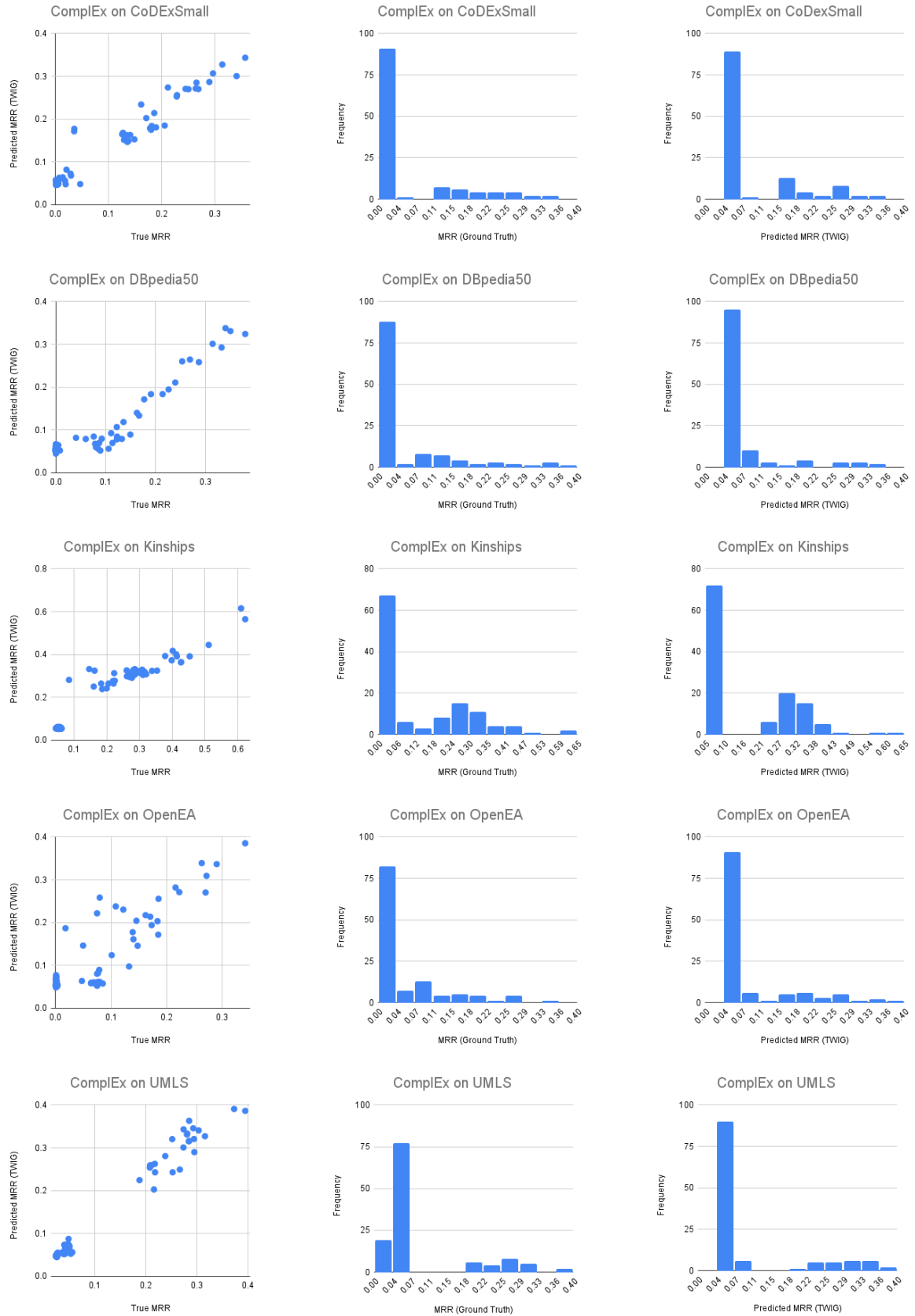


Figure 4.7: Details of TWIG on ComplEx. The first column shows scatter plots of all ground-truth MRR values vs TWIG’s predicted MRR values for all hyperparameter combinations in the hold-out test set. The second shows the distribution of the ground truth MRR values, and the third shows the distribution of TWIG’s predicted MRR values.

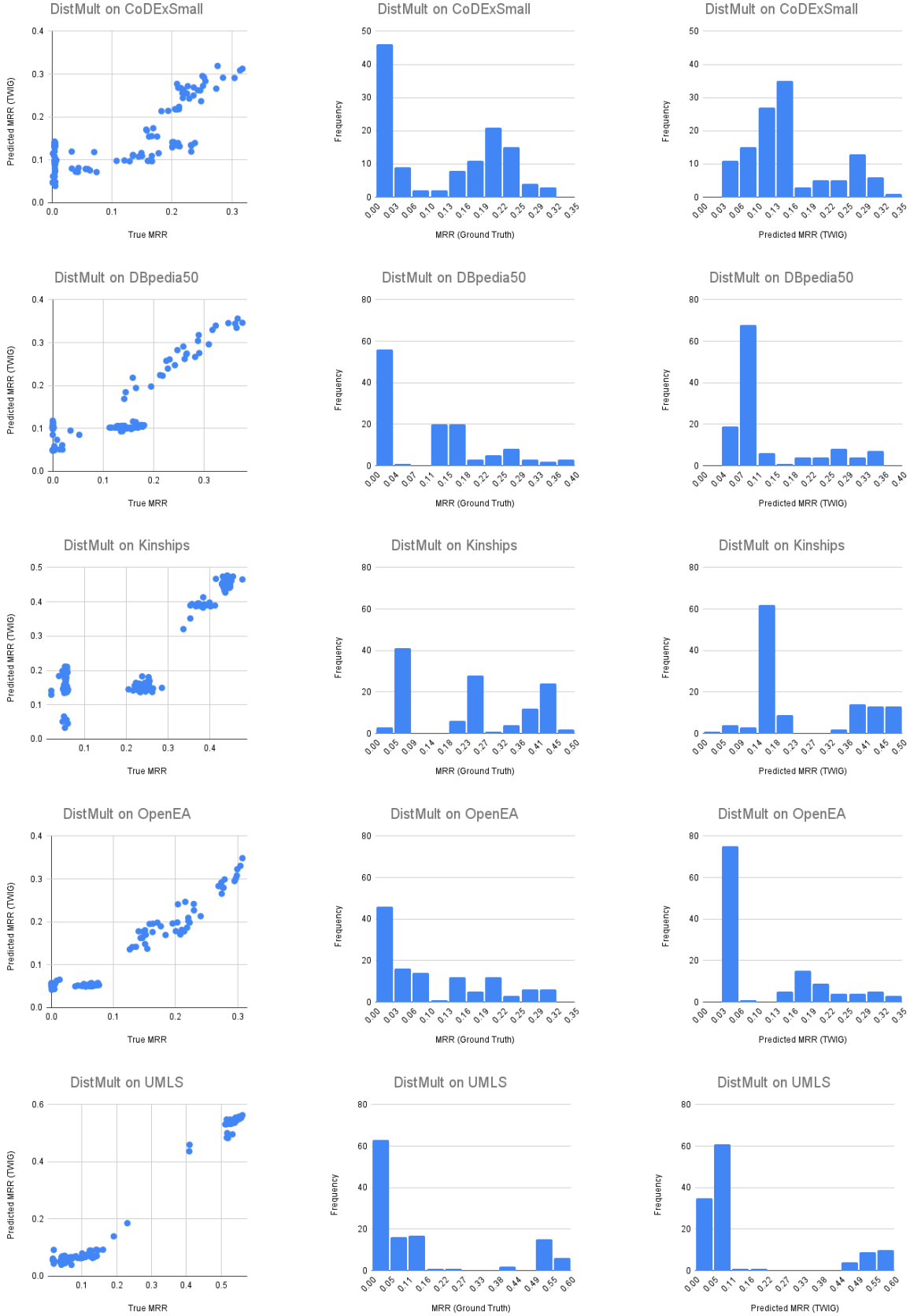


Figure 4.8: Details of TWIG on DistMult. The first column shows scatter plots of all ground-truth MRR values vs TWIG’s predicted MRR values for all hyperparameter combinations in the hold-out test set. The second shows the distribution of the ground truth MRR values, and the third shows the distribution of TWIG’s predicted MRR values.

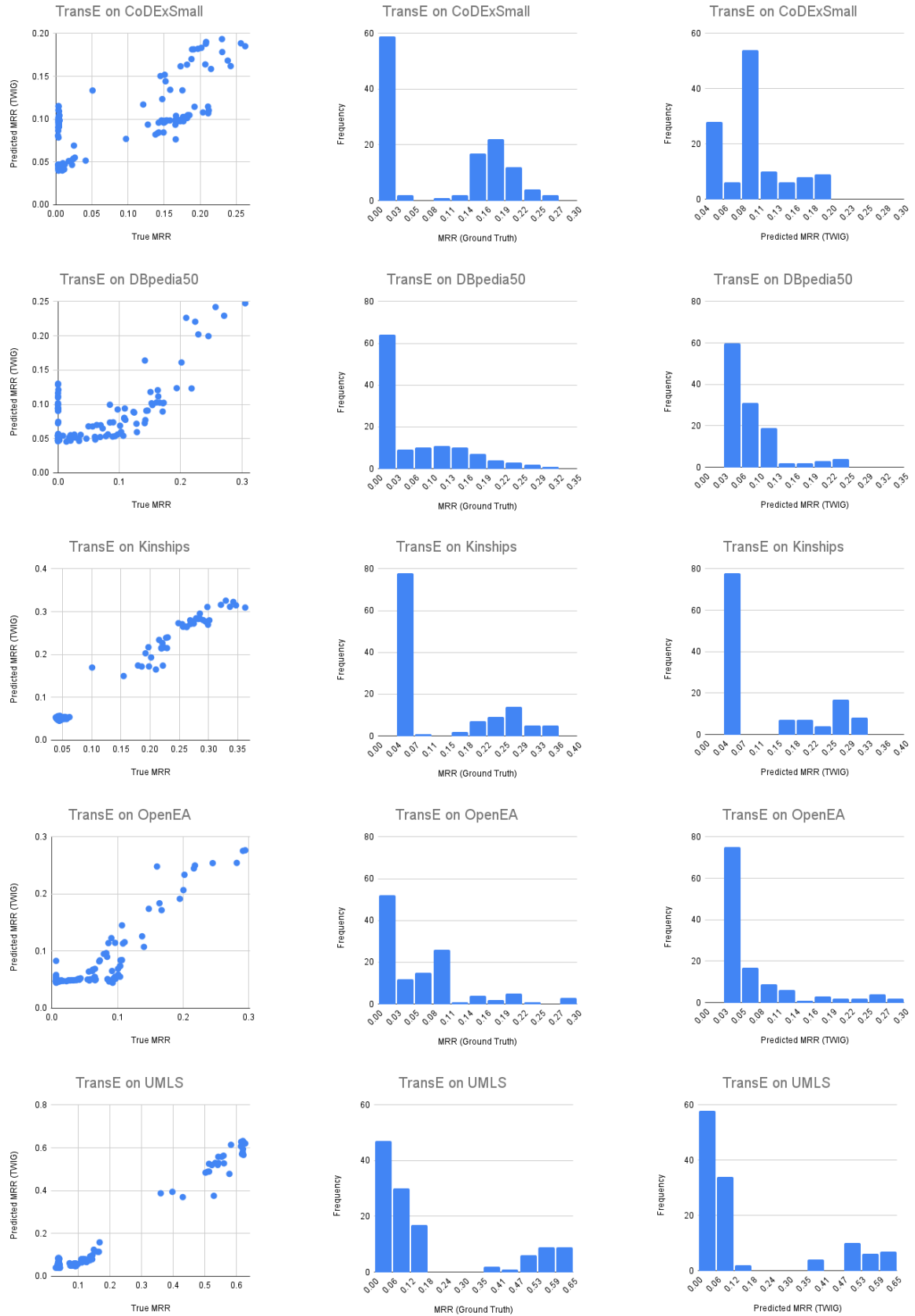


Figure 4.9: Details of TWIG on TransE. The first column shows scatter plots of all ground-truth MRR values vs TWIG’s predicted MRR values for all hyperparameter combinations in the hold-out test set. The second shows the distribution of the ground truth MRR values, and the third shows the distribution of TWIG’s predicted MRR values.

Overall, this data shows that TWIG’s performance is driven by a general ability to model the full range of ground-truth MRR values across the spectrum of low-performing and high-performing hyperparameter combinations. Further, TWIG’s ability to capture both pointwise and distributional aspects of MRR scores for various KG / KGEM pairs suggests that it is able to simulate the output and functioning of knowledge graph embedding models.

While this experiment does therefore technically answer the question of whether TWIG can accurately model hyperparameter preference and KGEM performance, it stops short of examining whether it is generalisable, or determining exactly what features in TWIG’s input lead to its success. As such, the following sections examine TWIG’s ability to generalise across multiple KGEMs and KGs at once. The results of these experiments show that TWIG has the ability to model link prediction in the cross-KG and cross-KGEM settings. Further experiments indicate that TWIG can effectively perform transfer learning, in which it leverages information from a pre-training phase to predict hyperparameter preference and KGEM performance from a much smaller set of observed hyperparameter combinations in the training set. Finally, a set of ablation studies is performed to provide a clear roadmap of which features TWIG uses for learning.

4.3.1 Cross-KGEM Evaluation of TWIG

In this section, TWIG is evaluated for its ability to make prediction across multiple KGEMs at once, but with a constant KG that all KGEMs are trained on. To do this, TWIG is evaluated in a setting where it is trained on 90% of the hyperparameter combinations for the KG for each KGEM, and then evaluated on the remaining hold-out 10%. The results of this experiment are given in Table 4.7. Note that finetuning experiments are not performed in this case because TWIG has already seen the 90% training set of hyperparameters before – it would simply be learning to predict the output of a new KGEM on each KG, similar to what was done to produce the results in Table 4.6.

	Complex	DistMult	TransE
CoDExSmall	0.41	0.41	0.44
DBpedia50	0.21	0.38	0.14
Kinships	0.88	0.63	0.71
OpenEA	0.18	0.64	0.67
UMLS	0.11	0.89	0.89

Table 4.7: Results of TWIG when trained and evaluated on 1 KG and all 3 KGEMs at once. All results are R2 values between predicted and ground-truth MRR values on a 10% hold-out test set.

These results indicate that cross-KGEM prediction is substantially harder than cross-KG prediction: looking at the results in Table 4.7 (cross-KGEM) and in Table 4.8 (cross-KG; in

the following section), it is clear that TWIG performs substantially better in the cross-KG setting. This is unsurprising – the structural features TWIG uses are meant to capture the variations in a KG. When the KG no longer varies, but the KGEM does, TWIG is suddenly asked to predict data that looks structurally identical and differs only in one hyperparameter value: the KGEM scoring function.

In light of the expected difficulty of this task, it is particularly notable that TWIG appears to do reasonably well on Kinships and UMLS – two datasets it was particularly effective on in the initial study of TWIG on every KG-KGEM pair (see Table 4.7). That said, while TWIG *can* operate in a cross-KGEM manner, it is fairly clear this the current formation of TWIG is *not optimal* for this purpose exactly for the reason that TWIG describes the qualities of KG structure more than it describes the properties of KGEM scoring functions. Nevertheless, this represents an interesting finding, and is left in this thesis explicitly by the author to promote future work in this direction.

4.3.2 Cross-KG Evaluation of TWIG

In this section, TWIG is evaluated for its ability to make prediction across multiple KGs at once, but with a constant KGEM to simulate. In order to do this, TWIG is evaluated in a setting where it is trained on 90% of the hyperparameter combinations used for running a KGEM on each KG; it is then evaluated on the remaining hold-out 10%. The results of this experiment are given in Table 4.8.

	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
ComplEx	0.77	0.53	0.94	0.51	0.93
DistMult	0.43	0.62	0.80	0.62	0.97
TransE	0.54	0.50	0.98	0.74	0.97

Table 4.8: Results of TWIG when trained and evaluated on 1 KGEM and all 5 KGs at once. All results are R2 values between predicted and ground-truth MRR values on a 10% hold-out test set.

Overall, these results show that TWIG is indeed able to learn to predict KGEM performance, on various different hyperparameter sets, on multiple KGs at once. However, its performance generally lags behind that of when TWIG is trained on each KG-KGEM pair one-at-a-time. This result is unsatisfying – if Structural Alignment holds, it would be expected that seeing additional examples of structure should improve, not worsen, TWIG’s performance. This effect is further explored in the following section.

4.3.3 Training TWIG on More Hyperparameter Experiments

While all of the prior experiments (and particularly the cross-KG experiments in the previous section) show that TWIG is able to predict hyperparameter preference and link prediction performance from KG structure, they have one theoretical shortcoming. As established in Section 4.2.1, different random initialisations result in widely different ranks being assigned to various link prediction queries. However, TWIG was never given enough information to model this effect, as it was always trained on one initialisation for each set of hyperparameters.

In order to address this potential impact of this, 3 additional hyperparameter grids were run using the KGEM ComplEx on all five KGs tested, resulting in a total of 4 replicates of each hyperparameter combination for each KG, differing only by random initialisation. This data was then used to train TWIG in the same manner as before: with a fixed set of hold-out hyperparameters. Note that, as before, the same hyperparameters were held out from all training data sources, such that the hyperparameter combinations in the 10% test set had never been seen at any point during training. The results of this experiment are given in Table 4.9.

	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
TWIG R2	0.95	0.72	0.98	0.80	0.96

Table 4.9: R2 values achieved by TWIG, simulating ComplEx, for each KG when trained jointly on the training sets of all KGs, using data from 4 hyperparameter grids run with different random initialisations for each KG. R2 is calculated between observed and predicted MRR values for each hyperparameter combination on each KG.

Comparing these results to those obtained by TWIG (simulating ComplEx) in any of the previous experiments, it can be seen that this new training protocol results in substantially increased performance. This suggests that seeing multiple replicates trained on different random initialisations imparts relevant and useful information to TWIG, beyond that contained in the results of a single hyperparameter grid alone.

Building on these results, TWIG was evaluated in the zero-shot and few-shot settings. In these settings, TWIG was trained with hyperparameter combinations from 4 of the 5 KGs (again using all 4 hyperparameter grids run with different random initialisations). As before, a 10% set of hyperparameter combinations was held out. The fifth KG was also held out. TWIG was then asked to predict the link prediction performance of all hyperparameter combinations on the fifth hold-out KG (including hyperparameter combinations it had seen in training and those it had not). This is referred to as the “zero-shot” setting, as TWIG was asked to make predictions about a dataset it had never seen before.

TWIG was also evaluated under the few-shot protocol, in which it was allowed to see either 5% or 25% of the hyperparameter combinations of the hold-out KG. It was then evaluated

Training KGs				Testing KG		
				0-shot	5%-shot	25%-shot
CoDExSmall	DBpedia50	Kinships	OpenEA	0.64	UMLS	0.97
0.95	0.81	0.98	0.83		0.91	
CoDExSmall	DBpedia50	Kinships	UMLS	0.54	OpenEA	0.97
0.95	0.85	0.94	0.96		0.77	
CoDExSmall	DBpedia50	OpenEA	UMLS	0.65	Kinships	0.99
0.97	0.83	0.83	0.86		0.90	
CoDExSmall	UMLS	Kinships	OpenEA	0.73	DBpedia50	0.86
0.95	0.93	0.95	0.88		0.81	
UMLS	DBpedia50	Kinships	OpenEA	0.73	CoDExSmall	0.98
0.98	0.88	0.98	0.89		0.96	

Table 4.10: R2 values achieved by TWIG (simulating ComplEx) on unseen hyperparameters, as well as in the 0-shot, 5%-shot, and 25%-shot settings on all combinations of 4 training KGs and 1 unseen KG. R2 is calculated between ground truth and predicted MRR values for each hyperparameter combination on each KG. All 4 hyperparameter grid runs using different random initialisations were used for training and testing TWIG.

on the remaining 95% or 75% of hyperparameters as normal. The results of all of these experiments are shown in Table 4.10.

The results in Table 4.10 are quite surprising: TWIG is able to predict the performance of ComplEx on unseen KGs in the zero-shot setting with similar efficacy to how well it can do so when only trained on one hyperparameter grid replicate of each KG-KGEM pair individually (see Table 4.6). For example, TWIG in the zero-shot setting matches TWIG in the single KG-KGEM pair setting for predictive performance on CoDExSmall and DBpedia50, and even exceeds its performance on OpenEA.

Similarly, TWIG in the 5%-shot and 25%-shot settings both match or outperform TWIG trained under any other training protocol tested here. Overall, the extremely high performance TWIG obtains in the few-shot setting suggests that TWIG is very readily able to use transfer learning to predict the performance of different hyperparameter sets on new KGs. Moreover, TWIG can do this while seeing only a very small portion of hyperparameter combinations on those KGs – from 0% (0-shot) to 25% (few-shot).

Finally, it is important to highlight that, as described in Table 2.1 and Table 2.2, these KGs come from different knowledge domains. UMLS is biological; Kinships is anthropological; CoDExSmall, DBpedia50, and OpenEA all model general knowledge. Despite this, zero-shot and few-shot learning work regardless of which KG is held out. For example, even when UMLS is omitted from training, a TWIG model trained on the output of ComplEx on the remaining 4 KGs is able to achieve an R2 score of 0.64 when predicting the performance of hyperparameter combinations on UMLS. The same effect holds for Kinships as the only anthropological dataset. This effect suggests that structural features are sufficient to model KGEM learning not only in the cross-KG context, but in the cross-domain context as well. This is a very important point, and will be discussed further in Chapter 6.

However, in order to fully demonstrate the merits of such a claim, an ablation study must be performed to show that TWIG’s performance in the cross-KG / cross-domain setting actually comes from learning structural features. The following section provides the results of several all-feature ablation studies on TWIG to provide further evidence that this conclusion is indeed correct.

4.3.4 TWIG Ablation Studies

As a final step of verification on TWIG, a full ablation study was performed. In this study, all features were ablated as follows:

- **Hyperparameter features.** Hyperparameter features were ablated individually. For example, in one run TWIG would ignore all loss function data, but use all other hyperparameter and structural features. In the next it would ignore the negative sampler, or the learning rate, and so on such that in every case, exactly one feature was removed.
- **Fine-grained structural features.** Fine-grained structural features were all ablated individually, following an identical protocol to that used for hyperparameter features.
- **Coarse-grained structural features.** Coarse-grained structural features were all ablated in pairs. This was done because each coarse-grained feature naturally comes in pairs – for example, *s mean deg nbr* and *o mean deg nbr*. Since these features calculate the exact same thing, just on different sides of the core triple, they were ablated together as a pair. Other than this, coarse-grained feature ablations were done identically to fine-grained feature ablations.

The results of all feature ablation studies for the KGEM ComplEx on all 5 KGs (CoDExS-mall, DBpedia50, Kinships, OpenEA, and UMLS) are done in both the single-KG setting and in the cross-KG setting in the following sections.

The results of feature ablations on other KGEMs (DistMult and TransE) are largely comparable to those on ComplEx and, as such, are omitted from this main text for brevity. They are included in full in Appendix A.

Ablating in the Single-KG Setting

TWIG was first ablated in the single-KG setting. In this setting, TWIG was trained to simulate ComplEx on each KG tested individually, following the exact same training protocol outlined for single KG-KGEM pair experiments in Section 4.3, except for the removal of various hyperparameter or structural features. A list of all features removed, and the resultant effects on TWIG’s performance, is given in Table 4.11.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.76	0.71	0.92	0.40	0.94
Hyperparameters					
loss	0.67	0.24	0.85	0.39	0.96
neg. sampler	0.20	0.45	0.90	0.42	0.59
lr	0.02	-0.19	-0.01	0.02	0.16
reg. coeff.	0.73	0.47	0.97	0.29	0.87
npp	0.79	0.61	0.94	0.19	0.95
margin	0.75	0.59	0.96	0.57	0.96
dimension	0.83	0.49	0.95	0.60	0.92
Aggregate Fts					
all fine-grained	0.79	0.73	0.96	0.68	0.83
all coarse-grained	0.79	0.66	0.96	0.52	0.96
Structure (fine)					
s deg	0.78	0.5	0.88	0.65	0.94
o deg	0.80	-0.18	0.95	0.44	0.97
p freq	0.82	0.28	0.98	0.65	0.98
s-p cofreq	0.81	0.57	0.95	0.62	0.97
o-p cofreq	0.75	0.32	0.99	0.58	0.98
s-o cofreq	0.77	0.44	0.98	0.18	0.95
Structure (coarse)					
s/o min deg nbr	0.78	0.43	0.98	0.71	0.94
s/o max deg nbr	0.76	0.42	0.98	0.56	0.93
s/o mean deg nbr	0.81	0.62	0.98	0.55	0.95
s/o num nbrs	0.76	0.46	0.96	0.53	0.97
s/o min freq rel	0.84	0.25	0.98	0.61	0.97
s/o max freq rel	0.72	0.38	0.98	0.59	0.96
s/o mean freq rel	0.83	0.49	0.98	0.72	0.97
s/o num rels	0.82	0.59	0.97	0.52	0.94

Table 4.11: The results of all feature ablation studies (for all KGs tested) when TWIG was trained to simulate ComplEx. All experiments are run in isolation on a single KG-KGEM pair, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2. Results that outperform TWIG when trained on all features are shown in bold.

Several trends become immediately clear from this data. First, hyperparameter features tend to be the most important to TWIG when run on a single KG-KGEM pair. Almost all hyperparameter features lead to significant reduction in TWIG’s performance when removed, indicating that they are important to achieving said performance. The three notable exceptions to this is embedding dimension, margin, and the number of negatives per positive (npp). When removed, in most cases, these actually lead to an increase in TWIG’s ultimate performance.

Conversely, structural data seems to be of little to no use to TWIG in the single-KG setting – in fact, for most KGs it simulates removing structural features tends to lead to an

increase in performance. This is particularly notable in the fact that, even when removing all fine-grained or all coarse-grained structural features, TWIG’s performance still tends to increase, not decrease. Notable exceptions (especially on DBpedia50) exist; however, the general trend of structural features being of little use remains. This directly implies that, in the case of simulating a single KG-KGEM pair, TWIG does not need structural knowledge to make accurate predictions.

While this at first seems contradictory to the Structural Alignment Hypothesis, this interpretation must be tempered by the fact that, in all cases tested, TWIG was trained on exactly one KG and one KGEM. Since the KG was constant, structure would be of little practical use. While structure varied within each KG at the level of individual link prediction queries TWIG was asked to predict, previous analysis shows that such predictions are subject to particularly high noise (see Section 4.2.1), which may obstruct learning. Further, since all structure comes from the same KG, it is never contrasted with other structures when learning in the single-KG setting. The result of this is that structure, while locally variable, is globally constant.

However, the utility of structure in some cases is also quite understandable – since TWIG simulates KGs at the level of individual link prediction queries, and as those queries can be biased by structure (as described in Section 2.4.1), it stands to reason that some elements of structure may remain useful to TWIG, even when structural features on a whole are less relevant than hyperparameter features.

Both of these effects – the strong impact of hyperparameter features and the low impact of structural features on TWIG’s learning – can be readily explained with an analysis of TWIG’s structure and hyperparameter data. In terms of hyperparameter effects, an analysis of the distribution of MRRs for fixed hyperparameter values is given in for ComplEx on UMLS in Table 4.15 and in Appendix C for all other KGs and KGEMs. Looking at this data, it is clear that certain hyperparameter combinations (such as learning rates of $1e-4$ or $1e-6$) in almost all cases lead to massively reduced KGEM performance, and a distributional shift towards generally lower MRR values. The result of this is that removing these hyperparameters from consideration will remove an element of data that is required for TWIG to accurately simulate KGEMs due to the nature of KGEM learning being so dependent on learning rate choice.

In other cases (such as embedding dimension), most KGEMs have little to no changes in performance at either the maximum level, or the distributional level, when it varies. As noted above, removing embedding dimension can actually lead to increased performance of TWIG in simulating KGEMs. Overall, this suggests that these low-variance hyperparameter features contribute largely to noise, as they cannot help explain why a KGEM’s MRR would change significantly.

Turning again to structural effects, the distributions of all structural features are given in Appendix B for every KG tested. This data shows that the values of almost all structural features observed across most datasets (with Kinships as a notable exception) are clearly non-uniform with a very strong right skew. Especially in the first quartile, these values are often near constant – meaning that they cannot contribute heavily to learning this difference between different triples in a graph when used as input to TWIG. While the remaining part of the distribution of structural features is generally rich in information, it is possible that this noise, combined with the difficulty in predicting individual link prediction ranks and the global consistency of KG structure in the single-KG case, result in structural features being a source of noise for TWIG in the single KG-setting.

While single-KG ablations do provide insight into how TWIG learns, they are necessarily incomplete because, as mentioned above, they consider structure that varied only locally (at the level of link prediction queries), not globally (at the level of the KG). In order to more directly test the role of structure in TWIG, a second round of ablation experiments was performed in the cross-KG setting, where KG structure would (necessarily) vary. These experiments, and their results, are given in the next section.

Ablating in the Cross-KG Setting

In order to determine the effect of structural features in TWIG’s ability to perform cross-KG simulation of KGEMs, a second round of ablations was performed in which TWIG was trained in the cross-KG setting as described in Section 4.3.2. In this setting, TWIG was trained on all 5 KGs at once, and then evaluated on the hold-out hyperparameter combination test sets for them. The results of these ablations for the KGEM ComplEx are given in Table 4.12. Once again, note that results for DistMult and TransE are given in Appendix A.

Three trends are immediately discernable from these results:

- **Higher influence of structural features.** Removing structural features, either individually or (especially) in aggregate, results in notable decreases in TWIG’s performance. This effect is much more pronounced than was observed in the single KG setting, where structural features had much lesser impact.
- **Variability in structural feature importance.** Which structural features resulted in increased (or decreased) learning when removed varies by KG. In only one case (*s-o-cofreq*) is removing a structural feature universally beneficial to TWIG’s learning.
- **Continued importance of hyperparameter features.** Hyperparameter features are critical to TWIG’s predictions, and results on hyperparameter importance generally mirror those of the single-KG ablations.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.77	0.53	0.94	0.51	0.93
Hyperparameters					
loss	0.56	0.42	0.93	0.16	0.96
neg. samp.	0.29	0.44	0.90	0.33	0.51
lr	0.04	-0.1	0.02	-0.19	0.20
reg. coeff.	0.62	-0.35	0.92	-0.48	0.72
npp	0.79	0.68	0.94	0.59	0.89
margin	0.72	0.49	0.97	0.54	0.92
dimension	0.74	0.58	0.92	0.56	0.89
Aggregate Fts					
all fine-grained	0.74	0.92	0.86	0.63	0.59
all coarse-grained	0.57	0.45	0.74	0.93	0.92
Structure (fine)					
s deg	0.82	0.63	0.92	0.63	0.95
o deg	0.79	0.41	0.96	0.41	0.82
p freq	0.79	0.66	0.88	0.58	0.94
s p cofreq	0.77	0.45	0.97	0.37	0.96
o p cofreq	0.76	0.57	0.98	0.43	0.92
s o cofreq	0.82	0.57	0.96	0.58	0.96
Structure (coarse)					
s/o min deg nbr	0.74	0.55	0.97	0.58	0.93
s/o max deg nbr	0.88	0.69	0.95	0.65	0.89
s/o mean deg nbr	0.80	0.71	0.96	0.63	0.73
s/o num nbrs	0.71	0.54	0.96	0.45	0.93
s/o min freq rel	0.76	0.58	0.98	0.58	0.97
s/o max freq rel	0.80	0.71	0.94	0.59	0.97
s/o mean freq rel	0.73	0.52	0.98	0.48	0.95
s/o num rels	0.77	0.49	0.93	0.39	0.94

Table 4.12: The results of all cross-KG feature ablation studies (for all KGs tested) when TWIG was trained to simulate ComplEx. All experiments are run in isolation on all KGs at once, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2. Results that outperform TWIG when trained on all features are shown in bold.

In the previous section, it was noted that the lack of global variation in KG structure (due to the use of a single KG only) resulted in structural features likely contributing more noise than signal to TWIG. In these cross-KG ablations, this trend has clearly reversed: in the cross-KG setting, KG structure is a necessary part of how TWIG learns to simulate KGEMs.

It is worth highlighting the case in which structural features are removed in aggregate – i.e. when all fine-grained or all coarse-grained structural features are removed. While in the single KG setting this had low impact, in this setting it has notable (and often high) impact across all KGs except OpenEA. DBpedia50 is a partial exception – removing all fine-grained features results in an increase to TWIG’s performance, but removing all coarse-

grained features hurts TWIG’s performance on it. This effect persists even though, in all cases, the removal of various structural features can lead to increased performance by TWIG. In other words, even though some individual structural features are not useful for TWIG to learn, in aggregate frequency based structural characteristics are very important to TWIG’s learning.

OpenEA and DBpedia50 merit discussion as an outliers to the otherwise clear trend of TWIG’s learning. Among all KGs tested, OpenEA and DBpedia50 are by far the most sparse. Looking at KG structural characteristics in Appendix B, Table B.4 (for OpenEA) and Table B.2 (for DBpedia50), it can be seen that both of these graphs have much lower minimum and first-quartile distributions of node degrees and edge frequencies than CoDExSmall, Kinships, and UMLS.

The fact that DBpedia50 and OpenEA have very different structures than the remaining KGs suggests that part of TWIG’s difficulty in learning them may arise from it having to deal with input data that varies so widely across so many different structures. The same structural feature values may represent different contexts in UMLS (for example) than OpenEA, resulting in them being useful for TWIG when simulating ComplEx on UMLS, but not useful when simulating ComplEx on OpenEA.

On the other hand, coarse-grained structural features of DBpedia50, as well as various structural features of OpenEA (i.e. *o deg*, *s-p cofreq*, *o-p cofreq*, *s/o num nbrs*, and *s/o num rels*) remain important for TWIG to simulate the output of ComplEx on those datasets. In no case in the cross-KG setting are structural features as a whole irrelevant. As such, the core hypothesis behind TWIG – that structure plays a major role in how KGEMs learn various KGs and is needed to simulate the process – holds.

As a final point, it is important to reiterate that, as outlined in the previous section and shown in Appendix C, some hyperparameter features lead to universally poor (or generally very good) results independent of other hyperparameters or structural elements. Such hyperparameters will, clearly, be critical for how TWIG models its data. As a result, the fact that across all experiments hyperparameter features seem to be more directly impactful on TWIG should not be taken as indicative of structural features being irrelevant or of less value. Instead, it should lead to an acknowledgement that the effects of structure can only be seen when learning actually happens – and if certain hyperparameter combinations obstruct learning, then structural effects on learning will necessarily not be present to observe.

4.3.5 A Final Note on TWIG

Taken altogether, TWIG is clearly very well able to model KGEM hyperparameter preference and link prediction performance as a function of hyperparameter features and graph structure.

In settings where TWIG is presented with most constant structure (i.e. in the single-KG setting), ablation studies show that it relies more on hyperparameter features to make its predictions (and does not make heavy use structural features). However, as KG structure varies (i.e. in the cross-KG setting) TWIG becomes much more reliant on KG structure in order to simulate KGEMs, rather than using hyperparameter features alone. This indicates that the process of simulating KGEMs via TWIG is a task that genuinely requires knowledge of both KGEM hyperparameters and KG structure, a finding that aligns with the expectations from the literature presented in Section 2.4. Finally, the fact that structural features are needed specifically in the cross-KG setting supports the Structural Alignment Hypothesis. Structure can be used – and in fact is needed – to effectively understand and model cross-KG trends in KGEM learning.

It is worth highlighting that TWIG was trained on a grid of fixed, clearly defined hyperparameter combinations. While TWIG can effectively predict the output of KGEMs trained on those hyperparameters, however, it was not built to predict the output of KGEMs trained on other hyperparameters – such as much higher embedding dimensions, or loss functions not considered herein. TWIG, in its current form, would not be expected to be able to predict the results of training on such alternate hyperparameter configurations. That said, TWIG’s high success in the 0-shot and few-shot settings suggest that adapting TWIG to do so – based on a new grid of different hyperparameter values or on hyperparameter performance results from a random search – should result in similar efficacy. Such experiments, not being necessary to support the core claim of the Structural Alignment Hypothesis, are left as future directions.

At this point, TWIG has one remaining issue – being a neural network it is, at its core, a black-box. That is to say, the mapping it learns between KG structure and hyperparameter performance is unknown, even though it is known that TWIG is able to make very reliable predictions based on what it learned.

There are many ways to address this. For the purpose of this thesis, the author has chosen to present a separate, manual analysis of KG structure and KGEM hyperparameters based on the data used to train TWIG. While this cannot explain what exactly TWIG does, the intent of this approach is to make structure-hyperparameter interaction maximally clear at a human level – and to leave optimised prediction of KGEM output to TWIG. This analysis can be found in full in the following section, Section 4.4.

4.4 Structure and Hyperparameter Analysis

This section is inspired by TWIG’s generally very strong ability to predict hyperparameter preference and link prediction performance, and performs a detailed manual analysis of what

elements of KG structure and KGEM hyperparameters contribute to this effect. Specifically, this section uses data from the previous hyperparameter grid experiments on the three chosen KGEMs (ComplEx, DistMult, and TransE) and the five chosen KGs (CoDExSmall, DBpedia50, Kinships, OpenEA, and UMLS) to examine:

- **Structural Correlation:** The correlation of all structural features describing each link prediction query to the rank assigned to that link prediction query. This is done to provide evidence for the effect of structure on link prediction performance directly.
- **Hyperparameter Ablations:** The performance of optimal hyperparameters for each KG-KGEM pair, as well as the performance of all hyperparameter sets differing in only one hyperparameter value. This is done to provide evidence for the effect of each hyperparameter choice in link prediction directly.
- **Structural Ablations:** The optimal hyperparameters and overall link prediction performance for each KGEM when it is trained on the entire training set as normal, but evaluated only on link prediction queries in which each structural feature is in the top 50% (or bottom 50%) of all values it can take. This is done to provide evidence of how changing structure can impact hyperparameter preference and link prediction performance.

This is undertaken for every KG-KGEM pair. The results of this are reported in the following section for ComplEx and UMLS. Results on all other KGs (CoDExSmall, DBpedia50, Kinships, OpenEA, and UMLS) and KGEMs (ComplEx, DistMult, and TransE) are included in Appendix C as they require significant page space.

A final section will follow the description of ComplEx and UMLS, drawing on data in this chapter and in the appendix, to present a unified view on structure, hyperparameter preference, and link prediction performance.

4.4.1 ComplEx on UMLS

This section describes data obtained from running all 1215 hyperparameter combinations on ComplEx (as the KGEM) and UMLS (as the KG). This case-study is presented in detail, and the general trends it finds will be placed in the broader context of all KG-KGEM pairs in the following section.

Correlation Analysis

Looking first at the structural data, a correlation analysis was performed to determine how well each structural feature used in TWIG aligned to link prediction performance. For this,

a list of all link prediction queries in UMLS’s validation set was obtained. The values of each structural feature in each link prediction query were then correlated (using Pearson’s r) to the rank each link prediction query obtained. This was done for all ranks (ignoring whether the subject or object was being corrupted) as well as for the ranks of only subject corruptions and only object corruptions. These correlations are shown for all structural features in Figure 4.13, with darker colours indicating higher absolute values of correlation. Note that all ranks used were obtained from the optimal hyperparameter set found for ComplEx on UMLS.

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.24	-0.29	-0.19
o deg	-0.27	-0.24	-0.31
p freq	-0.12	-0.12	-0.13
s-p cofreq	-0.21	-0.25	-0.17
o-p cofreq	-0.08	-0.04	-0.13
s-o cofreq	-0.17	-0.18	-0.15
s min deg nbr	-0.11	-0.06	-0.17
s max deg nbr	-0.34	-0.37	-0.32
s mean deg nbr	-0.3	-0.28	-0.33
s num s o cofreq	-0.24	-0.29	-0.2
s min freq rel	0.03	0.07	-0.01
s max freq rel	-0.24	-0.25	-0.23
s mean freq rel	-0.08	-0.05	-0.12
s num rels	-0.25	-0.29	-0.20
o min deg nbr	-0.08	-0.13	-0.02
o max deg nbr	-0.38	-0.36	-0.4
o mean deg nbr	-0.29	-0.3	-0.27
o num s o cofreq	-0.26	-0.22	-0.31
o min freq rel	0.13	0.12	0.15
o max freq rel	-0.35	-0.34	-0.37
o mean freq rel	-0.12	-0.13	-0.11
o num rels	-0.27	-0.24	-0.31

Table 4.13: ComplEx on UMLS: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

The first thing to note about the correlations shown is that 20 out of the 22 structural features examined negatively correlate to rank when looking at correlations to all link prediction queries. This means that as the values of those structural features increase, the rank assigned to the correct answer to link prediction queries decreases. Since lower ranks indicate better link prediction performance, this means that higher values for almost all structural features result in improved link prediction performance for ComplEx and UMLS. It is also notable that, for all structural features analysed, increasing values indicate greater connectivity in some manner – whether it be a node with more connections, a predicate with higher use, or

a greater co-frequency between different elements of a triple. These results therefore agree with previous observations in Section 2.4.1 that more dense regions of a graph tend to be learned better.

The two features that are exceptions to the negative-correlation trend are *s min freq rel* and *o min freq rel*, both of which are coarse-grained structural features calculated at the level of all triples connecting to the central triple from which a link prediction query was constructed. Both of these have comparatively low (absolute) correlation values. It is possible that such a (low) positive correlation could be spurious; if not, it may reflect a preference for diversity in local structure; i.e. having lower minimum and higher maximum connectivity around a link prediction query.

Of all correlations shown for the set of all link prediction queries, none exceed an absolute value of 0.40; the highest absolute-valued correlation is -0.38, for the structural feature *o max deg nbr*. A total of 12 had a correlation with absolute value above 0.20, and 4 had a correlation with absolute value above 0.30. This leaves 10 structural features with an absolute-valued correlation of under 0.20, of which 5 (out of 10) had a correlation with absolute value above 0.10. Overall, this means that correlation is at best moderate – no one feature selected can (on its own) fully predict the rank assigned to any one link prediction query. This agrees with previous observations in Figure 4.3 that ranked output of KGEMs is largely driven by randomness, and varies widely based on random initialisation even when hyperparameters remain identical.

Looking at *sided* correlations for predicting either the subject or the object only, the same general trend persists. That said, when the subject is being predicted, subject-side features tend to correlate better to rank; similarly, when the object is being predicted, object-side features tend to correlate better to rank. This is not very surprising, seeing as existing literature described in Section 2.4.1 has outlined similar dependence on the degree of the node being predicted.

Hyperparameter Ablations

Turning now to analysis of hyperparameter choice, a table with the optimal hyperparameters (on the first row) and their MRR for ComplEx on UMLS, as well as all the MRR of all possible single-hyperparameter alterations to that set, is given in Table 4.14. Hyperparameters that differ from the optimal values are shown in bold, and table rows are coloured in alternating grey or white to distinguish groups of rows in which different hyperparameters varied.

Overall, the most impactful hyperparameter (measured by change in MRR from the best observed MRR when it was varied) was the learning rate – choice of a lower learning rate led to substantially lower MRR values in both cases. The least impactful hyperparameter was the

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
BCE	Bernoulli	0.01	0.01	25	None	50	0.55
MRL	Bernoulli	0.01	0.01	25	0.5	50	0.19
MRL	Bernoulli	0.01	0.01	25	1	50	0.22
MRL	Bernoulli	0.01	0.01	25	2	50	0.22
CE	Bernoulli	0.01	0.01	25	None	50	0.29
BCE	Basic	0.01	0.01	25	None	50	0.52
BCE	Pseudo	0.01	0.01	25	None	50	0.02
BCE	Bernoulli	0.0001	0.01	25	None	50	0.05
BCE	Bernoulli	1e-06	0.01	25	None	50	0.05
BCE	Bernoulli	0.01	0.0001	25	None	50	0.31
BCE	Bernoulli	0.01	1e-06	25	None	50	0.31
BCE	Bernoulli	0.01	0.01	5	None	50	0.39
BCE	Bernoulli	0.01	0.01	125	None	50	0.22
BCE	Bernoulli	0.01	0.01	25	None	100	0.55
BCE	Bernoulli	0.01	0.01	25	None	250	0.54

Table 4.14: ComplEx on UMLS: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

embedding dimension, where all values tested (50, 100, and 250) led to similar performance of ComplEx on UMLS on the link prediction task. Most other hyperparameters had a middling effect. However, it is notable that most single-hyperparameter alternations led to an MRR value of around 0.30, whereas optimal values reached an MRR of 0.55. This indicates that change of even a single hyperparameter value can lead to a major loss of performance relative to the best hyperparameter combination, and suggests that ComplEx is particularly sensitive to hyperparameter choice on UMLS.

In order to further examine this effect, another view of this data is presented in Table 4.15. This view provides an overview of the overall distribution of MRRs achieved when each hyperparameter is fixed at a certain value. Therefore, it allows a direct description of how changing on hyperparameter values can result in shifted distributions of final MRR results. Note that all distributions are given in terms of extrema (minimum and maximum) and quartiles (25th percentile, median, and 75th percentile) because in most cases the data is clearly skewed and does not follow a Normal distribution.

Finally, it is worth noting that these experiments are not intended to show the maximal performance of ComplEx on UMLS, but rather to highlight differences in hyperparameter preference. The author makes no claim that these values are the best performance ComplEx can ever achieve on UMLS, but rather that these values are representative of the relative performance of different hyperparameter values for ComplEx on UMLS.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.02	0.04	0.05	0.06	0.27
Loss = BCE	0.02	0.04	0.05	0.06	0.55
Loss = CE	0.02	0.04	0.05	0.05	0.34
N. Samp = Basic	0.04	0.04	0.05	0.21	0.53
N. Samp = Bernoulli	0.04	0.05	0.05	0.21	0.55
N. Samp = Pseudo	0.02	0.03	0.04	0.05	0.06
LR = 0.01	0.02	0.03	0.21	0.26	0.55
LR = 0.0001	0.03	0.04	0.04	0.05	0.06
LR = 1e-06	0.04	0.04	0.05	0.05	0.06
Reg = 0.01	0.02	0.04	0.05	0.05	0.55
Reg = 0.0001	0.02	0.04	0.05	0.06	0.33
Reg = 1e-06	0.02	0.04	0.05	0.05	0.34
npp = 5	0.02	0.04	0.05	0.06	0.40
npp = 25	0.02	0.04	0.05	0.05	0.55
npp = 125	0.02	0.04	0.05	0.06	0.34
Mgn = None	0.02	0.04	0.05	0.05	0.55
Mgn = 0.5	0.02	0.04	0.05	0.05	0.26
Mgn = 1	0.02	0.04	0.05	0.06	0.27
Mgn = 2	0.02	0.04	0.05	0.06	0.27
Dim = 50	0.02	0.04	0.05	0.06	0.55
Dim = 100	0.02	0.04	0.05	0.05	0.55
Dim = 250	0.02	0.04	0.05	0.06	0.54

Table 4.15: Distribution of MRR scores obtained when running ComplEx on UMLS with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of ComplEx on UMLS in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Structure-Hyperparameter Interactions

The final aspect of ComplEx and UMLS that is examined here is the interaction between each KG structural feature and hyperparameter preference. To do this, a re-evaluation of ComplEx on UMLS is performed on link prediction queries in which each structural feature is lower than (or equal to) its median value, or above said median value. A search was then performed across all hyperparameter combinations for which one led to the best MRR for this specific structural subset of the KG. The structural features varied, optimal hyperparameters for each case, and MRRs achieved are shown in Table 4.16. The table rows alternate grey and white to distinguish each structural feature in the rows. “N/A” is inserted when the structural feature did not vary in the validation set, leading to a constant set of input values for which correlation is mathematically undefined.

It is important to note that *re-training was not done* in these cases – all of these re-evaluations were done by filtering the ranked list output of the highest-performing ComplEx

model based on the values of each structural feature. As such, it is intended that these values represent the differences in how a trained model performs on different subsets its KG, so as to indicate if certain graph structures are easier or harder to learn. Note also that the validation sets used in each case are (necessarily) different subsets of the main validation set, which means that direct comparison of their values must be done with care.

The results given show clear structure-based hyperparameter preference. In almost all cases, higher values of structural features (which all reflect higher graph connectivity in some manner) have a preference for higher embedding dimension values. This makes intuitive sense, as using more dimensions to embed data allows better representations of parts of the graph in which there is more data. The opposite also holds: more sparse regions of the graph, as detected by lower values of each structural feature, almost always prefer lower embedding dimensions. For example, below-median subject degrees are learned best on a dimension of 50, while above-median subject degrees are learned best on a dimension of 250.

The negative sampler chosen also sees some variation based on KG structure, an effect that was also observed in the results of Kotnis et al. (2017) [45]. The results presented here suggest that, when there is a difference in negative sampler preference between below-median and above-median structural subsets of the validation set, that above-median subsets tend to prefer a simpler (full random) negative sampler (“Basic”) as opposed to Bernoulli negative sampling. In this case, it is possible that (more advanced) Bernoulli sampling is more important when there is less data in the vicinity of a triple, and that when more information is available, a simpler negative sampler is able to perform better.

None of the other hyperparameter values see significant variation from the optimal hyperparameter values. This suggests that overall, the best hyperparameters for learning a KG also are best for learning most of its subsets.

Finally, in terms of performance, it is seen that in 11 (out of 21) cases shown, above-median values for structural features are associated with higher link prediction performance. In a further 2 cases, the reported MRR values are identical for both sets. This means, therefore, that in 8 cases performance actually improves on below-median values, which generally indicate greater sparsity of the KG. While previous work in the literature (see Section 2.4.1) suggest that lower connectivity should be more strongly aligned to poor learning, it is important to note here that only an above-median and below-median split is used. This means that the regions of a graph with the lowest connectivity are considered alongside those of more middling connectivity, which is likely the source of this observed disparity. It is finally notable that few of the structural changes involve a dramatic change in MRR – both above-median and below-median subsets of the validation set for each structural feature result in similar performance. This is actually somewhat expected; parts of the graph with low connectivity

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
s deg ≤ 84.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.49
s deg > 84.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.62
o deg ≤ 122.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.48
o deg > 122.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64
p freq ≤ 283.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.60
p freq > 283.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.50
s p cofreq ≤ 11.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.52
s p cofreq > 11.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.59
o p cofreq ≤ 11.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.56
o p cofreq > 11.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.54
s o cofreq ≤ 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.51
s o cofreq > 1.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.68
s min deg nbr ≤ 40.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.56
s min deg nbr > 40.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.56
s max deg nbr ≤ 304.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.52
s max deg nbr > 304.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.61
s mean deg nbr ≤ 132.5	BCE	Bernoulli	0.01	0.01	25	None	100	0.52
s mean deg nbr > 132.5	BCE	Bernoulli	0.01	0.01	25	None	50	0.60
s num s o cofreq ≤ 12.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.51
s num s o cofreq > 12.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.61
s min freq rel ≤ 35.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
s min freq rel > 35.0	BCE	Basic	0.01	0.01	25	None	250	0.56
s max freq rel	N/A							
s mean freq rel ≤ 255.6	BCE	Bernoulli	0.01	0.01	25	None	100	0.56
s mean freq rel > 255.6	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
s num rels ≤ 7.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.49
s num rels > 7.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.64
o min deg nbr ≤ 40.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.58
o min deg nbr > 40.0	BCE	Basic	0.01	0.01	25	None	250	0.54
o max deg nbr ≤ 304.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.50
o max deg nbr > 304.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.65
o mean deg nbr ≤ 130.8	BCE	Bernoulli	0.01	0.01	25	None	100	0.55
o mean deg nbr > 130.8	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
o num s o cofreq ≤ 15.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.48
o num s o cofreq > 15.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64
o min freq rel ≤ 42.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.59
o min freq rel > 42.0	BCE	Basic	0.01	0.01	25	None	250	0.53
o max freq rel	N/A							
o mean freq rel ≤ 272.4	BCE	Bernoulli	0.01	0.01	25	None	100	0.57
o mean freq rel > 272.4	BCE	Bernoulli	0.01	0.01	25	None	50	0.53
o num rels ≤ 8.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.49
o num rels > 8.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64

Table 4.16: ComplEx on UMLS: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

relative to one structural feature may well be highly connected with respect to others, as established in Mohamed et al. (2020) [58].

4.4.2 Bringing it all Together

Overall, the results on ComplEx and UMLS agree in broad terms with those reported for most other KG-KGEM pairs in Appendix A. As such, analysis here will aim to be broad rather than deep. That said, there are a few observations from other KG-KGEM combinations that are particularly of note.

Other KG-KGEM Pairs

For all KGs tested on ComplEx, structural correlations to rank tend to be somewhat weaker than those reported for ComplEx on UMLS. They do generally remain negative; however, CoDExSmall, Kinships, and OpenEA all have more positive correlation values than UMLS for ComplEx. Interestingly, on Kinships, structural correlations are near 0 in general.

Examining DistMult, correlations between each structural feature and rank output tend to be weaker – across all 5 KGs, near-0 correlations are common. However, in all cases, at least a couple of features (typically the fine-grained features) tend to have stronger correlations than the coarse-grained features that are calculated at a greater distance from the link prediction query in question. DistMult is also notable for having many more strong *positive* correlations, indicating that increasing various structural features’ values leads to increased rank and worse performance.

Turning at last to TransE, similar trends once again persist. Like DistMult and ComplEx, its correlation values tend to be lower on non-UMLS KGs. TransE tends to have almost all negative correlation values (with the exception of some values on OpenEA and UMLS). Finally, following the trend seen in DistMult, TransE also tends to have higher correlations between fine-grained structural features and rank than it has between coarse-grained features and rank.

Overall, these results suggest that which structure features correlate to rank depends both on the KG being used (as each KG has different structure) and on the KGEM used to learn it (as each KGEM learns in a different manner). Despite this, some general trends remain. These are, particularly,

- **Higher correlation from fine-grained features.** Subject / object degree, predicate frequency, and co-frequency features tend to correlate better to rank than any other features do.
- **Many low correlations.** Many correlations have low absolute values, suggesting

that the rank of a link prediction query is not determinable from any one structural feature alone. However, as TWIG has shown, using them in aggregate does allow such a prediction.

- **Generally negative correlations.** With some notable exceptions, most correlations are negative, which suggests that more dense regions of a KG in general are more readily learned by KGEMs.

Results on hyperparameter preference are essentially identical in principle to those found on ComplEx / UMLS for KGs learned by ComplEx: learning rate is by far the most impactful hyperparameter, and embedding dimension the least. Most other hyperparameter swaps lead to reduced performance at varying degrees (a change of 0.1 or 0.2 in MRR), but none are as universally impactful as learning rate.

Turning to DistMult, broadly similar effects are seen. Embedding dimension remains of little import, but the effect of learning rate is attenuated – while lower learning rates still do more poorly in general, DistMult is more tolerant of lower learning rates and can achieve middling MRRs with them (as compared to the MRR it obtains on its best hyperparameters on each KG). DistMult also seems to generally respond to hyperparameters less strongly than ComplEx, with many alternate combinations performing similar to the optimal hyperparameter combinations across the five KGs tested. TransE shows the same general trends as DistMult across all five KGs, except that the impact of the regularisation coefficient on it seems to be minimal – TransE performs similarly regardless of whether the regularisation coefficient is changed or not. Overall, the general trends observed from these experiments are:

- **High impact of the learning rate.** The learning rate is in almost all cases the single most impactful hyperparameter, and lower values of it are almost always linked to very poor link prediction performance.
- **Low Impact of embedding dimension.** The embedding dimension typically had the lowest impact on link prediction performance in all cases examined here – varying it from higher to lower resulted in minimal change in MRR.
- **Moderate impact of most other hyperparameters.** Most other hyperparameters have moderate impact on link prediction performance – changing them from their optimal values does result in decreased MRR, but not often as severely as is seen when learning rate is changed. That said, similar massive drops in performance can be seen at times when the negative sampler or loss function are changed.

- **Hyperparameter sensitivity dependent on KGEM, not KG.** The results presented here suggest that hyperparameter sensitivity is a function of the KGEM being used to learn, not the KG being learned. For example, DistMult tends to be less sensitive to different hyperparameter combinations than ComplEx does across the KGs tested. Note that hyperparameter *preference* is distinct from this, and can depend on both KG structure and on the KGEM scoring function.

It is important to highlight that the observed effect of hyperparameters is also dependent on the values that were chosen to test for each case. For example, had an embedding dimension of 1 been used, MRR results would have almost certainly been universally near-0 for all KGEMs and KGs tested. As such, all claims here must be understood to be valid in the domain of values tested for each hyperparameter only, not in a universal sense for all possible hyperparameter values.

Finally, turning to the results on the interaction between structural features and hyperparameter preference, it is seen that (as with ComplEx on UMLS), in most cases the optimal hyperparameters for each structural subset are similar or identical to those that are optimal for each KG as a whole. However, many exceptions to this exist, including cases on various KGs and KGEMs in which a majority of the preferred hyperparameter values change. Other trends found for ComplEx on UMLS (such as above-median values of structural features corresponding to preference for high embedding dimensions) do not necessarily hold across other KGs or other KGEMs. Overall, this variation makes drawing universal claims difficult. The overall results of this experiment are given below:

- **Tendency to prefer similar hyperparameters.** In many cases, the optimal hyperparameters for each subset are very similar to, or identical to, those that are optimal for the KG as a whole. However, exceptions to this rule exist on various KGs and KGEMs.
- **Variability by KG and KGEM.** The manner in which structure and hyperparameters interact varies both based on the KG being learned, and the KGEM used for learning. This effect is sufficiently strong and complex that few universal principles can be immediately extracted.
- **Variability by structural feature.** The results for above-median and below-median values for each structural feature vary across KG and KGEM, are not highly consistent.

A generalised summary of the dependencies found in this section is given in Figure 4.10. All edges represent a dependency of the object on the subject, and edges with like colours and styles are used to indicate dependency paths.

The meaning of each path, as well as its supporting evidence, is given below.

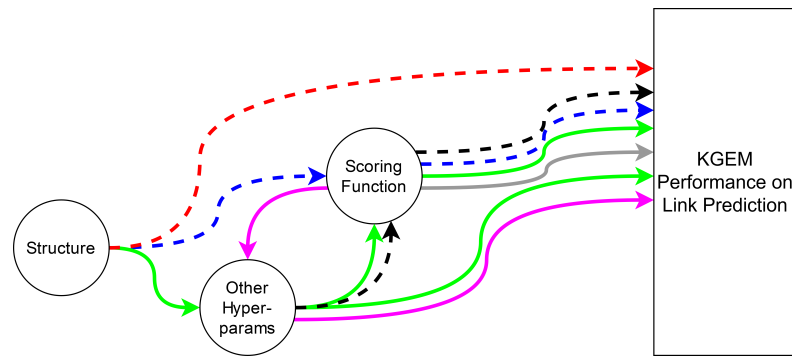


Figure 4.10: A graphical overview of the results of structural and hyperparameter analysis of KGEMs. An edge from a subject node to an object indicates that the subject influences the object. Edge colours and styles denote paths: for example, the solid green edge from “Structure” to “Other Hyperparams” to “Link Prediction Performance” indicates link prediction performance is dependent directly on hyperparameter choice, which creates an implicit dependency on structure which determines some level of hyperparameter preference.

- **Red, dashed:** Structure directly impacts the ability of KGEMs to learn parts of a graph, as illustrated in the state-of-the-art (see Section 2.4.1) and as evidenced in Table 4.13 (and in Appendix C).
- **Blue, dashed:** Varying KG structure impacts different KGEMs differently; see the difference in structural feature correlations for each KG-KGEM pair in Appendix C.
- **Green, solid:** Varying KG structure impacts hyperparameter preference. This change in hyperparameter preference has a direct impact on KGEM learning, and affects different KGEM scoring functions differently. This can be seen in the difference in hyperparameter preference of different graph sub-structures (such as in Table 4.16). Further data is given in Appendix C.
- **Purple, solid:** Changing the KGEM scoring function necessarily changes the optimal hyperparameters needed to learn, creating an scoring function-hyperparameters-performance dependency chain. This can be seen clearly in how different KGEMs have different hyperparameter preferences and different ultimate performance values. See Table 4.14 (and data for other KGs and KGEMs in Appendix C).
- **Grey, solid:** Changing the KGEM scoring function necessarily changes the range of MRR values that a given link predictor can achieve. This can be seen in the data tables for other KGs and KGEMs in Appendix C.

A Note on TWIG and Structural Alignment

As TWIG has shown a very strong ability to predict hyperparameter preference and link prediction performance from KG structure, the results here give insight into how this can

happen. It is of particular note that some structural features have moderate correlation on their own to link prediction performance; from TWIG’s results, they clearly have much stronger predictive power when used in aggregate. The trends in hyperparameter preference and structural influence on hyperparameters further explain what sort of data TWIG could be using to predict hyperparameter preference and link prediction performance.

Overall, between the results shown for TWIG in Section 4.3 and those presented in this section, there is substantial evidence that KGEM hyperparameter preference, and link prediction performance, can be modelled in terms of KG structural features. As such, there is evidence that Claim 1 of the Structural Alignment Hypothesis is true: hyperparameter preference in KGEMs, and their final performance on the link prediction task, can be expressed as a function of graph structure. The following chapter will examine Claim 2 of the Structural Alignment Hypothesis – that graph structural features in themselves contain enough information to perform link prediction on KGs.

5. TWIG-I

This chapter describes TWIG-I (Topologically-Weighted Intelligence Generation for Inference), an instantiation of the Structural Alignment Framework that performs the link prediction task, rather than simulating the output of link predictors (as TWIG does). As such, it is built to directly test the second claim of the Structural Alignment hypothesis: that link prediction itself can be performed as a function of graph structural features.

This chapter further describes an ablation study on all structural features to validate how each one in turn impacts TWIG-I’s performance and, therefore, which are most (or least) important to modelling the link prediction task. Specifically, the sections of this chapter are:

1. Section 5.1, which details the methodology used to build and evaluate TWIG-I.
2. Section 5.2, which contains a formulation of TWIG-I that uses the instantiation of the Structural Alignment Framework presented in Section 3.2.1.
3. Section 5.3, which provides a detailed study on which structural features contribute to TWIG-I’s performance, allowing for the refinement of the given instantiation of Structural Alignment to use a smaller, more effective set of features.
4. Section 5.4, which describes how TWIG-I can be applied in the transfer learning setting, and which characterises how effective TWIG-I is at transfer learning across different KGs for link prediction.

Identical to the previous chapter, all experiments were run across two different computers with different compute performance, which are listed below:

1. RTX 1070 with 8 GB vRAM and 16 GB RAM
2. RTX 3070-TI with 12 GB vRAM and 32 GB RAM

Note that some of the data and methods contained in this chapter have been published in a peer-reviewed venue by the author [79]. Finally, note that all code for TWIG-I can be found under an open-source licence at <https://github.com/Jeffrey-Sardina/TWIG-I>.

5.1 General Methodology of TWIG-I

To address how well a structural link predictor can perform link prediction, TWIG-I models link prediction in terms of a set of given structural features, such as those given in Table 3.1. Note, however, that there is no requirement that only these features be used; while these

methods are presented in terms of them, there is room to use a subset of them (as is done in Section 5.3) or to define entirely new feature sets (which is not done in this work).

The TWIG-I model, at its core, is a function that takes the form given in Equation 5.1:

$$Score = f(S_{triple}) \quad (5.1)$$

where f is the TWIG-I scoring function and S_{triple} is a set of structural features describing a triple. As is standard in link prediction, TWIG-I assigns a plausibility score to every triple, and then uses those to give ranked answers to link prediction queries.

Under this formulation, a huge number of functions could possibly stand in for f . In this work, f is expressed as a neural network with learnable weights. As the purpose of TWIG-I is to show that structure-based link prediction is *possible* (and specifically, as the purpose of TWIG-I is not to find the *optimal* way to do structure-based link prediction), f is modelled as a simple neural network of 3 dense layers. Dropout layers with probabilities of 0.01 each are inserted between the dense layers as a form of regularisation during training. A depiction of this model is given in Figure 5.1.

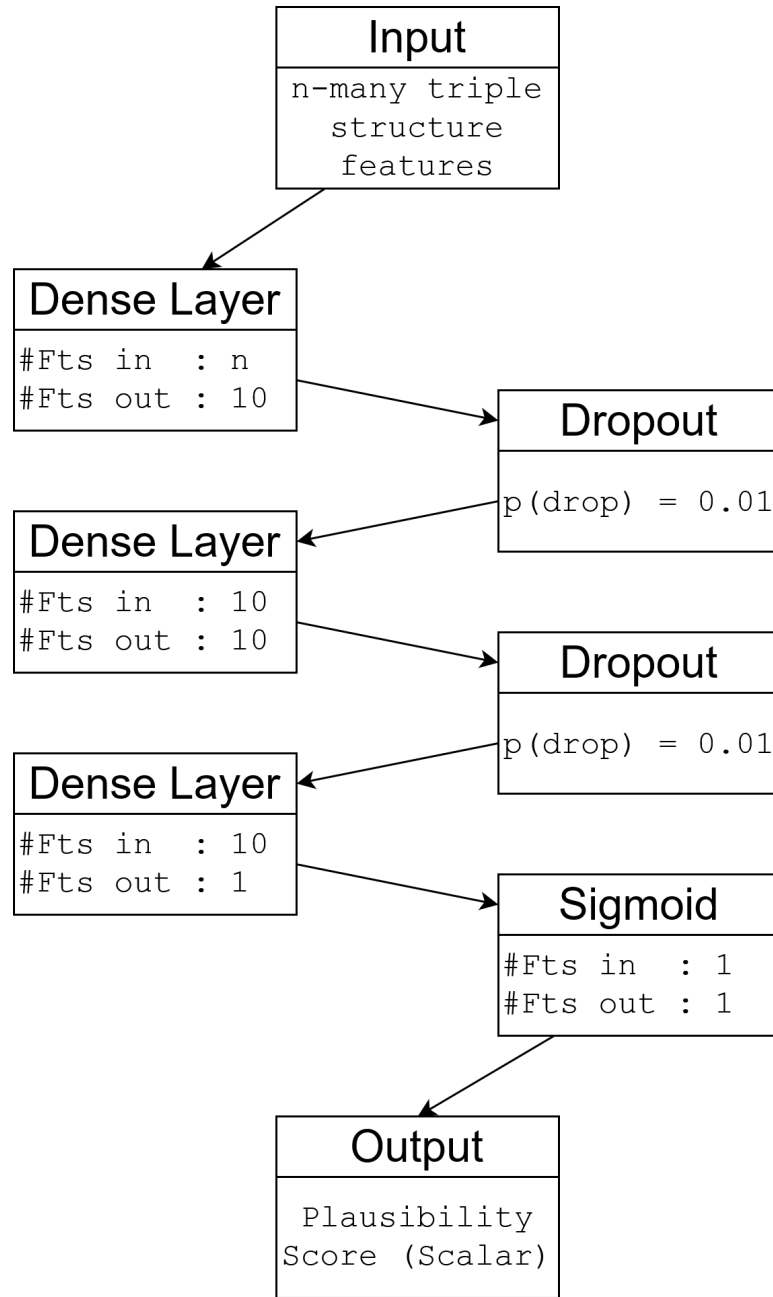
Specifically, the input of TWIG-I is z-score normalised structural feature values representing triples. Its output is, for each input triple, a single scalar plausibility score on the range $(0, 1)$, where higher scores indicate higher plausibility.

TWIG-I uses, in all cases, full random negative sampling, the Margin Ranking Loss loss function, and the Adam optimiser. Beyond these. TWIG-I has a few different hyperparameters that it requires to be defined. These are:

- **Batch Size:** The number of ground-truth triples to include in each batch during learning.
- **Learning Rate:** The learning rate to be used by TWIG-I’s optimiser.
- **Margin:** The margin used by the Margin Ranking Loss in TWIG-I. In practice, margin is always set to 0.1 in the experiments contained herein.
- **Negatives per Positive:** The number of negatives to sample for each positive triple.

TWIG-I is evaluated on four benchmark datasets: the standard FB15k-237 and WN18RR KGs, as well as the smaller benchmarks CoDExSmall and DBpedia50. The content of these datasets is given in Section 2.1.2. A brief characterisation of the size of the datasets is given in Table 5.1. Finally, note that all datasets used to benchmark TWIG-I are loaded using the PyKEEN library [3].

In all cases where a hyperparameter search is required, hyperparameter searching for TWIG-I is done using a grid search as defined in Section 2.3.3. When running hyperparameter

**Figure 5.1:** Depiction of the TWIG-I link prediction model.

Dataset	#Nodes	#Predicates	#Triples	Reference
FB15k-237	14505	237	310079	Toutanova et al. [93]
WN18RR	40559	11	92583	Toutanova et al. [93]
CoDExSmall	2034	42	36543	Safavi et al. [75]
DBpedia50	24624	351	34421	Shi et al. [88]

Table 5.1: An overview of the 4 benchmark datasets used to benchmark TWIG-I.

searches, candidate hyperparameter sets are trained for 20 epochs on the KG’s training set and then evaluated on the KG’s validation set to determine hyperparameter preference.

Finally, for baselines, the KGEMs ComplEx, DistMult, and TransE (described in Section 2.3.1) are used. When hyperparameter searches are run for them, they are run exactly as done when generating ground-truth data for TWIG: they are trained for 100 epochs on each

KG’s training set and then evaluated on the KG’s validation set to determine hyperparameter preference.

5.2 Evaluating Structural Alignment with TWIG-I

5.2.1 Experiments and Results

In this setting, the full set of 22 structural features given in the instantiation of the Structural Alignment Framework in Section 3.2.1 were used with TWIG-I to attempt to perform link prediction. This set of experiments aims to directly test Claim 2 of the Structural Alignment Hypothesis; that is, that link prediction itself can be performed as a function of graph structural features alone.

In order to do this, a hyperparameter search was performed for each KG that TWIG-I was trained on. The grid used to search for hyperparameters for FB15k-237 and WN18RR is given in Table 5.2; the grid used for CoDExSmall and DBpedia50 is given in Table 5.3. Note that a larger hyperparameter grid was used for the smaller KGs, since their smaller size allowed more hyperparameter combinations to be tried on limited computational power.

Hyperparameter	Values Searched
Negatives per Positive	30, 100, 500
Learning Rate	5e-3, 5e-4, 5e-5
Batch Size	128 (constant)
Margin	0.1 (constant)

Table 5.2: The hyperparameter grid used for grid searches on the larger benchmark KGs (FB15k-237 and WN18RR).

Hyperparameter	Values Searched
Negatives per Positive	30, 100, 500
Learning Rate	5e-3, 5e-4, 5e-5
Batch Size	64, 128, 256
Margin	0.1 (constant)

Table 5.3: The hyperparameter grid used for grid searches on the smaller benchmark KGs (CoDExSmall and DBpedia50).

The optimal hyperparameters found for TWIG-I on all KGs on the grids searched are reported in Table 5.4.

The KGEMs ComplEx, DistMult, and TransE were used as link prediction baselines. Results of each benchmark KGEM on FB15k-237 and WN18RR are taken from a previous benchmarking study by Ruffinelli et al. [73]. Results for KGEMs on CoDExSmall and DBpedia50 were obtained by running each KGEM on optimal hyperparameters for 1000 epochs, following the standard of previous publications on TWIG-I [79]. For reference, the

Dataset	npp	Learning Rate	Batch Size	Margin
FB15k-237	100	5e-4	128	0.1
WN18RR	500	5e-3	128	0.1
CoDExSmall	100	5e-3	64	0.1
DBpedia50	30	5e-3	128	0.1

Table 5.4: Hyperparameters selected for TWIG-I on each dataset for the standard training protocol. Note that, while the margin hyperparameter is shown for completeness, it was not varied during the hyperparameter searches. Note that “npp” refers to the number of negatives per positive.

hyperparameter grid used by KGEM baselines on CoDExSmall / DBpedia50 is given in Table 5.5, and the optimised hyperparameters for KGEMs is given in Table 5.6.

Hyperparameter	Values Searched
Loss Function	Margin Ranking, Binary Cross Entropy, Cross Entropy
Negative Sampler	Basic, Bernoulli, Pseudo-typed
Learning Rate	1e-2, 1e-4, 1e-6
Regulariser Coefficient	1e-2, 1e-4, 1e-6
Negatives per Positive	5, 25, 125
Margin	0.5, 1, 2
Embedding Dim	50, 100, 250

Table 5.5: The hyperparameter grid used for KGEMs on CoDExSmall and DBpedia50.

	Loss	Negative Sampler	lr	Reg Coeff	npp	Margin	Dim
ComplEx							
CoDExSmall	MRL	Bernoulli	1e-2	1e-2	125	2	100
DBpedia50	BCE	Basic	1e-2	1e-2	25	N/A	100
DistMult							
CoDExSmall	CE	Basic	1e-2	1e-2	125	N/A	250
DBpedia50	CE	Bernoulli	1e-2	1e-2	125	N/A	250
TransE							
CoDExSmall	MRL	Bernoulli	1e-2	1e-6	125	2	50
DBpedia50	CE	Bernoulli	1e-2	1e-2	125	N/A	250

Table 5.6: Hyperparameter values selected for each KGE model on each dataset. BCE = Binary Cross Entropy Loss; CE = Cross Entropy Loss; MRL = Margin Ranking Loss; npp = negatives per positive; lr = learning rate; reg coeff = regulariser coefficient; dim = embedding dimension

TWIG-I was trained on each KG on its optimal hyperparameters for 100 epochs using the standard link prediction evaluation protocol described in Section 2.2.2. The performance of TWIG-I, compared to all KGEM baselines on all four KGs used, is given in Table 5.7 in the following section.

5.2.2 Discussion of TWIG-I as a Link Predictor

The results shown in Table 5.7 lead to two clear conclusions:

- TWIG-I *can* perform link prediction using structural features only, and
- TWIG-I is *not optimal* at performing link prediction under its current combination of model formulation and structural feature use.

Dataset	FB15k-237	WN18RR	CoDExSmall	DBpedia50
TWIG-I	0.20	0.06	0.61	0.38
ComplEx	0.35	0.48	0.39	0.36
DistMult	0.34	0.45	0.34	0.39
TransE	0.31	0.23	0.28	0.31

Table 5.7: MRR performance of KGEMs vs TWIG-I in link prediction. The best results are shown in bold. Note again that results for ComplEx, DistMult, and TransE on FB15k-237 and WN18RR are as reported in Ruffinelli et al.’s benchmarking study [73].

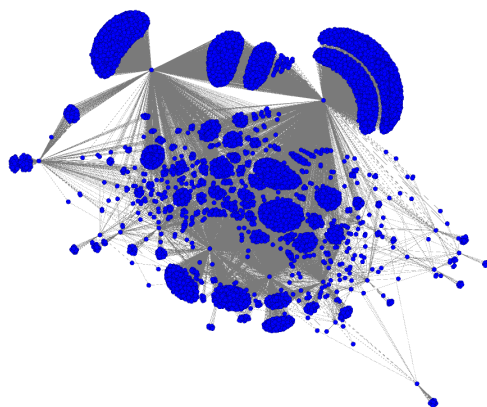
At this point, it is possible to validate Claim 2 as true: structural features are enough to perform link prediction on their own. However, the highly variable performance of TWIG-I, as well as its inability to learn effectively on WN18RR, leave much to be desired. TWIG-I only beats the state-of-the-art KGEMs in one case (CoDExSmall), matches it in one other case (DBpedia50) and lags behind it in two cases (on FB15k-237 and WN18RR, which are the standard link prediction benchmarks). The cause for such variable performance is similarly not clear from the results presented in Table 5.7 alone, except that TWIG-I seems to do better on the smaller KGs tested.

In order to explain these discrepancies, a structural analysis of all four KGs was performed in terms of the distribution of node degrees and relationship frequencies. This structural characterisation is shown in Table 5.8; in all cases, structural features are calculated from the KGs’ training sets only.

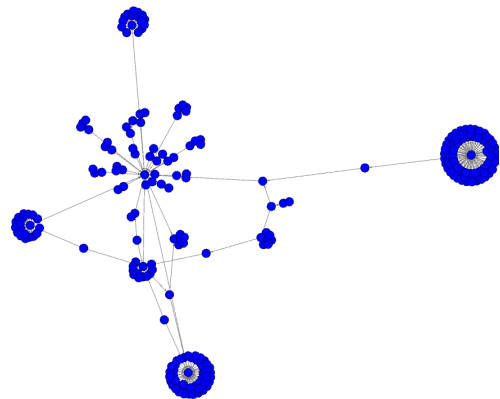
	FB15k-237	WN18RR	CoDExSmall	DBpedia50
Node Stats				
min degree	1	1	10	1
25% degree	11	2	15	1
50% degree	22	3	17	1
75% degree	41	5	25	2
max degree	7614	482	1008	781
Relation Stats				
min freq	37	80	1	1
25% freq	179	1030	28.25	3
50% freq	373	2921	143.5	10
75% freq	859	6109	370.5	46
max freq	15989	34796	10197	3006

Table 5.8: Structural characterisation of the training set of all four datasets on which TWIG-I was evaluated, in terms of node degree and relationship frequency. Percents in the table refer to percentiles; i.e. “25%” refers to the 25th percentile of degrees or relationship frequencies, respectively.

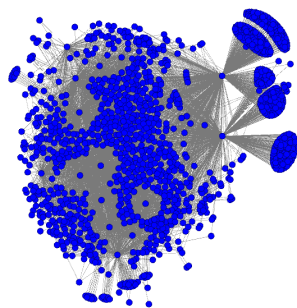
Further to this quantitative description, a qualitative view of the structure of each KG is provided in Figure 5.2. The figure shows sub-graphs of FB15k-237, WN18RR, CoDExSmall, and DBpedia50 centred on arbitrarily chosen nodes of median degree. For FB15k-237 and CoDExSmall, a 2-hop range around the chosen node is used. As WN18RR and DBpedia50 are substantially more sparse, a 6-hop region around the central node is used to allow a better visualisation of the overall graph structure. In all cases, the subgraph shown is extracted from the training set of each KG only. Visualisation was performed using Gephi [5].



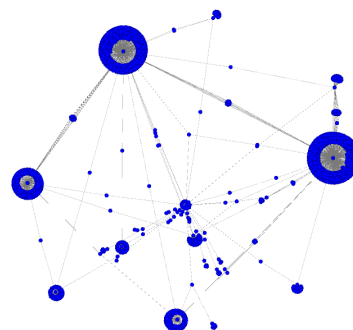
(a) Subgraph of FB15k-237.



(b) Subgraph of WN18RR.



(c) Subgraph of CoDExSmall.



(d) Subgraph of DBpedia50.

Figure 5.2: Visualisations of the structure of subgraphs of FB15k-237, WN18RR, CoDExSmall, and DBpedia50. For FB15k-237 and CoDExSmall, 2-hop subgraphs around an arbitrary node of median degree are shown. For WN18RR and DBpedia50, 6-hop subgraphs around an arbitrary node of median degree are shown as lower-hop subgraphs were too sparse. Visualisation was performed using Gephi [5].

From both Table 5.8 and Figure 5.2, it is clear that FB15k-237 and CoDExSmall are much more densely connected than either WN18RR or DBpedia50. Considering that structural features of lower-degree regions can easily end up being more uniform (as outlined in Section 3.2.2) it is plausible that this is part of the reason that TWIG-I does comparatively better

on FB15k-237 than WN18RR and on CoDExSmall than DBpedia50. This is corroborated by the observation that, as shown in Figure 5.2, both WN18RR and DBpedia50 have notably more uniform and more sparse structures than either FB15k-237 or CoDExSmall.

However, while this effect likely holds to some extent, it cannot explain why TWIG-I is able to learn to make predictions on DBpedia50 when it is unable to do so effectively on WN18RR. In fact, if connectivity were the only relevant variable, TWIG-I trained on WN18RR would have been expected to *outperform* TWIG-I on DBpedia50, as DBpedia50’s structure is much sparser across all degree percentiles examined in Table 5.8.

It is worth noting that WN18RR has notable higher relationship frequencies than DBpedia50, which sees many relationships used very infrequently. In line with this, CoDExSmall tends to have higher relationship frequency across all percentiles relative to FB15k-237. Yet while relationship frequencies can explain how TWIG-I learns FB15k-237 better than WN18RR, they cannot explain why TWIG-I performs comparatively better on DBpedia50 (which is less dense by both measures) than on WN18RR.

As such, despite the temptation to immediately ascribe TWIG-I’s variable performance to the node degree and relationship frequency, it appears that said structure features can only partially characterise TWIG-I’s performance. In order to better characterise what drives TWIG-I’s performance, a feature ablation study was performed to determine which structural features TWIG-I was actually using to make its prediction – and therefore, what sorts of structure (or lack thereof) would most directly impact its performance on link prediction. This ablation study is described in the following section.

5.3 TWIG-I Structural Feature Ablation Studies

In order to measure the impact of each structural feature on TWIG-I’s learning, an ablation study was performed on each dataset. In this ablation study, various structural features were removed from the TWIG-I model, and TWIG-I was then re-trained from scratch using only that reduced set of structural features. The features removed were as follows:

- **None:** in this setting, all 22 features were used for training to provide a baseline for each ablation setting.
- **All fine-grained features:** in this setting, all fine-grained features (i.e. *s deg*, *o deg*, *p freq*, *s-p co-freq*, *o-p co-freq*, and *s-o co-freq*) were removed at once, resulting in only 16 out of the original 22 structural features being used for training.
- **All coarse-grained features:** in this setting, all coarse-grained features were removed at once, leaving *only* the 6 fine-grained structural features to be used.

- **Individual fine-grained features:** in this setting, all 6 of the fine-grained features were removed one at a time, resulting in 6 different ablations. In all these cases, TWIG-I was trained only on the remaining 21 (out of the original 22) structural features.
- **Pairs of coarse-grained features:** in this setting, all 16 of the coarse-grained features were removed in pairs. For example both of the pair *s min deg neighbour* and *o min deg neighbour* were removed at the same time, since both represent the same structural metric (just calculated on different sides of the triple). This resulted in 8 different ablations. In all these cases, TWIG-I was trained only on the remaining 20 (out of the original 22) structural features.

Note that the manner in which fine-grained and coarse-grained structural features were ablated for TWIG-I is identical to the approach used in TWIG’s feature ablation study in Section 4.3.4.

All ablations were trained for 20 epochs on the training set of the respective KGs on the optimal hyperparameters found for them in Section 5.2. They were then evaluated in terms of MRR on the testing split of each KG in turn. The results of all of these experiments are shown in Table 5.9. Best results are shown with an *, and results that beat the TWIG-I model trained with all features are shown in bold.

Feature Removed	FB15k-237	WN18RR	CoDExSmall	DBpedia50
none	0.20	0.03	0.56	0.43
Aggregate Fts				
all fine-grained	0.01	0.00	0.33	0.02
all coarse-grained	0.06	0.09	0.10	0.16
Fine-grained Fts				
s deg	0.19	0.03	0.56	0.44
o deg	0.14	0.00	0.41	0.29
p freq	0.20	0.04	0.48	0.37
s-p cofreq	0.18	0.04	0.43	0.41
o-p cofreq	0.16	0.02	0.42	0.35
s-o cofreq	0.78*	0.48*	0.98*	0.26
Coarse-grained Fts				
s/o min deg neighbour	0.21	0.08	0.61	0.44
s/o max deg neighbour	0.21	0.04	0.38	0.42
s/o mean deg neighbour	0.22	0.04	0.35	0.46*
s/o num neighbours	0.02	0.01	0.14	0.20
s/o min freq rel	0.21	0.12	0.54	0.36
s/o max freq rel	0.20	0.06	0.55	0.39
s/o mean freq rel	0.19	0.06	0.45	0.43
s/o num rels	0.20	0.05	0.57	0.37

Table 5.9: Performance of TWIG-I on each benchmark KG in the ablation studies, given as MRR values. The best MRR values for each given KG are marked with an *; results that improve over the 22-feature baseline are shown in bold without *.

Interestingly, removing the feature *s-o cofreq* leads to (massive) improvements in MRR across all datasets tested except DBpedia50. In order to examine the reason for this effect, Table 5.10 provides data on the distribution of subject-object co-frequency values in the training and testing sets of each KG tested. From this table, one thing becomes immediately clear – almost all subject-object pairings occur only once in the training set, and only once in the testing set. This is immediately a concern from a machine learning perspective, as it means *s-o cofreq* essentially acts as a flag for whether a triple has been seen in the training set. For new triples that TWIG-I is meant to predict in the testing set, which connect subjects and objects not paired before, TWIG-I will be much more likely to reject these as plausible negatives because that is what it was trained to do. In other words, including the *s-o cofreq* feature allows overfitting of TWIG-I on the KGs’ training sets, and resultantly poor performance on the KGs’ testing sets.

s-o Co-Freq	FB15k-237	WN18RR	CoDExSmall	DBpedia50
Training Set				
min	1	1	1	1
25%	1	1	1	1
50%	1	1	1	1
75%	1	1	1	1
max	7	2	3	6
Testing Set				
min	1	1	1	1
25%	1	1	1	1
50%	1	1	1	1
75%	1	1	1	1
max	2	1	1	2

Table 5.10: Distribution of subject-object co-frequency values in the training and testing set of each dataset TWIG-I was trained on.

However, the *reduced* performance of DBpedia50 when *s-o cofreq* is removed cannot be explained as an overfitting effect. The ultimate source of this cannot be explained without a more detailed inspection of the content of the training and testing sets for DBpedia50, which is out of the scope of this work. Instead, this work aims to focus on the ability of a structure-based link predictor such as TWIG-I to perform link prediction task – which it still can do on DBpedia50 even when the feature *s-o cofreq* is removed, albeit with lesser efficacy.

Aside from *s-o cofreq*, very few of the other ablations result in consistent improvement in TWIG-I’s performance across the various KGs tested. The only other feature that is notable in improving performance when it is removed is *s/o min deg neighbour*, which leads to (typically small) improvements over baseline performance when it is removed. When removed, TWIG-I’s MRR on FB15k-237 and DBpedia50 only increases by 1 base point and TWIG-I’s performance on WN18RR and CoDExSmall increases by 5 base-points in MRR.

It is further notable that, as shown in Table 5.8, there are many more nodes at low degrees than there are at high degrees in all four KGs tested. As a result, it is likely that the feature *s/o min deg neighbour* has a tendency to contain the same (or similar) values, making it largely redundant during learning. This, in turn, explains why removing it could result in increased performance – as a largely redundant feature, it is possible that it would contribute more noise than signal to the learning process. Once again, however, this effect is generally weak, and overshadowed by the massive increase to performance obtained by removing the *s-o cofreq* feature.

The fact that removing *s-p cofreq* and *o-p cofreq* leads to a decrease in performance for TWIG-I is quite notable. The *s-p cofreq* and *o-p cofreq* features, by counting how often a predicate maps to various nodes as subjects or objects, necessarily also maps what nodes are (or are not) permitted in the subject or object position. In other words, they directly model the domain and range of each relationship. If either is 0, it suggests that the given subject / object is likely out of the domain / range of the predicate. The fact that TWIG-I’s performance drops when they are removed suggests that this level of (ontological) knowledge is relevant to how TWIG-I learns to predict new links.

Aside from *s-o cofreq* and *s/o min deg neighbour*, almost all other features generally lead to reduced performance of TWIG-I on most (or all) of the KGs tested when they are removed. This suggests that most of the features selected to be part of the Structural Alignment Framework are indeed useful for characterising the structure of a KG, and that they are sufficiently representative that they can be used to perform link prediction. It is of particular note that either removing all fine-grained features or all coarse-grained features results in dramatic reduction in performance of TWIG-I – suggesting that information about a triple *and* its broader structural context is critical to enable structure-based link prediction.

In light of these findings, a modified version of the Structural Alignment Framework is presented for TWIG-I in which the feature *s-o cofreq* is removed, resulting in only 21 total features. This modified setting for TWIG-I is evaluated in detail in the following section. The following section further details how TWIG-I can take advantage of its structure-based learning to directly enable cross-KG transfer-learning for link prediction.

5.4 Evaluating Transfer Learning with TWIG-I

TWIG-I’s formulation is directly and natively compatible with transfer learning. Since TWIG-I uses the same structural features to describe triples regardless of the KG that said triples come from, data from any KG can be input into it in exactly the same way. Moreover, TWIG-I does not produce learned embeddings – all of its parameters are shared. As such,

the same TWIG-I model can be later finetuned on a new KG – all this requires is giving it different triples as input to the exact same pretrained neural network to finetune it onto a new KG.

This is in contrast to KGEM models, where the learned parameters (the embeddings themselves) are KG-specific. For example, a ComplEx KGEM trained on FB15k-237 has no way to use those learned embeddings to make predictions about WN18RR, which contains an entirely disjoint set of node and edge types. As such, the property of enabling direct transfer learning is unique to TWIG-I – KGEMs do not allow transfer learning of knowledge from one KG to another.

In order to evaluate how well TWIG-I can perform transfer learning, TWIG-I is evaluated in the transfer learning setting on FB15k-237, WN18RR, CoDExSmall, and DBpedia50. It is trained for 10 epochs of pretraining on each (source) dataset, and then finetuned for a further 10 epochs on each other (target) dataset such that all KG pairs are evaluated. TWIG-I is also trained from scratch for 10 and 20 epochs. The TWIG-I version trained on 20 epochs from scratch represents a baseline that uses the same total number of epochs as a finetuned TWIG-I (which trained 10 epochs pre-training and 10 epochs finetuning). The TWIG-I version trained on 10 epochs from scratch represents a baseline that uses the same number of epochs on the target dataset, but begins learning with no prior knowledge from transfer learning.

In all cases, the *s-o co-freq* feature is removed, as it was found to lead to significant overfitting and reduced performance in Section 5.3. All hyperparameters found for finetuning from each source KG, to each target KG were determined by a unique hyperparameter search for each KG pair. All hyperparameter searches were run for a total of 5 epochs using the same hyperparameter grid as described in Table 5.3 (when finding hyperparameters for finetuning to CoDExSmall or DBpedia50) and Table 5.2 (when finding hyperparameters for finetuning to FB15k-237 or WN18RR). The optimal hyperparameters found from all hyperparameter searches are reported in Table 5.11.

While these hyperparameter results are not the purpose of this section, there are two main trends that are worth calling out explicitly:

1. **Dependency on the KG pair.** Hyperparameters vary based both on the KG that TWIG-I was trained on, and on the KG it is targeted to for transfer learning.
2. **Distinction from training from scratch.** The optimal hyperparameters found for training TWIG-I from scratch (shown in Table 5.4) are different than those that are optimal to transfer learn to the same target KG. In other words, optimal hyperparameters change when the initialisation state (pre-trained or from scratch) of TWIG-I changes.

Source KG	Target KG	npp	lr	b.s.	Margin
CoDExSmall	DBpedia50	30	5e-3	64	0.1
	FB15k-237	30	5e-5	128	0.1
	WN18RR	30	5e-4	128	0.1
DBpedia50	CoDExSmall	100	5e-4	128	0.1
	FB15k-237	500	5e-4	128	0.1
	WN18RR	100	5e-5	128	0.1
FB15k-237	CoDExSmall	30	5e-4	256	0.1
	DBpedia50	500	5e-4	64	0.1
	WN18RR	30	5e-4	128	0.1
WN18RR	CoDExSmall	500	5e-5	128	0.1
	DBpedia50	500	5e-4	128	0.1
	FB15k-237	500	5e-3	128	0.1

Table 5.11: All hyperparameter values selected for finetuning TWIG-I from a pretrained TWIG-I model on the source KG to a target KG. Note that, while the margin hyperparameter is shown for completeness, it was not varied during the hyperparameter searches. npp = negatives per positive; lr = learning rate; b.s. = batch size

The results of all transfer learning experiments are given in Table 5.12. Results for baseline experiments on ComplEx, DistMult, and TransE are also included for ease of comparison. Note again that KGEM performance on FB15k-237 and WN18RR is taken from a previous benchmarking study by Ruffinelli et al. [73].

	FB15k-237	WN18RR	CoDExSmall	DBpedia50
From FB15k-237	NA	0.73	0.99	0.31
From WN18RR	0.71	NA	0.02	0.28
From CoDExSmall	0.46	0.60	NA	0.38
From DBpedia50	0.61	0.46	0.27	NA
From Scratch, 10e	0.79	0.34	0.93	0.30
From Scratch, 20e	0.82	0.45	0.94	0.30
ComplEx Baseline	0.35	0.48	0.39	0.36
DistMult Baseline	0.34	0.45	0.34	0.39
TransE Baseline	0.31	0.23	0.28	0.31

Table 5.12: Transfer learning results for TWIG-I trained without the *s-o co-freq* feature, given as MRR values. All TWIG-I experiments were trained for a total of 20 epochs: 10 for pretraining and 10 for finetuning. TWIG-I trained from scratch for a total of 10 or 20 epochs, and KGEM baselines, are also given. Note that KGEM performance on FB15k-237 and WN18RR is taken from a previous benchmarking study by Ruffinelli et al. [73]. Best results overall are shown in bold.

Overall, transfer learning shows selective improvement over TWIG-I baselines (training for 20 epochs from scratch or training for 10 epochs from scratch), although this effect is not universal. It is first worth calling out that, using FB15k-237 as a source dataset (on which TWIG-I was pre-trained), and finetuning to any of the other 3 KGs tested leads to the universally best results of any TWIG-I model under any training condition. Secondly, CoDExSmall also has a similar effect – using CoDExSmall as a source dataset and either WN18RR or

DBpedia50 as a target dataset leads to increased performance over both TWIG-I baselines. These two observations are of particular note in light of the previous point (illustrated in Table 5.8 and Figure 5.2 in Section 5.2.2) that FB15k-237 and CoDExSmall have very similar structures. It is similarly noteworthy that FB15k-237 and CoDExSmall have substantially more heterogeneous structures than either DBpedia50 or WN18RR. As previously noted theoretically in Section 3.2.2 and empirically in Section 5.2, this greater connectivity and diversity of structure leads to substantively increased performance for TWIG-I.

Taken together, this evidence suggests that more heterogeneous source datasets result in generally better transfer learning results, regardless of the structure or the target dataset. The only potential exception to this hypothesis observed in this data here is that using CoDExSmall as a source dataset and FB15k-237 as a target dataset with TWIG-I did not lead to increased performance relative to TWIG-I baseline values. However, Table 5.8 shows that FB15k-237 is a substantially larger dataset with a wider range of possible connectivity patterns – particularly, the range of possible node degrees and predicate frequencies for FB15k-237 is much wider than that of CoDExSmall. As such, it is possible that the same principle actually still applies: that using a more heterogeneous source dataset leads to finetuning improvements on less heterogeneous target datasets.

It is similarly notable that in other cases where DBpedia50 and WN18RR were used as source datasets for pre-training, final fine-tuning results were poorer. As both of these graphs are very sparse and generally homogeneous in their structural patterns, it is again possible that this lack of variation gave TWIG-I a poor basis for transfer learning – resulting in generally worse performance in the transfer learning task as compared to using CoDExSmall and FB15k-237 as source KGs.

In all other cases, use of transfer learning was less effective than training even for just 10 epochs from scratch.

At a more general level, the results here suggest that there is an element of background knowledge about KGs that can be captured through mathematical descriptions of their structures. This is a surprising find, since this transfer learning was done even across KGs from different domains. For example, while FB15k-237 and CoDExSmall are both general knowledge datasets, they were still very strong source datasets for finetuning to WN18RR, which is specifically a linguistics KG. This ability to transfer learn even across knowledge domains reinforces the idea presented in this work of Structural Alignment – that the structure of a KG is necessarily connected to how well it and the information it contains can be learned.

TWIG-I in general shows very strong performance compared to KGEM baselines once the *s-o co-freq* feature is removed. In particular TWIG-I models trained from scratch (with no finetuning) outperform the KGEM state-of-the-art on FB15k-237 and on CoDExSmall.

This pattern is once again worth highlighting – that TWIG-I does relatively better on the most heterogeneous datasets tested compared to KGEM baselines.

Similarly, TWIG-I models trained from scratch on WN18RR and DBpedia50 underperform KGEM baselines. Despite this, they do match or out-perform TransE in both cases, and generally underperform the ComplEx and DistMult by a slight margin. On WN18RR, TWIG-I trained from scratch (for 20 epochs) achieves an MRR of 0.45; the strongest KGEM baseline (ComplEx) is only 0.03 higher at 0.48. On DBpedia50, TWIG-I achieves an MRR of 0.30, compared to 0.39 achieved by the strongest KGEM baseline (DistMult). The cause of this is likely the same – that in the absence of highly diverse structure, TWIG-I’s learning is attenuated. However, especially on WN18RR, TWIG-I remains competitive with the state-of-the-art.

Overall, the evaluation of TWIG-I described in this chapter, indicates that the structural characteristics chosen in this work are sufficient both to perform link prediction and to enable effective cross-KG transfer learning even between KGs from different knowledge domains. TWIG-I achieves results that are comparable to, or more performant than, the state-of-the-art in almost all cases when trained without finetuning. With fine-tuning from KGs with diverse structure, TWIG-I’s performance increases further above KGEM and TWIG-I baselines. TWIG-I’s ability to effectively apply transfer learning suggests that transfer learning for link prediction is a very viable direction for future work. Further exploration of this effect, and of its utility and generality to the field of KGs and link prediction, is left to future research.

It is finally notable that, while this chapter aims to present TWIG-I in terms of the Structural Alignment Hypothesis, one similar work in embedding-free link prediction has very recently been published – the Intersection Features model by Le et al. (2024) [48]. As TWIG-I is principally presented in this thesis to provide evidence for Structural Alignment, a comparison of TWIG-I to the Intersection Features model is beyond the scope of the main body of this work. However, as the Intersection Features model is of such direct relevance (and novelty) to the state-of-the-art in link prediction, a theoretical and empirical comparison is provided in Appendix D.

This chapter concludes the major experimental findings of this thesis. The following chapter, Chapter 6, explores the implications of this work, both in terms of how the scientific community understands knowledge graphs and how it understands the various algorithms (such as KGEMs and TWIG-I) built to perform inference on them.

6. Discussion and Conclusions

This thesis proposes the Structural Alignment Hypothesis – the idea that learning on knowledge graphs can be modelled as a function of knowledge graph structure. It then presents this hypothesis in terms of two concrete claims:

- **Claim 1:** that hyperparameter preference in knowledge graph embedding models, and their final performance on the link prediction task, is a function of graph structure, and
- **Claim 2:** that link prediction itself can be performed as a function of graph structural features.

Claim 1 is addressed through the creation of TWIG, a neural network built to predict KGEM performance as a function of KG structure and the hyperparameters used for learning. Results on TWIG, given in Chapter 4, indicate that this prediction can be made with very high efficacy. Moreover, it is demonstrated that TWIG can predict hyperparameter performance on even unseen KGs based on their structural patterns.

Claim 2 is addressed through the creation of TWIG-I, a link prediction model that uses a fixed set of structural features to perform link prediction directly (without using embeddings). Results from experiments on TWIG-I, given in Chapter 5, indicate that it can match or exceed state-of-the-art performance on standard link prediction datasets. Further, TWIG-I can be used in the transfer learning setting, and in some cases this leads to further increases in link prediction performance.

Both TWIG and TWIG-I are constructed with a specific instantiation of the Structural Alignment Hypothesis consisting of a set of 22 structural features based on the frequencies (and co-frequencies) of nodes and relations in a knowledge graph. All of these features are selected based on existing evidence in the literature for their relevance to the link prediction task, and this work as such provides further evidence that they are highly relevant to how the field characterises KGs, KGEMs, and link prediction.

Further, both TWIG and TWIG-I achieve particularly strong results; TWIG-I in particular sets a new value for state-of-the-art link prediction performance. That said, the purpose of the TWIG and TWIG-I studies was not to show optimality, but rather to show that structure-based simulation of KGEMs, and structure-based link prediction, are possible.

As such, and particularly given the success in evaluation of TWIG and TWIG-I, this thesis provides evidence to support both Claim 1 and Claim 2 of the Structural Alignment Framework outlined in Section 3.1 of Chapter 3, and therefore concludes that the Structural Alignment Hypothesis holds under the conditions outlined herein.

The remaining sections of this chapter discuss Structural Alignment, directions for future work, and further research performed by the author during the duration of his PhD. Specifically, these sections are:

- Section 6.1, which discusses the implications of Structural Alignment on how knowledge graphs, link prediction, hyperparameters, and graph learning generally are understood and modelled.
- Section 6.2, which discusses all research outputs by the author, including industry projects, patents, work as a reviewer, and other published papers not directly connected to the Structural Alignment Hypothesis.
- Section 6.3, in which the author gives ending remarks to this thesis.

6.1 Implications of Structural Alignment

The Structural Alignment Hypothesis has several implications for how KGEMs and link prediction are generally understood. These implications, and various interpretations of Structural Alignment, are described in the following sections. The author believes that the research outlined in this thesis, and its implications (listed in the following sections), will have a significant impact on the state-of-the-art and on the general development of the field.

6.1.1 Structure and Semantics

This thesis intentionally avoided discussion of a “structure vs semantics” distinction in its motivation, methods, and results due to the observation that structure as a concept can stand on its own without appeal to the concept of semantics. However, such a discussion cannot be avoided when addressing the implications of this work, especially because TWIG and TWIG-I show such strong performance while drawing only on a basis of structural features.

The results presented in this thesis provide evidence that the concept of semantics is unnecessary to either 1) simulate or 2) perform link prediction on knowledge graphs. After all, if semantics were necessary to perform either of these tasks, then semantics-free learners such as TWIG and TWIG-I should not have been able to perform them.

Yet while the claim “semantics is unnecessary” can (and does) follow to an extent, the author wishes to offer a separate perspective: semantics is unnecessary *but is probably useful*. Under this perspective, Structural Alignment is interpreted as a new baseline that represents, in essence, the best that can be done without semantics. It is almost certain that using structural knowledge and semantic knowledge would result in increased performance relative

to using structure alone. This directly implies that TWIG and TWIG-I act as structure-only, semantics-free baselines for the development of future (semantics-aware) link prediction methods.

The author proposes that Structural Alignment implies a hierarchy of concepts that can be used for learning knowledge graphs. This is presented as a pyramid of link prediction methodology, which is shown in Figure 6.1. In this view, the data (the KG itself) is the base of all learning. Just above data, structure is the level which is most commonly and most readily learned (i.e. by a wide variety of KGEMs, TWIG, and TWIG-I, among other methods). The layer above this is semantics and ontology – the aspects of a KG that give meaning to its structure. Finally, above that is a fourth layer, included to acknowledge that other higher-order concepts may come to the fore as the field advances.

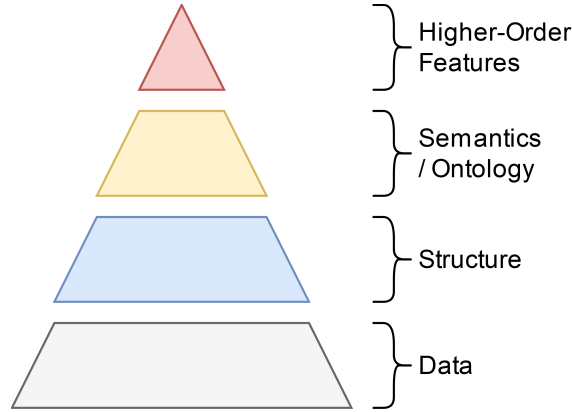


Figure 6.1: The Link Prediction Pyramid. Under Structural Alignment, the simplest (and most commonly learned) aspect of a KG for link prediction is structure. The next layer up above structure is semantics (and / or ontology) of a KG. Finally, space is left to indicate that other higher-order concepts may also be relevant to link prediction.

The Link Prediction Pyramid draws direct inspiration from the DIKW (Data-Information-Knowledge-Wisdom) Pyramid [72], which models the transition from Data to Information to Knowledge and finally to Wisdom as a hierarchical process in which higher-order information is continuously abstracted and used.

Akin to the DIKW Pyramid, the key idea of the Link Prediction Pyramid is that including information from each next layer should result in substantially increased link prediction performance; i.e. each layer is a baseline for the layer above it. A side-by-side representation of the Link Prediction and DIKW Pyramids is given in Figure 6.2.

For example, beating TWIG-I’s performance on link prediction represents beating a structure-only baseline. A model that beats TWIG-I and is semantically aware (i.e. is not a larger / better trained structural model) should be expected to perform (much) better than TWIG-I and reside above it in the pyramid. Similarly, creating a link predictor that TWIG cannot simulate using structural features implies that said link predictor is able to draw upon

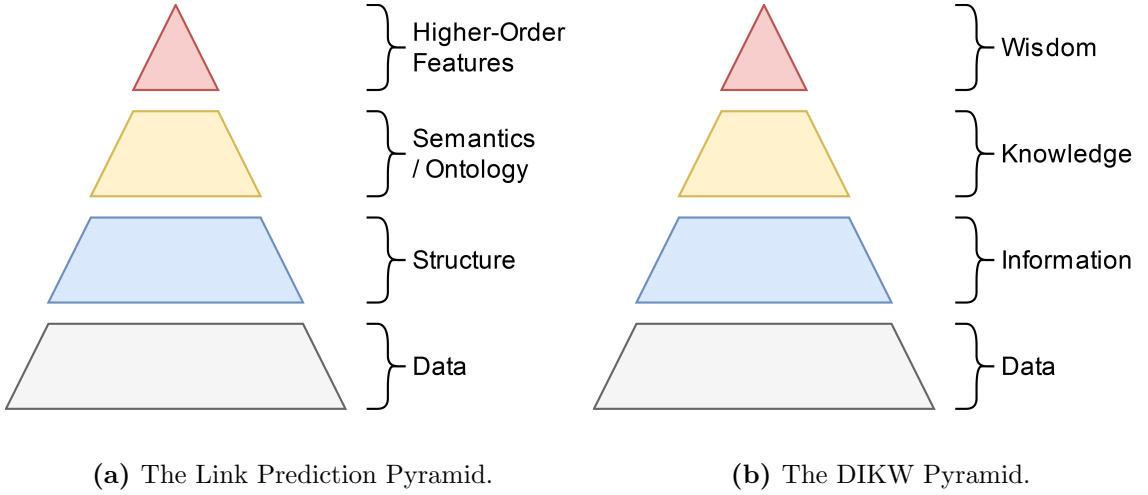


Figure 6.2: A side-by-side presentation of the Link Prediction Pyramid (left) and the DIKW Pyramid (right). Corresponding layers at higher levels represent abstraction of the layer below and the addition of further, higher-order concepts.

information which is non-structural in nature. This means that TWIG can actually serve as a test for semantic knowledge, and whether a new model can be placed in the semantic or the structural layer of the pyramid.

The author highlights that structure and semantics should not be considered to be contrary to each other. As highlighted in the Link Prediction Pyramid in Figure 6.1, semantics is a conceptual layer above structure – but in KGs, it is necessarily built on top of structure as well. The author considers it likely that semantics-aware methods built with a robust understanding of KG structure (along the lines of TWIG-I) would likely be a significant contribution to the state-of-the-art.

In fact, some recent works in neuro-symbolic link prediction have begun to explore semantics-aware link prediction. Neuro-symbolic models have been increasingly applied in recent works to the link prediction task, most commonly by extending KGEMs to integrate logical rules into the learning process [4, 29, 35, 53, 66, 69, 86, 87, 106, 107]. While further controlled comparisons and large-scale re-evaluations of these (in the same form as various existing mass KGEM benchmarking studies [2, 36, 38, 73]) are still needed to more clearly position these results in the state-of-the-art, and while these methods do not explicitly account for elements of KG structure, neuro-symbolic models generally show promise for advancing the state-of-the-art in link prediction techniques.

The author believes that further work both in structural representation and in semantic learning of KGs would be fruitful for the characterisation of, and advancement of, link prediction.

6.1.2 Hyperparameters and Model Creation

The finding that TWIG can predict the performance of various hyperparameter combinations, even on unseen KGs in the zero-shot and few-shot settings, leads to several insights into the nature of KGEM hyperparameters. The ability of TWIG to predict hyperparameter performance so reliably suggests that hyperparameter preference is a function of KG structure and the KGEM chosen. In other words, it implies that hyperparameters should be determinable in a pre-hoc manner once a KG and KGEM are selected.

However, while this work takes a first step towards such a determination, it cannot provide a full characterisation thereof. That notwithstanding, it is clear that a fine-tuned TWIG model can act as a replacement for a traditional hyperparameter search. The fact that TWIG can achieve quite accurate predictions of hyperparameter performance after seeing only 5% (or 25%) of a hyperparameter grid suggests that a short hyperparameter search on a small random subset of all possible hyperparameters, followed by inference using TWIG, is a possible alternative to established hyperparameter search methods. This must be taken with the understanding, however, that TWIG was evaluated in the few-shot setting for knowledge graphs that are small even compared to the standard benchmarks FB15k-237 and WN18RR. While there is no evidence to believe that TWIG would fail in such circumstances, it is possible that TWIG may require additional training or development in order to allow it to generalise across datasets of widely different sizes.

This leads into the broader question of link prediction model creation. If structural learners are desired, then taking a structure-informed approach to model design in the manner of TWIG for determining model components and model choices is a viable direction. This further implies that deeper analysis of existing scoring functions in terms of their ability to represent common structures would be a valuable direction for future work. However, as outlined in Figure 6.1, structure is better understood as a baseline for link prediction – not the optimal manner of solving it. When semantic-based learners are desired, a different perspective for model construction should be taken – to map model choices to aspects of KG semantics that may or may not also be directly represented in graph structure.

6.1.3 Ontologies and Learning

While TWIG and TWIG-I are presented as structural learners, it is important to highlight that some elements of KG structure correspond directly to KG ontology. For example, as noted in Chapter 5, the *s-p cofreq* and *o-p cofreq* structural features allow TWIG and TWIG-I to directly represent information on the domain and range of predicates. If either value is 0, this means that a certain node has never been observed as a subject / object of the predicate

in the training set, and suggests that it is (likely) out of its domain or range.

However, none of the KGEMs discussed here explicitly model for any level of KG ontology. While there is increasing work in this direction (see Kulmanov et al. (2019) [46], for example), most of the literature on link prediction remains non-ontological. The fact that TWIG can predict the output of KGEMs without use of any ontological features other than domain and range similarly suggests that such features of a KG are not heavily relied upon by these KGEMs in practice. This further means that these non-ontological KGEMs likely have to play catch-up to (attempt to) learn ontological knowledge, as well as to learn to represent the information content of a KG, before they can effectively perform link prediction.

Similarly, Chapter 5 suggests that TWIG-I’s ability to natively model domain and range is a contributing factor to its success in link prediction. This leads to the interesting irony that structure-based (and arguably semantic-less) learners such as TWIG-I may be more directly able to model aspects of ontological knowledge than standard KGEMs.

At a more general level, it is important to highlight a second aspect of structure which is not considered under Structural Alignment: logical structure. The annotation of graph structure in terms of rules (typically Horn clauses [28, 57]) or the logical properties of relations [2, 28, 34, 57, 107] represents a fundamentally different type of structure than that considered by Structural Alignment. The core of this difference is twofold:

1. Structural Alignment examines frequency-based, not logic-based, structure, and
2. Structural Alignment considers the structure around nodes, not just relationships.

For example, take the rule “if A is allies with B , and B is an enemy of C , then A is an enemy of C ”, which could possibly be determined for the example knowledge graph in Figure 1.1. This rule can be written mathematically as given in Equation 6.1:

$$(A, Ally-Of, B) \wedge (B, Enemy-Of, C) \implies (A, Enemy-Of, C) \quad (6.1)$$

where A , B , and C are variables for which any node in the graph can be substituted subject to the constraints of the rule. The direct result of this is that, in such a rule, structure is specified at the level of relationships only. This can be seen clearly in Figure 6.3, in which the rule is shown as a logical structure motif.

Contrasting this to the frequency-based structure of a triple, such as that given in Table 3.2 in Section 3.2.2, it can very readily be seen that these annotations of structure differ substantially from each other – both in what they annotate (only edges versus both nodes and edges) and how they annotate it (repeated motifs of logical structure versus frequency-based descriptions of graph connectivity).

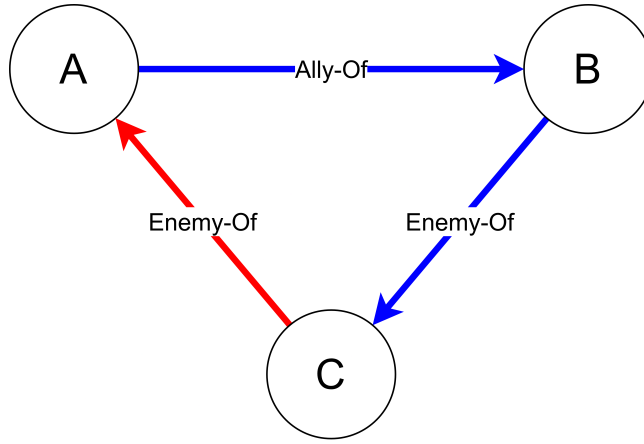


Figure 6.3: A graphical expression of the logical rule given in Equation 6.1. Edges in blue represent the antecedent, and the edge in red represents the consequent.

Whether Structural Alignment could be extended to elements of logical structure, as well as the implication of such an extension on understanding and modelling rule-based link predictors, is left as a direction for future research.

Further analysis of the overlap between ontology and structure, as well as the impact of both on learning, are likely to be highly fruitful future directions. Finally, as has been said many times in the literature, exploration of more ontology-based methods for learning KGs is certainly a fruitful future direction, regardless of whether that learning is based on KG structure, KG semantics, or a combination of the two.

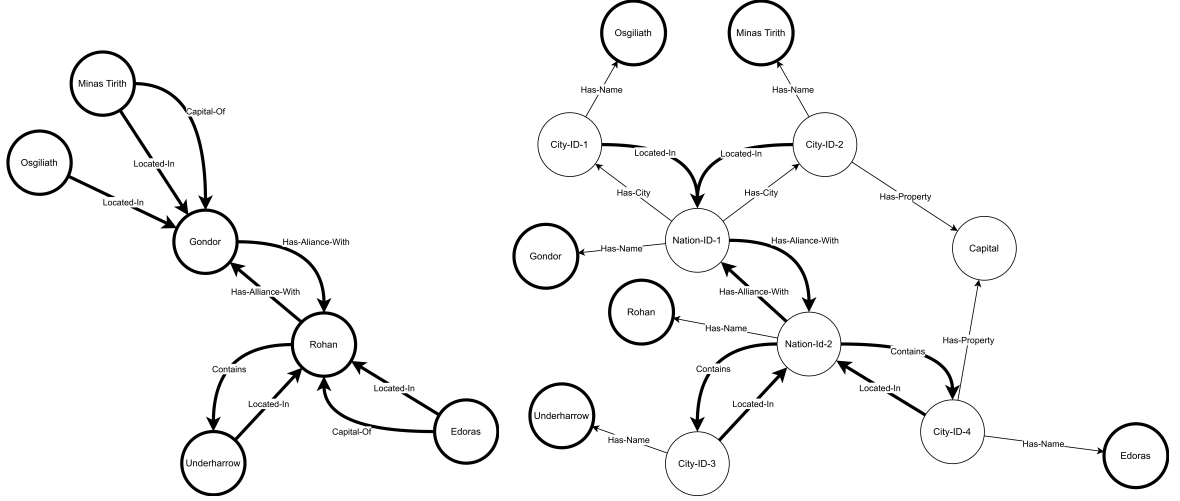
6.1.4 Implications for Knowledge Graph Creation

Structural Alignment has several significant implications for knowledge graph creation. First, it must be highlighted (as outlined in Section 3.2.1) that alterations to the single-hop structure around a triple are sufficient to change individual predictions in the link prediction task [7, 8, 64, 104]. This is the reason that the Structural Alignment Framework draws upon structural features from not just a triple, but from the 1-hop range around it as well. However, especially in very sparse knowledge graphs where higher-hop dependencies may be present, it is expected that using features from a larger range around a triple may be of increased importance for TWIG and TWIG-I. This effect can already be seen, for example, in TWIG-I’s substantially increased performance on more structurally diverse graphs relative to sparse graphs (see Section 5.4).

This dependency can be phrased not just as one of graph structure, but of graph design choices by knowledge graph creators as well. As highlighted in previous work by the author, many different graph structures can encode identical information content – but these do not result in identical performance of link predictors trained on them [78]. Considering this in the context of Structural Alignment, it can be said that dependency on KG structure is

necessarily a dependency on how knowledge graphs (and their creators) model information.

Take, for example, the two knowledge graphs shown in Figure 6.4. On the left-hand side, in Figure 6.4a, a more dense graph is shown. On the right, in Figure 6.4b, a more sparse graph, with identical information content, is shown. The difference between these graphs is one of the data modeller – whether to directly represent nodes by their plain-text names (which leads to a more compact, dense graph) or to use proxy identifier nodes and have plain-text names as properties of them (which leads to a more sparse graph).



(a) An example KG with generally more dense connectivity and compact representation of nodes and relations of interest. (b) An example KG with generally more sparse connectivity and substantially less compact representation of nodes and relations of interest.

Figure 6.4: An example of two knowledge graphs, based on the example in Figure 1.1. Nodes and edges present in both graphs are shown with bold lines; those only present in the right-hand graph are shown with thinner lines. In the left-hand graph, relations of interest (for example, *Located-In*) are directly able to be captured by single-hop link prediction in the form $(s, p, ?)$ or $(?, p, o)$. In right-hand graph, intermediary nodes result in sparsification and make single-hop link prediction less able to predict meaningful information from the graph.

As the choice of how to model the data in a graph impacts graph structure, Structural Alignment posits that it will directly impact graph learning. Specifically, Structural Alignment provides a tool for directly characterising the learnability of a graph from its structure. The core structural properties of graphs that lead to increased learnability, as implied by Structural Alignment, are as follows:

- **Density.** The state-of-the-art literature on KGEMs (see Section 2.4.1), and results in this thesis on TWIG (see Chapter 4) and TWIG-I (see Chapter 5), indicate very clearly that more densely connected elements of a graph (as reflected by higher node degree and higher relationship frequency, for example) result in increased learning capacity for the link prediction task. Similarly, KGEMs and TWIG-I are both shown to have lowered link prediction performance in the context of lower density. As such, increas-

ing graph density should be considered a desideratum for creating knowledge graphs whose intended application is as training data for the link prediction task under such structurally-influenced link predictors.

- **Compactness.** Similar to the above point, any knowledge graph intended for use in the (single-hop) link prediction task should ensure that all links that are desired to be predicted are directly modelled in the KG as single-hop relationships. For example, answering the query $(MinasTirith, Located-In, ?)$ is directly possible when the *Located-In* relationship is modelled in a compact manner (as in Figure 6.4a). However, if the nodes *MinasTirith* and *Gondor* are not connected compactly (as in Figure 6.4b), such a link prediction query becomes much harder, or in some cases impossible, to pose. Such structures can often result from reification, when higher order elements (such as named graphs) are present in the raw data but must be removed in order to create triples for standard (triples-based) link predictors [78]. Avoiding such reification should be considered a core desideratum for creating KGs that are directly and effectively applicable for use in link prediction pipelines.
- **Heterogeneity.** As noted theoretically in Section 3.2.2, and empirically in the various experiments in Chapter 5, TWIG-I requires heterogeneous structure in order to learn to predict links. After all, without structural heterogeneity, all frequency-based structural features would appear the same – therefore making it impossible for TWIG-I to distinguish distinct triples when scoring them. This effect does not seem to be restrictive, however, to KGEMs: in fact, results in Chapter 5 indicate that KGEMs can achieve comparatively higher performance on more homogenous graphs than they do on more heterogeneous ones. As such, this final desideratum is best seen as applicable specifically in the case that link predictors created based on a Structural Alignment Framework, such as TWIG-I, are used.

In a similar manner, when graphs are created with lower density, less compactness, and more homogeneity (as seen in Figure 6.4b), Structural Alignment suggests that link prediction performance on such graphs would generally decrease. It is similarly expected that further reduction of density, compactness, and heterogeneity would result in further decreases to link prediction performance. Simply put: Structural Alignment implies that the more a KG resembles Figure 6.4a, the more effectively it should be able to be learned; the more a KG resembles Figure 6.4b, the less effectively it should be able to be learned.

The author proposes that when the primary purpose of a KG is for use in link prediction, KG construction should be considered alongside link predictor construction in a single “learning fabric”, as opposed to as a distinct concept or step. Akin to what the author observed

for KGEM creation in previous work [77], it is quite possible that treating KG creation and link prediction as separate tasks is a sub-optimal configuration for link prediction application. Future research in this direction, and its broader impact on knowledge graph and link predictor design principles, is left as a future direction.

6.1.5 Other Future Directions

Aside from the implications of Structural Alignment listed above, several other future directions follow directly from this work. These are:

- **Analysis of other sets of structural features.** This thesis analyses only a single Structural Alignment Framework; however, many are possible based on different selections of structural features. Analysis of other such instantaneous of Structural Alignment, and how they contribute to the field of KGs and link prediction, is left as a future direction.
- **Wider and deeper analyses of TWIG and TWIG-I.** TWIG and TWIG-I were both analysed on a limited set of KGs and KGEMs. Analysing whether the patterns found in this work extend to others, and why, is left for future work. The author is particularly interested in the ability of TWIG to model non-KGEM learners, such as AMIE+ [28] or EL-embeddings [46], as both systems are explicitly logic-based and therefore may or may not be able to be understood in terms of frequency-based KG structural features.
- **Further analysis of TWIG and TWIG-I for transfer learning.** The author believes that research on the abilities and limitations of TWIG and TWIG-I to transfer-learn to new KGs would be of use. Particularly in the case of TWIG-I, which can directly perform link prediction, a broad ability to perform transfer learning could reduce the amount of time and energy needed for the link prediction pipeline while increasing predictive performance.
- **Evaluation of TWIG and TWIG-I using alternate learning architectures.** While TWIG and TWIG-I were trained as neural networks in this work, many other formulations are possible. Firstly, considering that TWIG and TWIG-I use carefully hand-crafted structural features as input, exploring natively explainable architectures (such as decision trees and support vector machines) would be very well merited. Further, exploring whether TWIG and TWIG-I could achieve better performance on more advanced neural architectures (such as attention layers and transformer blocks) would be similarly merited.

- **Exploration of the representational capacity of TWIG and TWIG-I.** TWIG (and especially TWIG-I) have fairly small neural architectures, especially compared to the number of learnable parameters typically used in KGEMs. As either model is scaled to simulate of KGEMs on larger KGs (in the case of TWIG) or to perform link prediction on larger KGs (in the case of TWIG-I), it is uncertain whether their neural architectures will also need to be widened and / or deepened in order to increase their representational capacity. Further research in this direction would be merited to better understand the capability and limitations of both models.
- **Pretraining TWIG-I on multiple KGs.** While TWIG is evaluated when pre-trained on multiple KGs and fine-tuned on one other KG, TWIG-I is only ever pretrained on one KG. Exploring the effect of pretraining TWIG-I on multiple different KGs is merited, especially in light of the observation that TWIG-I performs best in the fine-tuning setting when trained on a more diverse graph in the pre-training phase.
- **Structural Alignment models for other graph tasks.** The current instantiation of Structural Alignment was created specifically in the context of link prediction. However, many other graph tasks exist – such as node classification / regression and graph classification / regression. For such settings, it is likely that the structural features used to define a Structural Alignment Framework would have to change in order to better represent the properties of the item (such as the individual node, or the whole graph) being predicted. Exploring the impact of Structural Alignment on other such graph tasks is left as a future direction, and one in which the author sees significant potential.

Finally, one further future direction is worth calling out in particular detail. Graph Foundation Models (GFMs), whose goal is to enable large-scale pretrain-finetune pipelines for graph data, have recently gained significant interest [52, 101]. The Structural Alignment Framework, in directly and natively enabling cross-graph and cross-domain transfer learning on knowledge graphs, is situated in a similar conceptual space to these emerging approaches. It is the belief of the author that the exploration of structure-based graph foundation models, akin to the perspective taken by the Structural Alignment Hypothesis, would be a significant contribution to the field of graph learning in general.

It is possible that other directions may be inspired by the perspectives taken in the literature review, methods, results, and discussion of this thesis. The author makes no claim that this list of directions is complete, but rather presents it as a list of directions that he considers most likely to be fruitful.

6.2 Research Publications, Outputs and Awards

This section describes all of the research work that the author has completed, as well as relevant code / software systems that he has contributed to. An overview of all of the author’s peer-reviewed publications, as well as which aspect of this thesis they relate to, is given in Table 6.1. Note that the “Other” column is used for publications which do not have direct bearing on the concepts expressed in this work.

Publication	Graph Struct.	Hyper-param Pref.	Structure-based LP	Cross-KG Transfer Learning	Other
SeWeBMeDA (2021) [83]	X				
AICS (2022) [84]					X
ICSC (2024) [77]					X
ICSC (2024) [82]	X	X			
EMBC (2024) [78]					X
SEMANTiCS (2024) [79]			X	X	
AICS (2024) [80]	X	X		X	
ICSC (2025) [81]	X	X			
JoWS (2025) <i>Submitted</i>	X		X	X	

Table 6.1: An overview of all of the author’s peer-reviewed submissions and publications, and which topics in this thesis they address.

Peer-reviewed papers published by the author during the course of his PhD are further described in the following pages.

- **SeWeBMeDA at ESWC (2021) [83]**. This paper presents an analysis of the correlation between various graph structural features (which differ in part from those considered in this thesis) and ultimate KGEM performance. Its major result was to show that correlation can be found between graph connectivity and overall performance, but it was unable to make claims about hyperparameter preference.
- **AICS (2022) [84]**. This paper builds upon previously-published work on box embeddings for query answering on knowledge graphs [67] and performs an analysis of the role of attention mechanisms in box embeddings. The paper shows that using attention to approximate box embedding intersections actually outperformed using an exact geometric equation, and suggests that this indicates that the attention mechanism was learning more than vector geometry in how it creates box intersections.
- **IEEE ICSC (2024) [82]**. This paper presents the initial formulation of TWIG, which is largely similar to that reported in this thesis in Chapter 4.
- **IEEE ICSC (2024) [77]**. This paper gives an overview of the state-of-the-art in KGEMs and link prediction and lists the three major areas in which link prediction is

under-developed: expert knowledge integration, tolerance of heterogeneous structure, and modelling the relative importance of triples. The key contribution of this paper is a conceptual model constructed to aid the design of new link predictors that could address each of these issues as a joint problem with multiple effects, rather than as a set of disjoint problems.

- **EMBC (2024) [78]**. This paper outlines a new knowledge graph embedding framework, called NamE, that can extend all common (and most other) KGEMs to allow them to explicitly incorporate contextual knowledge in the form of named graph embeddings. The paper shows that this approach can lead to substantially improved performance over raw-KGEM baselines.
- **SEMANTiCS (2024) [79]**. This paper presents the initial formulation of TWIG-I, which is the same as that reported in this thesis in Chapter 5. The paper was shortlisted for the Best Student Paper at the conference.
- **AICS (2024) [80]**. This paper outlines how TWIG can be used in the 0-shot and few-shot settings, as described in this thesis in Section 4.3.3.
- **IEEE ICSC (2025) [81]**. This paper provides a broad survey of the structural and hyperparameter effects documented in state-of-the-art KGEM literature, as done in Section 2.4.1 and Section 2.4.2 of this thesis. The paper was shortlisted for the Best Paper at the conference.

In addition to these, at the time of submission, the author has submitted one article for journal review:

- **JoWS (Journal of Web Semantics) (2025) *Submitted***. This paper describes the full ablation study of TWIG-I, as well as how TWIG-I can be used in the transfer-learning setting, as outlined in Section 5.3 and Section 5.4 in this thesis. It has been submitted and is under review at the time of the submission of this thesis.

6.2.1 Code Contributions

There are two core code contributions of this thesis: TWIG and TWIG-I. The TWIG library contains a fully-featured, documented, and extensible implementation of TWIG and related tools. It can be found on:

- GitHub (<https://github.com/Jeffrey-Sardina/TWIG-TWM-dev>)
- PyPi (<https://pypi.org/project/twig-twm/>)

The TWIG-I library contains a fully-featured, documented, and extensible implementation of TWIG-I and related tools. It can be found on:

- GitHub (<https://github.com/Jeffrey-Sardina/TWIG-I>)
- PyPi (<https://pypi.org/project/twigi/>).

Finally, all data from all hyperparameter experiments can be found on FigShare at <https://figshare.com/s/7b2da136e05f3548399f>. This data is provided in a form directly usable by TWIG.

6.2.2 Listing of Research Outputs

Oral Presentation and Internal Abstract at the *Third IEEE UK&I YP Postgrad STEM Research Symposium* on the author’s Master’s work on knowledge graph embeddings under the title “Structural Characteristics of Knowledge Graphs Determine the Quality of Knowledge Graph Embeddings Across Model and Hyperparameter Choices”. The symposium information can be found here: https://ieeeukiyp.org/3rd_stem/schedule/

Poster Presentation and Internal Paper at the *2022 ADAPT Annual Scientific Conference* under the title “Knowledge Graph Embeddings and Graph Structure”, on a paper accepted to the conference under the same name.

Oral Presentation and Published Paper at the *5th Workshop on Semantic Web solutions for large-scale biomedical data analytics, part of the ESWC Conference*, under the title “Structural Characteristics of Knowledge Graphs Determine the Quality of Knowledge Graph Embeddings Across Model and Hyperparameter Choices”. The conference information can be found here: <https://sites.google.com/view/sewebmeda-2021/home>.

Award, Oral Presentation, and Published Abstract in Irish at the *2022 Irish Computational Biology and Genomics Symposium*, under the title “I dTreo an tIdirghníomhú idir Structúr Graif Eolais, Leabuithe Graif Eolais, agus Feidhmíocht a Shamhlú”. The research won the award for Best Overall Presentation in Irish.

Oral Presentation and Published Paper co-authored with Callie Sardina at the *30th Irish Conference on Artificial Intelligence and Cognitive Science*, under the title “Analysis of Attention Mechanisms in Box-Embedding Systems”.

Award, Oral Presentation, and Internal Paper co-authored with Matt Murtagh at the *2023 ADAPT Annual Scientific Conference* under the title “Graph Style Transfer: Stable Diffusion-based Creation and Enrichment of Knowledge Graphs”, on a paper accepted to the conference of the same name. Our work won the Best Overall Contribution Award for the conference.

Oral Presentation and Published Paper at the *18th IEEE International Conference on Semantic Computing* under the title “TWIG: Towards pre-hoc Hyperparameter Optimisation and Cross-Graph Generalisation via Simulated KGE models”. The conference information can be found here: <https://www.ieee-icsc.org/>.

Oral Presentation and Published Paper at the *18th IEEE International Conference on Semantic Computing* under the title “Veni, Vidi, Vici: Solving the Myriad of Challenges before Knowledge Graph Learning”. The conference information can be found here: <https://www.ieee-icsc.org/>.

Poster Presentation and Published Paper at the *46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* under the title “Name: Capturing Biological Context in KGEs via Contextual Named Graph Embeddings”. The conference information can be found here: <https://embs.org/2024/>.

Poster Presentation and Internal Paper at the *2024 ADAPT Annual Scientific Conference* under the title “TWIG-I: Structure-Based Knowledge Transfer and Cross-Graph Generalisation for Link Prediction”, on a paper accepted to the conference of the same name.

Poster Presentation and Internal Paper in Irish at the *2024 ADAPT Annual Scientific Conference* under the title “TWIG-I: Aistriú Eolais Bunaithe ar Struchtúr agus Ginearálú Tras-Ghraf i gComhair Réamhinsintí Nasc”, on a paper accepted to the conference of the same name. Note that the author was the driving factor behind opening the Conference to poster presentations in Irish, and that at least one other poster was presented in Irish there.

Oral Presentation and Internal Abstract Publication in Irish at the *Glór na dTaighdeoirí* conference at UCD, under the title “TWIG: Córas Foghlama do Ghraif Eolais Nua Bunaithe ar Intleacht Ionsamhlaithe”.

Award, Oral Presentation, and Published Paper to the *SEMANTiCS-2024 Conference* under the title “TWIG-I: Embedding-Free Link Prediction and Cross-KG Transfer Learning using a Small Neural Architecture”. The paper was shortlisted for the Best Student Paper at the conference. Conference information can be found here: <https://2024-eu.semantics.cc/>.

Oral Presentation and Published Paper at the *32nd Irish Conference on Artificial Intelligence and Cognitive Science*, under the title “Extending TWIG: Zero-Shot Predictive Hyperparameter Selection for KGEs based on Graph Structure”.

Oral Presentation and Published Paper at the *19th IEEE International Conference on Semantic Computing* under the title “A Survey on Knowledge Graph Structure and Knowledge Graph Embeddings”. The paper was shortlisted for the Best Paper at the conference. The conference information can be found here: <https://www.ieee-icsc.org/>.

6.2.3 Listing of Patents

As part of the author’s researcher programme under (through the D-REAL programme of Taighde Éirinn | Research Ireland), the author completed an industry placement with Accenture over a period of 6 months, from January 2023 to June 2023. As a part of this placement, the author contributed to several patents. These are listed below.

- Primary author on a patent on KGE Systems at Accenture regarding memory reduction and low-degree node / relation representation in knowledge graph embedding models. This patent was published under the title “Systems and methods for enhancing an artificial intelligence model via a multi-field embedding approach” [76].
- Primary author on a patent at Accenture regarding named graph embeddings for knowledge graph embedding models. This patent is connected to the aforementioned paper entitled “NamE: Capturing Biological Context in KGEs via Contextual Named Graph Embeddings” [78].
- Contributing author on a third patent on KGE Systems at Accenture regarding knowledge graphs and healthcare. This patent was published under the title “Electronic health records data summarization for graph machine learning” [31].

6.2.4 Work as a Reviewer

Finally, the author worked as a peer-reviewer for two years of his PhD. This work is listed below:

- **Workshop Reviewer** for the workshop CRUM@UMAP’24 (The Second Workshop on Context Representation in User Modeling at The 32nd ACM Conference On User Modeling, Adaptation And Personalization) as a member of the program committee.
- **Workshop Reviewer** for the workshop CRUM@UMAP’23 (The First Workshop on Context Representation in User Modeling at The 31st ACM Conference On User Modeling, Adaptation And Personalization) as a member of the program committee.

6.3 Final Remarks

The author welcomes any and all correspondence regarding this work. The author’s ORCID ID is 0000-0003-0654-2938, and he can be contacted at via email at jeffrey.sardina@gmail.com.

This thesis was written and published bilingually, with the main text in English and an extended summary in Irish. The Irish-language summary can be found under the title “Ailíniú Struchtúir agus Réamhinsint Nasc”. Ar scáth a chéile a mhairimid.

Appendix

Appendix Contents

A Further TWIG Ablations	151
A.1 TWIG on ComplEx	152
A.2 TWIG on DistMult	154
A.3 TWIG on TransE	156
B Structural Feature Distributions	158
B.1 CoDExSmall	159
B.2 DBpedia50	160
B.3 Kinships	161
B.4 OpenEA	162
B.5 UMLS	163
C Further KG-KGEM Data	164
C.1 ComplEx on CoDExSmall	165
C.2 ComplEx on DBpedia50	168
C.3 ComplEx on Kinships	171
C.4 ComplEx on OpenEA	174
C.5 ComplEx on UMLS	177
C.6 DistMult on CoDExSmall	180
C.7 DistMult on DBpedia50	183
C.8 DistMult on Kinships	186
C.9 DistMult on OpenEA	189
C.10 DistMult on UMLS	192
C.11 TransE on CoDExSmall	195
C.12 TransE on DBpedia50	198
C.13 TransE on Kinships	201
C.14 TransE on OpenEA	204
C.15 TransE on UMLS	207
D TWIG-I vs. the Intersection Features Model	210
D.1 Summary of the Intersection Features Model	210
D.2 Comparison with TWIG-I	211
D.3 Comparison of Empirical Performance	211
References	213

A. Further TWIG Ablations

This section provides the results of all TWIG feature ablations, as described in Section 4.3.4 in Chapter 4. The results on DistMult [102] and on TransE [11] are given here, and results on ComplEx [47, 94] from the main text are reproduced as well for completeness. Analysis of this data (including the general trends of results on DistMult and TransE) is given in Section 4.3.4, and as such is omitted from this Appendix.

A.1 TWIG on ComplEx

Two feature ablations were run to analyse how TWIG learns to simulate ComplEx. The first case, shown in Table A.1, shows the result of all feature ablations when ComplEx was trained on each KG in the single-KG setting as described in Section 4.3.4. The second case, shown in Table A.2, shows the results of feature ablations when TWIG was trained to simulate ComplEx in the cross-KG setting, as described in Section 4.3.4.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.76	0.71	0.92	0.40	0.94
Hyperparameters					
loss	0.67	0.24	0.85	0.39	0.96
neg. sampler	0.20	0.45	0.90	0.42	0.59
lr	0.02	-0.19	-0.01	0.02	0.16
reg. coeff.	0.73	0.47	0.97	0.29	0.87
npp	0.79	0.61	0.94	0.19	0.95
margin	0.75	0.59	0.96	0.57	0.96
dimension	0.83	0.49	0.95	0.60	0.92
Aggregate Fts					
all fine-grained	0.79	0.73	0.96	0.68	0.83
all coarse-grained	0.79	0.66	0.96	0.52	0.96
Structure (fine)					
s deg	0.78	0.5	0.88	0.65	0.94
o deg	0.80	-0.18	0.95	0.44	0.97
p freq	0.82	0.28	0.98	0.65	0.98
s p cofreq	0.81	0.57	0.95	0.62	0.97
o p cofreq	0.75	0.32	0.99	0.58	0.98
s o cofreq	0.77	0.44	0.98	0.18	0.95
Structure (coarse)					
s/o min deg nbr	0.78	0.43	0.98	0.71	0.94
s/o max deg nbr	0.76	0.42	0.98	0.56	0.93
s/o mean deg nbr	0.81	0.62	0.98	0.55	0.95
s/o num nbrs	0.76	0.46	0.96	0.53	0.97
s/o min freq rel	0.84	0.25	0.98	0.61	0.97
s/o max freq rel	0.72	0.38	0.98	0.59	0.96
s/o mean freq rel	0.83	0.49	0.98	0.72	0.97
s/o num rels	0.82	0.59	0.97	0.52	0.94

Table A.1: The results of all feature ablation studies (for all KGs tested) when TWIG was trained to simulate ComplEx. All experiments are run in isolation on a single KG-KGEM pair, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2 scores. Results that outperform TWIG when trained on all features are shown in bold.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.77	0.53	0.94	0.51	0.93
Hyperparameters					
loss	0.56	0.42	0.93	0.16	0.96
neg. samp.	0.29	0.44	0.90	0.33	0.51
lr	0.04	-0.1	0.02	-0.19	0.20
reg. coeff.	0.62	-0.35	0.92	-0.48	0.72
npp	0.79	0.68	0.94	0.59	0.89
margin	0.72	0.49	0.97	0.54	0.92
dimension	0.74	0.58	0.92	0.56	0.89
Aggregate Fts					
all fine-grained	0.74	0.92	0.86	0.63	0.59
all coarse-grained	0.57	0.45	0.74	0.93	0.92
Structure (fine)					
s deg	0.82	0.63	0.92	0.63	0.95
o deg	0.79	0.41	0.96	0.41	0.82
p freq	0.79	0.66	0.88	0.58	0.94
s p cofreq	0.77	0.45	0.97	0.37	0.96
o p cofreq	0.76	0.57	0.98	0.43	0.92
s o cofreq	0.82	0.57	0.96	0.58	0.96
Structure (coarse)					
s/o min deg nbr	0.74	0.55	0.97	0.58	0.93
s/o max deg nbr	0.88	0.69	0.95	0.65	0.89
s/o mean deg nbr	0.80	0.71	0.96	0.63	0.73
s/o num nbrs	0.71	0.54	0.96	0.45	0.93
s/o min freq rel	0.76	0.58	0.98	0.58	0.97
s/o max freq rel	0.80	0.71	0.94	0.59	0.97
s/o mean freq rel	0.73	0.52	0.98	0.48	0.95
s/o num rels	0.77	0.49	0.93	0.39	0.94

Table A.2: The results of all cross-KG feature ablation studies (for all KGs tested) when TWIG was trained to simulate ComplEx. All experiments are run in isolation on all KGs at once, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2 scores. Results that outperform TWIG when trained on all features are shown in bold.

A.2 TWIG on DistMult

Two feature ablations were run to analyse how TWIG learns to simulate DistMult. The first case, shown in Table A.3, shows the result of all feature ablations when DistMult was trained on each KG in the single-KG setting as described in Section 4.3.4. The second case, shown in Table A.4, shows the results of feature ablations when TWIG was trained to simulate DistMult in the cross-KG setting, as described in Section 4.3.4.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.49	0.65	0.74	0.87	0.98
Hyperparameters					
loss	0.34	0.17	0.68	0.78	0.90
neg. sampler	0.32	0.68	0.70	0.82	0.71
lr	0.18	0.16	0.06	0.02	0.15
reg. coeff.	0.45	0.65	0.82	0.80	0.93
npp	0.40	0.71	0.76	0.85	0.95
margin	0.41	0.61	0.75	0.88	0.97
dimension	0.53	0.55	0.76	0.90	0.98
Aggregate Fts					
all fine-grained	0.52	0.62	0.76	0.87	0.98
all coarse-grained	0.51	0.67	0.77	0.85	0.98
Structure (fine)					
s deg	0.36	0.77	0.76	0.84	0.97
o deg	0.33	0.75	0.83	0.83	0.97
p freq	0.56	0.67	0.77	0.84	0.97
s p cofreq	0.46	0.71	0.74	0.88	0.97
o p cofreq	0.48	0.62	0.76	0.82	0.98
s o cofreq	0.36	0.71	0.76	0.89	0.97
Structure (coarse)					
s/o min deg nbr	0.48	0.59	0.75	0.91	0.98
s/o max deg nbr	0.49	0.68	0.79	0.88	0.97
s/o mean deg nbr	0.57	0.68	0.76	0.90	0.98
s/o num nbrs	0.40	0.66	0.79	0.85	0.97
s/o min freq rel	0.51	0.66	0.68	0.89	0.96
s/o max freq rel	0.41	0.69	0.78	0.88	0.98
s/o mean freq rel	0.53	0.66	0.76	0.90	0.98
s/o num rels	0.46	0.65	0.73	0.85	0.97

Table A.3: The results of all feature ablation studies (for all KGs tested) when TWIG was trained to simulate DistMult. All experiments are run in isolation on a single KG-KGEM pair, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2 scores. Results that outperform TWIG when trained on all features are shown in bold.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.43	0.62	0.80	0.62	0.97
Hyperparameters					
loss	0.29	0.46	0.71	0.66	0.89
neg. samp.	0.22	0.56	0.78	0.82	0.70
lr	0.16	0.08	-0.15	0.02	0.17
reg. coeff.	0.31	0.55	0.71	0.74	0.95
npp	0.41	0.75	0.78	0.85	0.96
margin	0.45	0.64	0.74	0.86	0.96
dimension	0.42	0.67	0.70	0.85	0.95
Aggregate Fts					
all fine-grained	0.49	0.70	0.79	0.85	0.94
all coarse-grained	0.35	0.65	0.79	0.78	0.98
Structure (fine)					
s deg	0.45	0.60	0.69	0.83	0.96
o deg	0.49	0.70	0.79	0.89	0.97
p freq	0.35	0.67	0.80	0.78	0.96
s p cofreq	0.43	0.58	0.68	0.81	0.97
o p cofreq	0.48	0.66	0.72	0.83	0.96
s o cofreq	0.44	0.65	0.74	0.85	0.95
Structure (coarse)					
s/o min deg nbr	0.39	0.66	0.68	0.85	0.96
s/o max deg nbr	0.51	0.65	0.76	0.83	0.97
s/o mean deg nbr	0.49	0.70	0.78	0.86	0.97
s/o num nbrs	0.42	0.68	0.72	0.81	0.95
s/o min freq rel	0.47	0.65	0.74	0.84	0.98
s/o max freq rel	0.40	0.69	0.66	0.84	0.96
s/o mean freq rel	0.39	0.58	0.65	0.82	0.93
s/o num rels	0.45	0.61	0.80	0.78	0.97

Table A.4: The results of all cross-KG feature ablation studies (for all KGs tested) when TWIG was trained to simulate DistMult. All experiments are run in isolation on all KGs at once, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2 scores. Results that outperform TWIG when trained on all features are shown in bold.

A.3 TWIG on TransE

Two feature ablations were run to analyse how TWIG learns to simulate TransE. The first case, shown in Table A.5, shows the result of all feature ablations when TransE was trained on each KG in the single-KG setting as described in Section 4.3.4. The second case, shown in Table A.6, shows the results of feature ablations when TWIG was trained to simulate TransE in the cross-KG setting, as described in Section 4.3.4.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.43	0.43	0.98	0.73	0.97
Hyperparameters					
loss	0.45	0.35	0.91	0.5	0.97
neg. sampler	0.12	0.47	0.96	0.59	0.73
lr	0.29	0.17	-0.01	-0.79	0.13
reg. coeff.	0.52	0.37	0.99	0.67	0.97
npp	0.47	0.56	0.97	0.7	0.98
margin	0.45	0.50	0.99	0.71	0.97
dimension	0.49	0.48	0.97	0.8	0.97
Aggregate Fts					
all fine-grained	0.53	0.58	0.96	0.76	0.97
all coarse-grained	0.45	0.44	0.98	0.73	0.98
Structure (fine)					
s deg	0.44	0.53	0.99	0.78	0.98
o deg	0.48	0.47	0.99	0.69	0.98
p freq	0.60	0.41	0.99	0.8	0.97
s p cofreq	0.50	0.62	0.98	0.68	0.97
o p cofreq	0.43	0.48	0.99	0.75	0.97
s o cofreq	0.36	0.38	0.98	0.76	0.98
Structure (coarse)					
s/o min deg nbr	0.44	0.48	0.97	0.79	0.98
s/o max deg nbr	0.49	0.57	0.98	0.73	0.98
s/o mean deg nbr	0.55	0.41	0.99	0.79	0.96
s/o num nbrs	0.42	0.48	0.98	0.59	0.98
s/o min freq rel	0.53	0.44	0.98	-1.00	0.97
s/o max freq rel	0.44	0.54	0.99	0.77	0.97
s/o mean freq rel	0.51	0.41	0.98	0.7	0.95
s/o num rels	0.57	0.56	0.97	0.73	0.97

Table A.5: The results of all feature ablation studies (for all KGs tested) when TWIG was trained to simulate TransE. All experiments are run in isolation on a single KG-KGEM pair, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2 scores. Results that outperform TWIG when trained on all features are shown in bold.

Feature Removed	CoDExSmall	DBpedia50	Kinships	OpenEA	UMLS
none	0.54	0.50	0.98	0.74	0.97
Hyperparameters					
loss	0.41	0.37	0.90	0.56	0.95
neg. samp.	0.12	0.21	0.92	0.53	0.71
lr	0.23	0.17	0.00	0.11	0.16
reg. coeff.	0.30	0.36	0.95	0.61	0.94
npp	0.48	0.40	0.92	0.62	0.97
margin	0.43	0.24	0.96	0.42	0.96
dimension	0.47	0.58	0.75	0.71	0.97
Aggregate Fts					
all fine-grained	0.43	0.46	0.94	0.75	0.97
all coarse-grained	0.34	0.31	0.97	0.55	0.97
Structure (fine)					
s deg	0.53	0.46	0.99	0.74	0.98
o deg	0.31	0.34	0.94	0.64	0.97
p freq	0.45	0.25	0.95	0.59	0.98
s p cofreq	0.47	0.49	0.99	0.67	0.98
o p cofreq	0.36	0.44	0.95	0.59	0.95
s o cofreq	0.41	0.36	0.96	0.58	0.97
Structure (coarse)					
s/o min deg nbr	0.48	0.48	0.95	0.69	0.96
s/o max deg nbr	0.47	0.45	0.92	0.71	0.96
s/o mean deg nbr	0.55	0.53	0.96	0.80	0.98
s/o num nbrs	0.44	0.49	0.97	0.71	0.96
s/o min freq rel	0.44	0.41	0.88	0.66	0.98
s/o max freq rel	0.48	0.52	0.96	0.72	0.98
s/o mean freq rel	0.45	0.44	0.96	0.70	0.96
s/o num rels	0.49	0.40	0.93	0.70	0.95

Table A.6: The results of all cross-KG feature ablation studies (for all KGs tested) when TWIG was trained to simulate TransE. All experiments are run in isolation on all KGs at once, with the specified feature(s) removed. All ablations are grouped into either ablations of hyperparameter features, of fine-grained structural features, or of coarse-grained structural features. The first row show’s TWIG’s results with all features for reference, as reported in Section 4.3. All performance values are given as R2 scores. Results that outperform TWIG when trained on all features are shown in bold.

B. Structural Feature Distributions

This section gives a description of the distributions of all structural features in each of the 5 KGs used by TWIG or TWIG-I in this thesis. Each sub-section describes one dataset (i.e. CoDExSmall [75], DBpedia50 [88], Kinships [42], OpenEA [91], or UMLS [56]). All data is presented in terms of the minimum, 25th-percentile, median, 75th-percentile and maximum value of each structural feature used.

Features are calculated from all triples in the KG (across all training, testing, and validation splits). All structural features are calculated at the level of individual triples. Fine-grained features are those describing elements in the triple itself, coarse-grained features are aggregate statistics of the neighbourhood around said central triple. A full description of the meaning of each feature, and the reason for its use, is given in Section 3.2 in Chapter 3.

It can be observed that, in the general case, most features tend to follow a power law – the maximum value they obtain is typically far larger than that of even the 75th percentile, and values grow at a super-linear rate in almost all cases. Kinships is the one exception to this trend, where some of its values are clearly much more uniformly distributed. As such, analysis (even cursory) of this data must treat Kinships differently from the rest of the datasets described.

Note that all distributions are given in terms of extrema (minimum and maximum) and quartiles (25th percentile, median, and 75th percentile) because in most cases the data is clearly skewed and does not follow a Normal distribution. All data is given on the following pages.

B.1 CoDExSmall

Table B.1 provides a distributional description of all 22 structural features used by TWIG and TWIG-I to annotate CoDExSmall [75].

Features are calculated from all triples in the KG (across all training, testing, and validation splits). All structural features are calculated at the level of individual triples. Fine-grained features are those describing elements in the triple itself, coarse-grained features are aggregate statistics of the neighbourhood around said central triple. A full description of the meaning of each feature, and the reason for its use, is given in Section 3.2 in Chapter 3.

Feature	Min	25%	50%	75%	Max
Fine-grained Features					
s deg	10	15	18	39	1008
o deg	11	43	112	232	1008
p freq	15	1477	4985	10197	10197
s p cofreq	0	1	6	13	171
o p cofreq	0	29	72	196	676
s o cofreq	0	0	0	0	2
Coarse-grained Features					
s min deg neighbour	10	20	45	128	1008
s max deg neighbour	11	117	232	440	1008
s mean deg neighbour	11	73.3	136.8	242	1008
s num neighbours	1	1	2	4	50
s min freq rel	15	383	1648	4985	10197
s max freq rel	15	4985	5563	10197	10197
s mean freq rel	15	2656.5	5270.5	5837	10197
s num rels	1	1	2	2	5
o min deg neighbour	10	12	13	18	480
o max deg neighbour	11	22	29	174	1008
o mean deg neighbour	11	16.4	19	63.3	744
o num neighbours	1	3	7	15	50
o min freq rel	15	879	4985	10197	10197
o max freq rel	24	1648	5563	10197	10197
o mean freq rel	24	1477	4985	10197	10197
o num rels	1	1	1	2	5

Table B.1: An overview of the distribution of all structural features for all triples (in all the training, testing, and validation splits) in the KG CoDExSmall.

B.2 DBpedia50

Table B.2 provides a distributional description of all 22 structural features used by TWIG and TWIG-I to annotate DBpedia50 [88].

Features are calculated from all triples in the KG (across all training, testing, and validation splits). All structural features are calculated at the level of individual triples. Fine-grained features are those describing elements in the triple itself, coarse-grained features are aggregate statistics of the neighbourhood around said central triple. A full description of the meaning of each feature, and the reason for its use, is given in Section 3.2 in Chapter 3.

Feature	Min	25%	50%	75%	Max
Fine-grained Features					
s deg	1	1	2	3	22
o deg	1	3	15	188	781
p freq	9	233	458	1028	3006
s p cofreq	0	0	0	0.5	22
o p cofreq	0	0.5	7	59	701
s o cofreq	0	0	0	0	2
Coarse-grained Features					
s min deg neighbour	1	3	15	188	781
s max deg neighbour	1	3	15	188	781
s mean deg neighbour	1	3	15	188	781
s num neighbours	1	1	1	1	1
s min freq rel	9	233	458	1028	3006
s max freq rel	9	233	458	1028	3006
s mean freq rel	9	233	458	1028	3006
s num rels	1	1	1	1	1
o min deg neighbour	1	1	1	2	22
o max deg neighbour	1	1	2	3	22
o mean deg neighbour	1	1	2	3	22
o num neighbours	1	1	1	2	6
o min freq rel	9	198	443	1028	3006
o max freq rel	9	258	458	1739	3006
o mean freq rel	9	258	458	1028	3006
o num rels	1	1	1	1	5

Table B.2: An overview of the distribution of all structural features for all triples (in all the training, testing, and validation splits) in the KG DBpedia50.

B.3 Kinships

Table B.3 provides a distributional description of all 22 structural features used by TWIG and TWIG-I to annotate Kinships [42].

Features are calculated from all triples in the KG (across all training, testing, and validation splits). All structural features are calculated at the level of individual triples. Fine-grained features are those describing elements in the triple itself, coarse-grained features are aggregate statistics of the neighbourhood around said central triple. A full description of the meaning of each feature, and the reason for its use, is given in Section 3.2 in Chapter 3.

Feature	Min	25%	50%	75%	Max
Fine-grained Features					
s deg	148	161	164	168	174
o deg	148	161	164	168	174
p freq	34	367	404	663	1004
s p cofreq	0	4	6	9	21
o p cofreq	0	5	8	12	24
s o cofreq	0	0	0	0	0
Coarse-grained Features					
s min deg neighbour	148	152	153	155	160
s max deg neighbour	168	172	172	174	174
s mean deg neighbour	161.2	163.1	163.8	164.5	166.1
s num neighbours	12	18	20	23	30
s min freq rel	34	106	183	185	370
s max freq rel	663	1004	1004	1004	1004
s mean freq rel	378.4	448.9	473.5	494.7	568.2
s num rels	7	10	11	13	16
o min deg neighbour	148	152	153	155	160
o max deg neighbour	168	172	172	174	174
o mean deg neighbour	161.2	163.1	163.8	164.5	166.1
o num neighbours	12	18	20	23	30
o min freq rel	34	106	183	185	370
o max freq rel	663	1004	1004	1004	1004
o mean freq rel	378.4	455.4	472.9	495	568.2
o num rels	7	10	11	13	16

Table B.3: An overview of the distribution of all structural features for all triples (in all the training, testing, and validation splits) in the KG Kinships.

B.4 OpenEA

Table B.4 provides a distributional description of all 22 structural features used by TWIG and TWIG-I to annotate OpenEA [91].

Features are calculated from all triples in the KG (across all training, testing, and validation splits). All structural features are calculated at the level of individual triples. Fine-grained features are those describing elements in the triple itself, coarse-grained features are aggregate statistics of the neighbourhood around said central triple. A full description of the meaning of each feature, and the reason for its use, is given in Section 3.2 in Chapter 3.

Feature	Min	25%	50%	75%	Max
Fine-grained Features					
s deg	1	2	3	5	188
o deg	1	6	12	22	285
p freq	1	546	1242	1716	4788
s p cofreq	0	0	0	1	9
o p cofreq	0	2	5	11	241
s o cofreq	0	0	0	0	4
Coarse-grained Features					
s min deg neighbour	1	5	9	17	285
s max deg neighbour	1	7	14	26	285
s mean deg neighbour	1	7	12.2	22	285
s num neighbours	1	1	1	2	16
s min freq rel	1	399	1030	1716	4788
s max freq rel	1	714	1598	4788	4788
s mean freq rel	1	620	1311.5	2909	4788
s num rels	1	1	1	2	8
o min deg neighbour	1	1	2	3	32
o max deg neighbour	1	3	5	9	285
o mean deg neighbour	1	2.2	3.5	5.2	143
o num neighbours	1	1	3	5	29
o min freq rel	1	236	741	1632	4788
o max freq rel	1	741	1598	4788	4788
o mean freq rel	1	546	1242	2205	4788
o num rels	1	1	1	2	10

Table B.4: An overview of the distribution of all structural features for all triples (in all the training, testing, and validation splits) in the KG OpenEA.

B.5 UMLS

Table B.5 provides a distributional description of all 22 structural features used by TWIG and TWIG-I to annotate UMLS [56].

Features are calculated from all triples in the KG (across all training, testing, and validation splits). All structural features are calculated at the level of individual triples. Fine-grained features are those describing elements in the triple itself, coarse-grained features are aggregate statistics of the neighbourhood around said central triple. A full description of the meaning of each feature, and the reason for its use, is given in Section 3.2 in Chapter 3.

Feature	Min	25%	50%	75%	Max
Fine-grained Features					
s deg	6	59	84	182	306
o deg	7	59	122	223	306
p freq	6	153	283	455	803
s p cofreq	0	6	11	19	38
o p cofreq	0	5	11	23	115
s o cofreq	0	0	1	2	9
Coarse-grained Features					
s min deg neighbour	7	20	40	53	292
s max deg neighbour	20	299	304	306	306
s mean deg neighbour	15.5	120.6	132.5	154.3	292
s num neighbours	1	8	12	22	34
s min freq rel	6	25	35	51	455
s max freq rel	30	399	803	803	803
s mean freq rel	30	222.4	255.6	311.8	552.3
s num rels	1	4	7	10	16
o min deg neighbour	6	24	40	53	182
o max deg neighbour	16	299	304	306	306
o mean deg neighbour	15.5	120.6	130.8	148.5	218.5
o num neighbours	1	8	15	23	34
o min freq rel	6	27	42	56	455
o max freq rel	15	803	803	803	803
o mean freq rel	15	241.7	272.4	335	552.3
o num rels	1	4	8	11	16

Table B.5: An overview of the distribution of all structural features for all triples (in all the training, testing, and validation splits) in the KG UMLS.

C. Further KG-KGEM Data

This Appendix describes all further data on KG structure, hyperparameter preference, and link prediction performance that was used in the results and analysis in Section 4.4 in Chapter 4. As analysis of the overall trends in this data, as well as its impact on the Structural Alignment Hypothesis, has already been given there, it is not repeated here.

Each of the following sections contains the raw data used for that analysis on the three KGEMs (ComplEx [47, 94], DistMult [102], and TransE [11]) and the five KGs (CoDExSmall [75], DBpedia50 [88], Kinships [42], OpenEA [91], and UMLS [56]) that were considered. Finally, note that the data for ComplEx on UMLS is reproduced here for ease of reference.

C.1 ComplEx on CoDExSmall

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.13	-0.2	-0.02
o deg	-0.02	-0.01	-0.08
p freq	-0.05	-0.06	-0.05
s p cofreq	-0.14	-0.22	-0.02
o p cofreq	0.0	0.03	-0.08
s o cofreq	0.01	0.0	0.01
s min deg nbr	0.01	0.03	-0.06
s max deg nbr	-0.02	-0.02	-0.03
s mean deg nbr	0.01	0.03	-0.05
s num s o cofreq	-0.13	-0.19	-0.02
s min freq rel	-0.05	-0.07	-0.04
s max freq rel	0.02	0.04	-0.04
s mean freq rel	-0.03	-0.03	-0.05
s num rels	-0.02	-0.04	0.0
o min deg nbr	-0.11	-0.16	-0.01
o max deg nbr	-0.14	-0.21	-0.03
o mean deg nbr	-0.15	-0.23	-0.02
o num s o cofreq	-0.03	-0.02	-0.08
o min freq rel	-0.02	-0.02	-0.04
o max freq rel	-0.05	-0.07	-0.04
o mean freq rel	-0.04	-0.05	-0.04
o num rels	-0.02	-0.03	0.0

Table C.1: ComplEx on CoDExSmall: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
MRL	Bernoulli	0.01	0.01	125	2	100	0.36
BCE	Bernoulli	0.01	0.01	125	None	100	0.04
CE	Bernoulli	0.01	0.01	125	None	100	0.31
MRL	Basic	0.01	0.01	125	2	100	0.27
MRL	Pseudo	0.01	0.01	125	2	100	0.02
MRL	Bernoulli	0.0001	0.01	125	2	100	0.0
MRL	Bernoulli	1e-06	0.01	125	2	100	0.0
MRL	Bernoulli	0.01	0.0001	125	2	100	0.18
MRL	Bernoulli	0.01	1e-06	125	2	100	0.18
MRL	Bernoulli	0.01	0.01	5	2	100	0.3
MRL	Bernoulli	0.01	0.01	25	2	100	0.33
MRL	Bernoulli	0.01	0.01	125	0.5	100	0.28
MRL	Bernoulli	0.01	0.01	125	1	100	0.32
MRL	Bernoulli	0.01	0.01	125	2	50	0.35
MRL	Bernoulli	0.01	0.01	125	2	250	0.35

Table C.2: ComplEx on CoDExSmall: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.0	0.02	0.36
Loss = BCE	0.0	0.0	0.0	0.01	0.33
Loss = CE	0.0	0.0	0.0	0.03	0.34
N. Samp = Basic	0.0	0.0	0.0	0.13	0.34
N. Samp = Bernoulli	0.0	0.0	0.0	0.18	0.36
N. Samp = Pseudo	0.0	0.0	0.0	0.01	0.05
LR = 0.01	0.0	0.02	0.17	0.24	0.36
LR = 0.0001	0.0	0.0	0.0	0.0	0.01
LR = 1e-06	0.0	0.0	0.0	0.0	0.01
Reg = 0.01	0.0	0.0	0.0	0.03	0.36
Reg = 0.0001	0.0	0.0	0.0	0.01	0.32
Reg = 1e-06	0.0	0.0	0.0	0.01	0.32
npp = 5	0.0	0.0	0.0	0.02	0.33
npp = 25	0.0	0.0	0.0	0.03	0.34
npp = 125	0.0	0.0	0.0	0.03	0.36
Mgn = None	0.0	0.0	0.0	0.03	0.34
Mgn = 0.5	0.0	0.0	0.0	0.04	0.3
Mgn = 1	0.0	0.0	0.0	0.02	0.32
Mgn = 2	0.0	0.0	0.0	0.02	0.36
Dim = 50	0.0	0.0	0.0	0.02	0.35
Dim = 100	0.0	0.0	0.0	0.02	0.36
Dim = 250	0.0	0.0	0.0	0.02	0.35

Table C.3: Distribution of MRR scores obtained when running ComplEx on CoDExSmall with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of ComplEx on CoDExSmall in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	MRL	Bernoulli	0.01	0.01	125	2	100	0.36
s deg ≤ 18.0	MRL	Bernoulli	0.01	0.01	25	2	250	0.31
s deg > 18.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.41
o deg ≤ 112.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.33
o deg > 112.0	CE	Basic	0.01	0.01	125	None	100	0.40
p freq ≤ 4985.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.37
p freq > 4985.0	CE	Bernoulli	0.01	0.01	125	None	250	0.34
s p cofreq ≤ 6.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.33
s p cofreq > 6.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.39
o p cofreq ≤ 72.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.33
o p cofreq > 72.0	CE	Basic	0.01	0.01	125	None	100	0.39
s o cofreq ≤ 0.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.36
s o cofreq > 0.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.44
s min deg nbr ≤ 45.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.35
s min deg nbr > 45.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.37
s max deg nbr ≤ 232.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.32
s max deg nbr > 232.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.4
s mean deg nbr ≤ 136.8	MRL	Bernoulli	0.01	0.01	125	2	100	0.34
s mean deg nbr > 136.8	MRL	Bernoulli	0.01	0.01	125	2	100	0.38
s num s o cofreq ≤ 2.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.32
s num s o cofreq > 2.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.41
s min freq rel ≤ 1648.0	MRL	Bernoulli	0.01	0.01	125	2	250	0.35
s min freq rel > 1648.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.37
s max freq rel ≤ 5563.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.41
s max freq rel > 5563.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.28
s mean freq rel ≤ 5270.5	MRL	Bernoulli	0.01	0.01	125	2	100	0.37
s mean freq rel > 5270.5	MRL	Bernoulli	0.01	0.01	125	2	100	0.35
s num rels ≤ 2.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.35
s num rels > 2.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.39
o min deg nbr ≤ 13.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.34
o min deg nbr > 13.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.38
o max deg nbr ≤ 29.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.3
o max deg nbr > 29.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.42
o mean deg nbr ≤ 19.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.3
o mean deg nbr > 19.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.41
o num s o cofreq ≤ 7.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.33
o num s o cofreq > 7.0	CE	Basic	0.01	0.01	125	None	100	0.4
o min freq rel ≤ 4985.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.39
o min freq rel > 4985.0	MRL	Bernoulli	0.01	0.01	125	2	250	0.29
o max freq rel ≤ 5563.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.4
o max freq rel > 5563.0	MRL	Bernoulli	0.01	0.01	25	1	50	0.26
o mean freq rel ≤ 4985.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.39
o mean freq rel > 4985.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.32
o num rels ≤ 1.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.34
o num rels > 1.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.41

Table C.4: ComplEx on CoDexSmall: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.2 ComplEx on DBpedia50

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.13	-0.15	-0.1
o deg	-0.16	-0.13	-0.19
p freq	-0.12	-0.13	-0.12
s p cofreq	-0.03	-0.04	-0.02
o p cofreq	-0.13	-0.12	-0.15
s o cofreq	-0.22	-0.25	-0.18
s min deg nbr	-0.16	-0.13	-0.19
s max deg nbr	-0.16	-0.13	-0.19
s mean deg nbr	-0.16	-0.13	-0.19
s num s o cofreq	N/A	N/A	N/A
s min freq rel	-0.12	-0.13	-0.12
s max freq rel	-0.12	-0.13	-0.12
s mean freq rel	-0.12	-0.13	-0.12
s num rels	N/A	N/A	N/A
o min deg nbr	-0.11	-0.14	-0.07
o max deg nbr	-0.13	-0.13	-0.14
o mean deg nbr	-0.13	-0.15	-0.11
o num s o cofreq	-0.11	-0.09	-0.15
o min freq rel	-0.1	-0.11	-0.11
o max freq rel	-0.13	-0.11	-0.15
o mean freq rel	-0.12	-0.11	-0.13
o num rels	-0.06	-0.04	-0.09

Table C.5: ComplEx on DBpedia50: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
BCE	Basic	0.01	0.01	25	None	100	0.38
MRL	Basic	0.01	0.01	25	0.5	100	0.28
MRL	Basic	0.01	0.01	25	1	100	0.29
MRL	Basic	0.01	0.01	25	2	100	0.29
CE	Basic	0.01	0.01	25	None	100	0.16
BCE	Bernoulli	0.01	0.01	25	None	100	0.33
BCE	Pseudo	0.01	0.01	25	None	100	0.03
BCE	Basic	0.0001	0.01	25	None	100	0.0
BCE	Basic	1e-06	0.01	25	None	100	0.0
BCE	Basic	0.01	0.0001	25	None	100	0.23
BCE	Basic	0.01	1e-06	25	None	100	0.2
BCE	Basic	0.01	0.01	5	None	100	0.34
BCE	Basic	0.01	0.01	125	None	100	0.21
BCE	Basic	0.01	0.01	25	None	50	0.38
BCE	Basic	0.01	0.01	25	None	250	0.38

Table C.6: ComplEx on DBpedia50: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.0	0.09	0.37
Loss = BCE	0.0	0.0	0.0	0.01	0.38
Loss = CE	0.0	0.0	0.0	0.11	0.21
N. Samp = Basic	0.0	0.0	0.0	0.09	0.38
N. Samp = Bernoulli	0.0	0.0	0.0	0.1	0.37
N. Samp = Pseudo	0.0	0.0	0.0	0.07	0.19
LR = 0.01	0.0	0.09	0.11	0.17	0.38
LR = 0.0001	0.0	0.0	0.0	0.0	0.01
LR = 1e-06	0.0	0.0	0.0	0.0	0.01
Reg = 0.01	0.0	0.0	0.0	0.11	0.38
Reg = 0.0001	0.0	0.0	0.0	0.08	0.31
Reg = 1e-06	0.0	0.0	0.0	0.08	0.27
npp = 5	0.0	0.0	0.0	0.08	0.35
npp = 25	0.0	0.0	0.0	0.09	0.38
npp = 125	0.0	0.0	0.0	0.08	0.37
Mgn = None	0.0	0.0	0.0	0.09	0.38
Mgn = 0.5	0.0	0.0	0.0	0.09	0.32
Mgn = 1	0.0	0.0	0.0	0.08	0.36
Mgn = 2	0.0	0.0	0.0	0.09	0.37
Dim = 50	0.0	0.0	0.0	0.08	0.38
Dim = 100	0.0	0.0	0.0	0.09	0.38
Dim = 250	0.0	0.0	0.0	0.09	0.38

Table C.7: Distribution of MRR scores obtained when running ComplEx on DBpedia50 with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of ComplEx on DBpedia50 in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	BCE	Basic	0.01	0.01	25	None	100	0.38
s deg ≤ 2.0	BCE	Basic	0.01	0.01	25	None	50	0.34
s deg > 2.0	BCE	Basic	0.01	0.01	25	None	100	0.51
o deg ≤ 15.0	BCE	Basic	0.01	0.01	25	None	250	0.47
o deg > 15.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.34
p freq ≤ 458.0	BCE	Basic	0.01	0.01	25	None	250	0.43
p freq > 458.0	BCE	Basic	0.01	0.01	25	None	100	0.36
s p cofreq ≤ 0.0	BCE	Basic	0.01	0.01	25	None	50	0.41
s p cofreq > 0.0	BCE	Basic	0.01	0.01	25	None	250	0.33
o p cofreq ≤ 7.0	BCE	Basic	0.01	0.01	25	None	100	0.5
o p cofreq > 7.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.29
s o cofreq ≤ 0.0	BCE	Basic	0.01	0.01	25	None	250	0.28
s o cofreq > 0.0	BCE	Basic	0.01	0.01	5	None	250	0.79
s min deg nbr ≤ 15.0	BCE	Basic	0.01	0.01	25	None	250	0.47
s min deg nbr > 15.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.34
s max deg nbr ≤ 15.0	BCE	Basic	0.01	0.01	25	None	250	0.47
s max deg nbr > 15.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.34
s mean deg nbr ≤ 15.0	BCE	Basic	0.01	0.01	25	None	250	0.47
s mean deg nbr > 15.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.34
s num neighbours	N/A							
s min freq rel ≤ 458.0	BCE	Basic	0.01	0.01	25	None	250	0.43
s min freq rel > 458.0	BCE	Basic	0.01	0.01	25	None	100	0.36
s max freq rel ≤ 458.0	BCE	Basic	0.01	0.01	25	None	250	0.43
s max freq rel > 458.0	BCE	Basic	0.01	0.01	25	None	100	0.36
s mean freq rel ≤ 458.0	BCE	Basic	0.01	0.01	25	None	250	0.43
s mean freq rel > 458.0	BCE	Basic	0.01	0.01	25	None	100	0.36
s num rels	N/A							
o min deg nbr ≤ 1.0	BCE	Basic	0.01	0.01	25	None	50	0.33
o min deg nbr > 1.0	BCE	Basic	0.01	0.01	25	None	250	0.48
o max deg nbr ≤ 2.0	BCE	Basic	0.01	0.01	25	None	250	0.35
o max deg nbr > 2.0	BCE	Basic	0.01	0.01	25	None	100	0.45
o mean deg nbr ≤ 2.0	BCE	Basic	0.01	0.01	25	None	250	0.34
o mean deg nbr > 2.0	BCE	Basic	0.01	0.01	25	None	100	0.51
o num s o cofreq ≤ 1.0	BCE	Basic	0.01	0.01	25	None	100	0.43
o num s o cofreq > 1.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.33
o min freq rel ≤ 443.0	BCE	Basic	0.01	0.01	25	None	250	0.44
o min freq rel > 443.0	BCE	Basic	0.01	0.01	25	None	100	0.36
o max freq rel ≤ 458.0	BCE	Basic	0.01	0.01	25	None	250	0.43
o max freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.36
o mean freq rel ≤ 458.0	BCE	Basic	0.01	0.01	25	None	250	0.43
o mean freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.36
o num rels ≤ 1.0	BCE	Basic	0.01	0.01	25	None	100	0.4
o num rels > 1.0	MRL	Bernoulli	0.01	0.01	125	2	50	0.36

Table C.8: ComplEx on DBpedia50: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.3 ComplEx on Kinships

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.03	-0.04	-0.03
o deg	0.0	-0.01	0.01
p freq	-0.02	-0.02	-0.01
s p cofreq	-0.07	-0.06	-0.07
o p cofreq	-0.04	-0.04	-0.03
s o cofreq	N/A	N/A	N/A
s min deg nbr	0.01	0.01	0.02
s max deg nbr	-0.03	-0.04	-0.03
s mean deg nbr	0.06	0.05	0.07
s num s o cofreq	-0.02	-0.02	-0.03
s min freq rel	0.02	0.03	0.02
s max freq rel	0.02	0.01	0.03
s mean freq rel	0.01	0.02	0.01
s num rels	-0.04	-0.04	-0.04
o min deg nbr	0.03	0.05	0.02
o max deg nbr	-0.0	-0.0	-0.01
o mean deg nbr	0.02	0.02	0.02
o num s o cofreq	0.02	0.02	0.02
o min freq rel	0.0	-0.0	0.01
o max freq rel	0.03	0.04	0.03
o mean freq rel	0.03	0.02	0.04
o num rels	0.01	0.02	0.0

Table C.9: ComplEx on Kinships: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
BCE	Bernoulli	0.01	0.01	5	None	50	0.63
MRL	Bernoulli	0.01	0.01	5	0.5	50	0.31
MRL	Bernoulli	0.01	0.01	5	1	50	0.32
MRL	Bernoulli	0.01	0.01	5	2	50	0.38
CE	Bernoulli	0.01	0.01	5	None	50	0.41
BCE	Basic	0.01	0.01	5	None	50	0.61
BCE	Pseudo	0.01	0.01	5	None	50	0.44
BCE	Bernoulli	0.0001	0.01	5	None	50	0.05
BCE	Bernoulli	1e-06	0.01	5	None	50	0.05
BCE	Bernoulli	0.01	0.0001	5	None	50	0.44
BCE	Bernoulli	0.01	1e-06	5	None	50	0.41
BCE	Bernoulli	0.01	0.01	25	None	50	0.62
BCE	Bernoulli	0.01	0.01	125	None	50	0.16
BCE	Bernoulli	0.01	0.01	5	None	100	0.63
BCE	Bernoulli	0.01	0.01	5	None	250	0.63

Table C.10: ComplEx on Kinships: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.04	0.05	0.06	0.23	0.38
Loss = BCE	0.04	0.05	0.06	0.23	0.63
Loss = CE	0.05	0.05	0.06	0.3	0.53
N. Samp = Basic	0.05	0.05	0.06	0.3	0.63
N. Samp = Bernoulli	0.04	0.05	0.06	0.3	0.63
N. Samp = Pseudo	0.04	0.05	0.06	0.21	0.48
LR = 0.01	0.08	0.25	0.3	0.36	0.63
LR = 0.0001	0.04	0.05	0.05	0.06	0.06
LR = 1e-06	0.04	0.05	0.05	0.06	0.07
Reg = 0.01	0.04	0.05	0.06	0.26	0.63
Reg = 0.0001	0.05	0.05	0.06	0.25	0.49
Reg = 1e-06	0.04	0.05	0.06	0.25	0.52
npp = 5	0.04	0.05	0.06	0.26	0.63
npp = 25	0.04	0.05	0.06	0.27	0.63
npp = 125	0.04	0.05	0.06	0.21	0.53
Mgn = None	0.04	0.05	0.06	0.28	0.63
Mgn = 0.5	0.05	0.05	0.06	0.22	0.33
Mgn = 1	0.05	0.05	0.06	0.23	0.35
Mgn = 2	0.04	0.05	0.06	0.24	0.38
Dim = 50	0.04	0.05	0.06	0.25	0.63
Dim = 100	0.04	0.05	0.06	0.25	0.63
Dim = 250	0.04	0.05	0.06	0.25	0.63

Table C.11: Distribution of MRR scores obtained when running ComplEx on Kinships with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of ComplEx on Kinships in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	BCE	Bernoulli	0.01	0.01	5	None	50	0.63
s deg ≤ 164.0	BCE	Basic	0.01	0.01	25	None	50	0.62
s deg > 164.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.65
o deg ≤ 164.0	BCE	Basic	0.01	0.01	25	None	50	0.63
o deg > 164.0	BCE	Basic	0.01	0.01	5	None	250	0.64
p freq ≤ 404.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64
p freq > 404.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.65
s p cofreq ≤ 6.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.62
s p cofreq > 6.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.66
o p cofreq ≤ 8.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.62
o p cofreq > 8.0	BCE	Bernoulli	0.01	0.01	5	None	250	0.65
s o cofreq	N/A							
s min deg nbr ≤ 153.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.64
s min deg nbr > 153.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.63
s max deg nbr ≤ 172.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.63
s max deg nbr > 172.0	BCE	Bernoulli	0.01	0.01	5	None	250	0.64
s mean deg nbr ≤ 163.8	BCE	Bernoulli	0.01	0.01	5	None	50	0.64
s mean deg nbr > 163.8	BCE	Bernoulli	0.01	0.01	5	None	100	0.63
s num s o cofreq ≤ 20.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.64
s num s o cofreq > 20.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.64
s min freq rel ≤ 183.0	BCE	Basic	0.01	0.01	25	None	50	0.65
s min freq rel > 183.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.63
s max freq rel	N/A							
s mean freq rel ≤ 473.5	BCE	Basic	0.01	0.01	25	None	50	0.64
s mean freq rel > 473.5	BCE	Bernoulli	0.01	0.01	5	None	50	0.63
s num rels ≤ 11.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.64
s num rels > 11.0	BCE	Bernoulli	0.01	0.01	5	None	250	0.64
o min deg nbr ≤ 153.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.65
o min deg nbr > 153.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.62
o max deg nbr ≤ 172.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.63
o max deg nbr > 172.0	BCE	Bernoulli	0.01	0.01	5	None	50	0.64
o mean deg nbr ≤ 163.8	BCE	Bernoulli	0.01	0.01	5	None	100	0.63
o mean deg nbr > 163.8	BCE	Bernoulli	0.01	0.01	5	None	250	0.65
o num s o cofreq ≤ 20.0	BCE	Basic	0.01	0.01	5	None	250	0.62
o num s o cofreq > 20.0	BCE	Bernoulli	0.01	0.01	5	None	250	0.65
o min freq rel ≤ 183.0	BCE	Basic	0.01	0.01	25	None	50	0.65
o min freq rel > 183.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.64
o max freq rel	N/A							
o mean freq rel ≤ 472.9	BCE	Basic	0.01	0.01	25	None	50	0.63
o mean freq rel > 472.9	BCE	Bernoulli	0.01	0.01	5	None	100	0.63
o num rels ≤ 11.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.64
o num rels > 11.0	BCE	Bernoulli	0.01	0.01	5	None	250	0.63

Table C.12: ComplEx on Kinships: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.4 ComplEx on OpenEA

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.1	-0.13	-0.06
o deg	-0.07	-0.05	-0.12
p freq	0.12	0.14	0.09
s p cofreq	-0.04	-0.07	-0.0
o p cofreq	-0.04	-0.01	-0.09
s o cofreq	-0.18	-0.22	-0.14
s min deg nbr	-0.05	-0.02	-0.1
s max deg nbr	-0.06	-0.04	-0.1
s mean deg nbr	-0.06	-0.03	-0.11
s num s o cofreq	-0.05	-0.07	-0.02
s min freq rel	0.09	0.11	0.08
s max freq rel	0.09	0.11	0.07
s mean freq rel	0.1	0.12	0.08
s num rels	-0.07	-0.09	-0.05
o min deg nbr	-0.08	-0.13	-0.01
o max deg nbr	-0.04	-0.05	-0.03
o mean deg nbr	-0.05	-0.07	-0.02
o num s o cofreq	-0.09	-0.07	-0.13
o min freq rel	0.12	0.14	0.11
o max freq rel	0.08	0.09	0.06
o mean freq rel	0.11	0.13	0.1
o num rels	-0.08	-0.08	-0.1

Table C.13: ComplEx on OpenEA: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
BCE	Bernoulli	0.01	0.01	25	None	50	0.36
MRL	Bernoulli	0.01	0.01	25	0.5	50	0.26
MRL	Bernoulli	0.01	0.01	25	1	50	0.27
MRL	Bernoulli	0.01	0.01	25	2	50	0.29
CE	Bernoulli	0.01	0.01	25	None	50	0.14
BCE	Basic	0.01	0.01	25	None	50	0.34
BCE	Pseudo	0.01	0.01	25	None	50	0.02
BCE	Bernoulli	0.0001	0.01	25	None	50	0.0
BCE	Bernoulli	1e-06	0.01	25	None	50	0.0
BCE	Bernoulli	0.01	0.0001	25	None	50	0.18
BCE	Bernoulli	0.01	1e-06	25	None	50	0.17
BCE	Bernoulli	0.01	0.01	5	None	50	0.32
BCE	Bernoulli	0.01	0.01	125	None	50	0.01
BCE	Bernoulli	0.01	0.01	25	None	100	0.35
BCE	Bernoulli	0.01	0.01	25	None	250	0.35

Table C.14: ComplEx on OpenEA: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.0	0.07	0.3
Loss = BCE	0.0	0.0	0.0	0.01	0.36
Loss = CE	0.0	0.0	0.0	0.09	0.18
N. Samp = Basic	0.0	0.0	0.0	0.08	0.35
N. Samp = Bernoulli	0.0	0.0	0.0	0.07	0.36
N. Samp = Pseudo	0.0	0.0	0.0	0.05	0.23
LR = 0.01	0.0	0.07	0.08	0.14	0.36
LR = 0.0001	0.0	0.0	0.0	0.0	0.0
LR = 1e-06	0.0	0.0	0.0	0.0	0.0
Reg = 0.01	0.0	0.0	0.0	0.1	0.36
Reg = 0.0001	0.0	0.0	0.0	0.07	0.22
Reg = 1e-06	0.0	0.0	0.0	0.07	0.19
npp = 5	0.0	0.0	0.0	0.07	0.33
npp = 25	0.0	0.0	0.0	0.08	0.36
npp = 125	0.0	0.0	0.0	0.07	0.3
Mgn = None	0.0	0.0	0.0	0.07	0.36
Mgn = 0.5	0.0	0.0	0.0	0.07	0.26
Mgn = 1	0.0	0.0	0.0	0.07	0.29
Mgn = 2	0.0	0.0	0.0	0.07	0.3
Dim = 50	0.0	0.0	0.0	0.07	0.36
Dim = 100	0.0	0.0	0.0	0.07	0.35
Dim = 250	0.0	0.0	0.0	0.07	0.35

Table C.15: Distribution of MRR scores obtained when running ComplEx on OpenEA with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of ComplEx on OpenEA in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	BCE	Bernoulli	0.01	0.01	25	None	50	0.36
s deg ≤ 3.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.32
s deg > 3.0	BCE	Bernoulli	0.01	0.01	5	None	250	0.41
o deg ≤ 12.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.32
o deg > 12.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.39
p freq ≤ 1242.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.42
p freq > 1242.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.29
s p cofreq ≤ 0.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.41
s p cofreq > 0.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.26
o p cofreq ≤ 5.0	BCE	Basic	0.01	0.01	25	None	100	0.41
o p cofreq > 5.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.3
s o cofreq ≤ 0.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.23
s o cofreq > 0.0	BCE	Basic	0.01	0.01	5	None	250	0.76
s min deg nbr ≤ 9.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.33
s min deg nbr > 9.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.38
s max deg nbr ≤ 14.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.33
s max deg nbr > 14.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.38
s mean deg nbr ≤ 12.2	BCE	Bernoulli	0.01	0.01	25	None	50	0.33
s mean deg nbr > 12.2	BCE	Bernoulli	0.01	0.01	25	None	50	0.38
s num s o cofreq ≤ 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.36
s num s o cofreq > 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.35
s min freq rel ≤ 1030.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.42
s min freq rel > 1030.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.29
s max freq rel ≤ 1598.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.43
s max freq rel > 1598.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.26
s mean freq rel ≤ 1311.5	BCE	Bernoulli	0.01	0.01	25	None	50	0.43
s mean freq rel > 1311.5	BCE	Bernoulli	0.01	0.01	25	None	50	0.28
s num rels ≤ 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.33
s num rels > 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.41
o min deg nbr ≤ 2.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.31
o min deg nbr > 2.0	BCE	Bernoulli	0.01	0.01	5	None	100	0.45
o max deg nbr ≤ 5.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.32
o max deg nbr > 5.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.4
o mean deg nbr ≤ 3.5	BCE	Bernoulli	0.01	0.01	25	None	50	0.29
o mean deg nbr > 3.5	BCE	Bernoulli	0.01	0.01	25	None	50	0.42
o num s o cofreq ≤ 3.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.35
o num s o cofreq > 3.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.36
o min freq rel ≤ 741.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.45
o min freq rel > 741.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.26
o max freq rel ≤ 1598.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.4
o max freq rel > 1598.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.31
o mean freq rel ≤ 1242.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.42
o mean freq rel > 1242.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.28
o num rels ≤ 1.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.29
o num rels > 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.46

Table C.16: ComplEx on OpenEA: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.5 ComplEx on UMLS

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.24	-0.29	-0.19
o deg	-0.27	-0.24	-0.31
p freq	-0.12	-0.12	-0.13
s p cofreq	-0.21	-0.25	-0.17
o p cofreq	-0.08	-0.04	-0.13
s o cofreq	-0.17	-0.18	-0.15
s min deg nbr	-0.11	-0.06	-0.17
s max deg nbr	-0.34	-0.37	-0.32
s mean deg nbr	-0.3	-0.28	-0.33
s num s o cofreq	-0.24	-0.29	-0.2
s min freq rel	0.03	0.07	-0.01
s max freq rel	-0.24	-0.25	-0.23
s mean freq rel	-0.08	-0.05	-0.12
s num rels	-0.25	-0.29	-0.2
o min deg nbr	-0.08	-0.13	-0.02
o max deg nbr	-0.38	-0.36	-0.4
o mean deg nbr	-0.29	-0.3	-0.27
o num s o cofreq	-0.26	-0.22	-0.31
o min freq rel	0.13	0.12	0.15
o max freq rel	-0.35	-0.34	-0.37
o mean freq rel	-0.12	-0.13	-0.11
o num rels	-0.27	-0.24	-0.31

Table C.17: ComplEx on UMLS: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Meg	Dim	MRR
BCE	Bernoulli	0.01	0.01	25	None	50	0.55
MRL	Bernoulli	0.01	0.01	25	0.5	50	0.19
MRL	Bernoulli	0.01	0.01	25	1	50	0.22
MRL	Bernoulli	0.01	0.01	25	2	50	0.22
CE	Bernoulli	0.01	0.01	25	None	50	0.29
BCE	Basic	0.01	0.01	25	None	50	0.52
BCE	Pseudo	0.01	0.01	25	None	50	0.02
BCE	Bernoulli	0.0001	0.01	25	None	50	0.05
BCE	Bernoulli	1e-06	0.01	25	None	50	0.05
BCE	Bernoulli	0.01	0.0001	25	None	50	0.31
BCE	Bernoulli	0.01	1e-06	25	None	50	0.31
BCE	Bernoulli	0.01	0.01	5	None	50	0.39
BCE	Bernoulli	0.01	0.01	125	None	50	0.22
BCE	Bernoulli	0.01	0.01	25	None	100	0.55
BCE	Bernoulli	0.01	0.01	25	None	250	0.54

Table C.18: ComplEx on UMLS: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.02	0.04	0.05	0.06	0.27
Loss = BCE	0.02	0.04	0.05	0.06	0.55
Loss = CE	0.02	0.04	0.05	0.05	0.34
N. Samp = Basic	0.04	0.04	0.05	0.21	0.53
N. Samp = Bernoulli	0.04	0.05	0.05	0.21	0.55
N. Samp = Pseudo	0.02	0.03	0.04	0.05	0.06
LR = 0.01	0.02	0.03	0.21	0.26	0.55
LR = 0.0001	0.03	0.04	0.04	0.05	0.06
LR = 1e-06	0.04	0.04	0.05	0.05	0.06
Reg = 0.01	0.02	0.04	0.05	0.05	0.55
Reg = 0.0001	0.02	0.04	0.05	0.06	0.33
Reg = 1e-06	0.02	0.04	0.05	0.05	0.34
npp = 5	0.02	0.04	0.05	0.06	0.4
npp = 25	0.02	0.04	0.05	0.05	0.55
npp = 125	0.02	0.04	0.05	0.06	0.34
Mgn = None	0.02	0.04	0.05	0.05	0.55
Mgn = 0.5	0.02	0.04	0.05	0.05	0.26
Mgn = 1	0.02	0.04	0.05	0.06	0.27
Mgn = 2	0.02	0.04	0.05	0.06	0.27
Dim = 50	0.02	0.04	0.05	0.06	0.55
Dim = 100	0.02	0.04	0.05	0.05	0.55
Dim = 250	0.02	0.04	0.05	0.06	0.54

Table C.19: Distribution of MRR scores obtained when running ComplEx on UMLS with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of ComplEx on UMLS in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
s deg ≤ 84.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.49
s deg > 84.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.62
o deg ≤ 122.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.48
o deg > 122.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64
p freq ≤ 283.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.6
p freq > 283.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.5
s p cofreq ≤ 11.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.52
s p cofreq > 11.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.59
o p cofreq ≤ 11.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.56
o p cofreq > 11.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.54
s o cofreq ≤ 1.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.51
s o cofreq > 1.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.68
s min deg nbr ≤ 40.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.56
s min deg nbr > 40.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.56
s max deg nbr ≤ 304.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.52
s max deg nbr > 304.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.61
s mean deg nbr ≤ 132.5	BCE	Bernoulli	0.01	0.01	25	None	100	0.52
s mean deg nbr > 132.5	BCE	Bernoulli	0.01	0.01	25	None	50	0.6
s num s o cofreq ≤ 12.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.51
s num s o cofreq > 12.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.61
s min freq rel ≤ 35.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
s min freq rel > 35.0	BCE	Basic	0.01	0.01	25	None	250	0.56
s max freq rel	N/A							
s mean freq rel ≤ 255.6	BCE	Bernoulli	0.01	0.01	25	None	100	0.56
s mean freq rel > 255.6	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
s num rels ≤ 7.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.49
s num rels > 7.0	BCE	Bernoulli	0.01	0.01	25	None	250	0.64
o min deg nbr ≤ 40.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.58
o min deg nbr > 40.0	BCE	Basic	0.01	0.01	25	None	250	0.54
o max deg nbr ≤ 304.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.5
o max deg nbr > 304.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.65
o mean deg nbr ≤ 130.8	BCE	Bernoulli	0.01	0.01	25	None	100	0.55
o mean deg nbr > 130.8	BCE	Bernoulli	0.01	0.01	25	None	50	0.55
o num s o cofreq ≤ 15.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.48
o num s o cofreq > 15.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64
o min freq rel ≤ 42.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.59
o min freq rel > 42.0	BCE	Basic	0.01	0.01	25	None	250	0.53
o max freq rel	N/A							
o mean freq rel ≤ 272.4	BCE	Bernoulli	0.01	0.01	25	None	100	0.57
o mean freq rel > 272.4	BCE	Bernoulli	0.01	0.01	25	None	50	0.53
o num rels ≤ 8.0	BCE	Bernoulli	0.01	0.01	25	None	50	0.49
o num rels > 8.0	BCE	Bernoulli	0.01	0.01	25	None	100	0.64

Table C.20: ComplEx on UMLS: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.6 DistMult on CoDExSmall

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.12	-0.2	0.04
o deg	-0.01	0.02	-0.1
p freq	-0.07	-0.09	-0.06
s p cofreq	-0.14	-0.22	0.03
o p cofreq	0.01	0.05	-0.11
s o cofreq	-0.01	-0.02	0.01
s min deg nbr	0.03	0.06	-0.08
s max deg nbr	-0.01	-0.0	-0.08
s mean deg nbr	0.03	0.07	-0.09
s num s o cofreq	-0.12	-0.19	0.03
s min freq rel	-0.05	-0.06	-0.04
s max freq rel	0.0	0.01	-0.02
s mean freq rel	-0.04	-0.05	-0.04
s num rels	-0.01	-0.03	0.04
o min deg nbr	-0.1	-0.17	0.05
o max deg nbr	-0.11	-0.19	0.05
o mean deg nbr	-0.13	-0.22	0.07
o num s o cofreq	-0.01	0.01	-0.1
o min freq rel	-0.05	-0.06	-0.05
o max freq rel	-0.06	-0.09	-0.01
o mean freq rel	-0.06	-0.08	-0.03
o num rels	0.0	-0.0	0.03

Table C.21: DistMult on CoDExSmall: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
CE	Bernoulli	0.01	0.01	125	None	100	0.32
MRL	Bernoulli	0.01	0.01	125	0.5	100	0.21
MRL	Bernoulli	0.01	0.01	125	1	100	0.23
MRL	Bernoulli	0.01	0.01	125	2	100	0.22
BCE	Bernoulli	0.01	0.01	125	None	100	0.03
CE	Basic	0.01	0.01	125	None	100	0.32
CE	Pseudo	0.01	0.01	125	None	100	0.14
CE	Bernoulli	0.0001	0.01	125	None	100	0.21
CE	Bernoulli	1e-06	0.01	125	None	100	0.0
CE	Bernoulli	0.01	0.0001	125	None	100	0.29
CE	Bernoulli	0.01	1e-06	125	None	100	0.28
CE	Bernoulli	0.01	0.01	5	None	100	0.25
CE	Bernoulli	0.01	0.01	25	None	100	0.3
CE	Bernoulli	0.01	0.01	125	None	50	0.32
CE	Bernoulli	0.01	0.01	125	None	250	0.31

Table C.22: DistMult on CoDExSmall: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.16	0.22	0.3
Loss = BCE	0.0	0.0	0.0	0.05	0.26
Loss = CE	0.0	0.0	0.15	0.21	0.32
N. Samp = Basic	0.0	0.0	0.19	0.22	0.32
N. Samp = Bernoulli	0.0	0.0	0.2	0.23	0.32
N. Samp = Pseudo	0.0	0.0	0.04	0.13	0.18
LR = 0.01	0.0	0.06	0.21	0.24	0.32
LR = 0.0001	0.0	0.11	0.18	0.21	0.24
LR = 1e-06	0.0	0.0	0.0	0.0	0.01
Reg = 0.01	0.0	0.0	0.11	0.2	0.32
Reg = 0.0001	0.0	0.0	0.06	0.21	0.3
Reg = 1e-06	0.0	0.0	0.07	0.21	0.3
npp = 5	0.0	0.0	0.1	0.2	0.27
npp = 25	0.0	0.0	0.1	0.21	0.31
npp = 125	0.0	0.0	0.1	0.22	0.32
Mgn = None	0.0	0.0	0.03	0.19	0.32
Mgn = 0.5	0.0	0.0	0.14	0.23	0.3
Mgn = 1	0.0	0.0	0.16	0.22	0.3
Mgn = 2	0.0	0.0	0.16	0.2	0.3
Dim = 50	0.0	0.0	0.1	0.21	0.32
Dim = 100	0.0	0.0	0.1	0.21	0.32
Dim = 250	0.0	0.0	0.1	0.21	0.32

Table C.23: Distribution of MRR scores obtained when running DistMult on CoDExSmall with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of DistMult on CoDExSmall in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	CE	Bernoulli	0.01	0.01	125	None	100	0.32
s deg ≤ 18.0	CE	Basic	0.01	0.01	125	None	50	0.31
s deg > 18.0	MRL	Bernoulli	0.01	0.0001	125	0.5	250	0.35
o deg ≤ 112.0	CE	Bernoulli	0.01	0.01	125	None	100	0.28
o deg > 112.0	CE	Basic	0.01	0.01	125	None	250	0.37
p freq ≤ 4985.0	CE	Basic	0.01	0.01	125	None	100	0.36
p freq > 4985.0	MRL	Bernoulli	0.01	0.0001	125	0.5	100	0.29
s p cofreq ≤ 6.0	CE	Basic	0.01	0.01	125	None	50	0.33
s p cofreq > 6.0	MRL	Bernoulli	0.01	0.0001	125	0.5	250	0.32
o p cofreq ≤ 72.0	CE	Bernoulli	0.01	0.01	125	None	100	0.28
o p cofreq > 72.0	CE	Basic	0.01	0.01	125	None	50	0.38
s o cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	100	0.32
s o cofreq > 0.0	BCE	Bernoulli	0.01	1e-06	125	None	100	0.41
s min deg nbr ≤ 45.0	CE	Bernoulli	0.01	0.01	125	None	100	0.29
s min deg nbr > 45.0	CE	Basic	0.01	0.01	125	None	250	0.35
s max deg nbr ≤ 232.0	CE	Bernoulli	0.01	0.01	125	None	100	0.3
s max deg nbr > 232.0	CE	Basic	0.01	0.01	125	None	100	0.35
s mean deg nbr ≤ 136.8	CE	Bernoulli	0.01	0.01	125	None	100	0.29
s mean deg nbr > 136.8	CE	Basic	0.01	0.01	125	None	100	0.36
s num s o cofreq ≤ 2.0	CE	Bernoulli	0.01	0.01	125	None	100	0.31
s num s o cofreq > 2.0	MRL	Bernoulli	0.01	0.0001	125	0.5	250	0.35
s min freq rel ≤ 1648.0	CE	Basic	0.01	0.01	125	None	100	0.33
s min freq rel > 1648.0	MRL	Bernoulli	0.01	0.0001	125	0.5	100	0.31
s max freq rel ≤ 5563.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s max freq rel > 5563.0	CE	Basic	0.01	0.01	125	None	50	0.28
s mean freq rel ≤ 5270.5	CE	Basic	0.01	0.01	125	None	100	0.34
s mean freq rel > 5270.5	MRL	Bernoulli	0.01	0.0001	125	0.5	100	0.3
s num rels ≤ 2.0	CE	Bernoulli	0.01	0.01	125	None	100	0.32
s num rels > 2.0	CE	Basic	0.01	0.01	125	None	100	0.36
o min deg nbr ≤ 13.0	CE	Basic	0.01	0.01	125	None	50	0.34
o min deg nbr > 13.0	MRL	Bernoulli	0.01	0.0001	125	0.5	250	0.32
o max deg nbr ≤ 29.0	CE	Basic	0.01	0.01	125	None	50	0.31
o max deg nbr > 29.0	MRL	Bernoulli	0.01	0.0001	125	0.5	250	0.35
o mean deg nbr ≤ 19.0	CE	Basic	0.01	0.01	125	None	50	0.31
o mean deg nbr > 19.0	MRL	Bernoulli	0.01	0.0001	125	1	50	0.34
o num s o cofreq ≤ 7.0	CE	Bernoulli	0.01	0.01	125	None	100	0.28
o num s o cofreq > 7.0	CE	Basic	0.01	0.01	125	None	50	0.37
o min freq rel ≤ 4985.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
o min freq rel > 4985.0	CE	Basic	0.01	0.01	125	None	50	0.27
o max freq rel ≤ 5563.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
o max freq rel > 5563.0	CE	Basic	0.01	0.01	125	None	50	0.26
o mean freq rel ≤ 4985.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
o mean freq rel > 4985.0	MRL	Bernoulli	0.01	0.0001	125	0.5	100	0.28
o num rels ≤ 1.0	CE	Basic	0.01	0.01	125	None	50	0.32
o num rels > 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.33

Table C.24: DistMult on CoDExSmall: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.7 DistMult on DBpedia50

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	0.01	-0.01	0.03
o deg	-0.1	-0.07	-0.13
p freq	-0.11	-0.09	-0.14
s p cofreq	-0.06	-0.08	-0.05
o p cofreq	-0.15	-0.13	-0.17
s o cofreq	-0.2	-0.21	-0.2
s min deg nbr	-0.1	-0.07	-0.13
s max deg nbr	-0.1	-0.07	-0.13
s mean deg nbr	-0.1	-0.07	-0.13
s num s o cofreq	N/A	N/A	N/A
s min freq rel	-0.11	-0.09	-0.14
s max freq rel	-0.11	-0.09	-0.14
s mean freq rel	-0.11	-0.09	-0.14
s num rels	N/A	N/A	N/A
o min deg nbr	-0.08	-0.1	-0.05
o max deg nbr	0.02	0.0	0.04
o mean deg nbr	-0.04	-0.07	-0.02
o num s o cofreq	-0.05	-0.02	-0.08
o min freq rel	-0.09	-0.07	-0.12
o max freq rel	-0.04	-0.01	-0.07
o mean freq rel	-0.08	-0.05	-0.11
o num rels	0.04	0.06	0.02

Table C.25: DistMult on DBpedia50: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
MRL	Bernoulli	0.01	0.01	25	2	50	0.40
BCE	Bernoulli	0.01	0.01	25	None	50	0.04
CE	Bernoulli	0.01	0.01	25	None	50	0.38
MRL	Basic	0.01	0.01	25	2	50	0.35
MRL	Pseudo	0.01	0.01	25	2	50	0.28
MRL	Bernoulli	0.0001	0.01	25	2	50	0.17
MRL	Bernoulli	1e-06	0.01	25	2	50	0.0
MRL	Bernoulli	0.01	0.0001	25	2	50	0.25
MRL	Bernoulli	0.01	1e-06	25	2	50	0.24
MRL	Bernoulli	0.01	0.01	5	2	50	0.37
MRL	Bernoulli	0.01	0.01	125	2	50	0.37
MRL	Bernoulli	0.01	0.01	25	0.5	50	0.36
MRL	Bernoulli	0.01	0.01	25	1	50	0.38
MRL	Bernoulli	0.01	0.01	25	2	100	0.38
MRL	Bernoulli	0.01	0.01	25	2	250	0.35

Table C.26: DistMult on DBpedia50: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.15	0.2	0.4
Loss = BCE	0.0	0.0	0.0	0.02	0.21
Loss = CE	0.0	0.0	0.14	0.25	0.39
N. Samp = Basic	0.0	0.0	0.14	0.19	0.37
N. Samp = Bernoulli	0.0	0.0	0.14	0.2	0.4
N. Samp = Pseudo	0.0	0.0	0.12	0.16	0.3
LR = 0.01	0.0	0.17	0.23	0.29	0.4
LR = 0.0001	0.0	0.12	0.14	0.16	0.2
LR = 1e-06	0.0	0.0	0.0	0.0	0.01
Reg = 0.01	0.0	0.0	0.13	0.26	0.4
Reg = 0.0001	0.0	0.0	0.13	0.18	0.34
Reg = 1e-06	0.0	0.0	0.13	0.17	0.36
npp = 5	0.0	0.0	0.12	0.16	0.37
npp = 25	0.0	0.0	0.14	0.19	0.4
npp = 125	0.0	0.0	0.14	0.2	0.39
Mgn = None	0.0	0.0	0.01	0.16	0.39
Mgn = 0.5	0.0	0.0	0.14	0.18	0.37
Mgn = 1	0.0	0.0	0.15	0.2	0.39
Mgn = 2	0.0	0.0	0.15	0.21	0.4
Dim = 50	0.0	0.0	0.13	0.18	0.4
Dim = 100	0.0	0.0	0.13	0.18	0.38
Dim = 250	0.0	0.0	0.13	0.18	0.39

Table C.27: Distribution of MRR scores obtained when running DistMult on DBpedia50 with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of DistMult on DBpedia50 in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	MRL	Bernoulli	0.01	0.01	25	2	50	0.4
s deg ≤ 2.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.35
s deg > 2.0	CE	Bernoulli	0.01	0.01	125	None	250	0.53
o deg ≤ 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.47
o deg > 15.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.34
p freq ≤ 458.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.43
p freq > 458.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.39
s p cofreq ≤ 0.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.43
s p cofreq > 0.0	MRL	Bernoulli	0.01	0.01	25	2	50	0.35
o p cofreq ≤ 7.0	CE	Basic	0.01	0.01	25	None	50	0.49
o p cofreq > 7.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.31
s o cofreq ≤ 0.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.29
s o cofreq > 0.0	CE	Bernoulli	0.01	0.01	125	None	250	0.84
s min deg nbr ≤ 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.47
s min deg nbr > 15.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.34
s max deg nbr ≤ 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.47
s max deg nbr > 15.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.34
s mean deg nbr ≤ 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.47
s mean deg nbr > 15.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.34
s num neighbours	N/A							
s min freq rel ≤ 458.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.43
s min freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.39
s max freq rel ≤ 458.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.43
s max freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.39
s mean freq rel ≤ 458.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.43
s mean freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.39
s num rels	N/A							
o min deg nbr ≤ 1.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.35
o min deg nbr > 1.0	CE	Bernoulli	0.01	0.01	125	None	250	0.46
o max deg nbr ≤ 2.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.37
o max deg nbr > 2.0	CE	Bernoulli	0.01	0.01	25	None	50	0.46
o mean deg nbr ≤ 2.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.36
o mean deg nbr > 2.0	CE	Bernoulli	0.01	0.01	125	None	250	0.52
o num s o cofreq ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	250	0.43
o num s o cofreq > 1.0	CE	Bernoulli	0.01	0.01	25	None	50	0.35
o min freq rel ≤ 443.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.43
o min freq rel > 443.0	MRL	Bernoulli	0.01	0.01	25	2	100	0.39
o max freq rel ≤ 458.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.44
o max freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.39
o mean freq rel ≤ 458.0	MRL	Bernoulli	0.01	0.01	25	1	250	0.44
o mean freq rel > 458.0	MRL	Bernoulli	0.01	0.01	125	1	100	0.39
o num rels ≤ 1.0	MRL	Bernoulli	0.01	0.01	25	2	50	0.4
o num rels > 1.0	CE	Bernoulli	0.01	0.01	25	None	50	0.41

Table C.28: DistMult on DBpedia50: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.8 DistMult on Kinships

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.0	0.0	-0.01
o deg	0.03	0.01	0.04
p freq	0.24	0.23	0.25
s p cofreq	0.05	0.12	-0.02
o p cofreq	0.08	-0.0	0.14
s o cofreq	N/A	N/A	N/A
s min deg nbr	-0.03	-0.05	-0.02
s max deg nbr	-0.01	-0.01	-0.01
s mean deg nbr	0.05	0.05	0.06
s num s o cofreq	0.0	-0.01	0.01
s min freq rel	0.06	0.09	0.03
s max freq rel	0.05	0.05	0.06
s mean freq rel	0.09	0.1	0.07
s num rels	-0.06	-0.07	-0.04
o min deg nbr	0.01	0.03	-0.01
o max deg nbr	0.01	0.01	0.02
o mean deg nbr	-0.02	-0.03	-0.01
o num s o cofreq	0.02	0.03	0.01
o min freq rel	0.05	0.03	0.06
o max freq rel	0.05	0.06	0.05
o mean freq rel	0.09	0.1	0.09
o num rels	0.0	0.0	-0.0

Table C.29: DistMult on Kinships: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
CE	Bernoulli	0.01	0.01	25	None	100	0.48
MRL	Bernoulli	0.01	0.01	25	0.5	100	0.44
MRL	Bernoulli	0.01	0.01	25	1	100	0.45
MRL	Bernoulli	0.01	0.01	25	2	100	0.43
BCE	Bernoulli	0.01	0.01	25	None	100	0.21
CE	Basic	0.01	0.01	25	None	100	0.45
CE	Pseudo	0.01	0.01	25	None	100	0.42
CE	Bernoulli	0.0001	0.01	25	None	100	0.24
CE	Bernoulli	1e-06	0.01	25	None	100	0.06
CE	Bernoulli	0.01	0.0001	25	None	100	0.45
CE	Bernoulli	0.01	1e-06	25	None	100	0.46
CE	Bernoulli	0.01	0.01	5	None	100	0.46
CE	Bernoulli	0.01	0.01	125	None	100	0.46
CE	Bernoulli	0.01	0.01	25	None	50	0.47
CE	Bernoulli	0.01	0.01	25	None	250	0.47

Table C.30: DistMult on Kinships: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.05	0.06	0.24	0.39	0.46
Loss = BCE	0.02	0.05	0.05	0.17	0.44
Loss = CE	0.05	0.06	0.24	0.4	0.48
N. Samp = Basic	0.02	0.05	0.23	0.43	0.48
N. Samp = Bernoulli	0.02	0.05	0.23	0.43	0.48
N. Samp = Pseudo	0.02	0.05	0.23	0.36	0.42
LR = 0.01	0.02	0.38	0.43	0.44	0.48
LR = 0.0001	0.04	0.21	0.24	0.25	0.29
LR = 1e-06	0.04	0.05	0.05	0.06	0.06
Reg = 0.01	0.02	0.05	0.23	0.38	0.48
Reg = 0.0001	0.04	0.05	0.23	0.37	0.47
Reg = 1e-06	0.04	0.05	0.23	0.38	0.46
npp = 5	0.04	0.05	0.23	0.38	0.46
npp = 25	0.02	0.05	0.24	0.38	0.48
npp = 125	0.02	0.05	0.23	0.37	0.48
Mgn = None	0.02	0.05	0.06	0.31	0.48
Mgn = 0.5	0.05	0.06	0.24	0.39	0.45
Mgn = 1	0.05	0.06	0.24	0.39	0.46
Mgn = 2	0.05	0.06	0.24	0.39	0.46
Dim = 50	0.02	0.05	0.23	0.38	0.47
Dim = 100	0.02	0.05	0.23	0.38	0.48
Dim = 250	0.02	0.05	0.23	0.37	0.48

Table C.31: Distribution of MRR scores obtained when running DistMult on Kinships with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of DistMult on Kinships in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	CE	Bernoulli	0.01	0.01	25	None	100	0.48
s deg ≤ 164.0	CE	Basic	0.01	0.01	125	None	100	0.48
s deg > 164.0	CE	Bernoulli	0.01	0.01	25	None	100	0.5
o deg ≤ 164.0	CE	Bernoulli	0.01	0.01	25	None	100	0.5
o deg > 164.0	CE	Bernoulli	0.01	0.01	125	None	250	0.47
p freq ≤ 404.0	CE	Bernoulli	0.01	0.01	25	None	100	0.57
p freq > 404.0	MRL	Basic	0.01	0.01	125	1	250	0.38
s p cofreq ≤ 6.0	CE	Basic	0.01	0.01	125	None	100	0.51
s p cofreq > 6.0	CE	Basic	0.01	0.01	25	None	50	0.46
o p cofreq ≤ 8.0	CE	Basic	0.01	0.0001	125	None	250	0.51
o p cofreq > 8.0	CE	Bernoulli	0.01	0.01	25	None	100	0.45
s o cofreq	N/A							
s min deg nbr ≤ 153.0	CE	Bernoulli	0.01	0.01	25	None	100	0.48
s min deg nbr > 153.0	CE	Bernoulli	0.01	0.01	25	None	100	0.48
s max deg nbr ≤ 172.0	CE	Basic	0.01	0.01	125	None	100	0.48
s max deg nbr > 172.0	CE	Bernoulli	0.01	0.01	25	None	100	0.48
s mean deg nbr ≤ 163.8	CE	Basic	0.01	0.01	125	None	100	0.5
s mean deg nbr > 163.8	CE	Bernoulli	0.01	0.01	125	None	250	0.47
s num s o cofreq ≤ 20.0	CE	Bernoulli	0.01	0.01	25	None	100	0.49
s num s o cofreq > 20.0	CE	Basic	0.01	0.01	125	None	100	0.48
s min freq rel ≤ 183.0	CE	Bernoulli	0.01	0.01	25	None	100	0.49
s min freq rel > 183.0	CE	Bernoulli	0.01	0.01	25	None	100	0.45
s max freq rel	N/A							
s mean freq rel ≤ 473.5	CE	Bernoulli	0.01	0.01	25	None	100	0.51
s mean freq rel > 473.5	CE	Bernoulli	0.01	0.01	125	None	250	0.45
s num rels ≤ 11.0	CE	Bernoulli	0.01	0.01	125	None	250	0.48
s num rels > 11.0	CE	Basic	0.01	0.01	125	None	100	0.49
o min deg nbr ≤ 153.0	CE	Basic	0.01	0.01	125	None	250	0.5
o min deg nbr > 153.0	CE	Basic	0.01	0.01	25	None	250	0.46
o max deg nbr ≤ 172.0	CE	Bernoulli	0.01	0.01	25	None	100	0.49
o max deg nbr > 172.0	CE	Basic	0.01	0.01	125	None	100	0.48
o mean deg nbr ≤ 163.8	CE	Basic	0.01	0.01	125	None	100	0.47
o mean deg nbr > 163.8	CE	Bernoulli	0.01	0.01	25	None	100	0.5
o num s o cofreq ≤ 20.0	CE	Bernoulli	0.01	0.01	125	None	250	0.49
o num s o cofreq > 20.0	CE	Basic	0.01	0.01	125	None	100	0.48
o min freq rel ≤ 183.0	CE	Bernoulli	0.01	0.01	25	None	100	0.5
o min freq rel > 183.0	CE	Bernoulli	0.01	0.01	25	None	50	0.46
o max freq rel	N/A							
o mean freq rel ≤ 472.9	CE	Bernoulli	0.01	0.01	25	None	100	0.49
o mean freq rel > 472.9	CE	Basic	0.01	0.01	25	None	250	0.47
o num rels ≤ 11.0	CE	Basic	0.01	0.01	125	None	100	0.48
o num rels > 11.0	CE	Bernoulli	0.01	0.01	125	None	250	0.48

Table C.32: DistMult on Kinships: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.9 DistMult on OpenEA

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.05	-0.07	-0.02
o deg	-0.03	-0.02	-0.03
p freq	0.11	0.15	0.06
s p cofreq	-0.07	-0.08	-0.06
o p cofreq	-0.02	0.0	-0.04
s o cofreq	-0.19	-0.22	-0.16
s min deg nbr	-0.01	-0.01	-0.01
s max deg nbr	-0.03	-0.03	-0.03
s mean deg nbr	-0.02	-0.02	-0.02
s num s o cofreq	-0.02	-0.03	-0.01
s min freq rel	0.09	0.12	0.04
s max freq rel	0.09	0.13	0.04
s mean freq rel	0.1	0.14	0.04
s num rels	-0.04	-0.06	-0.02
o min deg nbr	-0.07	-0.1	-0.04
o max deg nbr	0.02	0.01	0.05
o mean deg nbr	-0.0	-0.02	0.03
o num s o cofreq	-0.05	-0.05	-0.06
o min freq rel	0.12	0.16	0.07
o max freq rel	0.08	0.11	0.03
o mean freq rel	0.11	0.15	0.06
o num rels	-0.05	-0.07	-0.03

Table C.33: DistMult on OpenEA: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
CE	Bernoulli	0.01	0.01	125	None	50	0.33
MRL	Bernoulli	0.01	0.01	125	0.5	50	0.3
MRL	Bernoulli	0.01	0.01	125	1	50	0.3
MRL	Bernoulli	0.01	0.01	125	2	50	0.3
BCE	Bernoulli	0.01	0.01	125	None	50	0.0
CE	Basic	0.01	0.01	125	None	50	0.31
CE	Pseudo	0.01	0.01	125	None	50	0.26
CE	Bernoulli	0.0001	0.01	125	None	50	0.06
CE	Bernoulli	1e-06	0.01	125	None	50	0.0
CE	Bernoulli	0.01	0.0001	125	None	50	0.25
CE	Bernoulli	0.01	1e-06	125	None	50	0.26
CE	Bernoulli	0.01	0.01	5	None	50	0.27
CE	Bernoulli	0.01	0.01	25	None	50	0.31
CE	Bernoulli	0.01	0.01	125	None	100	0.32
CE	Bernoulli	0.01	0.01	125	None	250	0.32

Table C.34: DistMult on OpenEA: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.06	0.16	0.31
Loss = BCE	0.0	0.0	0.0	0.01	0.19
Loss = CE	0.0	0.0	0.06	0.22	0.33
N. Samp = Basic	0.0	0.0	0.06	0.15	0.32
N. Samp = Bernoulli	0.0	0.0	0.05	0.16	0.33
N. Samp = Pseudo	0.0	0.0	0.05	0.13	0.28
LR = 0.01	0.0	0.15	0.21	0.26	0.33
LR = 0.0001	0.0	0.05	0.06	0.06	0.09
LR = 1e-06	0.0	0.0	0.0	0.0	0.0
Reg = 0.01	0.0	0.0	0.06	0.25	0.33
Reg = 0.0001	0.0	0.0	0.05	0.15	0.27
Reg = 1e-06	0.0	0.0	0.05	0.14	0.27
npp = 5	0.0	0.0	0.05	0.13	0.29
npp = 25	0.0	0.0	0.06	0.17	0.31
npp = 125	0.0	0.0	0.06	0.18	0.33
Mgn = None	0.0	0.0	0.0	0.08	0.33
Mgn = 0.5	0.0	0.0	0.06	0.16	0.3
Mgn = 1	0.0	0.0	0.06	0.16	0.31
Mgn = 2	0.0	0.0	0.06	0.17	0.31
Dim = 50	0.0	0.0	0.05	0.15	0.33
Dim = 100	0.0	0.0	0.05	0.15	0.32
Dim = 250	0.0	0.0	0.05	0.15	0.32

Table C.35: Distribution of MRR scores obtained when running DistMult on OpenEA with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of DistMult on OpenEA in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	CE	Bernoulli	0.01	0.01	125	None	50	0.33
s deg ≤ 3.0	CE	Bernoulli	0.01	0.01	125	None	50	0.29
s deg > 3.0	CE	Bernoulli	0.01	0.01	125	None	100	0.39
o deg ≤ 12.0	CE	Bernoulli	0.01	0.01	125	None	50	0.3
o deg > 12.0	CE	Bernoulli	0.01	0.01	125	None	50	0.36
p freq ≤ 1242.0	CE	Bernoulli	0.01	0.01	125	None	50	0.39
p freq > 1242.0	CE	Bernoulli	0.01	0.01	25	None	250	0.27
s p cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	50	0.38
s p cofreq > 0.0	CE	Bernoulli	0.01	0.01	125	None	100	0.23
o p cofreq ≤ 5.0	CE	Basic	0.01	0.01	125	None	250	0.39
o p cofreq > 5.0	CE	Bernoulli	0.01	0.01	125	None	50	0.27
s o cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	50	0.2
s o cofreq > 0.0	CE	Basic	0.01	0.01	125	None	250	0.71
s min deg nbr ≤ 9.0	CE	Bernoulli	0.01	0.01	125	None	50	0.31
s min deg nbr > 9.0	CE	Bernoulli	0.01	0.01	125	None	50	0.36
s max deg nbr ≤ 14.0	CE	Bernoulli	0.01	0.01	125	None	50	0.31
s max deg nbr > 14.0	CE	Bernoulli	0.01	0.01	125	None	50	0.35
s mean deg nbr ≤ 12.2	CE	Bernoulli	0.01	0.01	125	None	50	0.31
s mean deg nbr > 12.2	CE	Bernoulli	0.01	0.01	125	None	50	0.35
s num s o cofreq ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	50	0.34
s num s o cofreq > 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.32
s min freq rel ≤ 1030.0	CE	Bernoulli	0.01	0.01	125	None	50	0.39
s min freq rel > 1030.0	CE	Bernoulli	0.01	0.01	25	None	250	0.26
s max freq rel ≤ 1598.0	CE	Bernoulli	0.01	0.01	125	None	50	0.4
s max freq rel > 1598.0	CE	Bernoulli	0.01	0.01	125	None	50	0.24
s mean freq rel ≤ 1311.5	CE	Bernoulli	0.01	0.01	125	None	50	0.4
s mean freq rel > 1311.5	CE	Bernoulli	0.01	0.01	125	None	50	0.26
s num rels ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	50	0.32
s num rels > 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.37
o min deg nbr ≤ 2.0	CE	Bernoulli	0.01	0.01	125	None	50	0.28
o min deg nbr > 2.0	CE	Bernoulli	0.01	0.01	125	None	100	0.42
o max deg nbr ≤ 5.0	CE	Bernoulli	0.01	0.01	125	None	50	0.3
o max deg nbr > 5.0	CE	Bernoulli	0.01	0.01	125	None	50	0.37
o mean deg nbr ≤ 3.5	CE	Bernoulli	0.01	0.01	125	None	50	0.28
o mean deg nbr > 3.5	CE	Bernoulli	0.01	0.01	125	None	50	0.39
o num s o cofreq ≤ 3.0	CE	Bernoulli	0.01	0.01	125	None	50	0.33
o num s o cofreq > 3.0	CE	Bernoulli	0.01	0.01	125	None	50	0.32
o min freq rel ≤ 741.0	CE	Bernoulli	0.01	0.01	125	None	50	0.42
o min freq rel > 741.0	CE	Bernoulli	0.01	0.01	125	None	50	0.23
o max freq rel ≤ 1598.0	CE	Bernoulli	0.01	0.01	125	None	50	0.37
o max freq rel > 1598.0	CE	Bernoulli	0.01	0.01	125	None	50	0.28
o mean freq rel ≤ 1242.0	CE	Bernoulli	0.01	0.01	125	None	50	0.39
o mean freq rel > 1242.0	CE	Bernoulli	0.01	0.01	125	None	50	0.26
o num rels ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	50	0.26
o num rels > 1.0	CE	Bernoulli	0.01	0.01	125	None	50	0.43

Table C.36: DistMult on OpenEA: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.10 DistMult on UMLS

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.06	-0.21	0.12
o deg	-0.01	0.13	-0.18
p freq	0.27	0.34	0.2
s p cofreq	-0.09	-0.19	0.02
o p cofreq	0.12	0.26	-0.04
s o cofreq	-0.06	-0.08	-0.04
s min deg nbr	0.12	0.19	0.03
s max deg nbr	-0.05	-0.05	-0.06
s mean deg nbr	-0.01	0.06	-0.08
s num s o cofreq	-0.05	-0.19	0.11
s min freq rel	0.09	0.1	0.09
s max freq rel	0.04	0.04	0.04
s mean freq rel	0.11	0.17	0.05
s num rels	-0.07	-0.17	0.05
o min deg nbr	-0.03	-0.13	0.09
o max deg nbr	-0.03	-0.02	-0.05
o mean deg nbr	-0.03	-0.09	0.04
o num s o cofreq	-0.0	0.13	-0.16
o min freq rel	0.07	0.01	0.15
o max freq rel	0.01	-0.0	0.04
o mean freq rel	0.06	-0.04	0.18
o num rels	-0.01	0.09	-0.13

Table C.37: DistMult on UMLS: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
MRL	Bernoulli	0.01	0.01	5	0.5	100	0.57
BCE	Bernoulli	0.01	0.01	5	None	100	0.27
CE	Bernoulli	0.01	0.01	5	None	100	0.54
MRL	Basic	0.01	0.01	5	0.5	100	0.55
MRL	Pseudo	0.01	0.01	5	0.5	100	0.18
MRL	Bernoulli	0.0001	0.01	5	0.5	100	0.12
MRL	Bernoulli	1e-06	0.01	5	0.5	100	0.05
MRL	Bernoulli	0.01	0.0001	5	0.5	100	0.55
MRL	Bernoulli	0.01	1e-06	5	0.5	100	0.54
MRL	Bernoulli	0.01	0.01	25	0.5	100	0.56
MRL	Bernoulli	0.01	0.01	125	0.5	100	0.56
MRL	Bernoulli	0.01	0.01	5	1	100	0.56
MRL	Bernoulli	0.01	0.01	5	2	100	0.55
MRL	Bernoulli	0.01	0.01	5	0.5	50	0.56
MRL	Bernoulli	0.01	0.01	5	0.5	250	0.56

Table C.38: DistMult on UMLS: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.03	0.05	0.08	0.19	0.57
Loss = BCE	0.02	0.04	0.05	0.07	0.54
Loss = CE	0.03	0.05	0.09	0.23	0.57
N. Samp = Basic	0.02	0.05	0.1	0.51	0.56
N. Samp = Bernoulli	0.02	0.05	0.1	0.54	0.57
N. Samp = Pseudo	0.02	0.04	0.05	0.06	0.28
LR = 0.01	0.02	0.14	0.52	0.55	0.57
LR = 0.0001	0.02	0.04	0.08	0.11	0.18
LR = 1e-06	0.04	0.04	0.05	0.05	0.06
Reg = 0.01	0.02	0.05	0.06	0.21	0.57
Reg = 0.0001	0.02	0.05	0.06	0.14	0.57
Reg = 1e-06	0.02	0.05	0.06	0.13	0.57
npp = 5	0.02	0.05	0.06	0.18	0.57
npp = 25	0.02	0.05	0.06	0.17	0.57
npp = 125	0.02	0.05	0.06	0.15	0.57
Mgn = None	0.02	0.04	0.05	0.14	0.57
Mgn = 0.5	0.03	0.05	0.06	0.17	0.57
Mgn = 1	0.03	0.05	0.07	0.19	0.57
Mgn = 2	0.04	0.05	0.1	0.24	0.57
Dim = 50	0.02	0.05	0.05	0.15	0.57
Dim = 100	0.02	0.05	0.06	0.16	0.57
Dim = 250	0.02	0.05	0.06	0.16	0.57

Table C.39: Distribution of MRR scores obtained when running DistMult on UMLS with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of DistMult on UMLS in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.57
s deg ≤ 84.0	MRL	Bernoulli	0.01	0.01	125	2	250	0.59
s deg > 84.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.57
o deg ≤ 122.0	CE	Bernoulli	0.01	0.01	5	None	250	0.57
o deg > 122.0	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.59
p freq ≤ 283.0	MRL	Bernoulli	0.01	0.01	5	1	100	0.7
p freq > 283.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.44
s p cofreq ≤ 11.0	MRL	Bernoulli	0.01	0.01	5	1	250	0.59
s p cofreq > 11.0	MRL	Basic	0.01	0.01	125	0.5	250	0.57
o p cofreq ≤ 11.0	MRL	Bernoulli	0.01	1e-06	125	2	250	0.64
o p cofreq > 11.0	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.53
s o cofreq ≤ 1.0	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.56
s o cofreq > 1.0	MRL	Bernoulli	0.01	0.0001	125	0.5	250	0.62
s min deg nbr ≤ 40.0	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.61
s min deg nbr > 40.0	MRL	Bernoulli	0.01	0.01	125	2	250	0.55
s max deg nbr ≤ 304.0	MRL	Bernoulli	0.01	0.01	25	1	100	0.58
s max deg nbr > 304.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.57
s mean deg nbr ≤ 132.5	CE	Bernoulli	0.01	0.01	5	None	250	0.57
s mean deg nbr > 132.5	MRL	Bernoulli	0.01	0.01	5	1	250	0.58
s num s o cofreq ≤ 12.0	CE	Basic	0.01	0.0001	125	None	100	0.59
s num s o cofreq > 12.0	MRL	Bernoulli	0.01	0.01	5	0.5	250	0.56
s min freq rel ≤ 35.0	MRL	Bernoulli	0.01	1e-06	125	2	250	0.62
s min freq rel > 35.0	MRL	Bernoulli	0.01	0.01	5	1	250	0.54
s max freq rel	N/A							
s mean freq rel ≤ 255.6	MRL	Bernoulli	0.01	0.01	25	1	100	0.63
s mean freq rel > 255.6	MRL	Bernoulli	0.01	0.01	5	1	250	0.54
s num rels ≤ 7.0	MRL	Bernoulli	0.01	0.01	125	2	250	0.59
s num rels > 7.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.57
o min deg nbr ≤ 40.0	MRL	Bernoulli	0.01	1e-06	25	2	250	0.59
o min deg nbr > 40.0	CE	Basic	0.01	0.01	25	None	250	0.57
o max deg nbr ≤ 304.0	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.56
o max deg nbr > 304.0	MRL	Basic	0.01	0.01	25	0.5	250	0.61
o mean deg nbr ≤ 130.8	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.6
o mean deg nbr > 130.8	MRL	Basic	0.01	0.01	25	0.5	250	0.56
o num s o cofreq ≤ 15.0	MRL	Bernoulli	0.01	0.01	25	1	100	0.57
o num s o cofreq > 15.0	MRL	Bernoulli	0.01	0.01	5	0.5	100	0.58
o min freq rel ≤ 42.0	MRL	Bernoulli	0.01	0.01	125	2	250	0.62
o min freq rel > 42.0	MRL	Bernoulli	0.01	0.01	125	2	100	0.52
o max freq rel	N/A							
o mean freq rel ≤ 272.4	MRL	Bernoulli	0.01	1e-06	25	2	250	0.61
o mean freq rel > 272.4	CE	Bernoulli	0.01	0.01	5	None	250	0.54
o num rels ≤ 8.0	CE	Bernoulli	0.01	0.01	5	None	250	0.57
o num rels > 8.0	MRL	Bernoulli	0.01	1e-06	25	2	100	0.6

Table C.40: DistMult on UMLS: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.11 TransE on CoDExSmall

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.14	-0.22	0.07
o deg	-0.02	-0.0	-0.16
p freq	-0.12	-0.18	-0.04
s p cofreq	-0.16	-0.25	0.1
o p cofreq	-0.03	-0.02	-0.2
s o cofreq	0.0	-0.0	0.05
s min deg nbr	0.01	0.04	-0.12
s max deg nbr	-0.03	-0.03	-0.12
s mean deg nbr	0.01	0.03	-0.14
s num s o cofreq	-0.13	-0.21	0.07
s min freq rel	-0.08	-0.12	-0.03
s max freq rel	-0.03	-0.04	-0.05
s mean freq rel	-0.08	-0.11	-0.04
s num rels	-0.01	-0.02	0.03
o min deg nbr	-0.1	-0.18	0.11
o max deg nbr	-0.12	-0.19	0.08
o mean deg nbr	-0.14	-0.23	0.12
o num s o cofreq	-0.03	-0.02	-0.16
o min freq rel	-0.09	-0.13	-0.05
o max freq rel	-0.1	-0.15	0.01
o mean freq rel	-0.1	-0.14	-0.02
o num rels	0.02	0.03	0.06

Table C.41: TransE on CoDExSmall: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
MRL	Bernoulli	0.01	1e-06	125	2	50	0.27
BCE	Bernoulli	0.01	1e-06	125	None	50	0.19
CE	Bernoulli	0.01	1e-06	125	None	50	0.26
MRL	Basic	0.01	1e-06	125	2	50	0.23
MRL	Pseudo	0.01	1e-06	125	2	50	0.04
MRL	Bernoulli	0.0001	1e-06	125	2	50	0.19
MRL	Bernoulli	1e-06	1e-06	125	2	50	0.0
MRL	Bernoulli	0.01	0.01	125	2	50	0.26
MRL	Bernoulli	0.01	0.0001	125	2	50	0.27
MRL	Bernoulli	0.01	1e-06	5	2	50	0.19
MRL	Bernoulli	0.01	1e-06	25	2	50	0.25
MRL	Bernoulli	0.01	1e-06	125	0.5	50	0.22
MRL	Bernoulli	0.01	1e-06	125	1	50	0.25
MRL	Bernoulli	0.01	1e-06	125	2	100	0.27
MRL	Bernoulli	0.01	1e-06	125	2	250	0.27

Table C.42: TransE on CoDExSmall: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.02	0.17	0.27
Loss = BCE	0.0	0.0	0.13	0.16	0.19
Loss = CE	0.0	0.0	0.02	0.21	0.27
N. Samp = Basic	0.0	0.0	0.14	0.19	0.27
N. Samp = Bernoulli	0.0	0.0	0.15	0.19	0.27
N. Samp = Pseudo	0.0	0.0	0.01	0.02	0.19
LR = 0.01	0.01	0.04	0.17	0.21	0.27
LR = 0.0001	0.01	0.02	0.15	0.17	0.22
LR = 1e-06	0.0	0.0	0.0	0.0	0.01
Reg = 0.01	0.0	0.0	0.03	0.17	0.27
Reg = 0.0001	0.0	0.0	0.04	0.18	0.27
Reg = 1e-06	0.0	0.0	0.04	0.18	0.27
npp = 5	0.0	0.0	0.03	0.16	0.21
npp = 25	0.0	0.0	0.04	0.18	0.26
npp = 125	0.0	0.0	0.03	0.19	0.27
Mgn = None	0.0	0.0	0.11	0.18	0.27
Mgn = 0.5	0.0	0.0	0.02	0.14	0.23
Mgn = 1	0.0	0.0	0.02	0.16	0.26
Mgn = 2	0.0	0.0	0.04	0.19	0.27
Dim = 50	0.0	0.0	0.03	0.17	0.27
Dim = 100	0.0	0.0	0.03	0.17	0.27
Dim = 250	0.0	0.0	0.03	0.17	0.27

Table C.43: Distribution of MRR scores obtained when running TransE on CoDExSmall with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of TransE on CoDExSmall in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	MRL	Bernoulli	0.01	1e-06	125	2	50	0.27
s deg ≤ 18.0	CE	Bernoulli	0.01	0.01	125	None	50	0.29
s deg > 18.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.26
o deg ≤ 112.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.22
o deg > 112.0	MRL	Bernoulli	0.01	0.0001	125	2	250	0.34
p freq ≤ 4985.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.34
p freq > 4985.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.22
s p cofreq ≤ 6.0	CE	Bernoulli	0.01	0.01	125	None	50	0.3
s p cofreq > 6.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.26
o p cofreq ≤ 72.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.22
o p cofreq > 72.0	MRL	Bernoulli	0.01	1e-06	125	2	250	0.35
s o cofreq ≤ 0.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.27
s o cofreq > 0.0	MRL	Bernoulli	0.01	0.01	125	1	250	0.38
s min deg nbr ≤ 45.0	CE	Bernoulli	0.01	0.01	125	None	50	0.24
s min deg nbr > 45.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.32
s max deg nbr ≤ 232.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.25
s max deg nbr > 232.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.31
s mean deg nbr ≤ 136.8	CE	Bernoulli	0.01	0.01	125	None	50	0.24
s mean deg nbr > 136.8	MRL	Bernoulli	0.01	0.0001	125	2	250	0.32
s num s o cofreq ≤ 2.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.28
s num s o cofreq > 2.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.27
s min freq rel ≤ 1648.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.29
s min freq rel > 1648.0	MRL	Bernoulli	0.01	0.0001	125	2	100	0.26
s max freq rel ≤ 5563.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.28
s max freq rel > 5563.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.27
s mean freq rel ≤ 5270.5	CE	Bernoulli	0.01	0.0001	125	None	100	0.3
s mean freq rel > 5270.5	MRL	Bernoulli	0.01	1e-06	125	2	50	0.26
s num rels ≤ 2.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.28
s num rels > 2.0	CE	Basic	0.01	0.0001	125	None	100	0.28
o min deg nbr ≤ 13.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.32
o min deg nbr > 13.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.24
o max deg nbr ≤ 29.0	CE	Bernoulli	0.01	0.01	125	None	50	0.28
o max deg nbr > 29.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.27
o mean deg nbr ≤ 19.0	CE	Bernoulli	0.01	0.01	125	None	50	0.29
o mean deg nbr > 19.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.27
o num s o cofreq ≤ 7.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.23
o num s o cofreq > 7.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.34
o min freq rel ≤ 4985.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.3
o min freq rel > 4985.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.24
o max freq rel ≤ 5563.0	MRL	Bernoulli	0.01	0.0001	125	2	250	0.29
o max freq rel > 5563.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.25
o mean freq rel ≤ 4985.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.32
o mean freq rel > 4985.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.23
o num rels ≤ 1.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.29
o num rels > 1.0	MRL	Bernoulli	0.01	0.0001	125	2	100	0.24

Table C.44: TransE on CoDExSmall: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.12 TransE on DBpedia50

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.03	-0.05	-0.01
o deg	-0.08	-0.07	-0.12
p freq	-0.06	-0.11	0.05
s p cofreq	-0.07	-0.09	-0.02
o p cofreq	-0.08	-0.07	-0.1
s o cofreq	-0.15	-0.19	-0.1
s min deg nbr	-0.08	-0.07	-0.12
s max deg nbr	-0.08	-0.07	-0.12
s mean deg nbr	-0.08	-0.07	-0.12
s num s o cofreq	N/A	N/A	N/A
s min freq rel	-0.06	-0.11	0.05
s max freq rel	-0.06	-0.11	0.05
s mean freq rel	-0.06	-0.11	0.05
s num rels	N/A	N/A	N/A
o min deg nbr	-0.08	-0.12	0.01
o max deg nbr	-0.04	-0.04	-0.03
o mean deg nbr	-0.06	-0.1	-0.01
o num s o cofreq	-0.08	-0.08	-0.1
o min freq rel	-0.04	-0.09	0.06
o max freq rel	-0.04	-0.08	0.02
o mean freq rel	-0.04	-0.09	0.05
o num rels	-0.03	-0.02	-0.05

Table C.45: TransE on DBpedia50: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
CE	Bernoulli	0.01	0.01	125	None	100	0.33
MRL	Bernoulli	0.01	0.01	125	0.5	100	0.09
MRL	Bernoulli	0.01	0.01	125	1	100	0.14
MRL	Bernoulli	0.01	0.01	125	2	100	0.24
BCE	Bernoulli	0.01	0.01	125	None	100	0.1
CE	Basic	0.01	0.01	125	None	100	0.3
CE	Pseudo	0.01	0.01	125	None	100	0.11
CE	Bernoulli	0.0001	0.01	125	None	100	0.2
CE	Bernoulli	1e-06	0.01	125	None	100	0.0
CE	Bernoulli	0.01	0.0001	125	None	100	0.31
CE	Bernoulli	0.01	1e-06	125	None	100	0.29
CE	Bernoulli	0.01	0.01	5	None	100	0.21
CE	Bernoulli	0.01	0.01	25	None	100	0.29
CE	Bernoulli	0.01	0.01	125	None	50	0.31
CE	Bernoulli	0.01	0.01	125	None	250	0.3

Table C.46: TransE on DBpedia50: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.0	0.0	0.03	0.08	0.27
Loss = BCE	0.0	0.0	0.12	0.16	0.25
Loss = CE	0.0	0.0	0.1	0.2	0.33
N. Samp = Basic	0.0	0.0	0.06	0.14	0.32
N. Samp = Bernoulli	0.0	0.0	0.08	0.15	0.33
N. Samp = Pseudo	0.0	0.0	0.03	0.08	0.17
LR = 0.01	0.02	0.08	0.1	0.17	0.33
LR = 0.0001	0.0	0.02	0.07	0.15	0.22
LR = 1e-06	0.0	0.0	0.0	0.0	0.0
Reg = 0.01	0.0	0.0	0.05	0.11	0.33
Reg = 0.0001	0.0	0.0	0.05	0.13	0.32
Reg = 1e-06	0.0	0.0	0.05	0.13	0.31
npp = 5	0.0	0.0	0.04	0.1	0.28
npp = 25	0.0	0.0	0.06	0.12	0.31
npp = 125	0.0	0.0	0.05	0.13	0.33
Mgn = None	0.0	0.0	0.11	0.17	0.33
Mgn = 0.5	0.0	0.0	0.01	0.05	0.13
Mgn = 1	0.0	0.0	0.02	0.07	0.18
Mgn = 2	0.0	0.0	0.07	0.11	0.27
Dim = 50	0.0	0.0	0.05	0.11	0.31
Dim = 100	0.0	0.0	0.05	0.12	0.33
Dim = 250	0.0	0.0	0.05	0.12	0.32

Table C.47: Distribution of MRR scores obtained when running TransE on DBpedia50 with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of TransE on DBpedia50 in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	CE	Bernoulli	0.01	0.01	125	None	100	0.33
s deg ≤ 2.0	CE	Basic	0.01	0.01	25	None	100	0.31
s deg > 2.0	CE	Bernoulli	0.01	1e-06	125	None	250	0.41
o deg ≤ 15.0	CE	Basic	0.01	0.01	25	None	100	0.32
o deg > 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
p freq ≤ 458.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
p freq > 458.0	CE	Basic	0.01	0.0001	125	None	50	0.33
s p cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s p cofreq > 0.0	CE	Basic	0.01	1e-06	125	None	50	0.27
o p cofreq ≤ 7.0	CE	Basic	0.01	0.01	125	None	250	0.34
o p cofreq > 7.0	CE	Bernoulli	0.01	1e-06	125	None	100	0.34
s o cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	100	0.27
s o cofreq > 0.0	CE	Basic	0.01	0.01	125	None	250	0.57
s min deg nbr ≤ 15.0	CE	Basic	0.01	0.01	25	None	100	0.32
s min deg nbr > 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s max deg nbr ≤ 15.0	CE	Basic	0.01	0.01	25	None	100	0.32
s max deg nbr > 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s mean deg nbr ≤ 15.0	CE	Basic	0.01	0.01	25	None	100	0.32
s mean deg nbr > 15.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s num neighbours	N/A							
s min freq rel ≤ 458.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
s min freq rel > 458.0	CE	Basic	0.01	0.0001	125	None	50	0.33
s max freq rel ≤ 458.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
s max freq rel > 458.0	CE	Basic	0.01	0.0001	125	None	50	0.33
s mean freq rel ≤ 458.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
s mean freq rel > 458.0	CE	Basic	0.01	0.0001	125	None	50	0.33
s num rels	N/A							
o min deg nbr ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	50	0.31
o min deg nbr > 1.0	CE	Bernoulli	0.01	1e-06	125	None	50	0.36
o max deg nbr ≤ 2.0	CE	Bernoulli	0.01	0.0001	125	None	250	0.32
o max deg nbr > 2.0	CE	Bernoulli	0.01	0.01	125	None	100	0.39
o mean deg nbr ≤ 2.0	CE	Bernoulli	0.01	0.0001	125	None	250	0.31
o mean deg nbr > 2.0	CE	Bernoulli	0.01	1e-06	125	None	250	0.39
o num s o cofreq ≤ 1.0	CE	Bernoulli	0.01	0.0001	125	None	250	0.32
o num s o cofreq > 1.0	CE	Bernoulli	0.01	1e-06	125	None	100	0.39
o min freq rel ≤ 443.0	CE	Bernoulli	0.01	0.01	125	None	100	0.38
o min freq rel > 443.0	CE	Basic	0.01	0.0001	125	None	50	0.32
o max freq rel ≤ 458.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
o max freq rel > 458.0	CE	Basic	0.01	0.0001	125	None	50	0.33
o mean freq rel ≤ 458.0	CE	Bernoulli	0.01	0.01	125	None	100	0.36
o mean freq rel > 458.0	CE	Basic	0.01	0.0001	125	None	50	0.33
o num rels ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.32
o num rels > 1.0	CE	Basic	0.01	1e-06	125	None	250	0.39

Table C.48: TransE on DBpedia50: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.13 TransE on Kinships

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.03	-0.05	-0.01
o deg	0.04	0.06	0.02
p freq	-0.02	-0.03	-0.01
s p cofreq	-0.16	-0.25	-0.06
o p cofreq	-0.21	-0.12	-0.32
s o cofreq	N/A	N/A	N/A
s min deg nbr	-0.02	-0.03	-0.01
s max deg nbr	-0.01	-0.01	-0.0
s mean deg nbr	0.02	0.03	0.01
s num s o cofreq	-0.02	0.0	-0.05
s min freq rel	0.04	0.06	0.03
s max freq rel	0.01	-0.0	0.01
s mean freq rel	0.04	0.05	0.03
s num rels	-0.03	-0.03	-0.03
o min deg nbr	0.04	0.04	0.03
o max deg nbr	-0.0	-0.01	0.01
o mean deg nbr	-0.01	-0.01	-0.01
o num s o cofreq	-0.01	-0.01	-0.01
o min freq rel	0.04	0.04	0.04
o max freq rel	0.03	0.04	0.02
o mean freq rel	0.06	0.07	0.05
o num rels	-0.03	-0.03	-0.03

Table C.49: TransE on Kinships: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
CE	Basic	0.01	0.0001	25	None	100	0.36
MRL	Basic	0.01	0.0001	25	0.5	100	0.28
MRL	Basic	0.01	0.0001	25	1	100	0.28
MRL	Basic	0.01	0.0001	25	2	100	0.28
BCE	Basic	0.01	0.0001	25	None	100	0.23
CE	Bernoulli	0.01	0.0001	25	None	100	0.35
CE	Pseudo	0.01	0.0001	25	None	100	0.26
CE	Basic	0.0001	0.0001	25	None	100	0.05
CE	Basic	1e-06	0.0001	25	None	100	0.04
CE	Basic	0.01	0.01	25	None	100	0.34
CE	Basic	0.01	1e-06	25	None	100	0.36
CE	Basic	0.01	0.0001	5	None	100	0.31
CE	Basic	0.01	0.0001	125	None	100	0.36
CE	Basic	0.01	0.0001	25	None	50	0.33
CE	Basic	0.01	0.0001	25	None	250	0.33

Table C.50: TransE on Kinships: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.04	0.04	0.05	0.22	0.33
Loss = BCE	0.04	0.04	0.05	0.18	0.29
Loss = CE	0.04	0.04	0.05	0.27	0.36
N. Samp = Basic	0.04	0.04	0.05	0.26	0.36
N. Samp = Bernoulli	0.04	0.04	0.05	0.27	0.36
N. Samp = Pseudo	0.04	0.04	0.05	0.2	0.29
LR = 0.01	0.1	0.22	0.27	0.29	0.36
LR = 0.0001	0.04	0.05	0.05	0.05	0.06
LR = 1e-06	0.04	0.04	0.04	0.05	0.05
Reg = 0.01	0.04	0.04	0.05	0.21	0.36
Reg = 0.0001	0.04	0.04	0.05	0.23	0.36
Reg = 1e-06	0.04	0.04	0.05	0.23	0.36
npp = 5	0.04	0.04	0.05	0.21	0.32
npp = 25	0.04	0.04	0.05	0.22	0.36
npp = 125	0.04	0.04	0.05	0.23	0.36
Mgn = None	0.04	0.04	0.05	0.2	0.36
Mgn = 0.5	0.04	0.04	0.05	0.22	0.33
Mgn = 1	0.04	0.04	0.05	0.21	0.3
Mgn = 2	0.04	0.04	0.05	0.23	0.31
Dim = 50	0.04	0.04	0.05	0.22	0.36
Dim = 100	0.04	0.04	0.05	0.22	0.36
Dim = 250	0.04	0.04	0.05	0.22	0.35

Table C.51: Distribution of MRR scores obtained when running TransE on Kinships with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of TransE on Kinships in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	CE	Basic	0.01	0.0001	25	None	100	0.36
s deg ≤ 164.0	CE	Basic	0.01	1e-06	25	None	100	0.35
s deg > 164.0	CE	Basic	0.01	0.0001	125	None	100	0.38
o deg ≤ 164.0	CE	Basic	0.01	0.0001	25	None	100	0.39
o deg > 164.0	CE	Basic	0.01	1e-06	25	None	100	0.35
p freq ≤ 404.0	CE	Bernoulli	0.01	0.0001	125	None	50	0.35
p freq > 404.0	CE	Basic	0.01	0.0001	25	None	100	0.39
s p cofreq ≤ 6.0	CE	Bernoulli	0.01	0.0001	125	None	50	0.31
s p cofreq > 6.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.43
o p cofreq ≤ 8.0	CE	Bernoulli	0.01	0.0001	125	None	50	0.29
o p cofreq > 8.0	CE	Basic	0.01	0.0001	25	None	100	0.47
s o cofreq	N/A							
s min deg nbr ≤ 153.0	CE	Bernoulli	0.01	0.0001	125	None	50	0.37
s min deg nbr > 153.0	CE	Basic	0.01	0.0001	25	None	100	0.37
s max deg nbr ≤ 172.0	CE	Basic	0.01	0.0001	125	None	100	0.37
s max deg nbr > 172.0	CE	Basic	0.01	0.0001	25	None	100	0.37
s mean deg nbr ≤ 163.8	CE	Basic	0.01	0.0001	25	None	100	0.37
s mean deg nbr > 163.8	CE	Basic	0.01	0.0001	125	None	100	0.38
s num s o cofreq ≤ 20.0	CE	Basic	0.01	0.0001	125	None	100	0.38
s num s o cofreq > 20.0	CE	Bernoulli	0.01	0.0001	125	None	100	0.36
s min freq rel ≤ 183.0	CE	Bernoulli	0.01	0.0001	125	None	50	0.38
s min freq rel > 183.0	CE	Basic	0.01	1e-06	25	None	100	0.33
s max freq rel	N/A							
s mean freq rel ≤ 473.5	CE	Bernoulli	0.01	0.0001	125	None	50	0.38
s mean freq rel > 473.5	CE	Basic	0.01	0.0001	25	None	100	0.36
s num rels ≤ 11.0	CE	Basic	0.01	0.0001	125	None	100	0.36
s num rels > 11.0	CE	Basic	0.01	1e-06	25	None	100	0.38
o min deg nbr ≤ 153.0	CE	Basic	0.01	0.0001	125	None	100	0.37
o min deg nbr > 153.0	CE	Bernoulli	0.01	0.01	25	None	100	0.36
o max deg nbr ≤ 172.0	CE	Bernoulli	0.01	0.0001	125	None	50	0.36
o max deg nbr > 172.0	CE	Basic	0.01	0.0001	125	None	100	0.37
o mean deg nbr ≤ 163.8	CE	Basic	0.01	0.0001	125	None	50	0.35
o mean deg nbr > 163.8	CE	Basic	0.01	1e-06	25	None	100	0.39
o num s o cofreq ≤ 20.0	CE	Basic	0.01	1e-06	25	None	100	0.37
o num s o cofreq > 20.0	CE	Basic	0.01	0.0001	25	None	100	0.36
o min freq rel ≤ 183.0	CE	Bernoulli	0.01	1e-06	25	None	100	0.38
o min freq rel > 183.0	CE	Bernoulli	0.01	1e-06	25	None	50	0.35
o max freq rel	N/A							
o mean freq rel ≤ 472.9	CE	Basic	0.01	0.0001	25	None	100	0.37
o mean freq rel > 472.9	CE	Bernoulli	0.01	0.01	25	None	100	0.36
o num rels ≤ 11.0	CE	Bernoulli	0.01	1e-06	25	None	100	0.36
o num rels > 11.0	CE	Basic	0.01	0.0001	25	None	100	0.37

Table C.52: TransE on Kinships: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.14 TransE on OpenEA

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	-0.05	-0.06	-0.04
o deg	-0.02	-0.0	-0.06
p freq	0.09	0.12	0.05
s p cofreq	-0.09	-0.13	-0.02
o p cofreq	0.01	0.05	-0.06
s o cofreq	-0.14	-0.19	-0.08
s min deg nbr	-0.01	0.01	-0.05
s max deg nbr	-0.01	0.01	-0.05
s mean deg nbr	-0.01	0.01	-0.05
s num s o cofreq	-0.02	-0.02	-0.01
s min freq rel	0.07	0.09	0.04
s max freq rel	0.08	0.1	0.05
s mean freq rel	0.08	0.1	0.05
s num rels	-0.03	-0.05	-0.01
o min deg nbr	-0.08	-0.12	-0.03
o max deg nbr	-0.01	-0.02	-0.02
o mean deg nbr	-0.03	-0.04	-0.02
o num s o cofreq	-0.03	-0.02	-0.06
o min freq rel	0.08	0.11	0.04
o max freq rel	0.05	0.07	0.03
o mean freq rel	0.07	0.1	0.04
o num rels	-0.05	-0.07	-0.02

Table C.53: TransE on OpenEA: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
CE	Bernoulli	0.01	0.01	125	None	100	0.30
MRL	Bernoulli	0.01	0.01	125	0.5	100	0.15
MRL	Bernoulli	0.01	0.01	125	1	100	0.2
MRL	Bernoulli	0.01	0.01	125	2	100	0.26
BCE	Bernoulli	0.01	0.01	125	None	100	0.09
CE	Basic	0.01	0.01	125	None	100	0.29
CE	Pseudo	0.01	0.01	125	None	100	0.11
CE	Bernoulli	0.0001	0.01	125	None	100	0.12
CE	Bernoulli	1e-06	0.01	125	None	100	0.01
CE	Bernoulli	0.01	0.0001	125	None	100	0.29
CE	Bernoulli	0.01	1e-06	125	None	100	0.29
CE	Bernoulli	0.01	0.01	5	None	100	0.24
CE	Bernoulli	0.01	0.01	25	None	100	0.28
CE	Bernoulli	0.01	0.01	125	None	50	0.3
CE	Bernoulli	0.01	0.01	125	None	250	0.29

Table C.54: TransE on OpenEA: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.01	0.01	0.02	0.08	0.29
Loss = BCE	0.01	0.01	0.09	0.1	0.22
Loss = CE	0.01	0.01	0.07	0.12	0.3
N. Samp = Basic	0.01	0.01	0.04	0.11	0.29
N. Samp = Bernoulli	0.01	0.01	0.06	0.1	0.3
N. Samp = Pseudo	0.01	0.01	0.03	0.08	0.12
LR = 0.01	0.04	0.08	0.11	0.18	0.3
LR = 0.0001	0.01	0.02	0.04	0.09	0.13
LR = 1e-06	0.01	0.01	0.01	0.01	0.01
Reg = 0.01	0.01	0.01	0.04	0.09	0.3
Reg = 0.0001	0.01	0.01	0.04	0.1	0.29
Reg = 1e-06	0.01	0.01	0.04	0.1	0.29
npp = 5	0.01	0.01	0.04	0.08	0.26
npp = 25	0.01	0.01	0.04	0.1	0.29
npp = 125	0.01	0.01	0.04	0.11	0.3
Mgn = None	0.01	0.01	0.09	0.11	0.3
Mgn = 0.5	0.01	0.01	0.01	0.06	0.17
Mgn = 1	0.01	0.01	0.02	0.08	0.21
Mgn = 2	0.01	0.01	0.04	0.1	0.29
Dim = 50	0.01	0.01	0.04	0.1	0.3
Dim = 100	0.01	0.01	0.04	0.1	0.3
Dim = 250	0.01	0.01	0.04	0.1	0.29

Table C.55: Distribution of MRR scores obtained when running TransE on OpenEA with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of TransE on OpenEA in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	CE	Bernoulli	0.01	0.01	125	None	100	0.3
s deg ≤ 3.0	CE	Bernoulli	0.01	0.01	125	None	100	0.26
s deg > 3.0	MRL	Bernoulli	0.01	0.0001	125	2	100	0.36
o deg ≤ 12.0	CE	Bernoulli	0.01	0.01	125	None	100	0.25
o deg > 12.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
p freq ≤ 1242.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
p freq > 1242.0	CE	Basic	0.01	0.01	125	None	50	0.27
s p cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
s p cofreq > 0.0	CE	Bernoulli	0.01	1e-06	125	None	250	0.22
o p cofreq ≤ 5.0	CE	Bernoulli	0.01	0.01	125	None	100	0.33
o p cofreq > 5.0	CE	Bernoulli	0.01	1e-06	125	None	100	0.28
s o cofreq ≤ 0.0	CE	Bernoulli	0.01	0.01	125	None	50	0.2
s o cofreq > 0.0	MRL	Basic	0.01	1e-06	125	2	100	0.62
s min deg nbr ≤ 9.0	CE	Bernoulli	0.01	0.01	125	None	100	0.27
s min deg nbr > 9.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
s max deg nbr ≤ 14.0	CE	Bernoulli	0.01	0.01	125	None	100	0.27
s max deg nbr > 14.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
s mean deg nbr ≤ 12.2	CE	Bernoulli	0.01	0.01	125	None	100	0.26
s mean deg nbr > 12.2	CE	Bernoulli	0.01	0.01	125	None	100	0.34
s num s o cofreq ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.3
s num s o cofreq > 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.3
s min freq rel ≤ 1030.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
s min freq rel > 1030.0	CE	Basic	0.01	0.01	125	None	50	0.26
s max freq rel ≤ 1598.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s max freq rel > 1598.0	CE	Bernoulli	0.01	1e-06	125	None	50	0.24
s mean freq rel ≤ 1311.5	CE	Bernoulli	0.01	0.01	125	None	100	0.35
s mean freq rel > 1311.5	CE	Basic	0.01	0.01	125	None	50	0.26
s num rels ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.28
s num rels > 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.35
o min deg nbr ≤ 2.0	CE	Bernoulli	0.01	0.01	125	None	100	0.26
o min deg nbr > 2.0	MRL	Basic	0.01	1e-06	125	2	50	0.38
o max deg nbr ≤ 5.0	CE	Bernoulli	0.01	0.01	125	None	100	0.27
o max deg nbr > 5.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
o mean deg nbr ≤ 3.5	CE	Bernoulli	0.01	0.01	125	None	100	0.24
o mean deg nbr > 3.5	MRL	Bernoulli	0.01	0.0001	125	2	100	0.36
o num s o cofreq ≤ 3.0	CE	Bernoulli	0.01	0.01	125	None	100	0.29
o num s o cofreq > 3.0	CE	Bernoulli	0.01	1e-06	125	None	100	0.33
o min freq rel ≤ 741.0	CE	Bernoulli	0.01	0.01	125	None	100	0.37
o min freq rel > 741.0	CE	Bernoulli	0.01	0.01	125	None	100	0.23
o max freq rel ≤ 1598.0	CE	Bernoulli	0.01	0.01	125	None	100	0.33
o max freq rel > 1598.0	CE	Bernoulli	0.01	0.01	125	None	100	0.27
o mean freq rel ≤ 1242.0	CE	Bernoulli	0.01	0.01	125	None	100	0.34
o mean freq rel > 1242.0	CE	Basic	0.01	0.01	125	None	50	0.25
o num rels ≤ 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.24
o num rels > 1.0	CE	Bernoulli	0.01	0.01	125	None	100	0.39

Table C.56: TransE on OpenEA: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

C.15 TransE on UMLS

Struct Ft	All Ranks	Subject Ranks	Object Ranks
s deg	0.01	-0.08	0.07
o deg	-0.02	-0.01	-0.04
p freq	0.12	0.18	0.08
s p cofreq	-0.06	-0.23	0.07
o p cofreq	0.13	0.4	-0.08
s o cofreq	-0.03	-0.07	0.01
s min deg nbr	-0.03	0.05	-0.08
s max deg nbr	-0.03	-0.08	0.01
s mean deg nbr	-0.05	-0.05	-0.05
s num s o cofreq	0.01	-0.08	0.09
s min freq rel	0.08	0.21	-0.01
s max freq rel	-0.0	-0.03	0.03
s mean freq rel	0.07	0.15	0.0
s num rels	-0.03	-0.13	0.05
o min deg nbr	-0.04	-0.13	0.03
o max deg nbr	-0.04	-0.19	0.07
o mean deg nbr	-0.05	-0.2	0.07
o num s o cofreq	-0.02	-0.03	-0.02
o min freq rel	0.05	0.12	-0.0
o max freq rel	-0.1	-0.16	-0.05
o mean freq rel	-0.02	-0.01	-0.03
o num rels	-0.06	-0.12	-0.02

Table C.57: TransE on UMLS: Correlation of each structural feature of a link prediction query to the rank assigned to that link prediction query. Correlations to all ranks, or only to those for subject predictions or object predictions in turn, are shown. Darker colours indicate higher absolute values of correlation. Ft = feature; deg = degree; freq = frequency; nbr = neighbour; rel = relation.

Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
MRL	Bernoulli	0.01	1e-06	125	2	50	0.63
BCE	Bernoulli	0.01	1e-06	125	None	50	0.58
CE	Bernoulli	0.01	1e-06	125	None	50	0.62
MRL	Basic	0.01	1e-06	125	2	50	0.61
MRL	Pseudo	0.01	1e-06	125	2	50	0.11
MRL	Bernoulli	0.0001	1e-06	125	2	50	0.14
MRL	Bernoulli	1e-06	1e-06	125	2	50	0.04
MRL	Bernoulli	0.01	0.01	125	2	50	0.62
MRL	Bernoulli	0.01	0.0001	125	2	50	0.63
MRL	Bernoulli	0.01	1e-06	5	2	50	0.61
MRL	Bernoulli	0.01	1e-06	25	2	50	0.62
MRL	Bernoulli	0.01	1e-06	125	0.5	50	0.52
MRL	Bernoulli	0.01	1e-06	125	1	50	0.56
MRL	Bernoulli	0.01	1e-06	125	2	100	0.63
MRL	Bernoulli	0.01	1e-06	125	2	250	0.62

Table C.58: TransE on UMLS: Comparison of optimal hyperparameter configurations (top row) with all possible alternate configurations differing by one parameter value only. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = Margin; Dim = embedding dimension.

Setting	Min	25%	Median	75%	Max
Loss = MRL	0.03	0.04	0.07	0.13	0.63
Loss = BCE	0.03	0.04	0.09	0.31	0.6
Loss = CE	0.03	0.04	0.12	0.16	0.63
N. Samp = Basic	0.03	0.04	0.1	0.51	0.63
N. Samp = Bernoulli	0.03	0.04	0.11	0.53	0.63
N. Samp = Pseudo	0.03	0.04	0.04	0.08	0.33
LR = 0.01	0.06	0.16	0.52	0.6	0.63
LR = 0.0001	0.03	0.04	0.09	0.11	0.16
LR = 1e-06	0.03	0.04	0.04	0.04	0.05
Reg = 0.01	0.03	0.04	0.08	0.16	0.63
Reg = 0.0001	0.03	0.04	0.09	0.16	0.63
Reg = 1e-06	0.03	0.04	0.09	0.16	0.63
npp = 5	0.03	0.04	0.08	0.16	0.63
npp = 25	0.03	0.04	0.09	0.17	0.63
npp = 125	0.03	0.04	0.08	0.16	0.63
Mgn = None	0.03	0.04	0.09	0.25	0.63
Mgn = 0.5	0.03	0.04	0.07	0.1	0.55
Mgn = 1	0.03	0.04	0.07	0.12	0.58
Mgn = 2	0.03	0.04	0.11	0.14	0.63
Dim = 50	0.03	0.04	0.09	0.16	0.63
Dim = 100	0.03	0.04	0.08	0.16	0.63
Dim = 250	0.03	0.04	0.09	0.16	0.63

Table C.59: Distribution of MRR scores obtained when running TransE on UMLS with a given hyperparameter fixed and all others allowed to vary freely. Distributions are given as the minimum, 25th-percentile, median, 75th-percentile, and maximum values observed under the specified condition. For example, the row “loss = MRL” shows the distribution of MRR values for all runs of TransE on UMLS in which Margin Ranking Loss (MRL) was used. N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

Mode	Loss	N. Samp	LR	Reg	npp	Mgn	Dim	MRR
Overall	MRL	Bernoulli	0.01	1e-06	125	2	50	0.63
s deg ≤ 84.0	CE	Basic	0.01	0.0001	25	None	50	0.68
s deg > 84.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.61
o deg ≤ 122.0	MRL	Bernoulli	0.01	0.0001	25	2	100	0.67
o deg > 122.0	CE	Basic	0.01	0.01	25	None	250	0.62
p freq ≤ 283.0	MRL	Basic	0.01	1e-06	5	2	250	0.74
p freq > 283.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.53
s p cofreq ≤ 11.0	CE	Bernoulli	0.01	1e-06	125	None	100	0.66
s p cofreq > 11.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.62
o p cofreq ≤ 11.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.71
o p cofreq > 11.0	CE	Basic	0.01	0.0001	25	None	250	0.58
s o cofreq ≤ 1.0	MRL	Bernoulli	0.01	0.0001	25	2	100	0.65
s o cofreq > 1.0	CE	Basic	0.01	0.01	25	None	250	0.63
s min deg nbr ≤ 40.0	MRL	Bernoulli	0.01	1e-06	25	2	250	0.63
s min deg nbr > 40.0	MRL	Bernoulli	0.01	1e-06	5	2	100	0.66
s max deg nbr ≤ 304.0	CE	Basic	0.01	0.0001	25	None	250	0.65
s max deg nbr > 304.0	MRL	Bernoulli	0.01	0.0001	25	2	250	0.63
s mean deg nbr ≤ 132.5	MRL	Bernoulli	0.01	0.0001	125	2	50	0.62
s mean deg nbr > 132.5	MRL	Bernoulli	0.01	0.0001	25	2	100	0.67
s num s o cofreq ≤ 12.0	CE	Basic	0.01	0.0001	25	None	250	0.68
s num s o cofreq > 12.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.6
s min freq rel ≤ 35.0	MRL	Bernoulli	0.01	1e-06	125	2	100	0.65
s min freq rel > 35.0	MRL	Bernoulli	0.01	0.0001	25	2	100	0.63
s max freq rel	N/A							
s mean freq rel ≤ 255.6	CE	Basic	0.01	0.0001	25	None	250	0.65
s mean freq rel > 255.6	MRL	Bernoulli	0.01	0.0001	25	2	100	0.63
s num rels ≤ 7.0	MRL	Bernoulli	0.01	1e-06	25	2	250	0.67
s num rels > 7.0	MRL	Bernoulli	0.01	1e-06	125	2	50	0.62
o min deg nbr ≤ 40.0	CE	Basic	0.01	0.0001	25	None	250	0.63
o min deg nbr > 40.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.65
o max deg nbr ≤ 304.0	CE	Basic	0.01	1e-06	25	None	100	0.63
o max deg nbr > 304.0	MRL	Basic	0.01	1e-06	5	2	250	0.66
o mean deg nbr ≤ 130.8	CE	Basic	0.01	0.0001	25	None	250	0.66
o mean deg nbr > 130.8	MRL	Bernoulli	0.01	1e-06	125	2	50	0.63
o num s o cofreq ≤ 15.0	MRL	Bernoulli	0.01	1e-06	125	2	100	0.66
o num s o cofreq > 15.0	CE	Basic	0.01	0.0001	25	None	250	0.62
o min freq rel ≤ 42.0	CE	Basic	0.01	0.01	5	None	100	0.68
o min freq rel > 42.0	MRL	Bernoulli	0.01	0.0001	125	2	50	0.61
o max freq rel	N/A							
o mean freq rel ≤ 272.4	CE	Basic	0.01	1e-06	125	None	250	0.64
o mean freq rel > 272.4	MRL	Bernoulli	0.01	0.0001	25	2	100	0.65
o num rels ≤ 8.0	CE	Basic	0.01	0.0001	125	None	50	0.66
o num rels > 8.0	CE	Basic	0.01	0.0001	25	None	250	0.63

Table C.60: TransE on UMLS: Structure controlled analysis of hyperparameter preference and link prediction performance. deg = degree; freq = frequency; nbr = neighbour; rel = relation; N. Samp = negative sampler; LR = learning rate; Reg = regularisation coefficient; npp = negatives per positive; Mgn = margin; Dim = embedding dimension.

D. TWIG-I vs. the Intersection Features Model

The approach taken by TWIG-I for embedding-free, structure-based link prediction has notable similarities to (and differences from) a separate model developed independently by Le et al. (2024) called Intersection Features [48]. The Intersection Features model also performs embedding-free link prediction; however, its method is based not on frequency-based structural characteristics, but on the intersection of nodes and edges in the neighbourhood of the subject, predicate, and object in a triple [48]. A summary of the Intersection Features model is given below, followed by a theory-based comparison to the approach taken by TWIG-I. Finally, results of TWIG-I as presented in this thesis (under its best settings) and the Intersection Features system developed by Le et al. (under its best settings) are presented to provide a basis for understanding their relative strengths and weaknesses.

D.1 Summary of the Intersection Features Model

Le et al. (2024) [48] develop a novel link prediction paradigm based on graph intersections called Intersection Features. In essence, for every triple (s, p, o) , they calculate all nodes and edges that each triple element connects to within a k -hop radius [48]. They then use this to compute, for each triple, an estimation of the intersection of the overlap of the set of nodes and edges in the neighbourhood of the subject, predicate, and object [48]. This choice is motivated by the observation that, for true triples, there is typically much larger intersections in the neighbourhoods of each triple element (subject, predicate, and object) than is observed in any negative triple, a fact that they demonstrate empirically on 5 different KGs: NELL-995, WN18RR, YAGO3-10, FB15K-237, and FB15K.

They then perform link prediction using these features. The core of their method is a randomisation-based feature selection algorithm based on 3-way Jaccard similarity and 3-way union cardinality to select features that can characterise the intersection of the k -hop neighbourhoods of the subject, predicate, and object in each triple [48]. These features are calculated for all triples (positive and negative) and are then used as inputs into a single-dense-layer neural network that is trained to distinguish positive triples from negatives.

Their system results in massive increases in performance over baselines (including TransE, DistMult, and ComplEx) on 5 KGs: NELL-995, WN18RR, YAGO3-10, FB15K-237, and FB15K.

D.2 Comparison with TWIG-I

The intersection features that Le et al. (2024) choose are structural features and are very similar in principle to node-relation and node-node structural features highlighted in this thesis, even though they are calculated differently at a mathematical level [48]. In particular, while node-relation and node-node co-frequencies and intersection features both quantify how commonly different triple elements co-occur in a KG, the intersection features used by Le et al. (2024) are calculated based on the set of all nodes and edges in a KG with a k-hop distance of the subject, predicate, or object of a triple [48]. The intersection of *all* three sets is used as a measure of how likely that triple is to occur, with larger intersections indicating higher plausibility [48]. Since Le et al. (2024) considered unordered sets and set intersections, repeat observations of the same element are ignored [48].

This is in contrast to TWIG-I, where node-node and node-relation co-frequencies explicitly are meant to account repeat observations of the same elements. As such, while it is correct to say that TWIG-I operates on frequencies, it is not correct to say that the Intersection Features model uses frequency-based measures of graph structure [48]. Instead, Intersection Features uses set-theoretic measures of graph structure which relates to, but is not equivalent to, frequency-based measures of structure [48].

Looking beyond the level of features, it is notable that while TWIG-I uses a fixed set of features in all cases, that Le et al. (2024)’s system uses a randomisation algorithm to select what specific features to use [48]. This means that their system may generate different features for different graphs, and as a result may not allow native transfer learning in the way that TWIG-I does. The question of transfer learning is not addressed in Le et al. (2024), and as such it is uncertain if, or how well, their system could be developed to work in that setting [48].

Finally, while TWIG-I uses a three-layer neural network for scoring triples based on their features, the Intersection Features model uses a single dense layer to do so. Interestingly, both Intersection Features and TWIG-I are trained using the Sigmoid activation function in the final layer and margin-based loss.

D.3 Comparison of Empirical Performance

The results of TWIG-I in on FB15k-237 and WN18RR, as well as of the Intersection Features system, are given in Table D.1. In both cases, only the best-reported results of TWIG-I and the Intersection Features model are given.

From these results, it can be seen that TWIG-I significantly outperforms the Intersection

	FB15k-237	WN18RR
Intersection Features [48]	0.63	0.78
TWIG-I, Finetuned from FB15k-237	NA	0.73
TWIG-I, From Scratch, 20e	0.82	0.45

Table D.1: Comparison of the Link Prediction performance of TWIG-I and the Intersection Features system. The best results reported from Le et al.’s paper [48] on both FB15k-237 and WN18RR are shown. For TWIG-I, its best results on both datasets when trained from scratch (without transfer learning), and when trained with transfer learning, are shown. Full details on the evaluation of Le et al. (2024)’s system can be found in their paper [48], and full details on TWIG-I’s training can be found in Chapter 5 of this thesis.

Features model on FB15k-237 (by 0.19 points in MRR). However, TWIG-I lags notably behind it on WN18RR when trained from scratch (by 0.33 points in MRR). When trained in the transfer learning setting, TWIG-I achieves a much more similar performance on WN18RR, behind the Intersection Features model by only 0.05 points in MRR.

It is particularly of note that TWIG-I does comparatively better on the more structurally diverse KG (FB15k-237), while the Intersection Features model does comparatively better on the more structurally uniform (and less-well connected) KG (WN18RR). However, it is unclear if this is a general trend, or a chance occurrence, as TWIG-I and the Intersection Features system were not both evaluated on any other KG to provide a further basis for comparison.

Overall, both systems have comparably high performance. As both are also structure-based (and embedding-free) methods, this suggests that further work in structure-based link prediction is merited, and would likely lead to continued improvements in link prediction performance upon further research.

References

- [1] Akiba, Takuya, Sano, Shotaro, Yanase, Toshihiko, Ohta, Takeru, and Koyama, Masanori. “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.
- [2] Ali, Mehdi, Berrendorf, Max, Hoyt, Charles Tapley, Vermue, Laurent, Galkin, Mikhail, Sharifzadeh, Sahand, Fischer, Asja, Tresp, Volker, and Lehmann, Jens. “Bringing Light Into the Dark: A Large-Scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.12 (2022), pp. 8825–8845. DOI: [10.1109/TPAMI.2021.3124805](https://doi.org/10.1109/TPAMI.2021.3124805).
- [3] Ali, Mehdi, Berrendorf, Max, Hoyt, Charles Tapley, Vermue, Laurent, Sharifzadeh, Sahand, Tresp, Volker, and Lehmann, Jens. “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>.
- [4] Amayuelas, Alfonso, Zhang, Shuai, Rao, Xi Susie, and Zhang, Ce. “Neural methods for logical reasoning over knowledge graphs”. In: *International Conference on Learning Representations*. 2022.
- [5] Bastian, Mathieu, Heymann, Sebastien, and Jacomy, Mathieu. “Gephi: an open source software for exploring and manipulating networks”. In: *Proceedings of the international AAAI conference on web and social media*. Vol. 3. 1. 2009, pp. 361–362.
- [6] Belleau, François, Nolin, Marc-Alexandre, Tourigny, Nicole, Rigault, Philippe, and Morissette, Jean. “Bio2RDF: +Towards A Mashup To Build Bioinformatics Knowledge System”. In: *Journal of biomedical informatics* 41 (Apr. 2008), pp. 706–16. DOI: [10.1016/j.jbi.2008.03.004](https://doi.org/10.1016/j.jbi.2008.03.004).
- [7] Bhardwaj, Peru, Kelleher, John, Costabello, Luca, and O’Sullivan, Declan. “Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 8225–8239.
- [8] Bhardwaj, Peru, Kelleher, John, Costabello, Luca, and O’Sullivan, Declan. “Poisoning Knowledge Graph Embeddings via Relation Inference Patterns”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 1875–1888.

- [9] Bonner, Stephen, Kirik, Ufuk, Engkvist, Ola, Tang, Jian, and Barrett, Ian P. “Implications of topological imbalance for representation learning on biomedical knowledge graphs”. In: *Briefings in Bioinformatics* 23.5 (July 2022). bbac279. ISSN: 1477-4054. DOI: [10.1093/bib/bbac279](https://doi.org/10.1093/bib/bbac279). eprint: https://academic.oup.com/bib/article-pdf/23/5/bbac279/45937607/sup_main_bbac279.pdf. URL: <https://doi.org/10.1093/bib/bbac279>.
- [10] Bordes, Antoine, Glorot, Xavier, Weston, Jason, and Bengio, Yoshua. “A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation”. In: *Machine learning* 94 (2014), pp. 233–259.
- [11] Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. “Translating embeddings for modeling multi-relational data”. In: *Advances in neural information processing systems* 26 (2013).
- [12] Boschini, Armand. “Torchkg: Knowledge graph embedding in python and pytorch”. In: *arXiv preprint arXiv:2009.02963* (2020).
- [13] Bouchard, Guillaume, Singh, Sameer, and Trouillon, Théo. “On Approximate Reasoning Capabilities of Low-Rank Vector Spaces”. In: *AAAI Spring Symposia*. 2015. URL: <https://api.semanticscholar.org/CorpusID:13955854>.
- [14] Breit, Anna, Ott, Simon, Agibetov, Asan, and Samwald, Matthias. “OpenBioLink: a benchmarking framework for large-scale biomedical link prediction”. In: *Bioinformatics* 36.13 (2020), pp. 4097–4098.
- [15] Böhmann, Lorenz, Lehmann, Jens, and Westphal, Patrick. “DL-Learner—A framework for inductive learning on the Semantic Web”. In: *Journal of Web Semantics* 39 (2016), pp. 15–24.
- [16] Celebi, Remzi, Uyar, Huseyin, Yasar, Erkan, Gumus, Ozgur, Dikenelli, Oguz, and Dumontier, Michel. “Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction in realistic settings”. In: *BMC bioinformatics* 20 (2019), pp. 1–14.
- [17] Celebi, Remzi, Uyar, Huseyin, Yasar, Erkan, Gumus, Ozgur, Dikenelli, Oguz, and Dumontier, Michel. “Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction in realistic settings”. In: *BMC bioinformatics* 20 (2019), pp. 1–14.
- [18] Chandak, Payal, Huang, Kexin, and Zitnik, Marinka. “Building a knowledge graph to enable precision medicine”. In: *Scientific Data* 10.1 (2023), p. 67.

- [19] Claesen, Marc and De Moor, Bart. “Hyperparameter search in machine learning”. In: *arXiv preprint arXiv:1502.02127* (2015).
- [20] Cloud, The LoD. *The Linked Open Data Cloud*. URL: <https://lod-cloud.net/> (visited on 09/23/2024).
- [21] Costabello, Luca, Pai, Sumit, Le Van, Chan, McGrath, Rory, McCarthy, Nicholas, and Tabacof, Pedro. “AmpliGraph: a library for representation learning on knowledge graphs”. In: *Retrieved Oct 10* (2019), p. 2019.
- [22] Dave, Brandon, Christou, Antrea, and Shimizu, Cogan. “Towards Understanding the Impact of Graph Structure on Knowledge Graph Embeddings”. In: *International Conference on Neural-Symbolic Learning and Reasoning*. Springer. 2024, pp. 41–50.
- [23] Davis, Allan Peter, Grondin, Cynthia J, Johnson, Robin J, Sciaky, Daniela, McMorran, Roy, Wiegiers, Jolene, Wiegiers, Thomas C, and Mattingly, Carolyn J. “The comparative toxicogenomics database: update 2019”. In: *Nucleic acids research* 47.D1 (2019), pp. D948–D954.
- [24] Dettmers, Tim, Minervini, Pasquale, Stenetorp, Pontus, and Riedel, Sebastian. “Convolutional 2d knowledge graph embeddings”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [25] Dörpinghaus, Jens, Weil, Vera, Düing, Carsten, and Sommer, Martin W. “Centrality Measures in multi-layer Knowledge Graphs”. In: *Annals of Computer Science and Information Systems* (2022).
- [26] Falkner, Stefan, Klein, Aaron, and Hutter, Frank. “BOHB: Robust and efficient hyperparameter optimization at scale”. In: *International conference on machine learning*. PMLR. 2018, pp. 1437–1446.
- [27] Florea, Adrian-Catalin and Andonie, Razvan. “Weighted Random Search for Hyperparameter Optimization”. In: *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL* 14.2 (2019), pp. 154–169.
- [28] Galárraga, Luis, Teflioudi, Christina, Hose, Katja, and Suchanek, Fabian M. “Fast rule mining in ontological knowledge bases with AMIE+”. In: *The VLDB Journal* 24.6 (2015), pp. 707–730.
- [29] Garg, Dinesh, Ikbali, Shajith, Srivastava, Santosh K, Vishwakarma, Harit, Karanam, Hima, and Subramaniam, L Venkata. “Quantum embedding of knowledge for reasoning”. In: *Advances in Neural Information Processing Systems* 32 (2019).

- [30] Gualdi, Francesco, Oliva, Baldomero, and Piñero, Janet. “Predicting gene disease associations with knowledge graph embeddings for diseases with curtailed information”. In: *NAR Genomics and Bioinformatics* 6.2 (2024), lqae049.
- [31] Guéret, Christophe Dominique Marie, Costabello, Luca, and Sardina, Jeffrey. *Electronic health records data summarization for graph machine learning*. US Patent App. 18/333,875. Dec. 2024.
- [32] Himmelstein, Daniel Scott, Lizée, Antoine, Hessler, Christine, Brueggeman, Leo, Chen, Sabrina L, Hadley, Dexter, Green, Ari, Khankhanian, Pouya, and Baranzini, Sergio E. “Systematic integration of biomedical knowledge prioritizes drugs for repurposing”. In: *Elife* 6 (2017), e26726.
- [33] Hogan, Aidan, Blomqvist, Eva, Cochez, Michael, D’amato, Claudia, Melo, Gerard De, Gutierrez, Claudio, Kirrane, Sabrina, Gayo, José Emilio Labra, Navigli, Roberto, Neumaier, Sebastian, Ngomo, Axel-Cyrille Ngonga, Polleres, Axel, Rashid, Sabbir M., Rula, Anisa, Schmelzeisen, Lukas, Sequeda, Juan, Staab, Steffen, and Zimmermann, Antoine. “Knowledge Graphs”. In: *ACM Comput. Surv.* 54.4 (July 2021). ISSN: 0360-0300. DOI: [10.1145/3447772](https://doi.org/10.1145/3447772). URL: <https://doi.org/10.1145/3447772>.
- [34] Hubert, Nicolas, Monnin, Pierre, d’Aquin, Mathieu, Monticolo, Davy, and Brun, Armelle. “Pygraft: Configurable generation of synthetic schemas and knowledge graphs at your fingertips”. In: *European Semantic Web Conference*. Springer. 2024, pp. 3–20.
- [35] Jain, Nitisha, Tran, Trung-Kien, Gad-Elrab, Mohamed H, and Stepanova, Daria. “Improving knowledge graph embeddings with ontological reasoning”. In: *International Semantic Web Conference*. Springer. 2021, pp. 410–426.
- [36] Jain, Prachi, Rathi, Sushant, Chakrabarti, Soumen, et al. “Knowledge base completion: Baseline strikes back (again)”. In: *arXiv preprint arXiv:2005.00804* (2020).
- [37] Jiang, Xiaodong, Zhu, Ronghang, Li, Sheng, and Ji, Pengsheng. “Co-embedding of nodes and edges with graph neural networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).
- [38] Kadlec, Rudolf, Bajgar, Ondrej, and Kleindienst, Jan. “Knowledge Base Completion: Baselines Strike Back”. In: *ACL 2017* (2017), p. 69.
- [39] Kanehisa, Minoru and Goto, Susumu. “KEGG: Kyoto Encyclopedia of Genes and Ge-nomes”. In: *Nucleic acids research* 28.1 (2000), pp. 27–30.
- [40] Keeney, John, Roblek, Dominik, Jones, Dominic, Lewis, David, and O’Sullivan, Declan. “Extending siena to support more expressive and flexible subscriptions”. In:

- Proceedings of the second international conference on Distributed event-based systems*. 2008, pp. 35–46.
- [41] Kejriwal, Mayank. “Knowledge graphs: A practical review of the research landscape”. In: *Information* 13.4 (2022), p. 161.
 - [42] Kemp, Charles, Tenenbaum, Joshua B, Griffiths, Thomas L, Yamada, Takeshi, and Ueda, Naonori. “Learning systems of concepts with an infinite relational model”. In: *AAAI*. Vol. 3. 2006, p. 5.
 - [43] Kingma, Diederik P and Ba, Jimmy. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
 - [44] Kipf, Thomas N and Welling, Max. “Variational graph auto-encoders”. In: *arXiv preprint arXiv:1611.07308* (2016).
 - [45] Kotnis, Bhushan and Nastase, Vivi. “Analysis of the Impact of Negative Sampling on Link Prediction in Knowledge Graphs”. In: (Aug. 2017).
 - [46] Kulmanov, Maxat, Liu-Wei, Wang, Yan, Yuan, and Hoehndorf, Robert. “EL Embeddings: Geometric construction of models for the description logic EL++”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. 2019, pp. 6103–6109.
 - [47] Lacroix, Timothée, Usunier, Nicolas, and Obozinski, Guillaume. “Canonical tensor decomposition for knowledge base completion”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2863–2872.
 - [48] Le, Duy, Zhong, Shaochen, Liu, Zirui, Xu, Shuai, Chaudhary, Vipin, Zhou, Kaixiong, and Xu, Zhaozhuo. “Knowledge Graphs Can be Learned with Just Intersection Features”. In: *Forty-first International Conference on Machine Learning*.
 - [49] Le, Thanh, Nguyen, Hoang, and Le, Bac. “A survey of the link prediction on static and temporal knowledge graph”. In: *Journal of Research and Development on Information and Communication Technology* 2021.2 (2021), pp. 1–34.
 - [50] Lerer, Adam, Wu, Ledell, Shen, Jiajun, Lacroix, Timothee, Wehrstedt, Luca, Bose, Abhijit, and Peysakhovich, Alex. “Pytorch-biggraph: A large scale graph embedding system”. In: *Proceedings of Machine Learning and Systems* 1 (2019), pp. 120–131.
 - [51] Lewis, David, Keeney, John, O’Sullivan, Declan, and Guo, Song. “Towards a managed extensible control plane for knowledge-based networking”. In: *Large Scale Management of Distributed Systems: 17th IFIP/IEEE International Workshop on Distributed*

- Systems: Operations and Management, DSOM 2006, Dublin, Ireland, October 23-25, 2006. Proceedings 17*. Springer. 2006, pp. 98–111.
- [52] Liu, Jiawei, Yang, Cheng, Lu, Zhiyuan, Chen, Junze, Li, Yibo, Zhang, Mengmei, Bai, Ting, Fang, Yuan, Sun, Lichao, Yu, Philip S, et al. “Towards graph foundation models: A survey and beyond”. In: *arXiv preprint arXiv:2310.11829* (2023).
 - [53] Liu, Yushan, Hildebrandt, Marcel, Joblin, Mitchell, Ringsquandl, Martin, Raissouni, Rime, and Tresp, Volker. “Neural multi-hop reasoning with logical rules on biomedical knowledge graphs”. In: *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*. Springer. 2021, pp. 375–391.
 - [54] Mahdisoltani, Farzane, Biega, Joanna, and Suchanek, Fabian M. “A knowledge base from multilingual Wikipedias–yago3”. In: *Technical report, Telecom ParisTech* (2014).
 - [55] Mahdisoltani, Farzaneh, Biega, Joanna, and Suchanek, Fabian M. “Yago3: A knowledge base from multilingual wikipedias”. In: *CIDR*. 2013.
 - [56] McCray, Alexa T, Burgun, Anita, and Bodenreider, Olivier. “Aggregating UMLS semantic types for reducing conceptual complexity”. In: *Studies in health technology and informatics* 84.0 1 (2001), p. 216.
 - [57] Meilicke, Christian, Chekol, Melisachew Wudage, Ruffinelli, Daniel, and Stuckenschmidt, Heiner. “Anytime Bottom-Up Rule Learning for Knowledge Graph Completion”. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization. 2019, pp. 3137–3143.
 - [58] Mohamed, Aisha, Parambath, Shameem, Kaoudi, Zoi, and Aboulnaga, Ashraf. “Popularity agnostic evaluation of knowledge graph embeddings”. In: *Conference on Uncertainty in Artificial Intelligence (UAI)*. PMLR. 2020, pp. 1059–1068.
 - [59] Mohamed, Sameh K, Nounu, Aayah, and Nováček, Vít. “Biological applications of knowledge graph embedding models”. In: *Briefings in bioinformatics* 22.2 (2021), pp. 1679–1693.
 - [60] Mohamed, Sameh K, Nováček, Vít, and Nounu, Aayah. “Discovering protein drug targets using knowledge graph embeddings”. In: *Bioinformatics* 36.2 (2020), pp. 603–610.
 - [61] Mohamed, Sameh K., Nováček, Vít, Vandenbussche, Pierre-Yves, and Muñoz, Emir. “Loss Functions in Knowledge Graph Embedding Models”. In: *DL4KG@ESWC*. 2019.

- [62] Nickel, Maximilian, Murphy, Kevin, Tresp, Volker, and Gabrilovich, Evgeniy. “A Review of Relational Machine Learning for Knowledge Graphs”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 11–33. DOI: [10.1109/JPROC.2015.2483592](https://doi.org/10.1109/JPROC.2015.2483592).
- [63] Noy, Natalya F, Shah, Nigam H, Whetzel, Patricia L, Dai, Benjamin, Dorf, Michael, Griffith, Nicholas, Jonquet, Clement, Rubin, Daniel L, Storey, Margaret-Anne, Chute, Christopher G, et al. “BioPortal: ontologies and integrated data resources at the click of a mouse”. In: *Nucleic acids research* 37.suppl_2 (2009), W170–W173.
- [64] Pezeshkpour, Pouya, Tian, Yifan, and Singh, Sameer. “Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 3336–3347.
- [65] Portisch, Jan and Paulheim, Heiko. “The RDF2vec family of knowledge graph embedding methods”. In: *Semantic Web* Preprint (2024), pp. 1–32.
- [66] Qu, Meng, Chen, Junkun, Xhonneux, Louis-Pascal, Bengio, Yoshua, and Tang, Jian. “RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs”. In: *International Conference on Learning Representations*. 2020.
- [67] Ren, Hongyu, Hu, Weihua, and Leskovec, Jure. “Query2box: Reasoning Over Knowledge Graphs In Vector Space Using Box Embeddings”. In: *International Conference on Learning Representations (ICLR)*. 2020.
- [68] Ristoski, Petar, Rosati, Jessica, Di Noia, Tommaso, De Leone, Renato, and Paulheim, Heiko. “RDF2Vec: RDF graph embeddings and their applications”. In: *Semantic Web* 10.4 (2019), pp. 721–752.
- [69] Rivas, Ariam, Collarana, Diego, Torrente, Maria, and Vidal, Maria-Esther. “A neuro-symbolic system over knowledge graphs for link prediction”. In: *Semantic Web* 15.4 (2024), pp. 1307–1331.
- [70] Rossi, Andrea, Barbosa, Denilson, Firmani, Donatella, Matinata, Antonio, and Meri-
aldo, Paolo. “Knowledge Graph Embedding for Link Prediction: A Comparative Analysis”. In: *ACM Transactions on Knowledge Discovery from Data* 15 (Jan. 2021), pp. 1–49. DOI: [10.1145/3424672](https://doi.org/10.1145/3424672).
- [71] Rossi, Andrea and Matinata, Antonio. “Knowledge graph embeddings: Are relation-
learning models learning relations?” In: *EDBT/ICDT Workshops*. Vol. 2578. 2020.
- [72] Rowley, Jennifer. “The wisdom hierarchy: representations of the DIKW hierarchy”. In: *Journal of information science* 33.2 (2007), pp. 163–180.

- [73] Ruffinelli, Daniel, Broscheit, Samuel, and Gemulla, Rainer. “You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings”. In: *ICLR*. 2020.
- [74] Sadeghi, Afshin, Collarana, Diego, Graux, Damien, and Lehmann, Jens. “Embedding knowledge graphs attentive to positional and centrality qualities”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2021, pp. 548–564.
- [75] Safavi, Tara and Koutra, Danai. “CoDEX: A Comprehensive Knowledge Graph Completion Benchmark”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.
- [76] Sardina, Jeffrey, Bernardi, Alberto, and Guéret, Christophe. *Systems and methods for enhancing an artificial intelligence model via a multi-field embedding approach*. US Patent App. 18/351,959. Jan. 2025.
- [77] Sardina, Jeffrey, Costabello, Luca, and Guéret, Christophe. “Veni, Vidi, Vici: Solving the Myriad of Challenges before Knowledge Graph Learning”. In: *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*. IEEE. 2024, pp. 197–203.
- [78] Sardina, Jeffrey, Costabello, Luca, Guéret, Christophe, and Mohammadi, Hossein. “NameE: Capturing Biological Context in KGEs via Contextual Named Graph Embeddings”. In: *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2024, pp. 1–4.
- [79] Sardina, Jeffrey, Debnath, Alok, Kelleher, John D, and O’Sullivan, Declan. “TWIG-I: Embedding-Free Link Prediction and Cross-KG Transfer Learning Using a Small Neural Architecture”. In: *Knowledge Graphs in the Age of Language Models and Neuro-Symbolic AI*. IOS Press, 2024, pp. 106–122.
- [80] Sardina, Jeffrey, Kelleher, John D, and O’Sullivan, Declan. “Extending TWIG: Zero-Shot Predictive Hyperparameter Selection for KGEs based on Graph Structure”. In: *Irish Conference on Artificial Intelligence and Cognitive Science*. Springer. 2024.
- [81] Sardina, Jeffrey, Kelleher, John D, and O’Sullivan, Declan. “A Survey on Knowledge Graph Structure and Knowledge Graph Embeddings”. In: *2025 IEEE 19th International Conference on Semantic Computing (ICSC)*. IEEE. 2025.
- [82] Sardina, Jeffrey, Kelleher, John D, and O’Sullivan, Declan. “TWIG: Towards pre-hoc Hyperparameter Optimisation and Cross-Graph Generalisation via Simulated KGE Models”. In: *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*. IEEE. 2024, pp. 122–129.

- [83] Sardina, Jeffrey and O’Sullivan, Declan. “Structural Characteristics of Knowledge Graphs Determine the Quality of Knowledge Graph Embeddings Across Model and Hyperparameter Choices”. In: *SeWeBMeDA-2022@ESWC*. 2022.
- [84] Sardina, Jeffrey, Sardina, Callie, Kelleher, John D, and O’Sullivan, Declan. “Analysis of attention mechanisms in box-embedding systems”. In: *Irish Conference on Artificial Intelligence and Cognitive Science*. Springer. 2022, pp. 68–80.
- [85] Schlichtkrull, Michael, Kipf, Thomas N, Bloem, Peter, Van Den Berg, Rianne, Titov, Ivan, and Welling, Max. “Modeling relational data with graph convolutional networks”. In: *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*. Springer. 2018, pp. 593–607.
- [86] Sen, Prithviraj, Carvalho, Breno WSR, Abdelaziz, Ibrahim, Kapanipathi, Pavan, Luus, Francois, Roukos, Salim, and Gray, Alexander. “Combining rules and embeddings via neuro-symbolic ai for knowledge base completion”. In: *arXiv preprint arXiv:2109.09566* (2021).
- [87] Shengyuan, Chen, Cai, Yunfeng, Fang, Huang, Huang, Xiao, and Sun, Mingming. “Differentiable neuro-symbolic reasoning on large-scale knowledge graphs”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 28139–28154.
- [88] Shi, Baoxu and Weninger, Tim. “Open-world knowledge graph completion”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [89] Son, Jeongtae and Kim, Dongsup. “Applying network link prediction in drug discovery: an overview of the literature”. In: *Expert Opinion on Drug Discovery* 19.1 (2024), pp. 43–56.
- [90] Speer, Robyn, Chin, Joshua, and Havasi, Catherine. “Conceptnet 5.5: An open multilingual graph of general knowledge”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.
- [91] Sun, Zequn, Zhang, Qingheng, Hu, Wei, Wang, Chengming, Chen, Muhao, Akrami, Farahnaz, and Li, Chengkai. “A benchmarking study of embedding-based entity alignment for knowledge graphs”. In: *Proceedings of the VLDB Endowment* 13.12 (2020).
- [92] Tolkien, John Ronald Reuel. *The Fellowship of the Ring*. HarperCollins Publishers, 2008.
- [93] Toutanova, Kristina and Chen, Danqi. “Observed versus latent features for knowledge base and text inference”. In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Ed. by Alexandre Allauzen, Edward Grefenstette, Karl Moritz Hermann, Hugo Larochelle, and Scott Wen-tau Yih. Bei-

- jing, China: Association for Computational Linguistics, July 2015, pp. 57–66. DOI: [10.18653/v1/W15-4007](https://doi.org/10.18653/v1/W15-4007). URL: <https://aclanthology.org/W15-4007>.
- [94] Trouillon, Théo, Welbl, Johannes, Riedel, Sebastian, Gaussier, Éric, and Bouchard, Guillaume. “Complex embeddings for simple link prediction”. In: *International conference on machine learning*. PMLR. 2016, pp. 2071–2080.
- [95] Vashishth, Shikhar, Sanyal, Soumya, Nitin, Vikram, and Talukdar, Partha. “Composition-based Multi-Relational Graph Convolutional Networks”. In: *International Conference on Learning Representations*. 2019.
- [96] Wang, Meihong, Qiu, Linling, and Wang, Xiaoli. “A survey on knowledge graph embeddings for link prediction”. In: *Symmetry* 13.3 (2021), p. 485.
- [97] Wang, Quan, Mao, Zhendong, Wang, Bin, and Guo, Li. “Knowledge Graph Embedding: A Survey of Approaches and Applications”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743. DOI: [10.1109/TKDE.2017.2754499](https://doi.org/10.1109/TKDE.2017.2754499).
- [98] Wang, Zhen, Zhang, Jianwen, Feng, Jianlin, and Chen, Zheng. “Knowledge graph embedding by translating on hyperplanes”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. 1. 2014.
- [99] Wishart, David S, Feunang, Yannick D, Guo, An C, Lo, Elvis J, Marcu, Ana, Grant, Jason R, Sajed, Tanvir, Johnson, Daniel, Li, Carin, Sayeeda, Zinat, et al. “DrugBank 5.0: a major update to the DrugBank database for 2018”. In: *Nucleic acids research* 46.D1 (2018), pp. D1074–D1082.
- [100] Wu, Jiantao, Orlandi, Fabrizio, O’Sullivan, Declan, and Dev, Soumyabrata. “An ontology model for climatic data analysis”. In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. IEEE. 2021, pp. 5739–5742.
- [101] Xia, Lianghao, Kao, Ben, and Huang, Chao. “OpenGraph: Towards Open Graph Foundation Models”. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. 2024, pp. 2365–2379.
- [102] Yang, Bishan, Yih, Scott Wen-tau, He, Xiaodong, Gao, Jianfeng, and Deng, Li. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases”. In: *Proceedings of the International Conference on Learning Representations (ICLR) 2015*. 2015.
- [103] Zeiler, Matthew D. “ADADELTA: an adaptive learning rate method”. In: *arXiv preprint arXiv:1212.5701* (2012).

- [104] Zhang, Hengtong, Zheng, Tianhang, Gao, Jing, Miao, Chenglin, Su, Lu, Li, Yaliang, and Ren, Kui. “Data poisoning attack against knowledge graph embedding”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2019, pp. 4853–4859.
- [105] Zhang, Rui, Hristovski, Dimitar, Schutte, Dalton, Kastrin, Andrej, Fiszman, Marcelo, and Kilicoglu, Halil. “Drug repurposing for COVID-19 via knowledge graph completion”. In: *Journal of biomedical informatics* 115 (2021), p. 103696.
- [106] Zhang, Wen, Chen, Jiaoyan, Li, Juan, Xu, Zezhong, Pan, Jeff Z, and Chen, Huajun. “Knowledge graph reasoning with logics and embeddings: Survey and perspective”. In: *2024 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE. 2024, pp. 492–499.
- [107] Zhang, Wen, Paudel, Bibek, Wang, Liang, Chen, Jiaoyan, Zhu, Hai, Zhang, Wei, Bernstein, Abraham, and Chen, Huajun. “Iteratively learning embeddings and rules for knowledge graph reasoning”. In: *The world wide web conference*. 2019, pp. 2366–2377.
- [108] Zheng, Shuangjia, Rao, Jiahua, Song, Ying, Zhang, Jixian, Xiao, Xianglu, Fang, Evandro Fei, Yang, Yuedong, and Niu, Zhangming. “PharmKG: a dedicated knowledge graph benchmark for biomedical data mining”. In: *Briefings in bioinformatics* 22.4 (2021), bbaa344.
- [109] Zhou, Jie, Cui, Ganqu, Hu, Shengding, Zhang, Zhengyan, Yang, Cheng, Liu, Zhiyuan, Wang, Lifeng, Li, Changcheng, and Sun, Maosong. “Graph neural networks: A review of methods and applications”. In: *AI open* 1 (2020), pp. 57–81.
- [110] Zietz, Michael, Himmelstein, Daniel S, Kloster, Kyle, Williams, Christopher, Nagle, Michael W, and Greene, Casey S. “The probability of edge existence due to node degree: a baseline for network-based predictions”. In: *GigaScience* 13 (2024), giae001.