Learning Latent Concepts in Knowledge Graphs

Agnieszka Dobrowolska

Submitted in partial fulfillment of the requirements for the

MSc Machine Learning

at the

University College London

September 2020

Supervisors:
Dr. Pasquale Minervini
Dr. Antonio Vergari
Prof. Sebastian Riedel

Disclaimer: *This report is submitted as part requirement for the MSc Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.*

Abstract

Knowledge graphs (KGs) are graph-structured relational databases, which store factual information about the world as triples in the form of: (`subject`, `relation`, `object`). Despite their huge size, KGs are largely incomplete. The task of completing them, i.e. predicting missing links between the existing entities, is called link prediction and constitutes an active area of research. In this work we propose learning and explicitly modelling latent concepts in knowledge graphs as a means of enhancing the representations learnt by knowledge graph embedding (KGE) models. We show that leveraging the latent concept information leads to an improvement on link prediction tasks for benchmark datasets using ComplEx, a state-of-the-art KGE model. Our main contribution is a highly-versatile similarity-based approach for explicitly modelling concepts via data augmentation. The proposed approach is fully unsupervised and scales well with large datasets. Being a data augmentation approach, it is inherently model-agnostic and thus can be used alongside any existing KGE model. Moreover, it can implicitly introduce connectivity patters that are not otherwise captured by KGE models, such as rule-based information or observable graph patterns. We also find that invented concept entities act as attractors, pulling similar entities closer together in the embedding space and thus help to regularise the learnt representation. Beyond leading to better overall link prediction performance, harnessing concept information can also improve generalisation for triples with rare entities and predicates. Lastly, we also present a preliminary framework for generalising this approach via Concept Formation with EM (ConFormE) — a probabilistic framework for jointly learning concept clusters and KGE model parameters by maximising the likelihood of the data.

Acknowledgments

First, I would like to thank both of my supervisors - Dr. Pasquale Minervini and Dr. Antonio Vergari, who were incredibly generous with their time, for their invaluable insights, guidance and endless enthusiasm. Their expertise and determination to ask the right questions have been pivotal to this work.

I would also like to extend my thanks to everyone at the UCL NLP Group for their warm welcome and incorporating us, masters students, so seamlessly into the research team despite the challenges of remote communication. The opportunity to participate in weekly talks and discussions exploring new research areas has been important to my development in the field.

I am also immensely grateful to my family and friends who have supported and encouraged me at every step. This work could not have been completed without you.

Contents

1 Introduction 8

1.1 Motivation 8

1.1.1 A Case for Biologically-inspired Model Design 8

1.1.2 Concepts as Building Blocks 9

1.1.3 Why Learn Concepts in Knowledge Graphs? 9

1.1.4 Data Augmentation 10

1.1.5 Contribution 11

1.2 Aims 12

1.3 Thesis Outline 13

1.4 Notation and Terminology 13

1.4.1 Notation 13

1.4.2 Terminology 13

1.5 Code Repository 14

2 Concept Learning in Humans: Philosophy and Science 17

2.1 A Philosophical Perspective 17

2.2 Defining Concepts 18

2.3 Unsupervised Concept Learning in Humans 18

2.3.1 Can humans learn concepts in an unsupervised manner? . . . 19

2.3.2 Exemplar and Prototype Models 19

2.3.3 Concept Learning as Optimisation 20

2.3.4 Concept Learning as Cluster Formation 20

2.3.5 Incidental and Intentional Category Learning 21

3 Knowledge Graphs and Link Prediction 23
3.1 Frameworks for Factual Knowledge 23
3.2 Uses of Knowledge Graphs 24
3.3 The RDF Framework 25
3.4 Knowledge Graphs: Definitions 25
3.5 The Link Prediction Problem 26
3.6 Evaluating Link Prediction Models 27
3.6.1 Metrics 27
3.6.2 Other Criteria 28
3.7 Statistical Relational Learning for Link Prediction 29
3.8 Probabilistic Observable Models 31
3.9 Knowledge Graph Embeddings Models 32
3.9.1 Translational Models 34
3.9.2 Tensor Factorisation Models 35
3.9.3 ComplEx: The Learning Objective 37
3.10 Data Augmentation in Knowledge Graphs 38
4 Unsupervised Machine Learning: Clustering Algorithms 40
4.1 K-Means 41
4.2 Affinity Propagation 42
4.3 DBSCAN 43
4.4 Spectral Clustering 44
5 Concept-inspired Machine Learning 45
5.1 Concept Formation in Machine Learning 45
5.1.1 Inductive Concept Learning 45
5.1.2 Concept Learning on the Fly 46
5.1.3 Learning Hierarchies in Graphs 47
5.2 Concept Learning in Knowledge Graphs 47
5.2.1 Probabilistic Approaches 47
5.2.2 Non-Euclidean Embeddings for Hierarchy Modelling 48

5.2.3 Learning Hierarchies from Embeddings 49

6 Similarity-based Concept Learning in Knowledge Graphs 50

6.1 Propositionalisation 52

6.1.1 ComplEx Embeddings 52

6.1.2 Weisfeiler-Lehman Kernel 53

6.1.3 In-Range, In-Domain Embeddings 55

6.1.4 Random Paths 56

6.2 Link Prediction with Concept Augmentation 58

6.3 Concept Triples as Regularisers 59

7 ConFormE:

Concept Formation with EM 61

7.1 Joint Concept and Representation Learning 61

7.2 EM for Concept Learning in KGs 63

8 Datasets 65

8.1 Description of Datasets 65

8.2 Statistical Analysis of Data Distributions 66

8.2.1 UMLS 67

8.2.2 WN18RR 67

8.2.3 FB15K-237 68

8.2.4 YAGO3-10 69

9 Experimental Results and Analysis 73

9.1 Training ComplEx 73

9.2 Baselines 74

9.3 Preliminary Experiments 77

9.3.1 Clustering Algorithms for Concept Formation 77

9.3.2 Learning semantically-meaningful clusters 78

9.4 Random Clusters 79

9.5 Entity Representation 82

9.5.1 ComplEx Embeddings 82

9.5.2 Weisfeiler-Lehman Kernel 86

9.5.3 In-range, In-domain 89

9.5.4 Random Paths 92

9.6 Can oversampling reinforce the constraints? 95

9.7 Rank Analysis 97

9.8 Varying Rank 99

9.9 Default Reasoning with Concepts 102

9.10 Summary of Similarity-based Learning 104

9.11 Jointly Learning Concepts and Model Parameters 104

10 Conclusions 109

10.1 Summary 109

10.2 Future Work 111

Chapter 1

Introduction

1.1 Motivation

1.1.1 A Case for Biologically-inspired Model Design

Machine learning algorithms have recently shown a tremendous potential across a wide range of fields such as image recognition [He et al., 2015], language and speech modelling [Devlin et al., 2018, Waibel and Lee, 1990], self-driving cars [Eliot and Eliot, 2017] and anomaly detection [Wang et al., 2020b]. In fact, in some cases performance matches or even surpasses humans, being able to detect breast cancer more accurately than experienced radiologists [McKinney et al., 2020] or beating experts at complex strategy games [Silver et al., 2016]. Yet, despite these impressive achievements, there remains huge scope for further research. For instance, in many computational problems, such as language comprehension or scene understanding, algorithms are still outperformed by humans. This is because abilities such as causal and abstract reasoning, creative problem solving and use of analogy which constitute hallmarks of human intelligence, are notoriously difficult to inject into algorithms.

Hence, it appears intuitive to inspire and advance our model designs by 'reverse-engineering' human intelligence, especially for tasks at which humans excel, having spent millions of years evolving to perform them in an optimal way [Moravec, 1990].

1.1.2 Concepts as Building Blocks

Arguably, one of the most remarkable aspects of human intelligence is the capacity to learn so much from so little data, in a way that is exceptionally rapid and flexible [Lake et al., 2015]. What is it that allows humans to use the sparse observed data to infer such rich and accurate predictive representations? Many researchers [Pothos and Chater, 2002, Goodman et al., 2014, Lakoff and Johnson, 1980, Rosch et al., 1976] believe that man's ability to decompose knowledge into *concepts* plays a central role in this mechanism. Being able to consolidate experiences into elements that can be combined and reassembled in any arbitrary way provides humans with basic building blocks of thought, allowing us to create *generative* representations in a compositional fashion.

Secondly, we observe that human reasoning has statistical underpinnings, encoding beliefs of how likely a situation is, given what we already know. It is supposed that this *predictive* generalisation is enabled using concepts, by summarising typical distributions of objects and events which can then be combined in a combinatorial manner that can be extended to many different scenarios [Goodman et al., 2014].

Moreover, we observe that concepts based on direct abstractions of reality are organised into latent hierarchical structures i.e. organised into hierarchical taxonomic groups such as `rabbit` → `mammal` → `animal`, which allow for better generalisation [Rosch et al., 1976]. For instance, one may have never heard of a Pika before, but if one manages to classify it as a mammal, plausible assumptions about its behaviour can be made.

1.1.3 Why Learn Concepts in Knowledge Graphs?

Knowledge graphs (KG) are frameworks which allow for storing human knowledge in a structured way. They represent facts as triples in the form of (*subject*, *relation*, *object*), such as (`UCL`, `locatedIn`, `London`), where `UCL` and `London` are entities represented as graph nodes and `locatedIn` is a binary relation between them, corresponding to a labelled graph edge. While most commonly-used KGs are made up of billions of triples, they are vastly incomplete [Nickel et al., 2016]. Hence, the task of completing KGs by finding the missing link between existing entities, often referred to as *link prediction*, is an active area of research. Models developed specifically for link prediction tasks can be seen as generative models; capable of producing any arbitrary triple using the existing entities and predicates, and assessing the likelihood of it being true. In this paradigm, if we think of entities as analogous to first-level concepts since they correspond to abstractions of reality, we can see a resemblance between the generative processes in the human mind and in the link prediction model. However, the existing state-of-the-art knowledge graph link prediction models, which largely comprise of embedding models, focus on learning relation patterns, such as symmetry or inversion, and *"fail to model semantic hierarchies, which are common in real-world applications"* [Zhang et al., 2019]. Given how central concepts are to human cognition, we propose a new task of learning and explicitly modelling latent concepts in KGs.

1.1.4 Data Augmentation

One of the challenging aspects of link prediction in KGs is having to generalise from very little data — for while the KGs themselves can be huge, information about individual entities tends to be sparse, with most entities being connected with very few predicates (Section 8). As a result, powerful link prediction models, such as tensor factorisation models, easily overfit to the training data and struggle to generalise, positioning the entity vector representations too far apart in the embedding space.

One approach, which has been successfully applied to address the data scarcity problem across diverse fields of ML, is data augmentation. Broadly speaking, the aim of this technique is to artificially expand existing datasets. There are many potential advantages of this approach: it has been shown to make models more robust and less likely to overfit by acting as a regularizer, it can allow for encoding prior knowledge about the data and increase number of training examples [Dao et al., 2018].

In this work, we propose performing data augmentation in KGs by explicitly assigning each entity to an invented concept entity. We hypothesise that the cluster entities will act as ‘attractors’ and pull the participating entities closer together in the embedding space. This, in turn, should act as a regulariser and lead to improved link prediction performance.

1.1.5 Contribution

In this thesis we present a novel way for learning and explicitly modelling latent concepts in KGs. As our method relies on data augmentation, it can be easily integrated with any existing knowledge graph embedding model. Concepts are learnt in a fully unsupervised manner, hence prior knowledge about what concepts should be formed is not necessary. In contrast to the existing concept learning models for KGs, this method scales easily even to large KGs. We show that our approach is not only capable of learning semantically-meaningful concepts but also leads to significant improvements in link prediction on benchmark datasets.

- We present two distinct methods for concept learning. In the *similarity-based* concept learning approach, concepts are created prior to the link prediction task by generating vector representations of entities using propositionalisation. The embeddings are then clustered and concept assignment triples are introduced into the training dataset. We also propose generalisation of this approach — a method for jointly learning concepts clusters and model parameters which optimises link prediction performance. Specifically, we present an EM-like algorithm where cluster assignments are made in an iterative, probabilistic manner that maximises the likelihood of the data.

- While knowledge graphs were designed to be inherently semantically understandable, the process of link prediction is often not: most current state-of-the-art models learn vector representations for each entity and relation, and use a score function to estimate how likely a given triple is to be true. However in some fields, such as in the biomedical domain, explainability of predictions is of paramount importance. We show that by clustering model embeddings and exploring which entities are positioned closely to each other in embedding space, we can gain a qualitative understanding of the latent concepts learnt by the model.

- Exploratory statistical analysis of KGs is rarely performed in the field — we demonstrate the importance of this step and suggest it should become an industry standard.

- Lastly, performance of link prediction models is usually summarised by quoting test MRR. However, being a mean statistic, MRR gives limited information about the reciprocal rank distribution for different sub-populations. In this work, we wish to show that our understanding of model performance can be enhanced by looking beyond the mean.

1.2 Aims

The work presented in this thesis will attempt to answer the questions outlined below.

- Can we learn semantically-meaningful concepts in knowledge graphs? How do we assess whether they are meaningful?
- What effect does explicit modelling of concepts have on model performance in link prediction tasks?
- What is the effect of adding random concepts to the KGs?
- Does the number of concepts learnt matter? How do we form the optimum number of concepts?
- Which entity representations introduce new, latent information? Does the choice of clustering algorithm have an effect on the cluster quality and model performance?
- Can we extend our initial model to jointly learn concept assignments and model parameters?

1.3 Thesis Outline

We begin by exploring the process of concept formation in humans from a philosophical and biological perspective (Section 2). Next, we formally introduce knowledge graphs (Section 3), the link prediction problem (Section 3.5), the existing models for knowledge base completion (Section 3.7) and consider examples of successful data augmentation in KGs (Section 3.10). Section 4 reviews selected approaches for unsupervised learning using clustering algorithms. Next, we consider existing approaches for concept learning (Section 5) in both, broader fields of machine learning and those designed specifically for KGs. In Section 6 we present our main contribution, the similarity-based concept learning approach. In Section 7 we introduce an extension of our work for learning optimal concept assignments under a probabilistic framework. Section 8 introduces the benchmark datasets and presents a statistical analysis of entity and predicate distributions in each dataset. The experimental results and discussions are presented in Section 9 followed by a summary our findings and discussions of future work in Section 10.

1.4 Notation and Terminology

1.4.1 Notation

The mathematical notation used in this work is largely based on the conventions defined in Nickel et al. [2016], a recent survey of statistical relational learning for KGs, or introduced on the spot. A summary of our notation can be found in tables 1.1 and 1.2.

1.4.2 Terminology

Recently, there has been criticism towards the use of words such as *understanding*, *comprehension* or *recall of factual knowledge* in relation to machine learning models [Bender and Koller, 2020]. Such criticisms are certainly justified, as the human intelligence and reasoning abilities does not exist in a vacuum — they co-exist in a rich ecosystem with the memory, emotions, a constant stream of sensory experience, intentions, will, common sense, creativity, intuition and drives like curiosity. Hence, words such as 'understanding' have connotations far richer than what can be achieved using a series of isolated back-propagation passes. Nonetheless, there is a lack of succinct and well-defined terminology for referring to behaviours exhibited by machine learning algorithms. Hence, we would like to clarify that while we sparingly use terms which originated as describing human behaviour, such as *understanding* or *reasoning*, this is done merely for the ease of communication, as a loose metaphor applied to a different paradigm, and not meant as a declarative statement about models exhibiting human-like abilities.

1.5 Code Repository

The code accompanying this work can be found in the following public repository: https://github.com/AgaDob/concept_formation_in_knowledge_graphs.

Table 1.1: Summary of Notation

Symbol	Meaning
e	scalars
$\mathbf{e}$	column vectors of size N_1
$\mathbf{E}$	matrices of size $N_1 \times N_2$
$\underline{\mathbf{E}}$	tensors of size $N_1 \times N_2 \times N_3$
$\mathbf{E}_k$	k^{th} frontal slice of a tensor $\underline{\mathbf{E}}$, constituting a $N_1 \times N_2$ matrix
e_{ijk}	$(ijk)^{\text{th}}$ element of a tensor $\underline{\mathbf{E}}$
$[\mathbf{e}; \mathbf{d}]$	vertical stacking of vectors $\mathbf{e}$ and $\mathbf{d}$
$\mathbf{e} = \text{vec}(\mathbf{E})$	converting a $N_1 \times N_2$ matrix $\mathbf{E}$ into a $N_1 N_2$ vector by vertically stacking all columns
$\mathbf{e}^\top \mathbf{d} = \sum_{i=1}^{N} e_i d_i$	inner vector product of two N-sized vectors
$\mathbf{e} \circ \mathbf{d}$	Hadamard product which performs element-wise multiplication of two vectors
$\mathbf{ED}$	matrix multiplication
$\|\mathbf{e}\|_p = \sqrt{\sum_i e_i^p}$	L_p vector norm
$\|\mathbf{E}\|_F = \sqrt{\sum_i \sum_j e_{ij}^2}$	Frobenius matrix norm
$\mathbf{1}$	vector of ones
$\mathbf{I}$	identity matrix

Table 1.2: Summary of Knowledge Graph Notation

Symbol	Meaning
$\mathcal{G}$	knowledge graph (KG)
N_e	number of entities
N_r	number of relations
N_c	number of concepts
$\mathcal{E} = \{e_1, \ldots, e_{N_e}\}$	set of entities in a KG
$\mathcal{R} = \{r_1, \ldots, r_{N_r}\}$	set of relations in a KG
$\mathcal{C} = \{c_1, \ldots, c_{N_c}\}$	set of concept entities in a KG
$\mathcal{S} = \{S_1, \ldots S_{N_c}\}$	set of concept cluster sets, where $S_i \subseteq \mathcal{E}$, $S_i = \{e_1, \ldots, e_K\}, 1 \leq K \leq N_e$
$\mathcal{D}$	dataset of observed triples
$\mathbf{e}_i$	embedding representation of entity e_i
$\mathbf{c}_i$	embedding representation of concept entity c_i
$\mathbf{w}_i$	embedding of relation r_i
$\mathbf{E}$	entity embedding matrix
$\mathbf{C}$	concept entity embedding matrix
$\mathbf{W}$	relation embedding matrix

Chapter 2

Concept Learning in Humans: Philosophy and Science

We begin by exploring concept formation from a philosophical and biological perspective. In this chapter we present a review of literature that is broadly related to concept learning in humans which has inspired the method we present in this work.

2.1 A Philosophical Perspective

It is difficult to provide a general definition of a concept without already committing to a particular philosophical school of thought, hence we briefly summarise them here as defined in [Sullivan, 1957]:

- In *moderate realism*, based on the Aristotelian school of thought, concepts, also referred to as *universals*, are thought to exist solely in the mind of the knower. The term *moderate realism* thus arises from the notion that concepts truly exist but their existence does not extend beyond the mind of the knower.

- By contrast, in *absolute realism*, which is founded on Plato's philosophy and also known as *Platonic realism*, concepts are objective and exist outside of the human mind as ideals, though their existence is not spatio-temporal. According to Platonic philosophy, man is not capable of knowing things 'as they really are' — an idea which he famously explains using the *allegory of the cave* [Plato, 2010].

- In *conceptualism*, based on the work of Kant, it is supposed that the mind requires a 'form' or a 'category' onto which thoughts are cast. Thus, he asserts that concepts arise only in response to the the structural need of the mind. Hence, one's knowledge of reality can be very limited as every thought is recast according to the current configuration to the mind.

The theories discussed thereafter are most readily reconciled with Aristotelian philosophy, which it is more conservative in its assumptions and does not carry with it a fundamental mistrust in the observable world — a characteristic that is central to both, Platonic realism and conceptualism.

2.2 Defining Concepts

In the field of educational psychology, Bloom's taxonomy [LW et al., 2001] distinguishes between four categories of knowledge: factual, conceptual, procedural and meta-cognitive. Factual knowledge encompasses knowledge of facts, specific details and elements, and as such it does not promote understanding. Conceptual knowledge, meanwhile, is obtained through inferring or abstracting from facts. By observing that different instances share common attributes, general ideas such as *elephant* or *friendship* are formed. As such, they are broad and universal, supplying a framework for understanding and allow for transferring of existing knowledge to new situations.

Accordingly, we go on to define concepts as '*mental representations that are used to discriminate between objects, events, relationships, or other states of affairs*' which are learnt from observable data that can be noisy or sparse [Goodman et al., 2008].

2.3 Unsupervised Concept Learning in Humans

There exists a vast, and at times contradictory, deposit of literature on how concept formation occurs in human beings, with different theories developing across the fields of behavioural psychology, cognitive psychology and neuroscience. In this section we will review a selection of theories focused on *unsupervised* categorisation i.e. where the learner spontaneously establishes a classification for a set of items that is intuitive or natural, with no *a priori* structure. It is worth noting that the research concerning supervised concept learning is far more extensive than that into unsupervised learning, with the latter being much less understood. We are particularly interested in theories which have been tested using not only behavioural but also computational studies.

2.3.1 Can humans learn concepts in an unsupervised manner?

It is thought likely that conceptual structures in the human mind are constructed using both supervised and unsupervised learning mechanisms. Pothos and Chater, 2002, for instance, point out that when children learn new words they are often able to successfully generalise from a small number of 'labelled' examples [Xu and Tenenbaum, 2005]. This, in their view, suggests that observing unsupervised data (i.e. hearing many words with no knowledge of their meaning) gives rich prior constraints as to what categories are conceivable and can provide a valuable basis on top of which supervised learning can occur in a more efficient manner. Recent experiments reported in [Roads and Love, 2020] support the premise that meaningful information can indeed be inferred even when no explicit labels are provided.

2.3.2 Exemplar and Prototype Models

Questions concerning the *type* of memory representations created in the brain during concept formation have been at the heart of cognitive categorisation research for decades [Zeithamova et al., 2019]. Broadly speaking, there are two schools of thought. The exemplar model [Mack et al., 2013] supposes that the brain represents each concept using a specific concept example which it has encountered. Meanwhile, the prototype model [Bowman and Zeithamova, 2018] holds that concepts are represented using generalised representations, also called 'prototypes', which are constructed by abstracting across the exemplars. While the two hypothesis have been long thought of as contradictory, recent research by [Bowman and Zeithamova, 2019] shows that concept formation may involve both, specific and generalised representations, depending on the learning conditions thus reconciling the two theories.

2.3.3 Concept Learning as Optimisation

One way of modelling learning in humans is by considering the learning process as an optimisation process, driven by either maximising a value (reward) or minimising the surprise (cost, prediction error). This idea was famously encapsulated by Karl Friston using the Free Energy Principle [Friston, 2010]. As such, he proposes that many diverse brain theories can be unified by viewing the brain as '*a generative model of the world it inhabits*'. However, to date few detailed concept learning models have been proposed which use active inference [Smith et al., 2020], i.e. which are driven by minimisation of free energy.

2.3.4 Concept Learning as Cluster Formation

Perhaps one of the simplest cognitive models for modelling unsupervised concept formation in humans, proposed by [Pothos and Chater, 2002], poses it as a clustering problem, where entities are assigned to a cluster based on some notion of similarity. The process is driven be the *simplicity principle*: authors observe that when experiment participants were asked to classify objects created from a set of *separable* dimensions, the participants made their decisions based on a single dimension. It was only when the test objects were constructed such that it was not possible to split on a single feature that the participants considered multiple dimensions. This, in connection with the observation that simplicity may play an important part in human cognition [Chater, 1999], leads Pothos and Chater to believe that people prefer creating categories which provide the simplest representation of the participating items — a finding that was supported using four independent behavioural experiments with human subjects.

Independently, Love proposed SUSTAIN — a computational model of concept learning in humans [Love et al., 2004]. While it also poses concept learning as a clustering problem, it is inspired by a different hypothesis, based on the premise that category learning can be seen as compression. Every concept corresponds to the mean of the 'experiences' or 'observations' which participate in it. This can be formulated as an *on-line* learning cluster assignment problem where upon receiving a new observation, two different update steps are possible: an incremental update, where the new entity is assigned to an existing cluster and the mean of that cluster is updated, or, if the similarity between the cluster most similar to the current observation and the current observation is below a threshold, a new cluster is recruited, i.e. a new concept is created. SUSTAIN also allows for assigning different weights to the various dimensions of the observations to resemble the attention mechanisms in the brain. Despite the evident simplicity of the methods, recent experiments seem to suggest that the clustering representations faithfully resemble the patterns of activity observed in the medial temporal lobe [Mack et al., 2016, 2019]. There is evidence that there may exists human 'concept cells', where each individual cell corresponds to a specific concept, analogous to encoding a concept using a cluster in high-dimensional space [Mok and Love, 2019]. Moreover, [Love et al., 2004] makes a compelling argument in favour of cluster models: such models not only provide a flexible representational medium but also allow for building knowledge structures 'as needed' — starting with simple representations and increasing complexity as more data becomes available.

2.3.5 Incidental and Intentional Category Learning

In [Love, 2003], the author also explores the distinction between *incidental* and *intentional* unsupervised learning. In incidental unsupervised learning categories are created using only a similarity measure, with no intention or goal driving the learning process. Experimentally, this corresponds to the subjects not being aware that they are in a learning task and their effort being directed towards another task. In intentional unsupervised learning, on the other hand, the subjects are aware that they are in a learning task and they focus their attention on the learning task, actively searching for patterns. An interesting hypothesis by [Wattenmaker, 1999] suggests that incidental learning will be more based on shallow similarity-based comparisons, while intentional learning, in both supervised and unsupervised settings, is more likely to promote rule-based learning.

Chapter 3

Knowledge Graphs and Link Prediction

3.1 Frameworks for Factual Knowledge

The notion of a dataset is in many ways central to the field of Machine Learning. It provides a structure for storing information in a format that is both, human and machine readable. Many types of data can be stored as tabular datasets, where each datapoint can be treated independently. However, not all information can be readily adapted to this format. Consider for instance protein interactions, or social networks, or even factual statements such as 'bees make honey' — information that is by nature *relational*. The tabular format is inherently ill-suited to this setting. In response to a need for storing complex structured and unstructured information, Knowledge Bases (KBs) emerged. Knowledge Graphs (KGs) can be thought of graph-structured knowledge bases which provide a framework for storing and representing human knowledge about the world as entities and relationships between them, thus providing an intuitive way to store relational data.

3.2 Uses of Knowledge Graphs

This way of representing relational knowledge in a symbolic, human-readable form has a long-standing tradition in fields of logic and artificial intelligence [Davis et al., 1993]. Recently, there has been a growing interest in using KGs for driving forward various 'Big Data' applications in scientific and commercial settings [Noy et al., 2019]. Large-scale generic KGs have recently been collaboratively created, such as YAGO [Suchanek et al., 2007], DBpedia [Lehmann et al., 2014], NELL [Carlson et al., 2010], Freebase [Bollacker et al., 2008] and Wordnet [Miller et al., 1991], as well as domain-specific KGs such as those in the biomedical field: Hetionet [Himmelstein et al., 2017], LinkedLifeData [Momtchev et al., 2009], Neurocommons [Ruttenberg et al., 2009], Bio2RDF [Belleau et al., 2008] and most recently the COVID-19 Literature Knowledge Graph [Wang et al., 2020a]. Most search engines such as Google [Eder, 2012] and Microsoft's BING have integrated KGs into their retrieval models resulting in a significant increase in retrieval quality. The use of KGs in retrieval tasks allows for entity identification and disambiguation, as well as providing links to similar entities and enhancing query results with semantically-structured summaries. KGs also constitute repositories of structured knowledge which are essential for digital assistants, such as Alexa, Cortana and Siri. Other downstream tasks which utilise KGs include question answering [Ferrucci, 2012, Fader et al., 2014], visual relationship detection [Baier et al., 2018], word sense disambiguation [Agirre et al., 2014, Navigli and Velardi, 2005], co-reference resolution [Dutta and Weikum, 2015] and semantic parsing [Berant et al., 2013].

While in-depth considerations on the topic of Knowledge Graph construction are beyond the scope of this work, it is worth noting that this process is increasingly moving from a manual one towards an automated one. Broadly, there are two types of automated approaches: semi-structured and unstructured. Semi-structured approached extract triples from semi-structured text, such as Wikipedia info-boxes by making use of rules (either hand-crafted or learned) and regular expressions. Meanwhile unstructured approached use machine learning and natural language processing methods to obtain facts from unstructured text. Recent reviews on KB construction can be found in [Weikum and Theobald, 2010, Wu et al., 2018].

3.3 The RDF Framework

This work will loosely follow the *Resource Description Framework* (RDF) [1] which is a widely-accepted standard for knowledge representation on the Web. It is based on the idea of expressing facts as binary relationships in the form of subject-relation-object triples and defining specific classes for entities, and properties for relations.

While the existence of a particular triple always means that the participating entities are in a relationship of the given type (i.e. denotes a fact), the absence of a triple can be interpreted differently, depending on the assumptions we make. Under the **closed world assumption** triples which are not present in the graph indicate a false relationship. Meanwhile, under the **open world assumption** triples which are not present in the KG correspond to unknown knowledge, thus can be true or false. As large KGs tend to be very incomplete [Nickel et al., 2016] we choose the more cautious approach and make the open world assumption.

3.4 Knowledge Graphs: Definitions

In the RDF framework a knowledge graph (KG), $\mathcal{G}$, can be considered a labelled directed multi-graph where the vertices represent entities and each directed, labelled edge denotes a *relation* between two entities, also known as a predicate. A source vertex is known as a *subject* and the destination vertex as an *object*. Hence, an RDF (*subject*, *relation*, *object*) triple can be represented as (s, r, o). Let $\mathcal{E}$ be the set of all

[1] https://www.w3.org/standards/semanticweb/

entities in the KG:

$$\mathcal{E} = \{s \mid \exists(s, r, o) \in \mathcal{G}\} \cup \{o \mid \exists(s, r, o) \in \mathcal{G}\} \tag{3.1}$$

$$= \{e_1, \dots e_{N_e}\}, \text{ where } N_e = |\mathcal{E}| \tag{3.2}$$

and $\mathcal{R}$ the set of all relations:

$$\mathcal{R} = \{r \mid \exists(s, r, o) \in \mathcal{G}\} \tag{3.3}$$

$$= \{r_1, \dots, r_{N_r}\}, \text{ where } N_r = |\mathcal{R}| \tag{3.4}$$

Let $\mathcal{S} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ be the space of possible triples of $\mathcal{G}$. Hence, formally: $\mathcal{G} = \{(s, r, o)_i\} \subseteq \mathcal{S}$.

KGs usually obey type constraints; for instance, the `marriedTo` predicate can only connect two people, not things — they can be thought of as hard rules. Various *soft* statistical patterns that may not be universally true yet are of use for prediction tasks often are also present in KGs. One such property is *homophily* [Nickel et al., 2016] which refers to the tendency of similar entities to be related.

3.5 The Link Prediction Problem

Despite their large size, most KGs remain vastly incomplete [Nickel et al., 2016] and may even contain edges that are incorrect [Angeli and Manning, 2013]. For instance in DBpedia, a KG containing factual knowledge about the world, 58% of scientists are missing information about what they are known for, while 60% of people are missing information about their place of birth [Krompaß et al., 2015].

Finding all valid triples manually is intractable and impractical, especially given that most KGs are expanded on regular basis. This has motivated research into models capable of predicting new links between existing entities in KGs to discover new facts about the world and thus aid in completing KGs. This task is usually referred to as *Link Prediction* or *Knowledge Base Completion* (KBC). Formally, given a training set comprising a KG and a set of query triples for testing we want to construct models capable of accurately completing test triples of the form (?, relation, object) or (subject, relation, ?) — for example (`Edinburgh`, `CapitalOf`, `?`) where `?` is `Scotland`.

3.6 Evaluating Link Prediction Models

Before discussing the different link prediction models, it is essential to first define the desirable properties of such algorithms.

3.6.1 Metrics

Conventionally, the main criteria for assessing link prediction models revolve around evaluating their predictive power. One of the most commonly used metrics is the **Mean Reciprocal Rank** (MRR). First, for each test triple the true subject, s, is removed and the score function is used to assign a value to the triples completed with each of the entities $e \in \mathcal{E}$, where $\mathcal{E}$ is a set of all the entities present in the KG. Next, the assignments are ranked from highest to lowest and we compute the reciprocal ranks of the correct entities. This procedure is then repeated with the object entity for each of the triples. Lastly, we compute a mean of all of the reciprocal ranks and we obtain the MRR. Formally:

$$\mathrm{MRR} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \frac{1}{\mathrm{rank}_i} \tag{3.5}$$

where $|\mathcal{D}|$ is the number of test triples and rank_i corresponds to the position awarded to the correct candidate entity in a list of candidate entities ordered by their awarded scores for the i^{th} test triple.

This MRR is often referred to as *raw*, as the candidate triples are constructed using all of the entities $e \in \mathcal{E}$. However, there are problems with this formulation. To illustrate the issue, suppose we wish to compute the reciprocal rank for a triple (`Dog`, `IsA`, `Animal`) and corrupting the subject we have (`?`, `IsA`, `Animal`). It is likely that there are many other entities connected to the `Animal` entity using an `IsA` predicate, for instance (`Cat`, `IsA`, `Animal`). Suppose that the model gives a higher rank to the (`Cat`, `IsA`, `Animal`) triple instead — then we can end up penalising the model for ranking one correct answer over another.

Hence, in practice we compute the *filtered* MRR instead, which only considers the ranking of the correct triple against the incorrect triples. To achieve this, for every true (s, r, o) triple define $\mathcal{E}_T = \{s' \mid (s', r, o) \notin \mathcal{G}\} \cup \{s' \mid s' = s\}$, where s is the true subject. Next, only compute scores for candidate triples constructed using entities $e \in \mathcal{E}_T$, and repeat this procedure for the object corruption.

Another commonly used metric is the **HITS@K**, where K is usually {1, 3, 5, 10, 50, 100}. It is computed by counting how many correct triples are ranked in the top K positions and then dividing the sum by $|\mathcal{D}|$, the total number of test triples.

While it is common practice to compare performance of different models by considering MRR, we argue that using solely a mean statistic such as MRR to assess model's performance can hide discrepancies and gives a shallow insight into model performance. One way in which we can gain a deeper insight is to consider the MRR for different sub-population of test triples. For instance, suppose that we divide the test triples into *very rare*, *rare*, *common* and *very common*, based on the frequencies of their participating entities and predicates in the train set (a detailed consideration on how this can be done follows in Section 9.7). Next, by considering the reciprocal ranks for each of these sub-populations we gain a more profound insight into model's strengths and weaknesses. This is the approach we will follow in this work and we suggest that a more in-depth analysis of the reciprocal ranks should become an industry standard.

3.6.2 Other Criteria

Besides assessing model's ability to generalise, there are a number of other important criteria which should be taken into consideration.

As mentioned above, KGs tend to be very large and continually expanding, hence **efficiency** and **scalability** of these methods are of paramount importance. For instance, consider that KG benchmark datasets, such as WN18RR [Miller et al., 1991] which is made up of over 40k entities, consist only of a subset of the entire KG. What is more, while there exists a combinatorial number of *possible* triples, the fraction of triples which are in fact *true* remains tiny. Hence, one of the key challenges facing the community is developing methods capable of exploiting the sparsity of relationships in an efficient and scalable manner. In fact, an ideal model should scale at most linearly w.r.t the data size hence linearly w.r.t N_e, the number of entities, linearly w.r.t N_r, the number of predicates and linearly w.r.t N_d, the number of observed triples in the dataset. Moreover, as model **explainability** becomes an increasingly important issue in Machine Learning [Bianchi et al., 2020], especially in application such as biomedical research, it is important to consider the cost of trading interpretability of predictions for increased model performance. KGs are by nature designed to be human-readable but interpretability can be lost when using latent feature models, which are to some extent 'black box models', as opposed to mining observable patterns. Lastly, as automatic knowledge base construction gains traction, the probability of 'noisy' or false facts being added to the KGs increases. Hence, the **robustness** of these models to noise in the form of corrupted triples should also be assessed.

3.7 Statistical Relational Learning for Link Prediction

This work will focus on addressing the link prediction problem using the Statistical Relational Learning (SRL) framework [Getoor and Taskar, 2007]. Also known as probabilistic inductive logic programming, it brings together 3 disciplines: probability, logic, and learning [Raedt and Kersting, 2010]. Aims of SRL can be loosely summarised as learning representations to capture relationship patterns between nodes in graph-structured data.

As above, we assume a KG, $\mathcal{G}$, comprising a set of entities $\mathcal{E} = \{e_1, \dots e_{N_e}\}$ and relations $\mathcal{R} = \{r_1, \ldots, r_{N_r}\}$. At times rather than considering the KG as a graph it is more intuitive to group every *possible* triple $x_{sro} = (e_s, r_r, e_o)$ in an adjacency tensor $\underline{\mathbf{Y}} \in \{0,1\}^{N_e \times N_r \times N_e}$ where each dimension of the tensor corresponds to (*subject*, *relation*, *object*), respectively. As $\underline{\mathbf{Y}}$ is a random variable, its every realisation gives rise to a different KG. Each entry of the adjacency tensor $\underline{\mathbf{Y}}$ is then defined as follows:

$$y_{sro} = \begin{cases} 1, & \text{if the triple } x_{sro} \text{ exists} \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

where the interpretation of $y_{sro} = 0$ varies depending on what assumption about the world we make.

If we make a conditional independence assumption, we can define a joint distribution over $\underline{\mathbf{Y}}$:

$$\mathrm{P}(\underline{\mathbf{Y}} \mid \mathcal{D}, \mathbf{\Theta}) = \prod_{s=1}^{N_e} \prod_{r=1}^{N_r} \prod_{o=1}^{N_e} \mathrm{Ber}\left(y_{sro}; \sigma\left(f\left(x_{sro}; \mathbf{\Theta}\right)\right)\right) \tag{3.7}$$

where $\mathcal{D}$ is the set of observed triples, $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function and Ber represents the Bernoulli distribution:

$$\mathrm{Ber}(y \mid p) = \begin{cases} 1-p & \text{if } y = 0 \\ p & \text{if } y = 1 \end{cases} \tag{3.8}$$

Making the independence assumption allows us to define the **score function** $f(x_{sro}; \mathbf{\Theta})$ which operates on a given triple x_{sro} and measures how confident the model is about its existence, given the model parameters $\mathbf{\Theta}$. Learning a score function which assigns high scores to likely and existing triples and low scores to false triples is one of the central aims under the SRL framework.

SRL models for link prediction can be broadly divided into two categories:

- **Probabilistic Observable Models** assume that the y_{sro} are conditionally independent given the graph features. As such, they use the formulation from

Equation 3.7 and focus on capturing correlation between entities and predicates by using statistical models to mine *observable* patterns present in the graphs.

- **Latent Feature Models**, also known as **Knowledge Graph Embedding** models (KGE) or Energy-based models, aim to capture the correlation using *latent* variables and store it in entity and predicate vector representations. Rather than optimising Equation 3.7, KGE models optimise the score function $f(\cdot)$, where the definition of $f(\cdot)$ varies depending on the model. It is worth noting that one can still obtain the probabilities for score-based models using Platt scaling [Platt, 1999].

For the purposes of this project we require a link prediction model to serve as a baseline and to allow us to compare the performance pre and post explicitly augmenting the KG with concept entities. Hence, the selection criteria for such a model focus not so much on benchmark performance, though that is also worth considering, but more so on the *stability* of the model and its ability to model a wide range of relation types.

3.8 Probabilistic Observable Models

In Probabilistic Observable Models, also known as graph feature models, we extract the observable graph features to predict existence of unobserved edges. This approach has the advantage of making predictions which are entirely explainable using the existing triples.

One way to infer new triples in this way is by using rule-based, symbolic learning — making use of observed triples and logical inference. For instance, starting from a triple (`London`, `capital`, `UK`) one can infer that (`London`, `locatedIn`, `UK`). Such inference can be made using background information stored in logical axioms which determine that the capital of a country must also be located in the said country. Expressing this using first-order logic we have that $\forall X, Y : \text{capital}(X, Y) \implies \text{locatedIn}(X, Y)$. Logic-based link prediction can be performed either in a deterministic way, using logical deduction, or in a probabilistic manner to account for the uncertainty in data [Richardson and Domingos, 2006].

However, most KGs contain many observed triples and few (if any) rules or axioms [Dong et al., 2014] which seriously hinders the success of this approach. Hence, it is necessary to either handcraft the rules or infer them using techniques such as inductive logic programming (ILP) [Muggleton and de Raedt, 1994]. Yet even then, the learned rules usually only account for a subset of patterns present in KG and learning rules which are useful is often challenging. Moreover, this approach suffers from problems with complexity and does not scale well with large KGs [Trouillon et al., 2017].

Other observable models make use of homophily (Section 3.4) and attempt to measure similarities between different entities. Potential measures include using the neighbourhood of each node or via various graph paths analysis [Lü and Zhou, 2011]. A wide range of similarity indices has been proposed and while this approach is now widely used for relatively small KGs that contain only a single predicate, their performance on large multi-relational KGs has not shown much promise.

3.9 Knowledge Graph Embeddings Models

A more promising alternative is to represent entities and relations using low-dimensional embeddings (which can be vectors or matrices) and account for the existing triples by considering the latent features of entities. This family of models is known as Latent Feature Models or, more commonly, Knowledge Graph Embedding models (KGE) [Nickel et al., 2011, Bordes et al., 2013, Trouillon et al., 2016, Dettmers et al., 2017]. The theoretical considerations which justify this approach are beyond the scope of this work but can be found in [Orbanz and Roy, 2013].

In recent years state-of-the-art results on benchmark datasets have been achieved predominantly using KGE models. Moreover, in the context of this work, using KGE models gives us the ability to generate explicit representations for each entity, thus can be seen as one way of achieving propositionalisation i.e. capturing connectivity patterns in a vector form (see Section 6.1). Furthermore, by learning representations for the invented concept entities we could in principle use them in an inductive learning setting.

Broadly speaking, we can categorise KGE models into three groups: **translational models**, such as TransE [Bordes et al., 2013] and RotatE [Sun et al., 2019], **tensor factorization models**, such as RESCAL [Nickel et al., 2011], DistMult [Yang et al., 2014] and ComplEx [Trouillon et al., 2016], and **convolutional models**, such as ConvE [Dettmers et al., 2017]. It is worth bearing in mind the recent studies [Ruffinelli et al., 2020] which have shown that all of these models are highly sensitive to hyperparameter settings, loss function choices, training types (negative sampling vs 1vsAll scoring) and choice of regularizer, thus making comparing performance across different models challenging. In this work we are interested in choosing a model that is *stable* i.e. able to achieve consistently good performance across a wide range of datasets. As modern convolutional models such as ConvE can be particularly sensitive to training approaches [Ruffinelli et al., 2020], in this section we focus predominantly on the first two groups instead.

The main difference between the different translational and tensor factorization models lies in 1) the representation of entities and predicates — creating embeddings using vectors or matrices, or both; and 2) the choice of score function, i.e. way that the entity and relation embeddings are combined to represent a triple. In translational models, relations are thought of as 'translations' from the subject entity to the object entity, whereas tensor factorisation models commonly use a multiplicative approach instead and use matrices in vector space to represent relationships.

While some KGE models also incorporate external information such as textual embeddings [Xie et al., 2016], visual information [Wang et al., 2019] or numerical information [Garcia-Duran and Niepert, 2017] here we only consider 'pure' KGE models, which only use relational information.

Formally, $\mathbf{E} \in \mathbb{R}^{N_e \times k}$ denotes the entity embedding matrix, $\underline{\mathbf{W}} \in \mathbb{R}^{N_r \times k \times k}$ is the relations embedding tensor and $\mathbf{W} \in \mathbb{R}^{N_r \times k}$ is the relations embedding matrix (the use of either $\underline{\mathbf{W}}$ or $\mathbf{W}$ depends on the model). We will denote the vector embeddings for the subject s, relation r and object o using $\mathbf{e}_s$, $\mathbf{w}_r$ and $\mathbf{e}_o$ respectively, where usually $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \in \mathbb{R}^k$ and $k \in \mathbb{N}$. This dimension of the learned representation, k, is also sometimes referred to as *rank*.

3.9.1 Translational Models

In translational models, also known as *distance* models, the probability of a triple is a function of the distance separating the entity embeddings. Thus, entities whose latent representations lie close are more likely to have a relationship. The score function corresponds to some distance metric: $f(x, y) = -d(\mathbf{x}, \mathbf{y})$. For instance, in **TransE** [Bordes et al., 2013] entities and relations are assumed to satisfy $\mathbf{e}_s + \mathbf{w}_r \approx \mathbf{e}_o$ where $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_s \in \mathbb{R}^k$.

The corresponding score function is given by:

$$
\begin{aligned}
f^{\text{TransE}}(x_{sro}; \mathbf{\Theta}) &= -d\left(\mathbf{e}_s + \mathbf{w}_r, \mathbf{e}_o\right) && (3.9) \\
&= -\|\mathbf{e}_s + \mathbf{w}_r - \mathbf{e}_o\|_p && (3.10)
\end{aligned}
$$

where $p \in \{1, 2\}$ giving either the L1 or the L2 norm.

However, it was shown that TransE struggles with more complex multi-mapping relations [Wang et al., 2014] of the type $N - \text{to} - 1$ (e.g. `gender`), $1 - \text{to} - N$ (e.g. `bring_out`) and $N - \text{to} - N$ (e.g. `friends_with`). This is due to the way the score function is defined: suppose we have 2 triples, (`Edinburgh, locatedIn, Scotland`) and (`Edinburgh, locatedIn, Europe`). As the scoring function supposes that $\mathbf{e}_s + \mathbf{w}_r \approx \mathbf{e}_o$, the representations of `Scotland` and `Europe` would become similar, as the elements $\mathbf{e}_s$ and $\mathbf{w}_r$ are fixed in the formula. Newer models, such as TransH [Wang et al., 2014] and TransR [Lin et al., 2015] have been introduced specifically to address this issue.

More recently, **RotatE** [Sun et al., 2019] has been introduced to improve the modelling of symmetric and antisymmetric relations: every predicate is seen as a *rotation* in the complex space from the subject entity to the object entity. The resulting score function is:

$$f^{\text{RotatE}}(x_{sro};\mathbf{\Theta}) = -\|\mathbf{e}_s \circ \mathbf{w}_r - \mathbf{e}_o\|_1 \tag{3.11}$$

where $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \in \mathbb{C}^K$ and $\circ : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^k$ represents the Hadamard product which performs element-wise multiplication of two vectors. The authors also impose a constraint that $|w_{ri}| = 1$.

3.9.2 Tensor Factorisation Models

As introduced in Section 3.4, triples can be modelled using a binary tensor $\underline{\mathbf{Y}} \in \{0,1\}^{N_e \times N_r \times N_e}$, where $N_e = |\mathcal{E}|$ and $N_r = \mathcal{R}$, where:

$$y_{sro} = \begin{cases} 1, & \text{if the triple } x_{sro} \text{ exists} \\ 0, & \text{otherwise} \end{cases}$$

Tensor factorisation models can be thought of as inferring a predicted tensor $\hat{\underline{\mathbf{Y}}} \in \{0,1\}^{N_e \times N_r \times N_e}$ to estimate the true $\underline{\mathbf{Y}}$ from a set of observed triples. Models of this type are broadly based on the **Canonical Polyadic Decomposition** model (CP) by [Hitchcock, 1927] which decomposes a tensor using a latent matrix for each tensor dimension. It can be thought of as a generalisation of the singular value decomposition for matrices. As $\hat{\underline{\mathbf{Y}}}$ is a third order tensor, we require 3 latent matrices: one of entities as subject, one of relations and one for entities as objects. While CP performs very well on tensor decomposition tasks in general, in its original form it is not well suited for KGs because it learns two representations for every entity: one for the entity as a subject, and one for the entity as an object, which are entirely decorrelated. This makes generalisation difficult.

RESCAL [Nickel et al., 2011] was developed to tailor CP to the link prediction problem, using a single, unique vector embedding for each entity. Its corresponding score function takes the form of:

$$f^{\text{RESCAL}}(x_{sro}; \mathbf{\Theta}) = \mathbf{e}_s^\top \mathbf{W}_r \mathbf{e}_o \tag{3.12}$$

where $\mathbf{W}_r \in \mathbb{R}^{k \times k}$ gives the matrix representation of some relation r. Intuitively, entries i, j for the matrix indicate how much the latent features of entities i and j interact with each other using the k^{th} relation. While representing entities using unique vectors lead to significant improvements in performance, using a matrix to represent each relation made the time and space complexity of the scoring function quadratic w.r.t k and lead to overfitting.

DistMult [Yang et al., 2014] was introduced as a simplification of RESCAL, maintaining unique entity representations while using vectors to represent predicates by constraining $\mathbf{W}_r$ to be a diagonal matrix.

$$\begin{aligned} f^{\text{DistMult}}\left(x_{sro}; \mathbf{\Theta}\right) &= \langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle & (3.13) \\ &= \sum_{j=1}^{K} \mathbf{e}_{sj}, \mathbf{w}_{rj}, \mathbf{e}_{oj} & (3.14) \\ &= \mathbf{e}_s^\top (\mathbf{w}_r \circ \mathbf{e}_o) & (3.15) \end{aligned}$$

where $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \in \mathbb{R}^k$. A major problem with this model is the imposed symmetry under exchange of subject and object: consider that $f^{\text{DistMult}}(x_{sro}; \mathbf{\Theta}) = f^{\text{DistMult}}(x_{ors}; \mathbf{\Theta})$. Many relations are inherently antisymmetric and cannot be modelled using this assumption — for instance the model should not award the same score to (`smoking`, `causes`, `cancer`) and (`cancer, causes, smoking`).

To avoid enforcing this symmetry, [Trouillon et al., 2016] introduced **ComplEx** — an adaptation of DistMult that extends the domain of the embeddings from the real domain to the complex domain, allowing us to model antisymmetrical relationships. To compute the score function, we only use the real part of the vector product:

$$f^{\text{ComplEx}}(x_{sro}; \mathbf{\Theta}) = \operatorname{Re}\left(\langle \mathbf{e}_s, \mathbf{w}_r, \bar{\mathbf{e}}_o \rangle\right) \tag{3.16}$$

Note that now $\mathbf{E} \in \mathbb{C}^{N_e \times k}$ and $\mathbf{W} \in \mathbb{C}^{N_r \times k}$. $\mathrm{Re}\,(x)$ denotes the real part of a complex number $x \in \mathbb{C}$, while $\mathrm{Im}\,(x)$ represents the imaginary part. Hence, we have that $x = \mathrm{Re}(x) + i\,\mathrm{Im}(x)$ such that $\mathrm{Re}(x), \mathrm{Im}(x) \in \mathbb{R}$ and $i^2 = -1$.

This formulation allow us to introduce antisymmetry between the subject and object embeddings by using the complex conjugate of the embedding of the object entity. Formally, we define $\bar{\mathbf{e}}_o = \mathrm{Re}\,(\mathbf{e}_o) - i\,\mathrm{Im}\,(\mathbf{e}_o)\,,$ where $\mathrm{Re}\,(\mathbf{e}_o)\,, \mathrm{Im}\,(\mathbf{e}_o) \in \mathbb{R}^k$.

There are many advantages associated with this framework: the subject and object representations of an entity are not entirely decorrelated, as in CP, and neither are they forced to be identical, as in DistMult. Instead, they are tightly correlated but slightly different through the use of the complex conjugate to represent the object. Using a vector to represent the predicates results in linear time and space complexity. Moreover, this formulation allows ComplEx to learn how to exactly decompose every arbitrary KG [Trouillon et al., 2016]. In practical terms, this means that (when unregularised) ComplEx can achieve an MRR of 1 on the *training* set.

Recent experiments comparing different KGE models across a wide range of training approaches and hyperparameters have shown that ComplEx is one of the best performing models [Baier et al., 2018], especially when combined with new state-of-the-art regularisers such as N3 [Lacroix et al., 2018], and has shown most stable generalisation abilities on inductive reasoning tasks, on top being the only model capable of learning antisymmetric relationships [Trouillon et al., 2017]. Hence, in this work we use ComplEx to perform our experiments, but the methodology presented here could be readily applied to **any** existing KGE model.

3.9.3 ComplEx: The Learning Objective

ComplEx, as all KGE models, is trained by minimising a loss function defined over the KG, $\mathcal{G}$. Specifically, for each triplet $(s, r, o) \in \mathcal{G}$ and corruptions of the type $(s, r, ?)$ and $(?, r, o)$ we compute the scores of all candidate entities and pass the predicted scores via a SoftMax function. ComplEx has been shown to perform well with the multi-class log-loss, which can be formulated as $\ell_{\text{subject}}(x_{s,r,o}; \mathbf{\Theta})$ and $\ell_{\text{object}}(x_{s,r,o}; \mathbf{\Theta})$ for the subject and object, respectively, where $\mathbf{\Theta}$ jointly refers to the predicate and entity embeddings. As such, the aim is to find the entity and predicate embeddings, $\mathbf{E} \in \mathbb{C}^{N_e \times k}$ and $\mathbf{W} \in \mathbb{C}^{N_r \times k}$, that minimise the loss. The optimisation objective can then be formulated as:

$$\min_{\substack{\mathbf{E} \in \mathbb{C}^{N_e \times k} \\ \mathbf{W} \in \mathbb{C}^{N_r \times k}}} \mathcal{L}(\mathcal{G}; \mathbf{E}, \mathbf{W}) \tag{3.17}$$

where

$$\mathcal{L}(\mathcal{G}; \mathbf{E}, \mathbf{W}) = \sum_{(s,r,o) \in \mathcal{G}} \ell_{\text{object}}\left(s, r, o; \mathbf{E}, \mathbf{W}\right) + \ell_{\text{subject}}\left(s, r, o; \mathbf{E}, \mathbf{W}\right) + \alpha \Omega(\mathbf{E}, \mathbf{W}) \tag{3.18}$$

$\Omega(\cdot)$ represents a regularisation term, such as the nuclear N3 norm [Lacroix et al., 2018] and $0 \leq \alpha \leq 1$ is a parameter which controls the strength of the regularisation term.

3.10 Data Augmentation in Knowledge Graphs

Besides advancing the models themselves, one approach which has shown a lot of promise for improving link prediction performance is explicit augmentation of KGs. One of the benefits of data augmentation methods is that they are model agnostic, hence can be easily combined with any existing link prediction model. Here, we highlight two recent, distinct works which demonstrate benefits of explicit relation augmentation. Lacroix et al., 2018, propose a very simple yet surprisingly effective augmentation: introducing reciprocal relations. For every predicate, r, introduce its inverse r', hence for every triple (s, r, o) we now also have (o, r', s), which effectively doubles the size of the train dataset. This modification leads to an improvement on most benchmark datasets even with state-of-the-art KGE models.

Meanwhile, [Zhang et al., 2018] proposes learning and explicitly modelling hierarchical structure of the predicates. To this end, they generate relation embeddings using a KGE model and cluster them using K-means. They induce a three-layer hierarchical structure, consisting of a sub-relations (e.g. splitting relation `partOf` into `partOfResearchGroup` and `partOfCar`), relations and relation clusters, with a separate embedding being learnt at each layer. Hence, for a triple (s, r, o) the relation embedding $\mathbf{w}_r$ is given by a sum of embeddings from each of the three layers. The loss function is modified such that the embeddings at each hierarchy level have a different regularisation parameter. Their experimental results demonstrate that learning hierarchical information about relations in an unsupervised manner and explicitly leveraging it for link prediction tasks improves model performance. However, we note that the authors compare their results against weak baselines and provide no rationale to support their choice of clustering algorithm or modification of loss function.

Moreover, we note that there are no recent works which propose a successful *entity* augmentation technique for improving link prediction performance.

Chapter 4

Unsupervised Machine Learning: Clustering Algorithms

Having introduced the biological underpinning of human learning, the link prediction problem and the data augmentation task, we now introduce the next component which is central to our method — *unsupervised learning.* Since we have no prior knowledge as to what concepts we wish to learn, we require an unsupervised learning algorithm to capture the concept information. In this section we review selected ML approaches for unsupervised learning, focusing in particular on clustering algorithms.

The task of clustering, i.e. inferring class memberships from data with no knowledge of class labels, has been studied extensively in the past. Most clustering algorithms can be broadly said to define some measure of similarity and group items by maximising in-class similarity while minimising across-class similarity. This can be a challenging task as data is often high-dimensional, noisy and incomplete. Hence, the quality of clusters generated by a given algorithm can depend greatly on the data type, especially as most algorithms make some assumptions about the data and require specifying several parameters. Below, we give a brief theoretical overview of four clustering algorithms explored in our experiments and mention the known advantages and disadvantages of each method.

4.1 K-Means

One of the most commonly-used clustering algorithms is K-Means [MacQueen, 1967]. It is equivalent to a special case of the Expectation-Maximisation algorithm for learning Gaussian Mixture Models, where the covariance matrices are small, all-equal and diagonal. For a set of observations $\mathbf{e}_1, \ldots, \mathbf{e}_{N_e}$ where $\mathbf{e}_i \in \mathbb{R}^k$, the objective of K-means is to find sets $\mathcal{S} = \{S_1, \ldots, S_{N_c}\}$ which minimise the within-cluster sum of squares. Formally:

$$\arg\min_{\mathcal{S}} \sum_{i=1}^{N_c} \sum_{\mathbf{e} \in S_i} \|\mathbf{e} - \boldsymbol{\mu}_i\|^2 \tag{4.1}$$

where $\boldsymbol{\mu}_i$, often referred to as the centroid, represents the mean of vectors in S_i. This is achieved in an efficient way by randomly initialising the cluster centres and then iteratively:

1. Assigning each observation to the cluster whose cluster centroid is closest:

$$S_i \leftarrow \left\{\mathbf{e} \ : ||\mathbf{e} - \boldsymbol{\mu}_i||^2 \leq ||\mathbf{e} - \boldsymbol{\mu}_j||^2 \ \forall j, 1 \leq i, j \leq N_c\right\} \tag{4.2}$$

2. Computing new cluster centroids:

$$\boldsymbol{\mu}_i \leftarrow \frac{1}{|S_i|} \sum_{\mathbf{e}_j \in S_i} \mathbf{e}_j \tag{4.3}$$

Advantages: This approach is always guaranteed to converge, though the minima might be local. Moreover, it scales easily to large datasets and often is able to produce good quality clusters even when some assumptions do not hold.

Disadvantages: A key limitation of K-Means is its assumption that all the clusters are spherical i.e. have equal covariance matrices. This is a very general assumption which sometimes might not hold, leading to poor quality clusters. Moreover, K-Means is not guaranteed to find global minima unless the underlying distribution has a single minima and thus the quality of the final solution is dependent on the initialisation. Lastly, it can be sensitive to outliers.

4.2 Affinity Propagation

Affinity Propagation (AP) [Frey and Dueck, 2007] is founded on the idea of 'message passing' between data instances. It does so by finding 'exemplar' members to represent each cluster. The algorithm works by first selecting few initial 'exemplars' and then passing messages between pairs of observations. The messages express how appropriate one observation is as an exemplar for the other observation — this value is updated based on the information received from other pairs. The process is repeated iteratively until the values converge and then the final exemplars and clustering are chosen. There are two types of messages: the *responsibility* $r(i,j)$ collects evidence that observation j could be the exemplar for i. Meanwhile the *availability*, $a(i,j)$, complies the evidence that observation i should choose observation j as its exemplar. The availability also takes into account how many other samples are choosing j as their exemplar. This set-up encourages choosing exemplars such that they are similar to many observations and have been chosen as representative by numerous observations. Formally:

$$r(i,j) \leftarrow \mathrm{sim}(i,j) - \max_{j' \neq j} \left[a\left(i,j'\right) + \mathrm{sim}\left(i,j'\right) \right] \tag{4.4}$$

$$a(i,j) \leftarrow \min \left[0,\ r(j,j) + \sum_{i' : i' \notin \{i,j\}} r\left(i',j\right) \right] \tag{4.5}$$

where sim represents the similarity between observations i and j. The responsibility and availability values are computed iteratively, either for a fixed number of iterations or until convergence and can be stored as matrices $\mathbf{R}$ and $\mathbf{A}$. In practice, a damping factor is also introduced to prevent numerical instabilities during the iteration process. Once the updates have finished, the so-called criterion matrix is computed, which is simply: $\mathbf{C} = \mathbf{R} + \mathbf{A}$. Lastly, the exemplar for each observation i

is chosen as $\underset{j}{\operatorname{argmax}}\, \mathbf{c}_{ij}$.

Advantages: One of the benefits of AP is that the number of clusters does not need to be specified, but is instead inferred from the data.

Disadvantages: As the time and memory complexity can be high, it can have problems scaling to larger datasets.

4.3 DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [Ester et al., 1996] algorithm was developed to allow discovering clusters of arbitrary shape by considering a density-based notion of clusters. Each cluster is seen as made up of 2 parts: a set of *core samples*, i.e. observations located in areas of high density according to some distance metric, and a set of non-core observations, called *neighbours* which lie on the outskirts of the cluster. To formalise this definition, we introduce two parameters: `eps` and `min_samples`. If there exists a sample such that within a distance of `eps` there exist at least `min_samples` other samples, we define this instance as a the core sample. Therefore, we can consider the search for the core sample as a search for a dense region of the vector space.

Advantages: One of the distinguishing features of DBSCAN is the notion of noise and the ability to automatically detect outliers. Moreover, it is capable of finding arbitrarily shaped clusters and the number of clusters is automatically detected by the algorithm.

Disadvantages: One of drawbacks associated with using DBSCAN is its sensitivity to parameter choice, especially `eps`. Choosing a slightly too small `eps` value can cause all of the data to be labelled as noise, while a slightly too big `eps` will assign all data points to the one cluster. Although there exist some heuristics for choosing an appropriate value such as the so-called elbow plots [Rahmah and Sitanggang, 2016], they require producing a plot for every representation and analysing it manually. Furthermore, DBSCAN struggles to cluster datasets with large differences in densities between the clusters as it becomes impossible to find a suitable combination of `min_samples` and `eps` for all clusters.

4.4 Spectral Clustering

Broadly speaking, the Spectral Clustering approach [Ng et al., 2001] aims to create k clusters by clustering the first k eigenvectors of the Laplacian of the similarity matrix of the data.

An intuitive way to think of this approach is to consider what happens when we project our data to the eigenspace: the eigenvectors constitute the axes of the new space and the length of the axes is determined by the eigenvalues. In the eigenspace, the original datapoints tend to align with the new axes. As a result, we obtain the principal components of our data which are orthogonally separated and the noise in the data is reduced.

Formally, given a set of observations $x_1, \ldots, x_n$, a similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric matrix where each entry $\mathbf{A}_{ij} \geq 0$ gives a measure of similarity between the i^{th} and the j^{th} instances. Here, we define the Laplacian matrix, $\mathbf{L}$, as:

$$\mathbf{L} := \mathbf{D} - \mathbf{A} \tag{4.6}$$

where the entries of the diagonal matrix $\mathbf{D}$ are the degrees of each node: $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. We next compute the first k eigenvectors $(\mathbf{u}_i)_{i=1,\ldots,k}$ of $\mathbf{L}$ by solving the eigenvalue problem. Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ be a matrix of eigenvectors matrix where the columns correspond to eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$. Then, define vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$ where $\mathbf{y}_i \in \mathbb{R}^k$ is the i^{th} row of $\mathbf{U}$ and cluster these using an algorithm such as K-Means. Hence, Spectral Clustering can be seen as K-Means with an additional preliminary data transformation. The change of representation makes use of the properties of the graph Laplacians which makes the clusters trivial to detect.

Chapter 5

Concept-inspired Machine Learning

We have now discussed four distinct areas: concept learning in humans, knowledge graphs and KGEs, data augmentation and unsupervised learning. The method we present in Section 6 integrates approaches from each of these fields for explicitly augmenting KGs with concepts learnt in an unsupervised manner. In this section we summarise the *existing* approaches for concept learning and comment on their suitability for our problem setting. We begin by considering concept learning models in the broader field of machine learning and later focus on those designed specifically for KGs.

As discussed earlier (Section 2.2), *concepts* here are defined as representations of observed data which capture attributes shared by a group of instances.

5.1 Concept Formation in Machine Learning

5.1.1 Inductive Concept Learning

Some researchers [Goodman et al., 2008] attempt modelling concept learning as an inductive, rule-based process, capable of constructing rich, structured concepts from sparse data. They argue that the key to modelling this mechanism lies in integration of rule-based, symbolic learning and statistical inference. To this end, they propose a rule-based model which brings together Bayesian inference and a grammaticallystructured disjunctive hypothesis space [Goodman et al., 2008]. It is designed primarily to model supervised learning tasks, though the authors suppose that it could also be used to model unsupervised tasks by using a 'strong sampling' likelihood. However, this approach still requires the objects to be assigned to existing categories, although the assignments remain hidden to the model throughout the entire training procedure. Hence, this method cannot be adapted to a fully unsupervised learning scenario, in contrast to the method we propose in Section 6.

5.1.2 Concept Learning on the Fly

The model proposed by [Mordatch, 2018] aims to learn concepts *on the fly*, where the process is driven by an execution-time energy minimisation process, analogous to the biological theory proposed by [Friston, 2010]. Specifically, the author focuses on the task of learning 'relation' concepts, such as *near, above* and *between* from observing complex interactions of multiple entities. The model learns concepts in a way that allows them to be combined recursively and compositionally, and is capable of both: concept generation and concept identification. He achieves this by exploiting a core property of the energy function — its smoothness — to take derivatives of it at inference time and use those gradients as part of the model. Crudely, one first assumes that there exists a 'good' energy function and uses it to obtain better concept codes, world states and attention masks by gradient descending on the energy function. Next, one uses the loss function (which is also a function of the energy function) to obtain better model parameters and learn a better energy function. Together, the two steps create an iterative process which is bound to converge. However, once again there is no obvious way of applying this approach to a fully unsupervised setting, where no example concepts are shown. Moreover, while the model performs exceptionally well on the toy examples, applying this method to a larger dataset would pose serious scalability issues as the inference-time optimisation is very computationally intensive. Meanwhile, the approach we propose here can scale easily even to large KGs.

5.1.3 Learning Hierarchies in Graphs

Shifting to a different paradigm, Ying et al., 2018 propose DIFFPOOL — a method for learning concept hierarchies from graph-structured data [Ying et al., 2018]. They propose using a graph neural network (GNN) and introduce a novel differentiable graph pooling module. At each layer of the network DIFFPOOL makes a differentiable soft cluster assignment for each node. The clusters formed at layer i then serve as coarsened input at layer $i+1$. Hence, the differentiable graph pooling module can be seen as a graph analogue of the pooling operations found in CNNs, achieved by building a hierarchical multi-layer scaffold on top of the underlying graph. The effectiveness of the method is measured by analysing model performance on a graph classification task — in comparison to the graph classification benchmarks, the authors observe on average a 5-10% increase in accuracy. This highlights potential benefits of performing data augmentation and explicitly modelling cluster nodes. While here this approach is applied in a graph classification setting, at first it appears that it could be readily applied to link prediction in KGs, since they can be seen as labelled multigraphs. Moreover, the learning of clusters is achieved in a fully unsupervised manner — something that is at the core of our aims. Unfortunately, there is not easy way of applying a GNN to a benchmark KG as it requires an adjacency matrix to operate. These in turn are huge for KGs due to the large number of entities, making the computation intractable. Our method circumvents this problem by avoiding the use $N_e \times N_e$ matrices for measuring similarity between entities.

5.2 Concept Learning in Knowledge Graphs

5.2.1 Probabilistic Approaches

The problem of clustering entities (and relations) in Knowledge Graphs has been addressed using three similar probabilistic models, based on the SRL framework. The Infinite Relational Model (IRM) [Kemp et al., 2006] is a non-parametric generative Bayesian model which learns single cluster assignments. Concurrently, [Xu et al., 2012] proposed the Infinite Hidden Relational Model, which supports multiple types of relations. The Multiple Relational Clusterings model (MRC) [Richardson and Domingos, 2006] builds upon the IRM by learning multiple cross-cutting clusterings, i.e. allowing each object to belong to more than one cluster. To achieve this, it introduces finite, function-free second-order Markov Logic. In fact, most relational problems (and most statistical models) can be formulated as Markov Logic Networks (MLNs). In this paradigm, a knowledge base can be seen as a set of hard constraints, hence accounting for its brittleness. This problem can be circumvented by making the constraints soft, such that violations of the constraints are less likely, but not impossible. In MLNs this is done by assigning a weight to each formula. As such, this work aims to provide the foundations for statistical predicate invention in KGs, with the methods described in the paper focusing on unary predicate invention. There are many advantages to this formulation, such as that the number of clusters does not need to be specified and the model favours simple clusterings, only introducing new clusters as needed. While this is a powerful idea, a clear disadvantage of using MLNs is that probabilistic inference is generally NP-hard and requires approximations. Meanwhile, the framework we propose is simpler and does not commit to full First Order Logic, making for a more efficient approach. To illustrate the computational burden, consider that to train either the IRM or the MRC on a small knowledge graph, such as UMLS [McCray, 2003] (135 entities) took **10 hours**, while our framework requires under 2 minutes. The difference becomes even more pronounced when you consider that the datasets we are primarily interested in, such as FB15K-237 [Toutanova and Chen, 2015], are made of up **at least** 14k entities.

5.2.2 Non-Euclidean Embeddings for Hierarchy Modelling

Rather than focusing on concept learning, some approaches consider how best to capture *existing* concept hierarchies in a latent way, using representations generated by Knowledge Graph Embedding models. Nickel and Kiela, 2017 observe that Euclidean space is inherently ill-suited for modelling hierarchies since with increasing levels of branching the space shrinks, hence reducing the expressive power of the embeddings.

As a remedy, they propose utilising hyperbolic geometry, specifically the Poincaré Ball: a surface where distances grow exponentially with increasing distance from the centre. This formulation allows for modelling potentially infinite levels of branching with significantly fewer dimensions that in Euclidean geometry. A similar approach, albeit using the polar coordinate system, has also been proposed recently [Zhang et al., 2019]. However, in both of these approaches the hierarchies are provided in the dataset, rather than learned. Meanwhile, our method allows us to *learn* the hierarchies in a fully unsupervised way, with no prior knowledge of what concepts should be formed.

5.2.3 Learning Hierarchies from Embeddings

A preliminary study by [Nickel, 2011] suggests that the KGE models can, to some extent, capture a latent concept hierarchy even without having it explicitly provided. To demonstrate this, the authors choose a dataset with an existing taxonomy and then attempt to rebuild the hierarchy using a fully unsupervised approach. Specifically, they train RESCAL [Nickel et al., 2011] and then cluster the resulting entity embeddings. The results of their experiments suggest that it is indeed possible to learn hierarchies from KGE embeddings in a fully unsupervised manner. Nonetheless, the study only considered a single, small dataset, a single clustering approach and a single representation method. Our work could perhaps be seen as extension of this study since we consider the task of unsupervised concept learning for a range of large datasets, various clustering algorithms and experiment with different representation methods. However, while the scope in [Nickel, 2011] is limited to answering the question '*can we learn concepts from KGE embeddings?*', our objectives have the task of link prediction at its centre, asking '*what is the effect of explicitly harnessing concept information on link prediction performance?*'.

Chapter 6

Similarity-based Concept Learning in Knowledge Graphs

As observed in the previous chapter, the existing approaches for modelling concept formation in KGs are either not scalable [Ying et al., 2018, Xu et al., 2012, Kemp et al., 2006] or simply model existing hierarchies using latent variables [Nickel and Kiela, 2017, Zhang et al., 2019] rather than learning concepts directly from the data in a fully unsupervised manner. Moreover, there are have been no experiments to analyse the effect of explicitly modelling latent concepts as entities on link prediction performance.

In this section we introduce a novel approach for learning latent concepts in knowledge graphs in a fully unsupervised manner. The framework is model-agnostic, thus can be easily integrated with any existing link prediction model. Moreover, it easily scales even to large datasets and our experiments show that explicitly exploiting the latent concept structure can lead to an improvement on link prediction tasks.

The method we propose is loosely inspired by the biological theory of learning as compression [Love et al., 2004], which views concept formation as a clustering process. More specifically, it is analogous to incidental unsupervised concept learning since the cluster assignments are made based purely on similarity and not driven by any downstream task.

1. **Propositionalisation**

   First, we choose a representation method and for every entity $e \in \mathcal{E}$ obtain vector embeddings: $\{\mathbf{e}_1, \ldots, \mathbf{e}_{|N_e|}\}$ where $\mathbf{e}_i \in \mathbb{R}^k$. In this work we experiment with the following representations: ComplEx embeddings (where $\mathbf{e}_i \in \mathbb{C}^k$), the Weisfeiler-Lehman Kernel (where the representation is latent), a rule-based representation and a random paths representation.

2. **Clustering**

   Next, we choose a method for measuring similarity between the vectors and assigning each entity to a cluster. The assignments are made such that each entity belongs to exactly one cluster. Hence, we can consider the clustering step as a partitioning of N_e entities $e_i \in \mathcal{E}$ into N_c sets $\mathcal{S} = \{S_1, \ldots, S_{N_c}\}$, $N_c \leq N_e$. The number of clusters, N_c, can be either defined *a priori* and treated as a hyperparameter, or it can be automatically chosen by the clustering algorithm. Here, we consider the following algorithms: K-Means, Spectral Clustering, DBSCAN and Affinity Propagation.

3. **Data Augmentation**

   We then augment the dataset by introducing the clusters as new concept entities. We denote the set of concept entities in the KG using $\mathcal{C} = \{c_1, \ldots, c_{N_c}\}$ and use a new `isA_concept` predicate. We augment the original KG $\mathcal{G}$ by connecting each concept c_j to every entity e_i which has been assigned to it. Formally: $\mathcal{G}' \leftarrow \mathcal{G} + \{(e_i, \texttt{isA\_concept}, c_j)\ \forall e_i \in S_j, \forall c_j \in \mathcal{C}\}$.

4. **Train and Evaluate KGE Model on Link Prediction**

   A KGE model is then trained on the augmented dataset for the link prediction task and we obtain embeddings for the original entities $\{\mathbf{e}_1, \ldots, \mathbf{e}_{N_e}\}$, the concept entities $\{\mathbf{c}_1, \ldots, \mathbf{c}_{N_c}\}$ and the relations $\{\mathbf{r}_1, \ldots, \mathbf{r}_{N_r}\}$.

For the remainder of this chapter we explore each step in greater detail and summarise the method in Algorithm 2.

6.1 Propositionalisation

While there exist clustering algorithms which can operate directly on graphs by partitioning them [Blondel et al., 2008], they usually do not scale well with large datasets. Standard clustering algorithms, on the other hand, such as K-Means scale much better but require as input vector observations and assume that the data are *i.i.d.* Hence, applying them to graph-structured data is challenging as graph nodes are inherently linked to each other, violating the *i.i.d* assumption. The family of methods concerned with transforming relational data such that it can be represented using *i.i.d* feature vectors is known as *Propositionalisation* [Kramer et al., 2001]. The key challenge facing these methods is constructing features which capture the rich connectivity patterns in a low-dimensionality vector. In this work we experiment with four different propositionalisation methods. Below, we describe the procedures for obtaining each of the representations.

6.1.1 ComplEx Embeddings

The vector embeddings generated when training a KGE model can be seen as one way of achieving propositionalisation — deriving a vector representation for each entity. As noted above, the results from a preliminary study [Nickel, 2011] suggest that it is indeed possible to extract concepts by clustering KGE embeddings. While the authors experiment with RESCAL [Nickel et al., 2011], which is one of the earliest tensor factorisation models, here we use ComplEx (Section 3.9.2), one of the current state-of-the-art models. To obtain the embeddings we train ComplEx on the original dataset and then save the embeddings which achieve highest MRR on the held-out validation set. We extract both: the real part and the imaginary part of the embeddings, without making a distinction between the two. Detailed discussions about the training procedure, such as the choice of hyperparameters, will follow in Section 9. It is worth noting, however, that KGE models are known to place similar entities close to each other in embedding space [Nickel et al., 2016], hence already modelling concepts in a latent manner to some extent. Thus, in terms of pushing similar entities closer together in the embedding space, introducing concepts learnt from the ComplEx representation might introduce little new information, as the clustered entities are already close to each other in ComplEx's embedding space. Therefore, for the other propositionalisation methods we will choose approaches that are likely to introduce information which KGE models struggle to capture.

6.1.2 Weisfeiler-Lehman Kernel

Computing a graph kernel i.e. a kernel which can be applied directly to graph-structured data, can also be seen as one way of achieving propositionalisation. Consider that kernels project data into a high-dimensional *implicit* feature space and measure similarity between objects by computing the inner product in the Hilbert space. Thus, the nodes are also projected into some embedding space, though the mapping is implicit. Graph kernels belong to a special family of kernels called R-convolution kernels [Haussler, 1999] which can be defined on discrete objects that are made up of discrete components. As such, they operate by considering all possible pairs of decompositions of such objects. A key notion in graph kernels is that an instance in a graph can be represented using a small subgraph, relying on the fact that such a subgraph can encompass useful information about the instance and its position in the graph as a whole. In this setting, similarity between two instances can be measured by extracting their respective subgraphs and then apply a graph kernel.

Based on the Weisfeiler-Lehman isomorphism test, the Weisfeiler-Lehman (WL) kernel [Shervashidze et al., 2011] answers the question '*are two (sub)graphs topologically identical?*' by computing the number of subtrees which the two graphs share. A subtree is a subgraph with a specific root node and no cycles — hence, a subtree of $\mathcal{G}$ can be thought of as a subset of connected distinct nodes with a tree structure. To compare the subgraphs the WL kernel uses a 're-labelling' procedure: for each vertex in the graph a new multiset label is constructed using the labels of its neighbours. The vertex is then given a new abridged label, created by concatenating the original vertex label with the new (sorted) multiset label and compressing the result. The kernel computation is then reduced to simply counting the node labels common to both graphs at each iteration step.

Formally, we define the n^{th} iteration of the relabelling of two graphs $\mathcal{G}$ and $\mathcal{G}'$ as $\mathcal{G}_n = (V, E, \ell_n)$ and $\mathcal{G}'_n = (V', E', \ell'_n)$ where V is the set of vertices, E the set of edges and $\ell : (V \cup E) \times \mathbb{N} \to \Sigma$ is a labelling function from edges or vertices and depth index $j \in \mathbb{N}$ to the set of labels Σ. The index j measures the depth of subgraph at which the given node or edge occurred. It is also worth noting that V and E do not change in a sequence of relabelling and explicit inclusion of these terms is merely conventional. The WL kernel is then defined as:

$$k_{WL}^{(h)}(\mathcal{G}, \mathcal{G}') = \sum_{n=0}^{h} k_\delta(\mathcal{G}_n, \mathcal{G}'_n) \tag{6.1}$$

where h is the number of relabelling iterations and k_δ is the Dirac kernel:

$$k_\delta(\mathcal{G}_n, \mathcal{G}'_n) = \sum_{v \in V} \sum_{v' \in V'} \delta(\ell(v), \ell'(v')) \tag{6.2}$$

which yields 1 if the labels are the same and 0 if the equality does not hold. Hence, it is essentially a counting kernel.

Usually, graph kernels are computed by first extracting a set of subgraphs and then computing the kernels on them. In this work, we instead make use of the Fast Approximation of the WL Kernel [de Vries, 2013], which is designed specifically for more efficient computation of WL kernels on knowledge graphs. While many works on graph kernels are interested in comparing different graphs, in the context of KGs the focus is usually on measuring similarity between entities which all exist in the same graph. As a results, the subgraphs we extract often share some vertices and edges. Hence, rather than explicitly extracting a set of subgraphs, [de Vries, 2013] show that it is more efficient to perform the kernel computation directly on the whole KG graph while retaining subgraph information. For details concerning the implementation of this approach, see [de Vries, 2013].

6.1.3 In-Range, In-Domain Embeddings

While latent feature models are effective at capturing global connectivity patterns and perform efficiently when a small number of latent variables, k, can be used to explain the triples, there nonetheless exist numerous cases in which observable feature models perform better [Nickel et al., 2016]. For instance, tensor factorisation models are known to struggle when the triples contain many 'strong connected' elements, such as a (`Monica`, `marriedTo`, `Mark`) triple. Meanwhile, predicting this relation with observable feature models using rule-based and Inductive Logic Programming (ILP) approaches is easy — one could for instance infer the triple from existence of the path `Monika` $\xrightarrow{\text{parentOf}}$ `Simon` $\xleftarrow{\text{parentOf}}$ `Mark`. Combining latent and graph-based approaches is a promising area of research, which has shown to result in increased predictive power. The method we propose here — learning concepts from representations built using rule-based reasoning and training a latent feature model on the augmented dataset — can be seen as a novel way of marrying these two complimentary approaches.

ILP approaches hinge upon defining a set of first-order clauses, which in turn induce a disjuctive hypothesis. To begin with, we induce two very simple clauses, `in_range` and `in_domain`, which test whether a given entity participates in a triple with relation r as either object or subject. Formally:

$$\texttt{in\_range}(e, r) = \begin{cases} True & \text{if } \exists\, (e, r, o) \in \mathcal{G}\text{, for any } o \\ False & \text{otherwise} \end{cases} \tag{6.3}$$

$$\texttt{in\_domain}(e, r) = \begin{cases} True & \text{if } \exists\, (s, r, e) \in \mathcal{G}\text{, for any } s \\ False & \text{otherwise} \end{cases} \tag{6.4}$$

For a given entity e and some relation r, we construct a vector $\mathbf{p} \in \mathbf{B}^2$ such that:

$$\begin{aligned} &(\texttt{in\_range}(e, r) \rightarrow \mathbf{p}_1 = 1) \wedge (\neg\texttt{in\_range}(e, r) \rightarrow \mathbf{p}_1 = 0) \\ &(\texttt{in\_domain}(e, r) \rightarrow \mathbf{p}_2 = 1) \wedge (\neg\texttt{in\_domain}(e, r) \rightarrow \mathbf{p}_2 = 0) \end{aligned} \tag{6.5}$$

Each entity e is then represented by a vector $\mathbf{e}$ created by concatenating $\mathbf{p}$ vectors which test the participation of e for every predicate $r \in \mathcal{R}$. Hence, $\mathbf{e} \in \mathbf{B}^{2N_r}$ where $N_r = |\mathcal{R}|$ is the number of relations in the graph.

6.1.4 Random Paths

Generating Random Paths

Recent work on case-based reasoning approaches [Das et al., 2020] highlights the predictive potential of travelling along multi-hop random paths in a KG. Inspired by this approach, we propose constructing a representation by first generating for each entity, e, n random paths starting at e of maximum length L, where a path is defined as a list of triples of the form $(e_i, r_j, e_k, direction)$. Direction specifies whether the we travel along the relation in forward or inverse direction. The paths are generated such that we ensure to avoid an immediate reversal step, i.e. avoid steps of the type $(e_1, r, e_2), (e_2, r^{-1}, e_1)$. We also check whether loops have occurred, where we define loops as having travelled along a segment $(e_1, r_1, e_2, r_2, e_3, r_3, e_4)$ more than once in a single path. If the path is of length greater than 6 and we have encountered a loop, the path is terminated. A simplified algorithm for obtaining paths is shown in Algorithm 1.

Algorithm 1 Generate Random Paths

```
function GenerateRandomPaths($\mathcal{G}$)
    for $e \in \mathcal{E}$ do
        for $i \in \{1, \ldots, n\}$ do
            for $j \in \{1, \ldots, L\}$ do
                $outgoingEdges \leftarrow$ GetEdges($e$)
                if $previousEdge$ is not None then
                    $outgoingEdges$.Remove(Inverse($previousEdge$))
                $newEdge \leftarrow$ RandomChoice($outgoingEdges$)
                $paths$.Append($newEdge$)
                $previousEdge \leftarrow newEdge$
                if Length($path > 6$ and DetectLoops($path$) then
                    Break

function DetectLoops(path)
    return True if loops in $path$, else False

function Inverse($edge : (e_1, r, e_2, direction)$)
    return $(e_1, r, e_2, -direction)$

function GetEdges($e$)
    return a set of all outgoing edges from entity $e$, $\{(e, r_i, e_j, direction)\}$, where
  $direction \in \{-1, 1\}$ specifies whether the relation $r$ is forward or reciprocal.
```

Generating Representation

There exist different approaches which could be used to construct vector representations of entities from random paths: for instance one could represent entity e' using a vector $\mathbf{e}' \in \mathbb{R}^{N_e}$ where each entry is a count how many times each of the entities $e \in \mathcal{E}$ was visited in the n paths starting from e'. However, the issue with this representation is that the dimensions of the embedding matrix $\mathbf{E} \in \mathbb{R}^{N_e \times N_e}$ would be very large — an issue which we are precisely trying to avoid. An alternative approach is to instead represent the entities by considering the *predicates* along which we travel during the paths. As usually $N_r << N_e$, this approach is more efficient. We will distinguish as to whether we have travelled along a predicate in the forward or inverse direction, hence the resulting embeddings are $\mathbf{e} \in \mathbb{R}^{2N_r}$, where each entry e_i is given by the number of times we have travelled along relation r_i across the n paths. In some ways, this representation can be seen as an extension of the more primitive representation proposed in 6.1.3.

6.2 Link Prediction with Concept Augmentation

Once the propositionalisation step is performed and each of the entities has been represented using a feature vector, similar entities are then grouped into concept clusters. In this work we experiment with four different clustering algorithms (Chapter 4) as we are generating representations using distinctly different methods. Thus, clustering approaches which might work well for some representations might be sub-optimal for others, especially as we have no *a priori* expectations of what the clusters should look like.

Once we partition the entities into sets of clusters $\mathcal{S} = \{S_1, \ldots, S_{N_c}\}$, we then construct a set of new triples $\{(e_i, \texttt{isA\_concept}, c_j)\ \forall e_i \in S_j, \forall c_j \in \mathcal{C}\}$, explicitly assigning each pre-existing entity to a new concept entity and connecting them with a new `isA_concept` predicate. The set of new triples is then concatenated with the original training dataset. The decision to introduce a new predicate to denote concept assignment was made in light of the fact that while in some datasets there already exists an `isA` type predicate, that is not the case for all datasets. Introducing a new predicate in some datasets but not in others would render performance comparisons between different datasets challenging.

One of the advantages of this approach is that any KGE model can be trained on the augmented dataset with practically no adaptations required. While implementation details are discussed in the following section, here we point out that one modification is required for the performance evaluation. Namely, when computing the MRR for the validation and test sets, we only use the embeddings of the original entities as candidate answers to prevent the model from predicting a concept entity as one of the missing entities.

Algorithm 2 Similarity-based Concept Learning in KGs

```
Require $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where $\mathcal{E}$, $\mathcal{R}$ are sets of all entities and relations

procedure ConceptLearning
    $\mathbf{E} \leftarrow \texttt{Propositionalisation}(\mathcal{G})$
    $\mathcal{S} \leftarrow \texttt{Clustering}(\mathbf{E})$
    $\mathcal{G}' \leftarrow \mathcal{G} + \{(e_i, \texttt{isA\_concept}, c_j)\ \forall e_i \in S_j, \forall c_j \in \mathcal{C}\}$
    Initialise $\mathbf{E}, \mathbf{C}, \mathbf{W}$
    for $n$ epochs do
        $\mathbf{E}', \mathbf{C}', \mathbf{W}' \leftarrow \texttt{KGE}(\mathcal{G}', \mathbf{E}, \mathbf{C}, \mathbf{W})$
```

6.3 Concept Triples as Regularisers

In this section we provide an intuition as to how augmenting the KG with concept triples can be seen as a form of regularisation. First, we note that the new, augmented KG, $\mathcal{G}'$, is a subset of the original KG, $\mathcal{G}$:

$$\mathcal{G}' \leftarrow \mathcal{G} + \{(e_i, \texttt{isA\_concept}, c_j)\ \forall e_i \in S_j, \forall c_j \in \mathcal{C}\} \tag{6.6}$$

$$\therefore \mathcal{G} \subset \mathcal{G}' \tag{6.7}$$

Since the loss function is a sum of the losses for all of the training triples, we can re-write the multi-class log-loss for the augmented KG:

$$\mathcal{L}(\mathcal{G}'; \mathbf{E}', \mathbf{W}') = \sum_{(s,r,o) \in \mathcal{G}'} \ell_{\text{object}}\left(s, r, o; \mathbf{E}', \mathbf{W}'\right) + \ell_{\text{subject}}\left(s, r, o; \mathbf{E}', \mathbf{W}'\right) + \alpha\Omega(\mathbf{E}', \mathbf{W}') \tag{6.8}$$

$$= \sum_{(s,r,o) \in \mathcal{G}} \ell_{\text{object}}\left(s, r, o; \mathbf{E}, \mathbf{W}\right) + \ell_{\text{subject}}\left(s, r, o; \mathbf{E}, \mathbf{W}\right) + \alpha\Omega(\mathbf{E}, \mathbf{W}) \tag{6.9}$$

$$+ \sum_{(s,r,o) \in \mathcal{G}' \ \mid \ (s,r,o) \notin \mathcal{G}} \ell_{\text{object}}\left(s, r, o; \mathbf{E}', \mathbf{W}'\right) + \ell_{\text{subject}}\left(s, r, o; \mathbf{E}', \mathbf{W}'\right) \tag{6.10}$$

$$+ \alpha\Omega(\mathbf{C}, \mathbf{W}_{\texttt{isA\_concept}}) \tag{6.11}$$

where $\mathbf{E}'$, $\mathbf{W}'$ are the entity and predicate embeddings for all entities and predicates in $\mathcal{G}'$, $\mathbf{E}$, $\mathbf{W}$ are the entity and predicate embeddings for all entities and predicates in $\mathcal{G}$ and $\mathbf{C}, \mathbf{W}_{\texttt{isA\_concept}}$ are the embeddings of the concept entities and the `isA_concept` predicate.

Hence, we see that the augmented training loss decomposes to give the training loss of the original KG (see Equation 3.18) with additional terms. Therefore, introducing the concept triples corresponds to introducing additional terms into the loss function. It is worth noting that here we apply the same regularisation strength, α, to both, concept embeddings and original entity embeddings but given more time it would be interesting to experiment with using a different regularisation strength for the concept representations.

Chapter 7

ConFormE: Concept Formation with EM

7.1 Joint Concept and Representation Learning

Thus far, we have first and foremost tried to answer the question: *can explicit modelling of concept entities lead to an improvement on link prediction performance?* To this end, we focused on generating entity representations via propositionalisation and then making the assignments using a purely similarity-based approach. In this chapter, we wish to extend the question and ask: *can we **jointly** learn cluster assignments and representations that are **optimal** for link prediction tasks?*

One way this could be achieved is by performing active, rather than static, propositionalisation, as proposed by [Landwehr et al., 2006]. In active propositionalisation one learns a representation which maximises the performance on a downstream task. However, in our case, this approach would require first learning a representation, clustering it, augmenting the dataset, then training the model till convergence and using the feedback to update the representation, and repeating this in an iterative manner. Aside from the fact that there is no obvious way to augment the initial representation, such as a random paths representation, based on link prediction performance, such a method would also be highly inefficient as each training iteration would involve training ComplEx for around 100 or more epochs, depending on the dataset.

Instead, we propose learning concepts at train time and thus learning cluster assignments and model parameters jointly. From a statistical perspective, we could consider cluster assignments as latent variables. Formally, let Z be a random variable denoting the cluster assignment, with $z \in \{c_1, \ldots, c_{N_c}\}$. We can then define a joint distribution $p(X, Z)$ over the observed triples, X, and the latent variables, Z, and a likelihood function $L(\mathbf{\Theta}; X, Z)$ where $\mathbf{\Theta}$ are the KGE model parameters. As such, we wish to compute the model parameters which maximise the marginal likelihood of the observed data.

While computing marginal likelihood directly is usually intractable, we can use the Expectation-Maximisation (EM) algorithm [Dempster et al., 1977] to estimate the model parameters. EM finds the local maximum likelihood estimates of model parameters in an iterative way, by repeating two steps: the Expectation (E) step and the Maximisation (M) step. In the standard EM setting the E-step involves computing the expectation of the log-likelihood function evaluated using current estimates of the parameters. As such, the cluster assignments are made in a *soft*, probabilistic manner. The M-step then involves computing the model parameters which maximise the expected log-likelihood from the E-step. The two steps are repeated iteratively till convergence.

Adapting the *soft* EM regime to our problem is challenging. Specifically, i) the partition function for the soft assignments in the E step and ii) the maximisation in the M step pose serious complexity issues. To solve i), we propose using a *hard* EM implementation: rather than computing all the conditional probabilities of the cluster memberships, we simply assign the entity to its most likely concept cluster. This is a common practical solution, used for instance in the K-Means algorithm (Section 4) and it has also been successfully implemented for probabilistic concept learning by [Kok and Domingos, 2007]. Moreover, it offers the advantage that the M step can be reinterpreted in the usual link prediction scenario — learning optimal model parameters using the augmented dataset.

7.2 EM for Concept Learning in KGs

We will now outline the details of Concept Formation with EM (ConFormE) — a hard EM-like algorithm for jointly learning cluster assignments and KGE model parameters. We denote the original graph as $\mathcal{G}$ and the augmented graph as $\mathcal{G}'$. Concepts as clusters, i.e. sets of entities, are denoted as $S_1, \ldots, S_{N_c}$, while concepts as latent variables and instances in the KG are written as $\mathcal{C} = \{c_1, \ldots, c_{N_c}\}$.

- **E-step**: The E-step corresponds to estimating the cluster memberships for each of the entities. The aim is to find the cluster assignment c which satisfies $\max_c p(Z = c|e_i)$ for each entity e_i. To estimate this conditional probability we use the ComplEx score function which is the real part of the trilinear vector product in the complex space. For simplicity, we will denote it using:

$$\begin{aligned}\text{score}(e_i, r_k, e_j) &= f^{\text{ComplEx}}(x_{ijk}; \mathbf{\Theta}) \\ &= \text{Re}(\langle \mathbf{e}_i, \mathbf{w}_k, \bar{\mathbf{e}}_j \rangle)\end{aligned}$$

  Since $p(Z = c|e_i) \propto \exp\{-\text{score}(e_i, \texttt{isA\_concept}, c)\}$ [Platt, 1999] we can simply make a hard assignment:

$$\begin{aligned}c^* &= \underset{c}{\text{argmax}}\ p(Z = c|e_i),\ \forall e_i \in \mathcal{E}, \forall c \in \mathcal{C} &\quad (7.1)\\ &= \underset{c}{\text{argmax}}\ \exp\{-\text{score}(e_i, \texttt{isA\_concept}, c)\} &\quad (7.2)\end{aligned}$$

- **M-step**: In the M-step we should be computing the model parameters, i.e. the embeddings of the original entities, $\mathbf{E}$, concept entities, $\mathbf{C}$, and relations, $\mathbf{R}$, by maximising the expected log-likelihood $E_{z \sim p(Z=k|\mathcal{E})}[p(X, Z)]$. The formulation of $p(X, Z)$ is somewhat difficult as our observed variables, X, are triples. Nonetheless, if we interpret $p(X, Z)$ as the joint distribution of all existing triples along with the latent concepts, then $p(X, Z)$ simply corresponds to the distribution of the new dataset, augmented according to the assignments learnt in step E. Hence, in the M-step we simply explicitly model the latent variables (Equation 7.3) and run n training epochs, performing backpropagation to find new ComplEx embeddings (Equation 7.4).

$$\mathcal{G}' \leftarrow \mathcal{G} + \{(e_i, \mathtt{isA\_concept}, c_j)\ \forall e_i \in S_j, \forall c_j \in \mathcal{C}\} \tag{7.3}$$

$$\mathbf{\Theta}' \leftarrow \text{ComplEx}(\mathcal{G}', \mathbf{\Theta}) \tag{7.4}$$

Algorithm 3 Concept Formation with EM

```
1:  Initialise the parameters $\mathbf{\Theta} := \mathbf{E}, \mathbf{C}, \mathbf{W}$ where $\mathbf{E} \in \mathbb{C}^{N_e \times k}$, $\mathbf{C} \in \mathbb{C}^{N_c \times k}$ and
    $\mathbf{W} \in \mathbb{C}^{N_r \times k}$, and cluster assignment sets $S_1, \ldots, S_{N_c}$

2:  function ConFormE($\mathbf{\Theta}, \mathcal{G}, S_1, \ldots, S_{N_c}$)
3:      for $n$ iterations do
4:          E-step
5:          for $e \in \mathcal{E}$ do                                  ▷ Make hard assignments
6:              $c^* \leftarrow \underset{c}{\text{argmax}}\ \exp\{-\text{score}(e_i, ISA, c)\},\ \forall c \in \mathcal{C}$
7:          M-step
8:          $\mathcal{G}' \leftarrow \mathcal{G} + (\{e_i, \text{ISA}, c_j\}, j = \{1, \ldots, N_c\}, \forall e_i \in S_j)$   ▷ Augment the graph
9:          for $t$ epochs do
10:             $\mathbf{\Theta}' \leftarrow \text{KGE}(\mathcal{G}', \mathbf{\Theta})$                ▷ Train a KGE model
11:     Evaluate $\mathbf{\Theta}'$ using $\mathcal{G}$
```

In Algorithm 3 we summarise the entire ConFormE procedure. In comparison with Algorithm 2, we introduce new parameters n and t, where n is the total number of iterations and t is the number of ComplEx training epochs between each E-step. We note that the previously presented similarity-based concept learning approach can be seen as simply performing the M-step (steps 7-11) with $n = 1$ and $t \leq 100$. Meanwhile, in ConFormE we will consider $t \in \{1, 2, 3\}$ and use n such that the total number of ComplEx training steps is ≤ 100.

In terms of the space complexity of this algorithm, the M-step is simply the complexity of the score function in a standard ComplEx training procedure i.e. $\mathcal{O}(N_e + N_c k + N_r k)$, except that we have $(N_e + N_c)$ instead of N_e, where N_e is the number of original entities and N_c is the number of concept entities. The E-step, on the other hand, is simply $\mathcal{O}(N_e N_c)$.

Chapter 8

Datasets

To evaluate the performance of our proposed approach, we used a number of benchmark datasets. In this section we briefly describe the type of information each KG stores, justify our dataset choices, and then perform some data exploration to gain an understanding of the data distributions and foresee potential challenges specific to each dataset.

8.1 Description of Datasets

Due to the computational expense associated with large datasets, we used a smaller dataset for preliminary experiments. Commonly used small benchmark KGs include UMLS [McCray, 2003], Kinship [Denham, 1973] and Nations [Rummel, 1999]. We decided to choose UMLS (Unified Medical Language System) — a biomedical ontology made up of binary predicates, such as `causes` or `interacts_with` and entities which correspond to high-level biomedical concepts, such as `amino_acid` or `virus`. The reasons for this choice are twofold: firstly, it is the largest out of the small datasets, consisting of 135 entities, 46 predicates and 5215 training triples. Moreover, by experimenting with this dataset, we gain an insight into whether our approach has potential to be applied in the biomedical domain and tested on larger biomedical KGs in the future.

As one of our aims is to develop a method that scales well with data size, in our experiments we focus predominantly on large datasets which are commonly used in evaluating the state-of-the-art. FB15K and WN18 were some of the first publicly-available large KGs. FB15K is a subset of Freebase [Bollacker et al., 2008], a KG made up of general facts about the world, while WN18 is a subset of Wordnet [Miller et al., 1991], a dataset containing semantic knowledge of words. However, in the recent years it has been pointed out that both of these datasets consist of test triples which are essentially inverses of those present in the training set [Dettmers et al., 2017, Toutanova and Chen, 2015]. This in turn makes the learning task much easier, as the model can learn that, for e.g. relationships `film/film_producer` and `film_producer/film` are inverses of each other. Indeed, it is possible to achieve very high MRR of 0.86 and 0.95 on FB15K and WN18 respectively [Lacroix et al., 2018]. To address this issue, two new datasets were released: WN18RR [Dettmers et al., 2017] and FB15K-237 [Toutanova and Chen, 2015], which are subsets of the original ones but without triples that can be easily solved. Another popular dataset is YAGO3-10 [Mahdisoltani et al., 2015], a subset of YAGO which holds descriptive information about individuals, such as their profession, citizenship or gender. It is the largest KG we will consider in this work, though it is worth noting that all of these datasets are still relatively small subsets of the entire KG — for instance as of 2019 YAGO3 consists of more than 120M facts[1], while YAGO3-10 contains only 1M.

8.2 Statistical Analysis of Data Distributions

In order to interpret experimental results, it is essential to first attain an understanding of the statistical distributions of the datasets. To this end, we explore the distributions of entities and predicates for each of the datasets. A summary of basic statistics can be found in Table 8.1.

[1] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Train	Validation	Test
UMLS	135	46	5215	652	661
WN18RR	40,493	11	86,835	3,034	3,134
FB15K-237	14,541	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Table 8.1: Number of entities, predicates, and train, validation and test triples in the benchmark datasets.

8.2.1 UMLS

In Figure 8-1 we explore the distribution of the entities and predicates in the UMLS training set. Figure 8-1a shows in how many triples each entity participated in the training set — we observe that the distribution has a fairly long tail, however each entity occurs at least three times in the training set (Table 8.2), with the majority of the entities participating in under 100 triples as seen more clearly in 8-1b. The mean entity frequency is 77, however, they are a few entities which participate in a very large number of triples, the maximum here being 306. This pushes up the mean, as seen from the median which is at 58. The predicates, of which there are 46, are on the whole quite well represented, though one relation only occurs once in the entire training set and the distribution in 8-1c has a fairly long tail. While the median number of occurrences is 45 for the predicates (with a mean of 113), one of the predicates occurs 803 times.

8.2.2 WN18RR

WN18RR consists of 40k entities and 11 predicates. The entity distribution is very skewed as seen in Figure 8-2a, with some entities occurring up to 482 times while 384 entities do not participate in any train triples. However, the majority of entities participate in 1-5 triples, with the mean being 4.2 and median of 3 (Table 8.2). Hence, the data about each of the entities is quite sparse. Meanwhile, the distribution of the predicates is also skewed, with a maximum frequency of 35k and a minimum of 80.

Statistics of Occurrences in Train Set

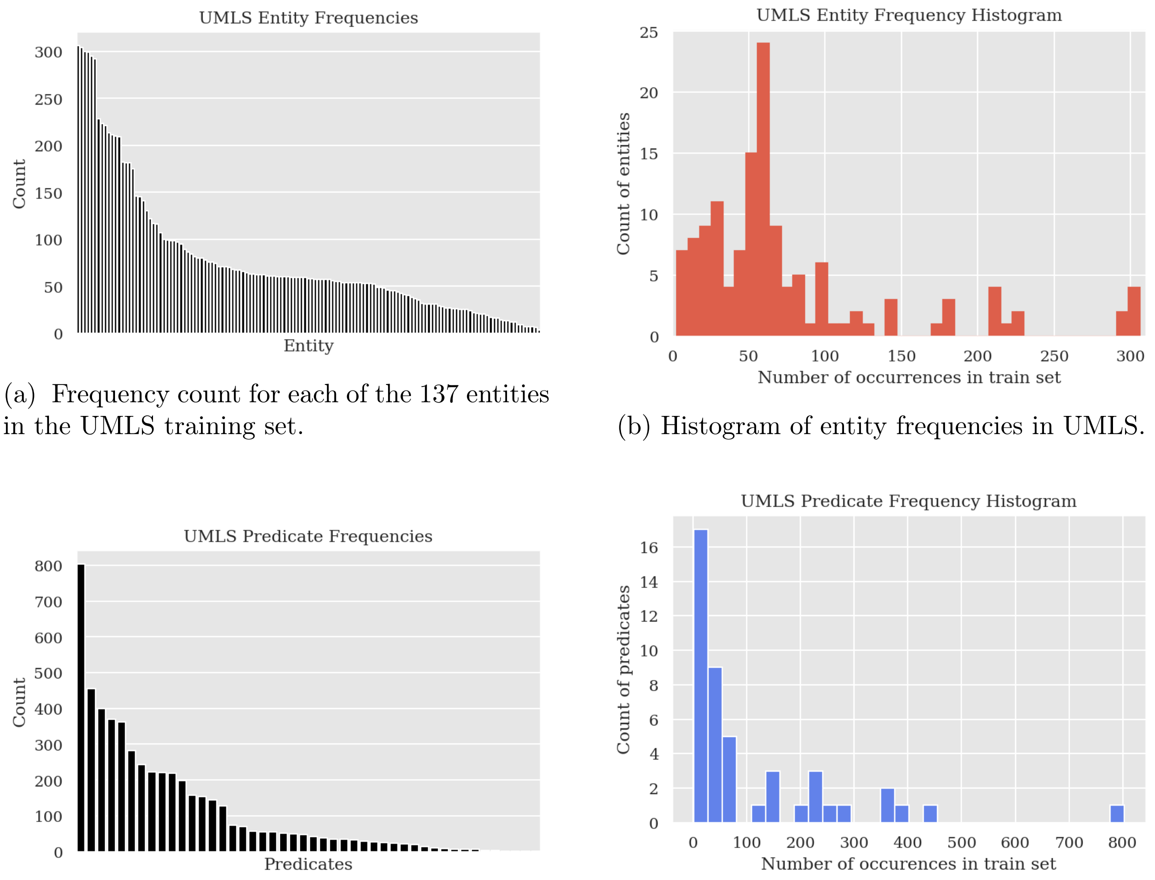
	UMLS		WN18RR		FB15K-237		YAGO3-10	
	Entities	Relations	Entities	Relations	Entities	Relations	Entities	Relations
Mean	77	113	4.2	7894	37	1148	17.5	29163
Median	58	45	3	2921	22	373	10	3757
Max	306	803	482	34796	7613	15989	61044	373783
Min	3	1	0	80	0	37	0	19

Table 8.2: Summary statistics of the entity and predicate distributions in the training set for the benchmark datasets. For example, each entity in WN18RR on average participates in 4.2 training triples.

This is reflected in the disparity between the mean and the median, with the former being 7894 and the latter 2921 — this is visualised best in Figure 8-2c.

8.2.3 FB15K-237

In FB15K-237 we also see a large skew (Figure 8-3a), with some entities participating in up to 7613 triples, while 36 are not present in any training triples (Table 8.2). However, here, the average number of triples each entity participates in is significantly higher, with a mean of 37 and a median of 22. The bigger challenge associated with this KG is the large number of predicates, of which there are 237. One can think of link prediction as multi-task learning, since we are simultaneously learning a single set of embeddings which we will use to predict for different predicates, and each predicate can be seen as a different 'task', requiring a different representation. From this perspective, link prediction for datasets such as FB15K-237 is challenging, even when all of the predicates are well-represented, as they are here, with a mean frequency of 1148 and a median of 373. While the frequencies of most predicates lie under 4k, there is also a considerable number of predicates which occur very commonly, with one occurring 16k times — although that corresponds to only 5.9% of the training set.



(a) Frequency count for each of the 137 entities in the UMLS training set.

(b) Histogram of entity frequencies in UMLS.

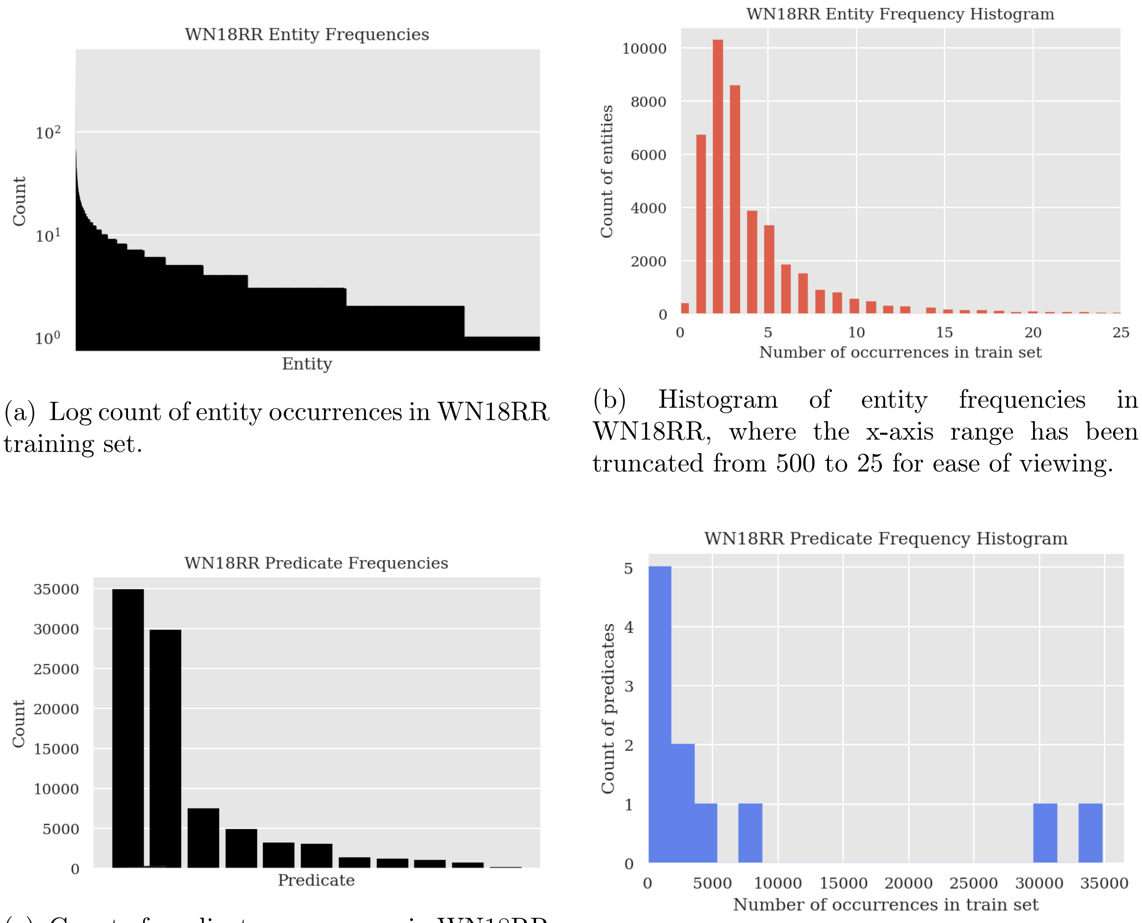
(c) Counts of predicate occurrences in the UMLS training triples.

(d) Histogram of predicate frequencies in UMLS.

Figure 8-1: UMLS entity and predicate distributions in the train set.

8.2.4 YAGO3-10

YAGO3-10 is so called because each of its entities participates in at least 10 triples across the train, validation and test sets. It is a very large dataset, with 123k entities, 37 predicates and over 1M training triples. Each of the entities participates on average in 17.5 triples and a median of 10, with 39 entities that participate in no training triples and a few entities occurring very frequently, with a maximum of 61k (Table 8.2). From the visualisation in Figure 8-4b we see that most entities participate in under 40 triples. The distribution of the 37 predicates is also heavily right skewed, as shown in Figure 8-4c, with the most frequent entity participating in 373k triples, while the rarest one takes part in 19. While this is rarely done in industry, one of the questions we wish to answer in this work is whether one can gain a better under-



(a) Log count of entity occurrences in WN18RR training set.

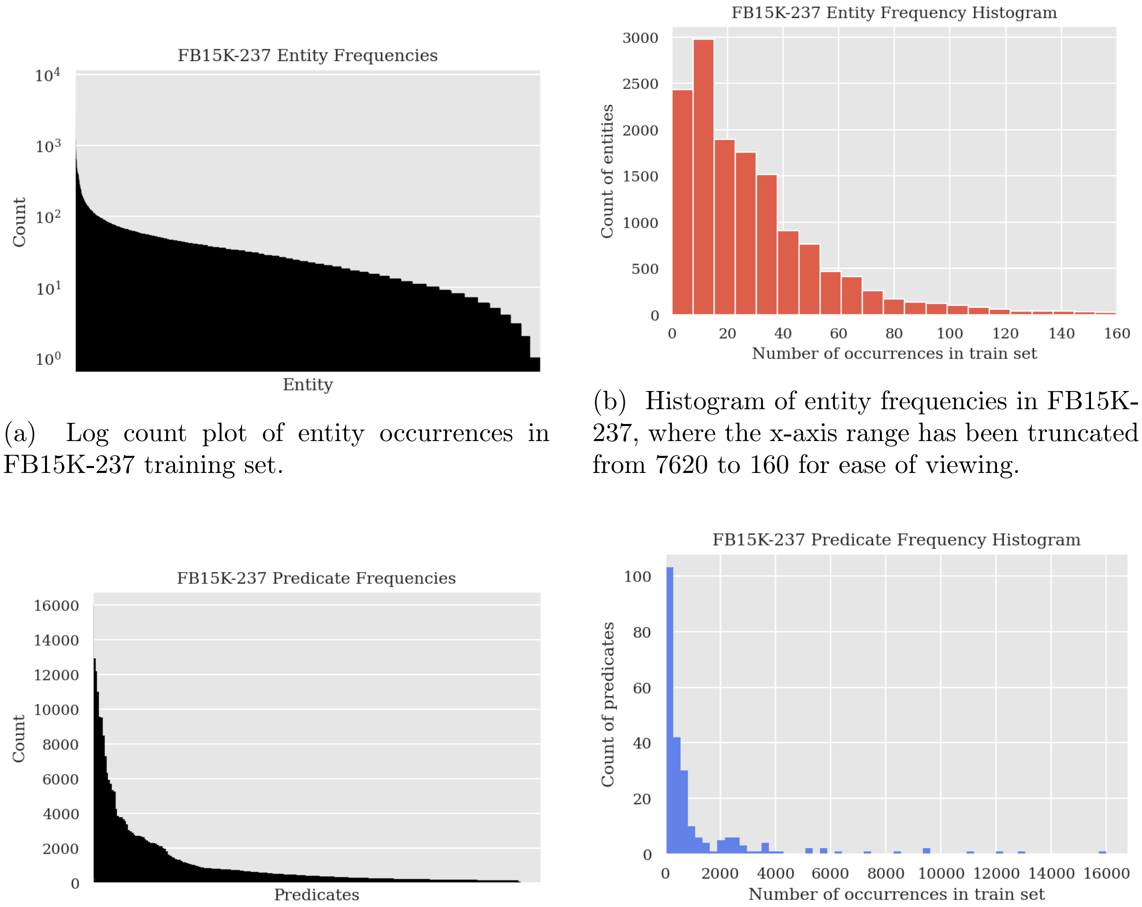
(b) Histogram of entity frequencies in WN18RR, where the x-axis range has been truncated from 500 to 25 for ease of viewing.

(c) Count of predicate occurrences in WN18RR training set.

(d) Histogram of the predicate frequencies in WN18RR training set.

Figure 8-2: WN18RR entity and predicate distributions in the train set.

standing of the strengths and shortcomings of the model performance by analysing the predictive performance of different types of predicates, e.g. for those that are very rare and very common.



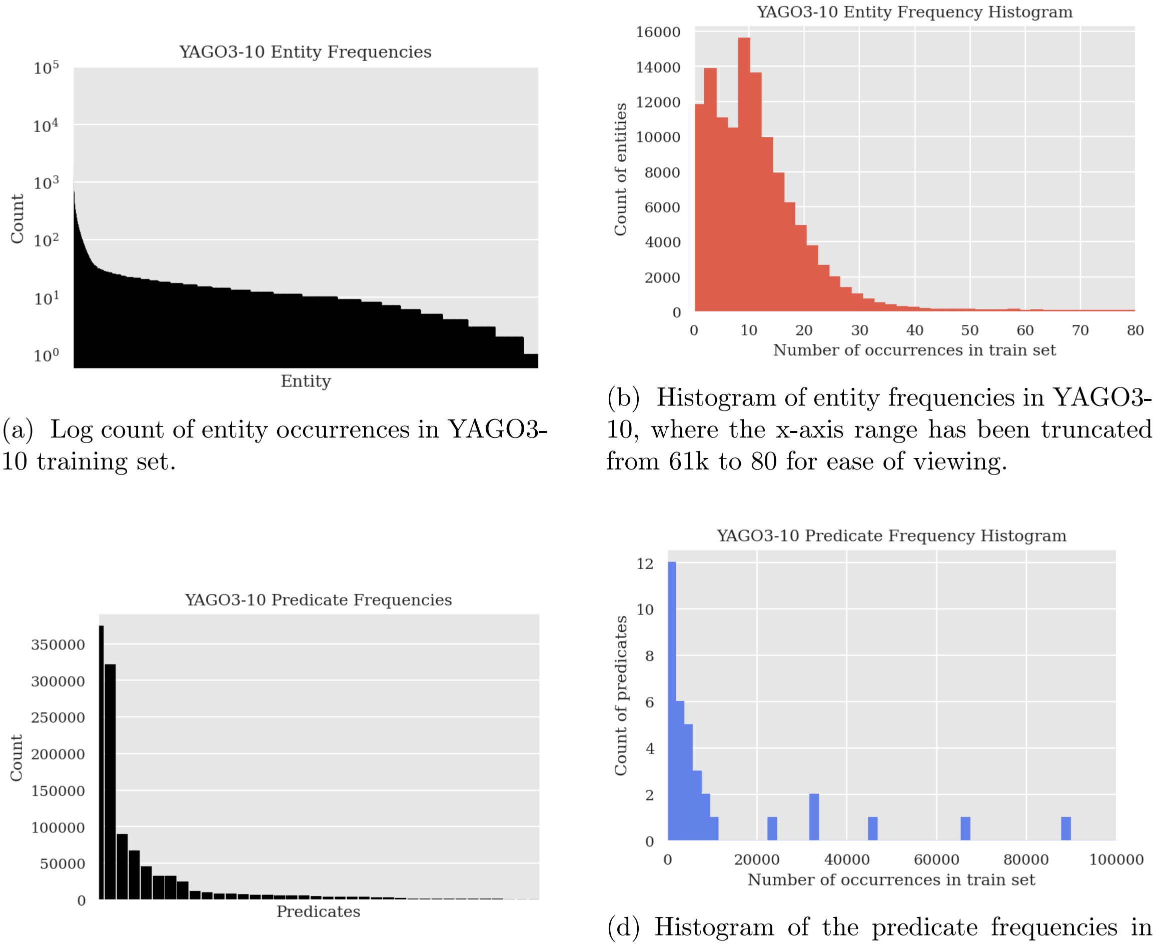
(a) Log count plot of entity occurrences in FB15K-237 training set.

(b) Histogram of entity frequencies in FB15K-237, where the x-axis range has been truncated from 7620 to 160 for ease of viewing.

(c) Count of predicate occurrences in FB15K-237 training set.

(d) Histogram of the predicate frequencies in FB15K-237.

Figure 8-3: FB15K-237 entity and predicate distributions in the train set.



(a) Log count of entity occurrences in YAGO3-10 training set.

(b) Histogram of entity frequencies in YAGO3-10, where the x-axis range has been truncated from 61k to 80 for ease of viewing.

(c) Count of predicate occurrences in YAGO3-10 training set.

(d) Histogram of the predicate frequencies in YAGO3-10, where the x-axis range has been truncated from 374k to 100k for ease of viewing.

Figure 8-4: FB15K-237 entity and predicate distributions in the train set.

Chapter 9

Experimental Results and Analysis

In this section we outline the technical details of our implementation and present experimental results. Rather than merely attempting to beat a benchmark, we are interested in gaining a deeper understanding of the impact that this explicit dataset augmentation has on the predictive power of the model. To this end, the experiments we present in this section are designed to answer specific questions about concept formation in knowledge graphs. In Sections 9.1 to 9.10 we experiment with using the similarity-based concept learning approach and then provide preliminary experimental results for ConFormE in Section 9.11.

9.1 Training ComplEx

We begin by introducing the experimental set-up followed in this work. The code implementation utilised here is built on top of a general link prediction framework developed by [Lacroix et al., 2018].

All of the metrics reported in this work are obtained as follows: during training we compute the validation and test MRRs — this is done every 3 epochs as it can be relatively expensive to compute, depending on the test size. We do not compute the train MRR at train time because, even for WN18RR, it requires > 4 minutes for a single computation. Instead, we use the train loss as a proxy to monitor performance on the train set. At the end of the training, we extract the highest validation MRR achieved and report the corresponding test MRR.

The training time of ComplEx can vary considerably with dataset size. Using a GPU with around 16GB RAM, UMLS needs under 3 minutes, WN18RR takes around 1h, FB15K-237 around 1.5-2h, while YAGO3-10 requires a minimum of 16h. Hence, many of the experiments here are performed using a larger hyperparameter range for WN18RR and FB15K-237, and a reduced range for YAGO3-10 due to time and computational costs.

9.2 Baselines

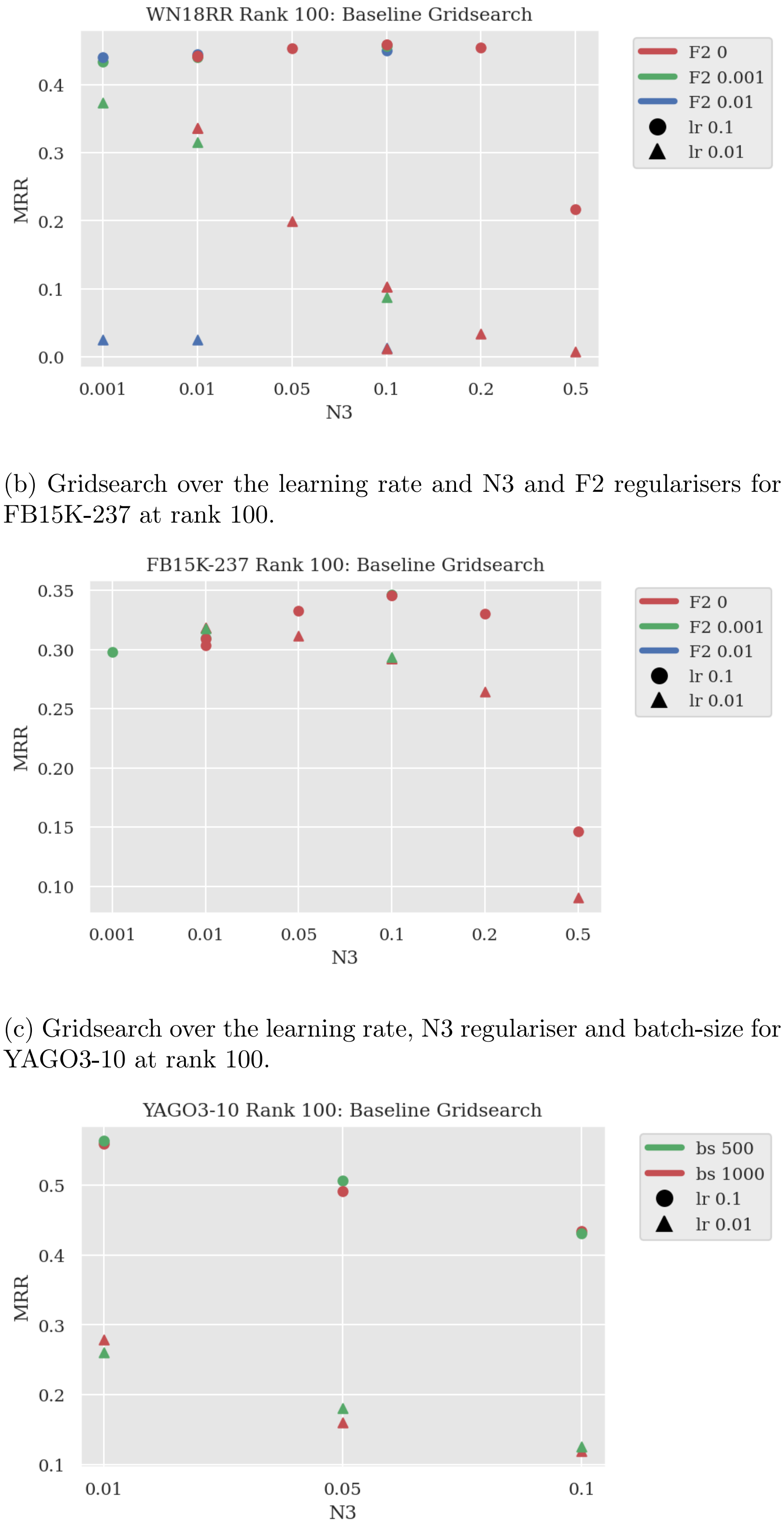
While concept invention in KGs is in itself a valuable task, our primary objective here is harnessing the rich latent information for the link prediction task. To this end, in order to make any comparative statements, we first need to measure the performance of ComplEx on the original datasets. Most of our experiments will focus on the three large datasets, WN18RR, FB15K-237 and YAGO3-10. It is worth noting that ComplEx easily achieves test MRR of > 0.95 on UMLS as it is a very small dataset, thus we will only be using it for proof-of-concept experiments rather than expecting to improve upon it. For each of the datasets, we perform a grid search over the hyperparameters. The values used to guide our hyperparameter search are motivated by [Lacroix et al., 2018], which have recently reported new state-of-the-art results on ComplEx. However, we note that [Lacroix et al., 2018] deploy reciprocal relations which we, for the most part, do not. Hence, we do not aim to replicate their results, although our baselines (Table 9.1) are not far off. As such, we use the standard cross-entropy loss and Adagrad optimiser. We limit the embedding size to 100 to begin with, making the assumption that if we observe significant improvements for a specific rank, we will observe a similar trend for other embedding sizes. We will test this hypothesis later on by repeating our final best-performing model for a range of embedding sizes. We experiment with using two different regularisers, the Frobenius (F2) and Nuclear 3 (N3) norms, and we vary the learning rate. We also explore two different batch-sizes for YAGO3-10, 500 and 1000, while keeping a constant batch-size of 500 for WN18RR and FB15K-237. For each of the datasets, we extract the combination which achieves the highest validation MRR and re-run it five times, each time with different seeds. Lastly, we report the mean and standard deviation of the test MRR from repeated runs as the baseline.

The baseline gridsearches for rank 100 for WN18RR, FB15K-237 and YAGO3-10 are shown in Figure 9-1, with the final results quoted in Table 9.1. We observe that a higher learning rate of 0.1 is optimal for all three datasets. We note that the F2 regulariser has little effect on both FB15K-237 and WN18RR, with the best performance being achieved by using an F2 of 0. For this reason, we limit our gridsearch for YAGO3-10 to only the N3 norm. Both WN18RR and FB15K-237 perform best with an N3 regularisation of 0.1, while YAGO3-10 performs better with a lower regularisation of 0.01.

For the remainder of this work, unless stated explicitly otherwise, we use the baseline hyperparameters as default.

	WN18RR	**FB15K-237**	**YAGO3-10**
MRR	0.462±0.002	0.3455±0.0004	0.559±0.002
H@1	0.428±0.003	0.2535±0.0007	0.489±0.003
H@3	0.475±0.001	0.379±0.001	0.601±0.002
H@5	0.498±0.002	0.4454±0.0005	0.64±0.002
H@10	0.528±0.002	0.5322±0.0006	0.685±0.001
H@50	0.597±0.004	0.713±0.0005	0.767±0.002
H@100	0.625±0.004	0.7806±0.0003	0.8±0.002

Table 9.1: Baseline results using ComplEx rank 100. Each baseline is a mean of 5 re-runs of the best hyperparameter combination.



(a) Gridsearch over the learning rate and N3 and F2 regularisers for WN18RR at rank 100.

(b) Gridsearch over the learning rate and N3 and F2 regularisers for FB15K-237 at rank 100.

(c) Gridsearch over the learning rate, N3 regulariser and batch-size for YAGO3-10 at rank 100.

Figure 9-1: Hyperparameter baseline gridsearch for WN18RR, FB15K-237 and YAGO3-10.

Table 9.2: Clustering Algorithms

Algorithm	**Parameters**	**Geometry Metric**
K-Means	`n_clusters`	Euclidean
Affinity Propagation	`damping`	Euclidean or pre-computed
DBSCAN	`eps` and `min_samples`	Euclidean or any pairwise
Spectral Clustering	`n_clusters`	nearest neighbours, rbf or pre-computed

9.3 Preliminary Experiments

9.3.1 Clustering Algorithms for Concept Formation

In this section, we discuss some of the practical advantages and challenges encountered with each clustering method. We begin by considering four different algorithms: K-Means, Spectral Clustering, DBSCAN and Affinity Propagation. Out of these, K-Means and Spectral Clustering require the number of clusters to be specified, while DBSCAN and Affinity Propagation do not. Moreover, the latter two can be used to cluster a similarity matrix, such as Kernel Representation. The parameters and metrics possible for each algorithm are summarised in Table 9.2. All of the implementations were done using the `sklearn` library[1] and the default parameters were used, unless stated otherwise.

The challenges with using DBSCAN soon became apparent. While DBSCAN does not require the number of clusters to be specified, it does require two hyperparameters: `eps` and `min_samples`. We found that the algorithm is very sensitive to choice of hyperparameters, and choosing either of the two even slightly sub-optimally results in either no clusters being formed, or all entities being assigned to a single cluster — Figure 10-1 in the Appendix illustrates the drastically different results achieved with small parameter changes. Since we are using many different entity representations in this work — for instance, for random paths we consider 8 different representa-

[1] https://scikit-learn.org/stable/modules/clustering.html

tions for each dataset — fine-tuning the clustering algorithm to find the exact set of hyperparameters is a very expensive and time-consuming procedure. Hence, despite DBSCAN being able to produce high quality clusters when carefully fine-tuned, we find that its sensitivity to parameter choice is ill-suited to our problem.

Experimenting with clustering larger datasets, we found that Affinity Propagation was significantly slower than K-Means or Spectral Clustering while creating clusters of similar quality. Thus, we decided to narrow our algorithm choices to K-Means and Spectral Clustering for most experiments, though we make use of Affinity Propagation for clustering kernel matrices as it allows for input matrices of this type. Heuristic experiments with different metrics lead to little change in cluster assignments, hence we have used the default metrics throughout the experiments, which are Euclidean distance for K-Means and nearest-neighbours graph distance for Spectral Clustering.

9.3.2 Learning semantically-meaningful clusters

Firstly, we are interested in performing a proof-of-concept experiment to verify whether it is possible to learn semantically-meaningful concepts in a fully unsupervised way from embeddings constructed using only relational data. Assessing quality of a clustering in a *quantitative way* is a challenging task. In the case where one has *a priori* knowledge of what categories the entities should belong to, a simple solution is to compute the point-wise mutual information [Church and Hanks, 1989] between the true assignments and the clustering results. In fact, clustering entity embeddings to assess whether the embeddings model has learnt a semantically-meaningful latent representation could constitute a downstream task in itself. However, there is no existing information on concept assignments available for any of the benchmark datasets. Hence, we resort to first inspecting cluster coherence *qualitatively*.

To this end, we utilise the UMLS dataset since it contains only 134 entities, allowing for easy manual analysis. To train ComplEx, we use N3 = 0.005, F2 = 0, k = 100, a batch-size of 100, a learning rate of 0.1 and a maximum of 100 epochs. We next cluster the embeddings corresponding to the highest dev MRR (which in turn correspond to test MRR of 0.945), using K-Means and specifying `n_clusters`

= 10. The resulting cluster assignments can be found in Table 9.3. We find that the resulting clusters are indeed meaningful with, for instance, Concept 1 capturing the notion of an *animal* or Concept 4 representing *activity or behaviour*.

9.4 Random Clusters

Having verified in a *manual* way that it is indeed possible to learn meaningful clusters for small datasets, we note that doing so for larger datasets, such as YAGO3-10 which contains 123k entities, is intractable. Moreover, even if one was to analyse a small sample of cluster assignments, this approach provides no quantitative measure of *meaningfulness*. Instead, we propose assessing cluster quality via comparing link prediction performance of a model trained on a dataset augmented with **learned** clusters to that of a model trained using **random** concepts. To create a random clusters baseline we first define an arbitrary number of clusters and assign each entity to one such cluster at random. Next, we train a number of ComplEx models on datasets augmented with random triples and observe the effect on link prediction performance. These results will then serve us as a reference for comparing result distributions attained from training on a dataset with supposedly meaningful concept triples. If the performance on link prediction will be distinctly different, we can assume that the concepts learned from the proposed representation are not random but to some extent meaningful. Intuitively, we would expect the performance to decrease upon introducing random concepts since we are effectively introducing noise and pushing entities towards random directions in the embedding space.

For WN18RR and FB15K-237 we experiment with changing the regulariser strength, since the regularisation which performed best for the original dataset may not be optimal for the augmented dataset. However, the experiments in Figure 9-2 indicate that, in both cases, baseline regularisation is preferred. As expected, we find that introducing random concept triples into the KG leads to a decrease in model performance. However, it is somewhat surprising how robust ComplEx appears to be random concept triples. Consider, for instance, WN18RR, which has 87k training

Table 9.3: Example of cluster assignments generated for UMLS by clustering ComplEx embeddings using K-Means, for $n\_clusters = 10$.

Concept 1
alga
amphibian
animal
archaeon
bacterium
bird
fish
fungus
human
invertebrate
mammal
organism
plant
reptile
rickettsia_or_chlamydia
vertebrate
virus

Concept 2
amino_acid_peptide_or_protein
carbohydrate
chemical_viewed_functionally
chemical_viewed_structurally
eicosanoid
element_ion_or_isotope
food
inorganic_chemical
lipid
nucleic_acid_nucleoside_or_nucleotide
organic_chemical
organophosphorus_compound
steroid

Concept 3
health_care_related_organization
organization
professional_society
self_help_or_relief_organization

Concept 4
activity
behavior
daily_or_recreational_activity
diagnostic_procedure
educational_activity
governmental_or_regulatory_activity
health_care_activity
individual_behavior
laboratory_procedure
machine_activity
molecular_biology_research_technique
occupational_activity
research_activity
social_behavior
therapeutic_or_preventive_procedure

Concept 5
acquired_abnormality
anatomical_abnormality
anatomical_structure
body_part_organ_or_organ_component
cell
cell_component
embryonic_structure
fully_formed_anatomical_structure
gene_or_genome
tissue

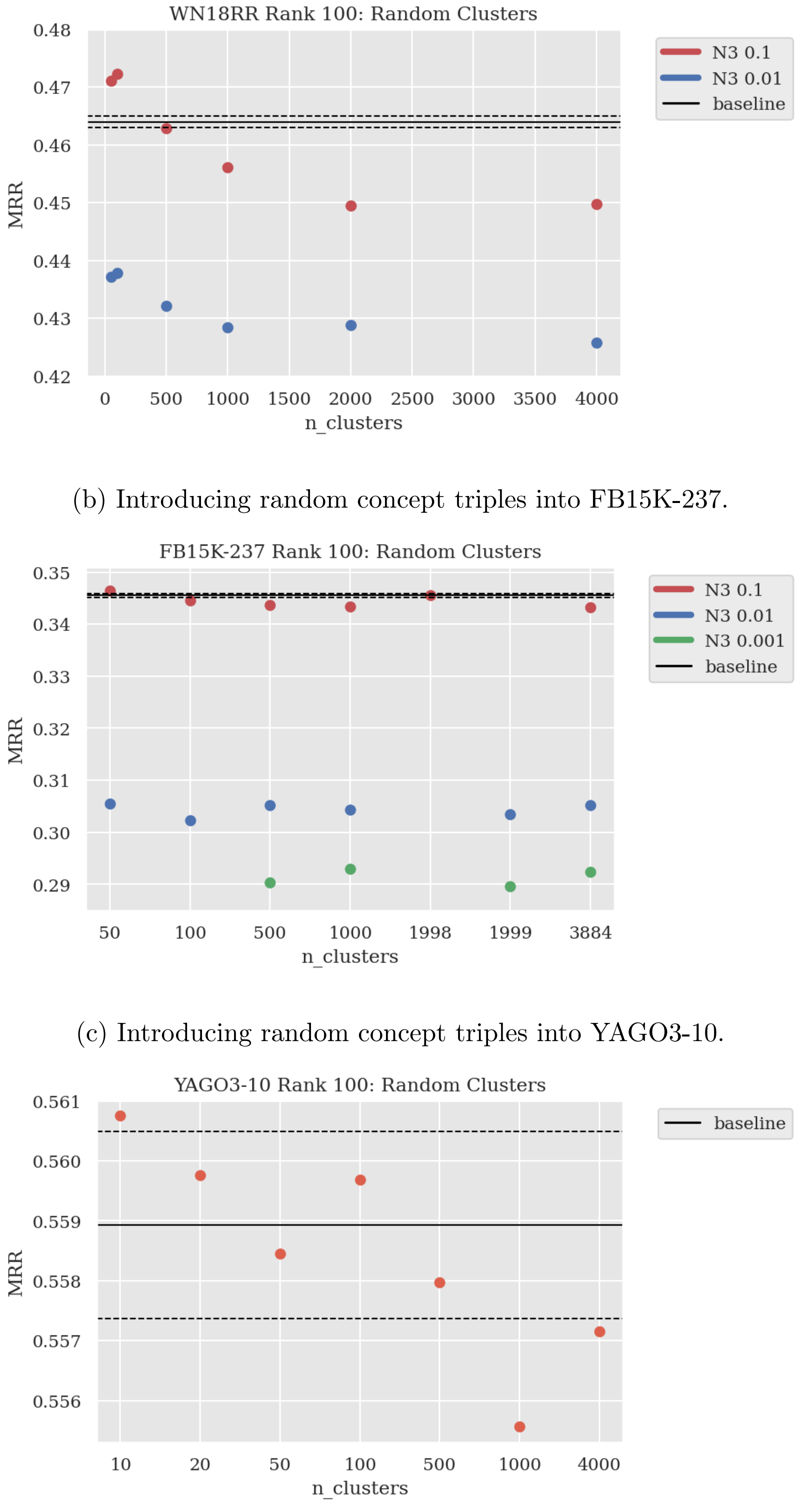
Concept 6
biologic_function
cell_function
cell_or_molecular_dysfunction
congenital_abnormality
disease_or_syndrome
experimental_model_of_disease
genetic_function
injury_or_poisoning
mental_or_behavioral_dysfunction
mental_process
molecular_function
natural_phenomenon_or_process
neoplastic_process
organ_or_tissue_function
organism_function
pathologic_function
phenomenon_or_process
physiologic_function

Concept 7
biomedical_occupation_or_discipline
classification
clinical_attribute
clinical_drug
conceptual_entity
drug_delivery_device
entity
environmental_effect_of_humans
event
finding
human_caused_phenomenon_or_process
intellectual_product
laboratory_or_test_result
manufactured_object
medical_device
occupation_or_discipline
organism_attribute
physical_object
regulation_or_law
research_device
sign_or_symptom
substance

Concept 8
age_group
family_group
group
patient_or_disabled_group
population_group
professional_or_occupational_group

Concept 9
antibiotic
biologically_active_substance
biomedical_or_dental_material
body_substance
chemical
enzyme
hazardous_or_poisonous_substance
hormone
immunologic_factor
indicator_reagent_or_diagnostic_aid
neuroreactive_substance_or_biogenic_amine
pharmacologic_substance
receptor
vitamin

Concept 10
amino_acid_sequence
body_location_or_region
body_space_or_junction
body_system
carbohydrate_sequence
functional_concept
geographic_area
group_attribute
idea_or_concept
language
molecular_sequence
nucleotide_sequence
qualitative_concept
quantitative_concept
spatial_concept
temporal_concept


(a) Introducing random concept triples into WN18RR.

(b) Introducing random concept triples into FB15K-237.

(c) Introducing random concept triples into YAGO3-10.

Figure 9-2: Performance of ComplEx trained on datasets where each entity has been assigned to a random cluster.

triples and 40k entities. By assigning each entity to a random concept cluster, we introduce 40k new 'noisy' triples into the dataset — in other words, 32% of the training dataset is now meaningless. Yet, even with 4000 clusters, the test MRR drops at most by 0.015 to 0.45. Another unexpected finding is that adding a small number of random clusters, $N_c < 50$, can in fact lead to a marginal improvement on all three datasets. One plausible explanation for this phenomenon is that the few concept entities, even when random, act as a regulariser — pulling entities closer together in the embedding space.

9.5 Entity Representation

In this section, we will focus on assessing quantitatively what effect explicit data augmentation has on link prediction performance. Specifically, we are interested in discovering how changing entity representation, clustering algorithm, and number of clusters will influence the predictive power of the model. However, all of these choices can be considered as hyperparameters to some extent and can potentially be co-dependent — perhaps the optimal number of clusters may vary, depending on the representation? Moreover, the ComplEx hyperparameters which gave best performance when training on the original dataset may not be optimal when training on the a dataset. To answer these questions, we need to perform a number of quite large gridsearches. We begin by considering a wide range of hyperparameter values and proceed to narrow the search for the consecutive experiments to reduce the time and computational costs.

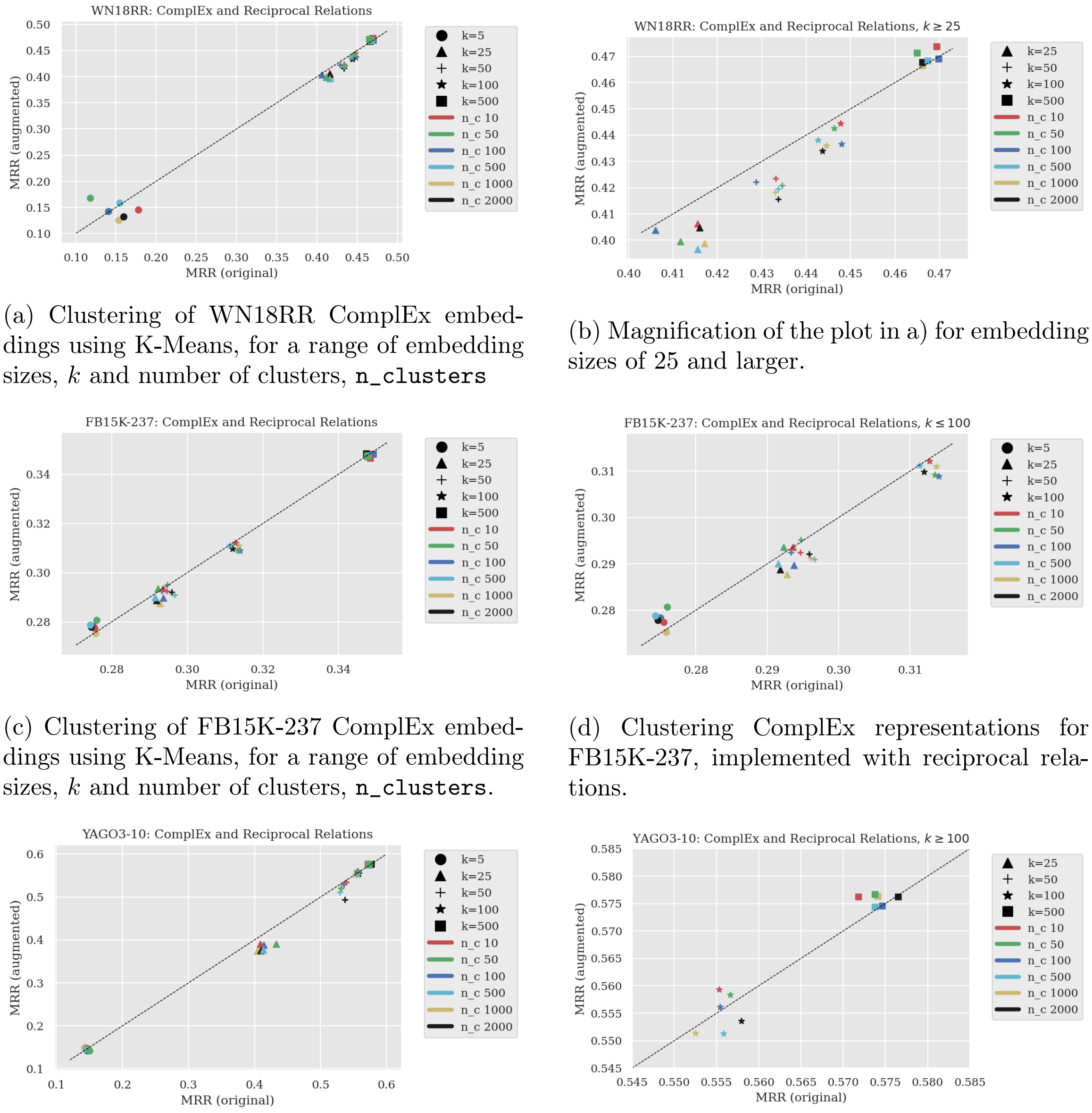
9.5.1 ComplEx Embeddings

The first set of results presented here is a gridsearch which was carried out in the early stages of the project as an initial heuristic. It is in some ways sub-optimal, primarily because we started by using the code provided by [Lacroix et al., 2018] which introduces reciprocal relations into the dataset. While we do not expect that the trends observed for a dataset with reciprocal relations will be drastically different from those observed for a dataset without them, it is nonetheless an unnecessary component, especially as we wish to start with the 'simplest model that works' first and increase the complexity as needed. Thus, while for all the other experiments we use a code implementation that does not introduce reciprocal relations, the results presented in Figure 9-3 were obtained with an implementation that does use reciprocal relations.

To obtain ComplEx embeddings we train a model of rank k using the best hyperparameters for that rank, i.e. the baseline hyperparameters. In the initial large gridsearch (Figure 9-3) we use the embeddings obtained at the end of the training process, while in the later experiments (Figure 9-4) we extract the embeddings corresponding to the highest validation MRR. Next, the embeddings are clustered, the new triples are introduced into the dataset, and a new ComplEx model — also of rank k — is trained on the augmented dataset.

In the first set of experiments, rather than using a single baseline for each embedding size, for each individual training we compare the MRR pre and post augmentation for a given seed. The two sets of MRRs are then compared, as shown in bi-sector plots in Figure 9-3, where the MRR on the x-axis corresponds to the test MRR obtained by training on the original dataset, and the MRR on the y-axis to that obtained by training on the augmented data. Hence, improvements upon the baseline appear above the $y = x$ line. In these experiments we use only one clustering algorithm, K-Means, and no hyperparameter search is performed for the augmented dataset. Instead, we use the hyperparameters reported in [Lacroix et al., 2018] as optimal for both, original and augmented datasets. For all three datasets we experiment with five different embedding sizes, $k \in \{5, 25, 50, 100, 500\}$ and vary the cluster number: `n_clusters` $\in \{10, 50, 100, 500, 1000, 2000\}$.

While we expected to observe either a marked increase or decrease in the performance metrics, instead what we observe is that the modification makes very little difference, with most points distributed to some extent normally about the baseline. Ideally, we would have liked to have repeated the entire gridsearch using the standard implementation, i.e. without the reciprocal relations. While this has not been possible



(a) Clustering of WN18RR ComplEx embeddings using K-Means, for a range of embedding sizes, k and number of clusters, `n_clusters`

(b) Magnification of the plot in a) for embedding sizes of 25 and larger.

(c) Clustering of FB15K-237 ComplEx embeddings using K-Means, for a range of embedding sizes, k and number of clusters, `n_clusters`.

(d) Clustering ComplEx representations for FB15K-237, implemented with reciprocal relations.

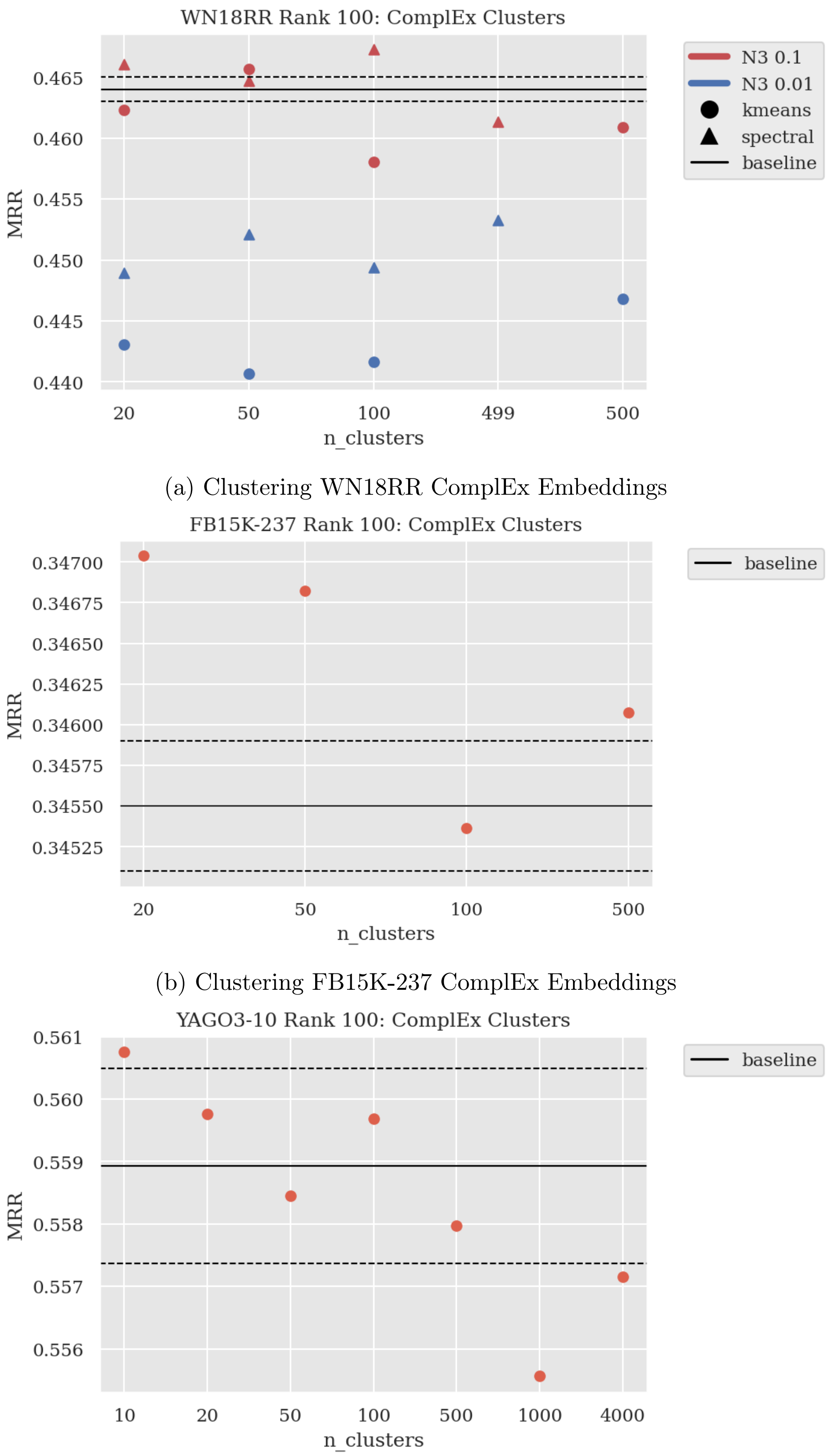
(e) Clustering of YAGO3-10 ComplEx embeddings using K-Means, for a range of embedding sizes, k and number of clusters, `n_clusters`.

(f) Clustering ComplEx representations for YAGO3-10, implemented with reciprocal relations.

Figure 9-3: Large gridsearch over rank size and number of clusters with clusters learned from ComplEx representation. Implemented with reciprocal relations.

due to time constraints, we present just a few combinations for embedding size of 100 in Figure 9-4.

Without the use of reciprocal relations we also observe a similar lack of trend — addition of clusters can lead to a small increase or decrease in test MRR, but the effect is not the same for all embedding sizes. This finding is quite surprising, especially as



(a) Clustering WN18RR ComplEx Embeddings

(b) Clustering FB15K-237 ComplEx Embeddings

(c) Clustering YAGO3-10 ComplEx Embeddings

Figure 9-4: Performance of models trained on datasets with clusters learnt from ComplEx embeddings, using the standard implementation (no reciprocal relations).

a recent work [Zhang et al., 2018] reported an improvement by clustering the TransE *relation* embeddings and training TransE on the new dataset. However, firstly, it is worth noting that the baselines quoted in that paper are much lower than the current state-of-the-art, such as [Lacroix et al., 2018] or those used in this work. Secondly, the geometries learned by ComplEx and TransE are distinctly different. While it is beyond the scope of this work, it would be instructive to repeat the experiments presented here using a translational model, such as TransE or RotatE.

One way to explain these findings is by considering that the entities which we have classified as being similar using a clustering algorithm have been placed in close proximity by ComplEx itself during the first training run. To illustrate this, we plot a t-SNE visualisation of ComplEx embeddings for UMLS (Figure 9-5) where ComplEx has been trained on the original dataset, and the entities are coloured according to cluster assignments generated using K-Means. Hence, we are 1) extracting the latent structure already learnt by the model; 2) explicitly modelling it; and 3) expecting that explicitly reinforcing the latent structure will lead to an improvement in performance. Indeed, the experimental results suggest that the latent concept information which we explicitly introduce is already captured in the initial ComplEx embeddings. Therefore, in the following sections we experiment with using other entity representations which may capture connectivity patterns that ComplEx is not capable of.

9.5.2 Weisfeiler-Lehman Kernel

In this section we describe the technical details concerning the construction of the WL Kernel (see Section 6.1.2 for theoretical background) for node entities in KGs. We adopt the publicly available implementation of the WL Kernel for the RDF framework[2] based on the work of [de Vries, 2013].

Obtaining this entity representation has been challenging in many ways. Firstly, there are few existing adaptations of kernel methods which can be readily used for multiply connected graphs that are also potentially not fully connected — both of these features are characteristic of large KGs. Moreover, computation times have

[2] https://github.com/deeplego/wl-graph-kernels

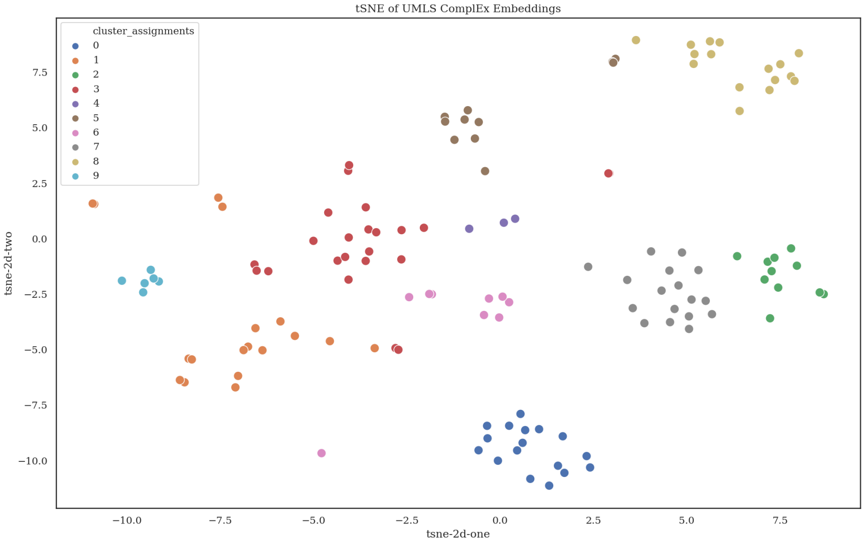


Figure 9-5: t-SNE visualisation of UMLS ComplEx embeddings, coloured according to K-Means assignments.

been a major obstacle. To compute the WL kernel, we first extract a small subgraph to represent each entity. The size of this subgraph is controlled by a hyperparameter `subgraph_depth`, which typically ranges from 1 to 6. The other hyperparmeter in this representation is the number of re-labelling iterations, `n_iter`, where `n_iter` $\in \mathbb{Z}_+$. The time complexity increases significantly with both, increasing `subgraph_depth` and `n_iter`. Computing a WL Kernel of `subgraph_depth` $= 1$ and `n_iter` $\in \{0, 1, 2, 3\}$ took over 2 days for both, FB15K-237 and WN18RR. It is worth noting here that while FB15K-237 has fewer entities ($N_e = 14951$) than WN18RR ($N_e = 40943$), FB15K-237 is more densely connected, with each entity being connected on average to 37 other entities, as opposed to 4 for WN18RR. For this reason, the subgraphs for FB15K-237 are larger than those for WN18RR. Indeed, attempts at computing `subgraph_depth` > 1 for FB15K-237 have crashed at 26% after three days. While it has been possible to construct WL Kernels for WN18RR for higher subgraphs depths, we stumbled upon another difficulty which made it impossible to use any of the kernel matrices generated for WN18RR. Consider the kernel matrix is by definition $\mathbf{K} \in \mathbb{R}^{N_e \times N_e}$. In case of WN18RR, that corresponds to an embedding matrix that consists of $40943^2 \approx 1.7$ billion entries, resulting in file size of > 4GB. For comparison, ComplEx embeddings we consider in this work are **at most** of rank 2000, and since they consists of the real and imaginary part the total embedding size is 4000. This results in an embedding matrix of 163 million entries, which is a tenth of the kernel matrix. Attempts at loading and clustering kernels matrices for WN18RR have been unsuccessful due to the high RAM requirements. However, it has been possible to compute and cluster a few kernel matrices for FB15K-237, as it consists of significantly fewer entities, using `subgraph_depth` $= 1$ and `n_iter` $\in \{0, 1, 2\}$.

Computing a kernel matrix can be thought of as constructing entity representations in a high-dimensional latent space and measuring similarity between them. Thus, each entry of the matrix $\mathbf{K}_{ij}$ is a measure of similarity between entities i and j. To cluster this representation we require an algorithm which accepts a 'pre-computed' similarity matrix as input, such as Affinity Propagation or DBSCAN. In this work, as discussed earlier, we choose Affinity Propagation (AP). As AP does not require the number of clusters to be specified and is deterministic we only perform a single clustering for each kernel representation, using the default parameters of 0.5 for `damping`. We also experimented with normalising the kernel matrix, as recommended in the code implementation.

The results on link prediction, as shown in Figure 9-6, first of all clearly show that higher (baseline) regularisation of $N3 = 0.1$ performs better. We found that AP was not able to find unique cluster centres for the normalised kernels, hence the results shown here are only for unnormalised kernels. For `n_iter` $= 0$, i.e. a kernel generated with no re-labelling steps, AP finds 462 unique cluster centres and the link prediction performance is not significantly different from training on a standard dataset. This is to some extent what we have expected, as it is the re-labelling step that allows for a more meaningful comparison of subgraphs. Affinity Propagation has not been able to cluster the representation generated using `n_iter` $= 1$, assigning each entity to a single cluster. However, somewhat surprisingly, this lead the biggest improvement in the link prediction performance during this experiment, despite the fact that this modification could have been easily achieved without the WL kernel. Also, it should

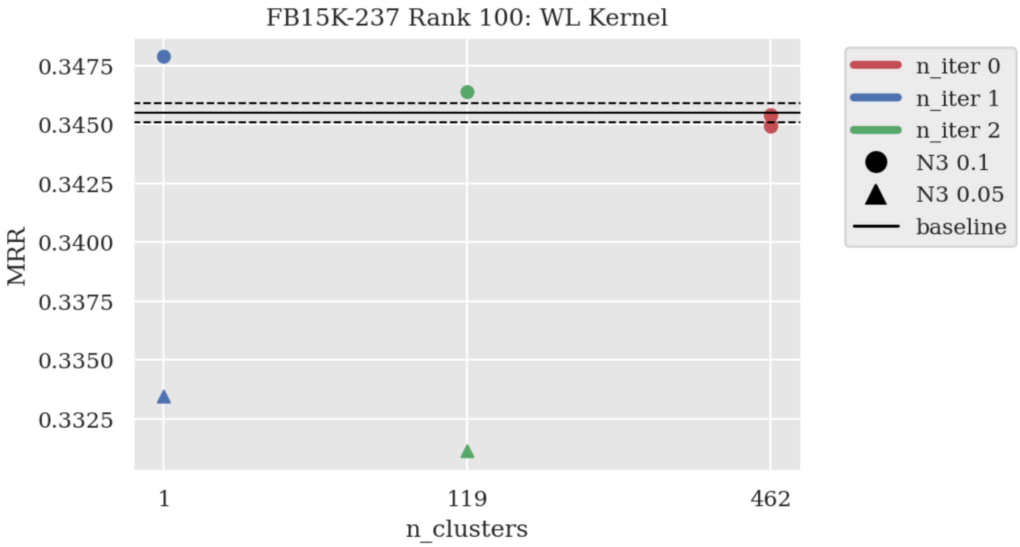


Figure 9-6: Training ComplEx on dataset augmented using clusters learnt from the Weisfeiler-Lehman kernel for the FB15K-237 dataset, where the representations were created using maximum subgraph depth of 1 and clustering was performed using Affinity Propagation, with a damping factor of 0.5.

be stressed that even when constructing a single concept cluster, the test MRR is not significantly higher than the best test MRR achieved when clustering ComplEx embeddings. Lastly, when creating a kernel representation using `n_iter` = 2, AP finds 119 clusters but we observe negligible, if any, improvement.

Consequently, we can firstly infer that clustering kernel representations does not show much promise in terms of improving generalisation for link prediction, suggesting that using this representation does not enrich the representations learnt by ComplEx. Aside from the link prediction performance results, this experiment also demonstrates the challenges associated with learning over large KGs, which render some of the most commonly used and theoretically-backed approaches, such as kernel methods, intractable due to long computational times and large storage requirements associated with $N_e \times N_e$ matrices.

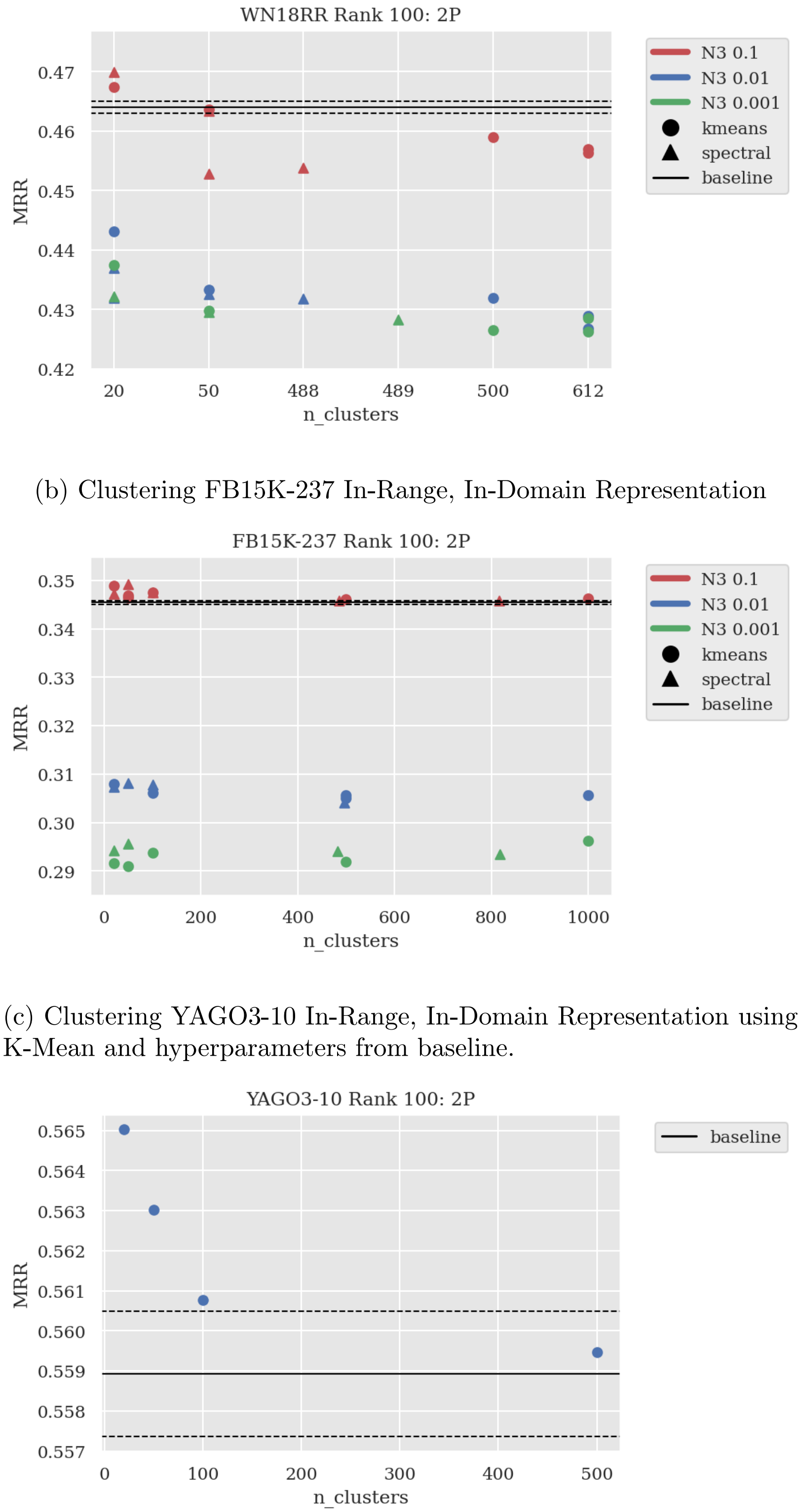
9.5.3 In-range, In-domain

As seen in the previous section, scalability and representation sizes are of paramount importance for large knowledge graphs, especially as the datasets we experiment with here are merely subsets of the entire KGs. Experiments presented in this section explore the use of In-range, In-domain representation inspired by rule-based approaches, as described in Section 6.1.3. We will at times refer to it using the abbreviation 2P, which arises from the representation size being twice the number of the predicates in the given dataset. Being one of the simplest possible representations, it is fast and efficient to construct, taking only seconds even for YAGO3-10. Here, we generate 2P embeddings for all three large KGs and cluster them, experimenting with both Spectral Clustering and K-Means, and exploring a wider range of regularisations for WN18RR and FB15K-237. Due to the computational time associated with training YAGO3-10 we use the hyperparameters which performed best for the unaugmented dataset.

Upon generating this representation we note that the resulting embedding matrix is sparse for most entities. This is likely because the entities tend to 1) participate in few relations (see Datasets Section 8 for detailed statistics); and 2) even if an entity participates in multiple triples, the predicates are often repeated. Therefore, we suspect that this representation might be too primitive to capture insights that are not already captured by ComplEx.

The results, as shown in Figure 9-7, indeed indicate that this representation is too naive. In the first place, we observe that both Spectral Clustering and K-Means have difficulties creating 500 clusters, especially for WN18RR (Figure 9-7a) in which the entities are connected to very few predicates. From test MRRs we infer that the concept triples introduced into the train dataset do not add significant insights that have not already been captured by ComplEx, though we do observe a decided preference towards a smaller number of clusters across all 3 datasets. We also once again observe that the regularisations which performed best on the original datasets perform best on the modified KGs for WN18RR and FB15K-237, and note that there is no significant difference in performance between K-Means or Spectral clustering.

(a) Clustering WN18RR In-Range, In-Domain Representation



(b) Clustering FB15K-237 In-Range, In-Domain Representation

(c) Clustering YAGO3-10 In-Range, In-Domain Representation using K-Mean and hyperparameters from baseline.

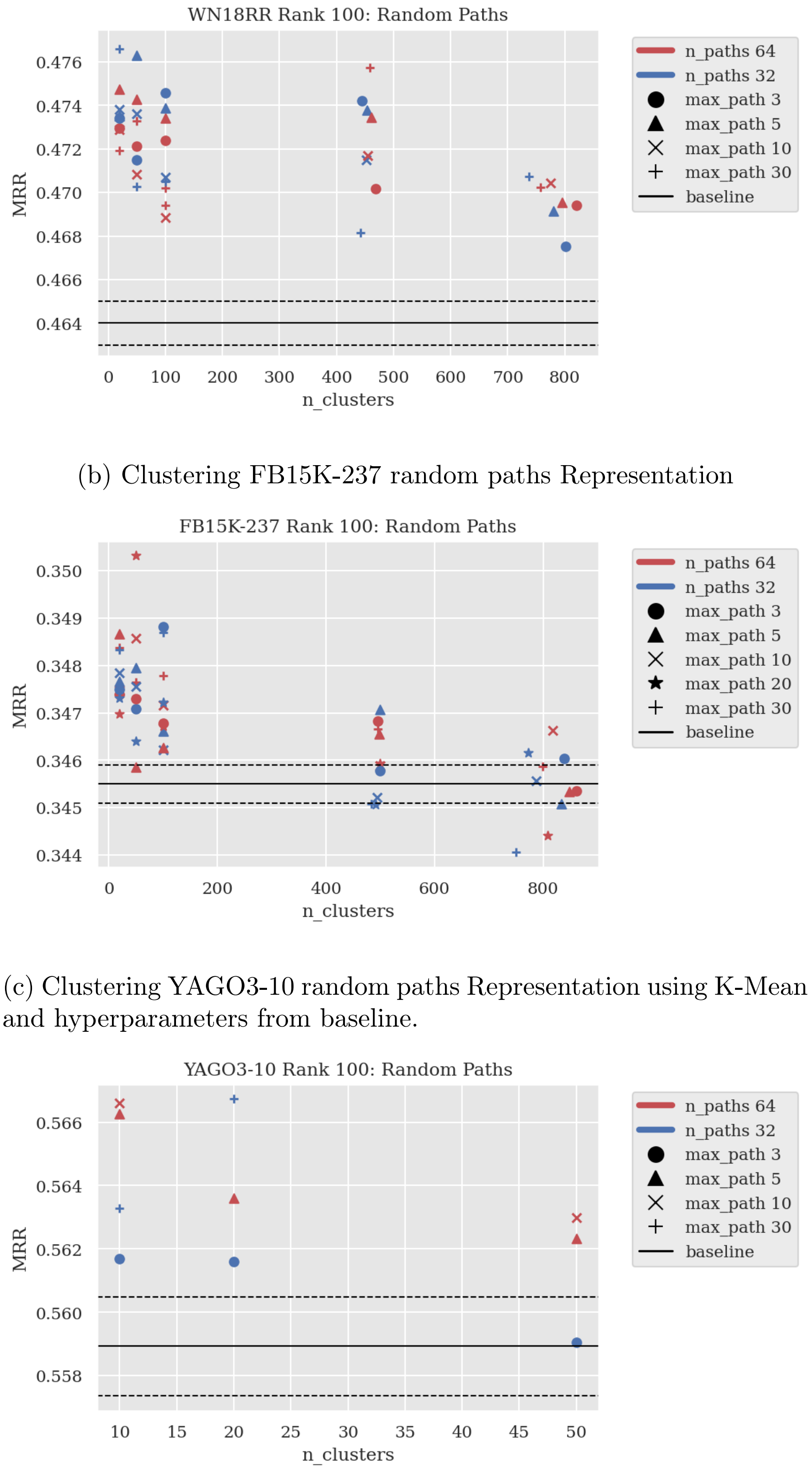
Figure 9-7: Performance of models trained on datasets with clusters learnt from In-Range, In-Domain Representation

9.5.4 Random Paths

One of the weakness of ComplEx, and most feature-based models, is that they only consider the data as sets of triples. Therefore, they do not take into account potentially useful information that is implicitly present in the KG structure in e.g. multi-hop paths between entities. One way to capture this information is by sampling random paths, also known as random walks. This representation can be seen as an extension of 2P: while in 2P we only consider the predicates directly connected to each entity, in random paths we randomly visit the neighbouring entities and record the predicates we encounter. It is worth noting that, in contrast to 2P, random paths requires constructing an adjacency dictionary storing information about the direct neighbourhood of each node, which can be quite computationally expensive to create. However, this structure only needs to be generated once for each KG and can be readily re-used to quickly and efficiently generate any number of random paths of desirable length. The method described in Section 6.1.4 allows for specifying three different hyperparameters: the number of paths walked to represent each entity, `n_paths`, the minimum length of path, `min_path`, and the maximum path length, `max_path`. Choosing a suitable range for these parameters *a priori* has been challenging. Typically, one can use graph measures, such as graph diameter, to estimate a the size of the graph or compute the mean Dijkstra path [Dijkstra, 1959] for all nodes in the graph, where the Dijkstra path is the shortest path between two nodes in a graph. However, all of these measures assume that the graph is fully connected and that is not the case for large KGs. Therefore, we use two heuristics to guide our hyperparameter choice: first, we sample around 80 thousand node pairs from each graph and compute the shortest path between them, discarding the pairs between which no path exists. We find that across all three datasets the longest shortest path is around 13-15 hops. Secondly, we consult the hyperparameters quoted in literature. DeepWalk [Perozzi et al., 2014] is an algorithm which samples random walks from KGs and generates node embeddings from these using a deep learning framework. The authors claim that it is sufficient to sample around 32 to 64 random walks from each node and suggest 40 steps as a suitable length in most cases. Consequently, we decide to experiment with `min_path` = 2, `max_path` $\in \{3, 5, 10, 20, 30\}$ and `n_paths` $\in \{32, 64\}$, limiting the number of representations for YAGO3-10 due to the longer training times.

Considering simultaneously a range of path lengths and number of path samples, alongside with a varying number of clusters, regulariser strengths and clustering algorithms results in a very large parameter space. As our results obtained thus far suggest that there is no clear preference between Spectral Clustering and K-Means we restrict our experiments here to only Spectral Clustering. Moreover, as in the above experiments we have consistently found that the optimum regulariser strengths for the unaugmented dataset corresponded to the optimum regulariser strengths for the augmented datasest, we use a single N3 strength for each dataset.

The results obtained for link prediction (Figure 9-8) obtained by training ComplEx on the augmented datasets show a lot of promise. For WN18RR, every path representation, even when clustered into a large number of clusters, leads to an improvement upon the baseline. We observe a similar trend for FB15K-237, though the increase is not as large, especially for large numbers of clusters. It is important to note that the two datasets are very different, with FB15K-237 being made up of a huge number of predicates — it is possible that the parameter ranges chosen for random paths are sub-optimal for FB15K-237 and we do not capture the connectivity patterns very well. One quite remarkable aspect about the improvement seen for WN18RR is that the random paths embeddings are of dimension $\mathbf{e} \in \mathbb{R}^{2 \times N_r}$. For WN18RR $N_r = 11$, making the resulting embedding of rank 22. It is quite surprising that the information obtained from clustering such low rank representation leads to significant increase in performance. Meanwhile, for FB15K-237 the embedding vectors are much larger, with $N_r = 237$, bringing the embedding size to $\mathbf{e} \in \mathbb{R}^{474}$. The results for YAGO3-10 also show promise, with nearly all representations outperforming the baseline. Comparing the test MRR results for random paths (Figure 9-8) and random clusters (Figure 9-2), we observe that they are markedly different, which suggests that we are not introducing noise, but rather meaningful information into the model. We also illustrate some example concepts learned for WN18RR in Table



(a) Clustering WN18RR random paths Representation

(b) Clustering FB15K-237 random paths Representation

(c) Clustering YAGO3-10 random paths Representation using K-Mean and hyperparameters from baseline.

Figure 9-8: Performance of models trained on datasets with clusters learnt from random paths representations

9.4. While the improvement for any of the dataset are by no means huge, they are nonetheless significant, and show that explicitly introducing cluster entities into the dataset can be beneficial.

Table 9.4: Example fragments of prototypical concepts learnt for WN18RR using random paths representation and Spectral Clustering.

Concept 1	**Concept 2**	**Concept 3**	**Concept 4**
exhale	accurate	desmidiaceae	american_civil_war
pant	inaccurate	mastigophora	american_revolution
pufe	inactive	dinoflagellata	arab-israeli_war
blow	adequate	zoomastigina	balkan_wars
wear	maladroit	polymastigina	battle_of_wake
tease	advantageous	phytomastigina	chinese_revolution
abstract	adventurous	cryptophyta	english_civil_war
shake	affirmative	sporozoa	korean_war
nooze	negative	telosporidia	napoleonic_wars
nap	afraid	malacopterygii	punic_war
wake_up	unafraid	cypriniformes	seven_years_war
wash_up	aggressive	cyprinidae	spanish-american_war
invigorate	unagitated	abramis	thirty_years_war
frown	alert	catostomidae	vietnam_war
scowl	alive	cyprinodontidae	world_war_i
goap	altruistic	poeciliidae	world_war_ii

9.6 Can oversampling reinforce the constraints?

One could think of concept memberships as constraints — assigning each entity to a specific region of embedding space. However, while in mathematics constraints such as e.g. Lagrangian constraints are always enforced, when training a KGE model the concept assignment constraint is only imposed when a concept triple is present in the batch sampled during training. As noted above, introducing concept triples into WN18RR introduces a large number of new triples. Meanwhile, for FB15K-237, where the number of train triples is 272k and number of entities is 15k, upon augmentation the new concept triples constitute only 5.1% of the new training set. This poses a question: is the reason for smaller increase in performance for FB15K-237 upon introducing clusters learnt from random paths caused by the ratio of *concept triples* to *rest of the training set* being significantly smaller for this dataset, in comparison with WN18RR?

In an attempt to answer this, we will experiment with altering the fraction of concept triples sampled during training. Once we cluster random paths embeddings, we add each triple to the dataset once and then uniformly sample additional triples. The number of concept triples added is controlled by a parameter α, which denotes the ratio of *concept triples* to *all train triples*. Let N be the number of original train triples and C the number of concept triples added to the training data. Then $\alpha = \frac{C}{C+N}$. In order to vary alpha, we compute the corresponding $C = \frac{\alpha N}{1-\alpha}$.

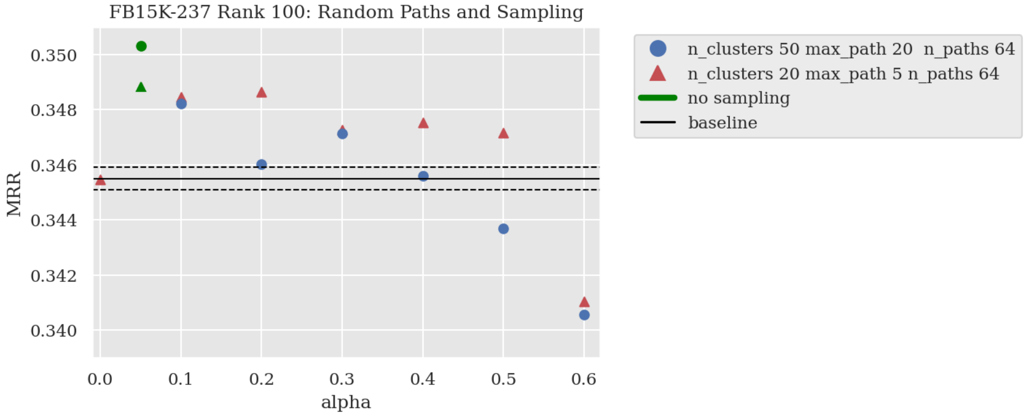


Figure 9-9: Oversampling concept triples learnt from random paths representation for FB25K-237.

The results (Figure 9-9) indicate that oversampling of concept triples has no beneficial effect on link prediction performance, suggesting that even when the ratio is small, the concept assignment constraints are still enforced. This leads us to suppose that the lower gains on FB15K-237 are perhaps due to sub-optimal representation.

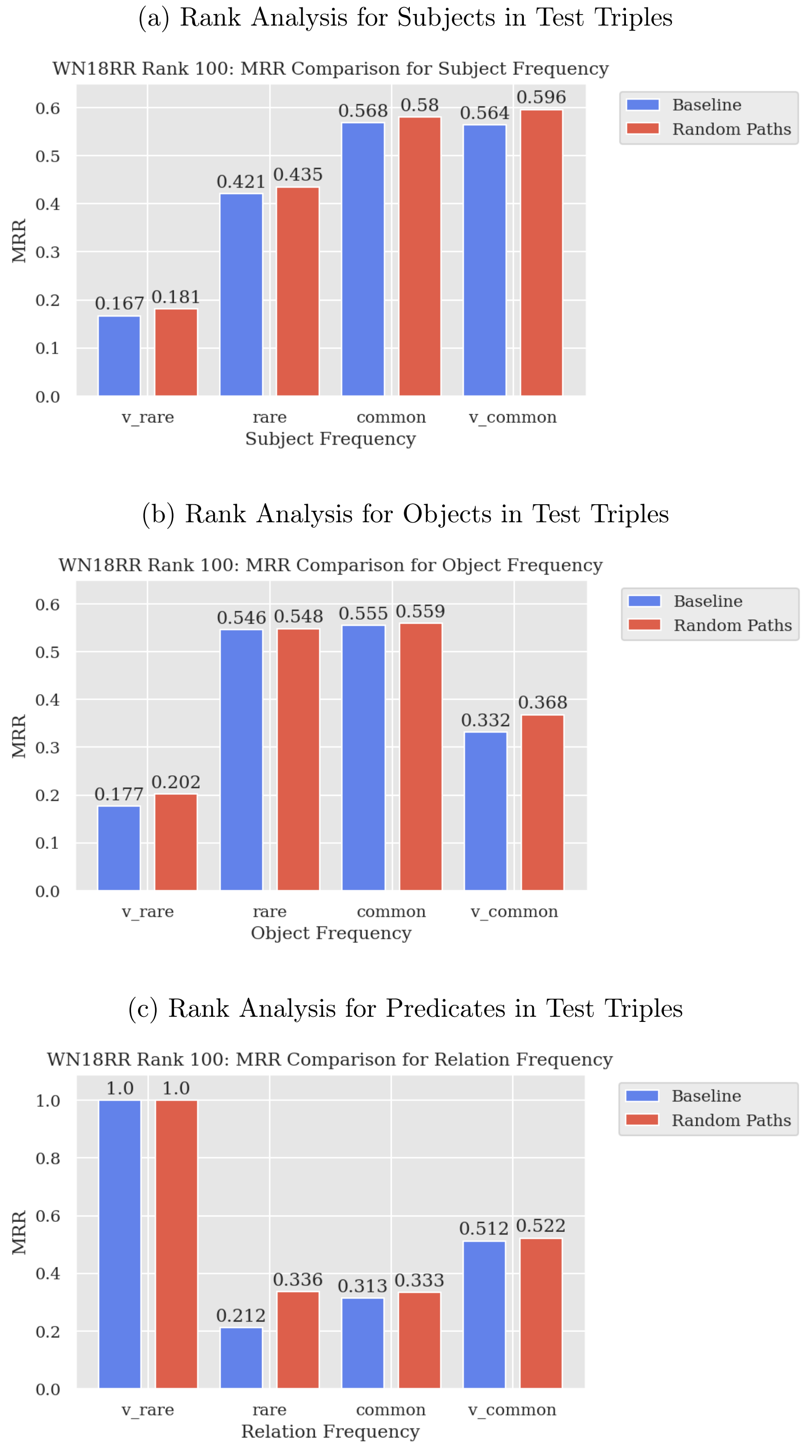
Table 9.5: Frequency bins for WN18RR, where N is the count of train triples the given entity or predicate participate in.

Frequency	**Entity**	**Predicate**
v rare	$N \leq 1$	$N < 100$
rare	$1 < N < 3$	$100 < N < 1000$
common	$3 \leq N < 15$	$1000 < N < 10000$
v common	$N \geq 15$	$N > 10000$

9.7 Rank Analysis

The improvements on test MRR suggest that introducing clusters learnt from random paths representation introduces new information that ComplEx has not been able to capture otherwise. However, looking at a single number, such as mean of reciprocal ranks, gives one a rather shallow insight into model performance. Perhaps a model trained on the augmented dataset learned to predict perfectly on a common predicate at the expense of performing worse on a rare predicate. In this experiment, we want to gain a deeper understanding of the effect that dataset augmentation has on model's predictive power. To this end, we categorise each entity and predicate as either *very rare*, *rare*, *common* and *very common*. To construct the frequency bins shown in Table 9.5 we study the entity and predicate distributions (Section 8), though the divisions are of course to some extent arbitrary and could potentially be improved upon. Next, we compare the test MRR achieved for each sub-population using the baseline model and using ComplEx trained on a dataset with clusters learnt from random paths representation, where we use `max_path`=5, `n_paths`=32, `n_clusters`= 20, Spectral Clustering and baseline hyperparameters for training ComplEx.

Analysing the MRR for different sub-populations (Figure 9-10) reveals some very interesting insights. Firstly, we observe that the augmented dataset model achieves higher performance on every sub-population, except for the very rare predicates for which the baseline also achieves perfect prediction. Some of the biggest increases are seen for triples with rare predicates, where we see an increase of over 0.1, and



(a) Rank Analysis for Subjects in Test Triples

(b) Rank Analysis for Objects in Test Triples

(c) Rank Analysis for Predicates in Test Triples

Figure 9-10: Comparison of MRR for different frequency-based sub-populations in WN18RR, comparing baseline ComplEx and ComplEx trained on clusters learned from random paths representation.

for triples with very common and very rare objects. As WN18RR contains only 11 predicates, we also visualise the MRR for each of the predicates in Figure 9-11. For each relation, we either observe a distinct improvement or no change.

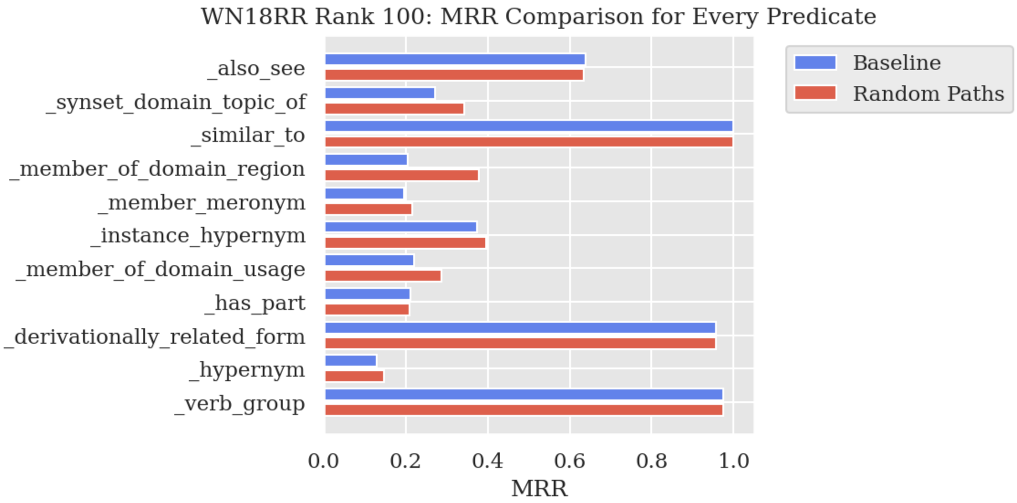


Figure 9-11: MRR comparison for all predicates in WN18RR, comparing baseline ComplEx with ComplEx trained on clusters learned from random paths representation.

We also wish to investigate which triples observe the biggest change in performance, whether as an improvement or as a decline. To this end, for every test triple we compute the difference between the reciprocal rank that the baseline model and augmented model achieved and we rank the triples in a descending order according to absolute difference in reciprocal rank, as shown in Table 9.6. We observe that out of the 30 biggest changes in ranks, 29 of them are changes where the augmented model overwhelmingly improves upon the baseline.

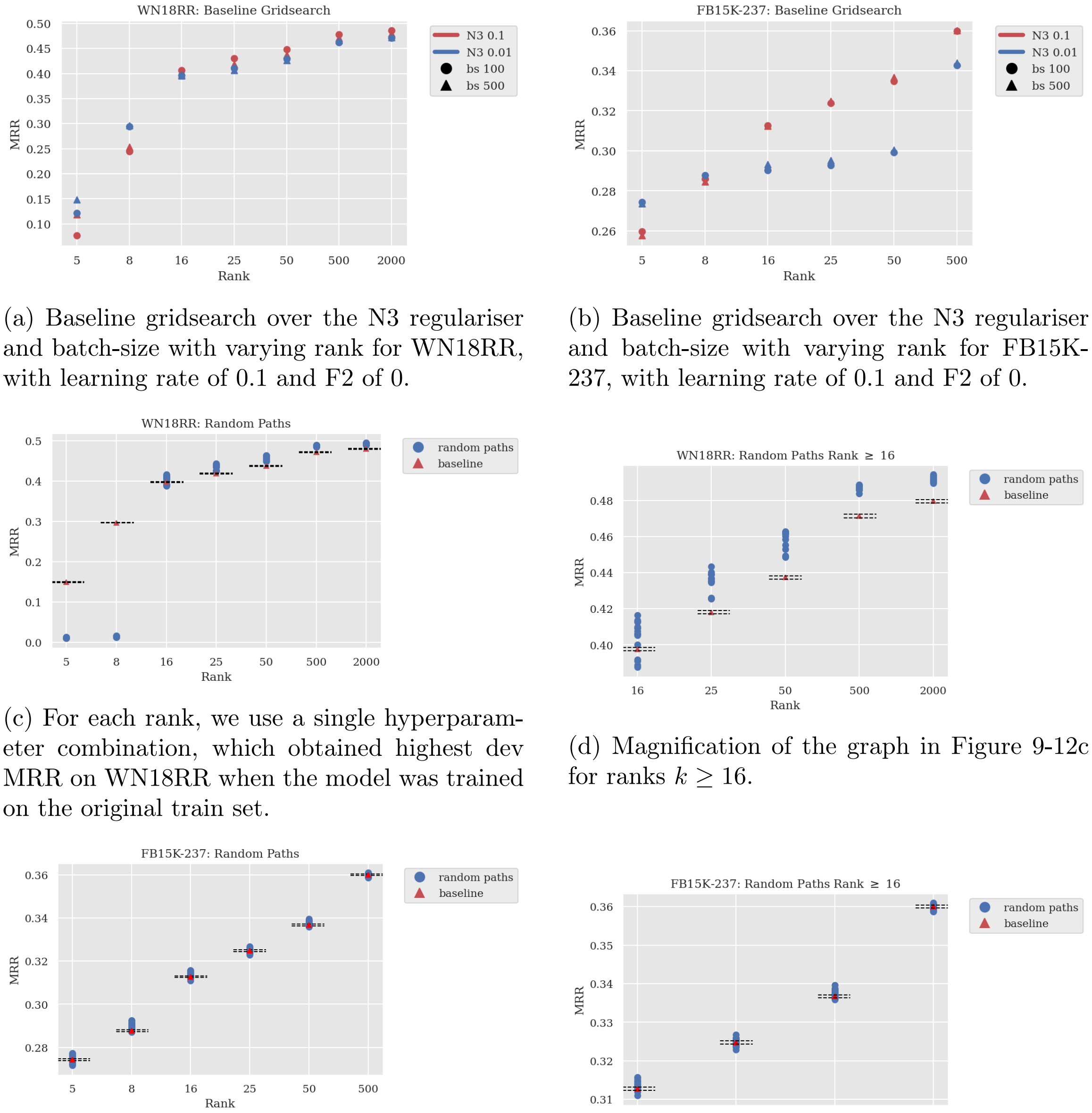
9.8 Varying Rank

While we have seen some significant improvements, thus far our experiments have been restricted to looking at a rank size of 100 for all of the datasets. In this section, we will repeat the baseline and random paths experiments for $k \in \{5, 8, 16, 25, 50, 500, 2000\}$. Due to time constraints and the computational cost associated with training YAGO3-10, we restrict our experiments here to WN18RR and FB15K-237. The gridsearch for every rank here is smaller than that for $k = 100$ — we will only consider changing the $N3$ regulariser and the batch-size, keeping the learning rate constant and maintaining F2 of 0. Although the grid-search is far from extensive, it is worth noting that the results we obtain are either very close or equal to those quoted in [Lacroix et al., 2018]. Moreover, our aim here is primarily to verify whether the trends we have thus far observed for $k = 100$ apply to a wider range of embedding sizes. We hypothesise that even when the hyperparameters are somewhat sub-optimal for both, original and unaugmented datasets, introducing the concept entities will lead to an improvement in performance.

The **baseline** results, as shown in Figures 9-12a and Figure 9-12b, reveal that higher values of N3 work better for $k \geq 16$ for both datasets. Changing the batch-size seems to be of little or no consequence for FB15K-237, while for WN18RR the smaller embedding models perform better with a batch-size of 500 and higher rank models with a batch-size of 100.

To observe the effect of inserting concept clusters, for each rank we use the same hyperparameters which gave best validation MRR on the original train set. For WN18RR we experiment with the following random paths of $(\texttt{max\_path\_len}, \texttt{n\_paths})$: $(10, 32)$, $(5, 64)$, $(30, 32)$, $(20, 64)$ and $\texttt{n\_clusters} \in \{10, 20, 50, 100, 500\}$. We chose the random paths representations to maximise variety between them, while being guided by performance on $k = 100$. Using similar approach, for WN18RR we experiment with random paths $(5, 32)$, $(5, 64)$, $(30, 32)$, $(30, 64)$ and $\texttt{n\_clusters} \in \{20, 50, 100, 500\}$.

In the case of WN18RR, we observe an interesting trend: for ranks $k = 8$ and $k = 16$ introducing clusters learnt from any random paths representation leads to a steep decrease in test MRR, while for larger embedding sizes, $k \geq 16$, there exists at least some representations which lead to a considerable increase in test MRR. Comparing plots 9-12a and 9-12c, we notice a striking similarity: the same ranks for which the model performs better with stronger regularisation also perform better with explicit cluster entities in the training set. This supports our hypothesis that explicitly modelling concept entities can act as a regulariser, potentially by pulling



(a) Baseline gridsearch over the N3 regulariser and batch-size with varying rank for WN18RR, with learning rate of 0.1 and F2 of 0.

(b) Baseline gridsearch over the N3 regulariser and batch-size with varying rank for FB15K-237, with learning rate of 0.1 and F2 of 0.

(c) For each rank, we use a single hyperparameter combination, which obtained highest dev MRR on WN18RR when the model was trained on the original train set.

(d) Magnification of the graph in Figure 9-12c for ranks $k \geq 16$.

(e) For each rank, we use a single hyperparameter combination, which obtained highest dev MRR on FB15K-237 when the model was trained on the original train set.

(f) Magnification of the graph in Figure 9-12e for ranks $k \geq 16$.

Figure 9-12: Exploring the effect of introducing clusters from random paths representation into WN18RR and FB15K-237 for different embedding sizes. Figures a) and b) show the baseline ComplEx gridsearch while remaining figures consider results with concept entities.

similar entities closer in the embedding space.

One plausible explanation as to why introducing clusters leads to a decrease in performance for low-rank models, such a $k = 5$ and $k = 8$, is that the model already has few latent parameters to capture the complexity of the data. Upon augmenting the dataset, it also has to use these parameters to learn the cluster membership predicate, potentially at the cost of sacrificing performance on other predicates.

Magnifying the results for $k \geq 16$ in Figure 9-12d, we observe that the increase in test MRR can be quite significant, especially for $k = 25$ and $k = 50$, where we see an increase of over 0.02. Meanwhile, for FB15K-237 (Figure 9-12e and Figure 9-12f) we observe a similar trend though the gain on the test MRR is smaller than that for WN18RR. This is in line with the earlier observations, once again suggesting that the random paths representations explored here do not capture very well the connectivity patterns in FB15K-237, given the very large number of predicates in this dataset (> 200).

9.9 Default Reasoning with Concepts

Consider that link prediction with KGE models is inherently transductive: all entities, including those in the test set, are seen at train time so that their representations can be learned. As such, predicting in an inductive setting, with entities not seen at train time, is only possible by either generating random representations or re-training the model for n epochs with the new entities, where n is usually around 100.

Default reasoning [Reiter, 1980] is a field of logic concerned with reasoning using default assumptions. In the context of this work, perhaps default reasoning with concept representations could provide an efficient way for performing link prediction in an inductive setting. The representation of the concept the given entity belongs to could be seen as a default representation — capturing the common attributes shared by its participating entities. It is worth noting that such a procedure is much more time and cost efficient — assigning an entity to a concept cluster by generating e.g. a random paths representation could be achieved in a fraction of the time required to train a link prediction model to convergence.

However, verifying whether this approach is feasible first requires answering an important question: are the concept *representations* learnt by ComplEx *meaningful*? To determine this, we carry out a proof-of-concept experiment. Using the best hyperparameters from 9.5.4 we train a ComplEx model for an augmented WN18RR datasets, with `n_clusters` $\in \{1000, 4000, 10000\}$. At the end of the training we swap the representations of all entities for the representations of their respective concepts and predict on the held-out validation and test sets. For comparison, we also generate *random* concept representations and also measure their performance on the test set. The results are shown in Table 9.7, with each experiment being repeated five times.

Three conclusions can be drawn from the experimental results. Firstly, there is a clear increase in predictive power as the number of concept entities is increased for the ComplEx concept representations. This is what we would except, since the clusters will be more fine-grained. Secondly, we observe that there is a clear difference between the performance achieved using random concept representations and that obtained using ComplEx representations, with the later achieving MRR over two degrees of magnitude higher when the number of concept clusters is 10k. Lastly, we note that there is nonetheless a large gap between the performance that can be achieved using bespoke ComplEx representations, for which our method achieves a test MRR > 0.47 and that which we can achieve using even fine-grained clusters. However, in this experiment we have swapped *all* of the entity representations for their concept representations. An interesting future experiment could involve predicting on triples where only one of the entity representations has been swapped for its default embedding.

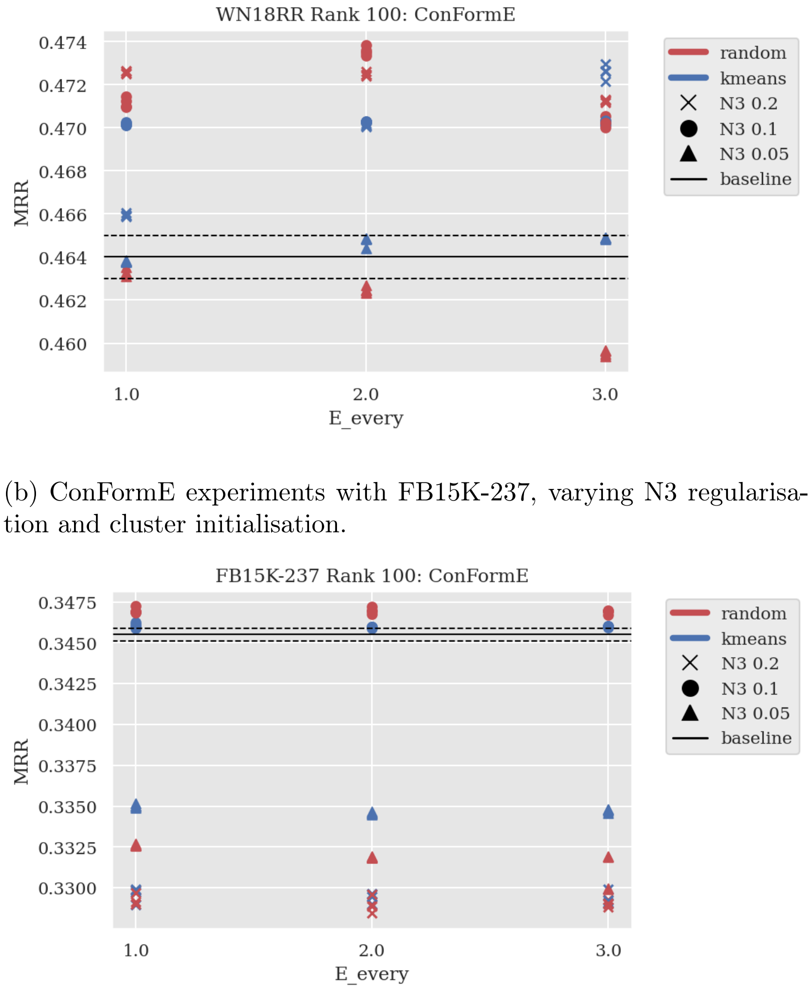
While here we have only experimented with using the ComplEx concept representation, there are other ways in which concepts could potentially be utilised for the default reasoning task. For instance, to make a prediction for a triple with an unseen entity one could substitute for its representation the embeddings of other entities in its cluster. Next, one could sum the scores each of the co-clustered entities awards to each of the candidate completions and choose the candidate entity with the highest total score. Another, similar alternative is to represent the unseen entity using the averaged embeddings of the entities in the given cluster.

9.10 Summary of Similarity-based Learning

In Table 9.8 we summarise the best performance achieved for rank 100 using each of the concept formation approaches. Firstly, we observe that the best results achieved for **every** augmentation method outperform the baseline, with varying margin. We also note that the approach which delivers best results depends largely on the dataset. This is not surprising, given how different each dataset is. The biggest overall increase is seen for WN18RR. One possible explanation for this could be based around the sparsity of data for WN18RR entities, with each entity participating on average in just 4 train triples. As such, potentially explicit modelling of the latent concepts learned by travelling along random walks in the graph enables better generalisation. In the case of FB15K-237 we note that the best results were obtained using the 2P representation, with random paths also performing competitively. In the case of YAGO3-10 ComplEx embeddings performed marginally better than Random Paths. However, it is worth remembering that the selection of path representations used for YAGO3-10 was much smaller than for the other datasets.

9.11 Jointly Learning Concepts and Model Parameters

In this section we present some preliminary experiments with ConFormE (Section 7). We begin by setting up an experiment with WN18RR and FB15K-237, introducing 50 concept entities and using the hyperparameters which consistently gave best results thus far. The cluster assignments are initialised either randomly or using K-Means clustering of random paths representation. The augmentation step is performed every t ComplEx epochs, where $t \in \{1, 2, 3\}$. As preliminary experiments with UMLS suggested that the model might be overfitting to the train data with baseline regularisation, we also consider N3 $\in \{0.05, 0.1, 0.2\}$. We then run each algorithm and print out at every epoch 1) the number of assignment changes and 2) number of existing concepts.



(a) ConFormE experiments with WN18RR, varying N3 regularisation and cluster initialisation.

(b) ConFormE experiments with FB15K-237, varying N3 regularisation and cluster initialisation.

Figure 9-13: Preliminary experiments with ConFormE for WN18RR and FB15K-237, where *E_ every* indicates the number of ComplEx training epochs between each E-step and two different cluster initialisation are explored: random clusters and clusters learnt from a random paths representation using K-Means. Every experiment is initialised with 50 concepts and repeated five times, as shown.

A number of interesting observations surfaced while monitoring the train-time performance. Firstly, we observe that the number of concept assignment changes in the first few epochs is significantly higher for randomly initialised concepts in comparison with those initialised using random paths representation. Beyond around 5-10 epochs the number of assignment changes for each of the initialisation decreases quite rapidly, with only 5 - 40 entities changing membership and in most cases no entities change membership beyond 30 epochs. We also note that the decrease in the number of entities that change assignment is strictly monotonic over time.

The initial number of clusters drops during training, usually to around 20-25 concepts, depending on the regularisation and number M-steps between every E-step. Hence, ConFormE can be seen as a generalisation of the method proposed in Section 6, where the number of clusters is fixed and the cluster assignments are determined using a purely similarity-based metric, with no consideration of link prediction performance. Meanwhile, ConFormE allows for the number of clusters to decrease during training based on the link prediction performance.

The results on link prediction (Figure 9-13) show a lot of promise: we have been able to demonstrate that learning with ConFormE is able to outperform learning with vanilla ComplEx on both datasets when ConFormE is initialised with 50 random clusters and N3 of 0.1. We note that the number of M-steps between every E-step has little effect on the link prediction performance, except for WN18RR where the highest result was achieved when performing the E-step every two epochs. However, the difference is quite small and the experiment should be repeated to verify this finding. Across the different regularisation strengths there seems to be no clear trend as to which is more favourable: a random or a non-random initialisation. However, in conjunction of with N3 of 0.1 (which has performed best on all experiments in this work for WN18RR and FB15K-237), a random initialisation outperforms a non-random one, which is somewhat surprising. One possible explanation for this phenomenon may emerge by drawing a parallel to clustering with classical K-Means: the algorithm is only guaranteed to find local minima — an issue that is usually addressed by using random initialisations, often with multiple restarts.

Table 9.6: Comparison of reciprocal ranks achieved by the baseline model and augmented model for WN18RR, ordered by descending absolute difference in reciprocal rank. Columns *s*, *r* and *o* indicate how common are the subject, relation and object in the given triple. The *L/R* column indicates whether we are corrupting the LHS i.e. subject (s) or the RHS i.e. object (o). *Base rank* and *model rank* display the reciprocal rank achieved by the baseline model and the ComplEx model trained on the dataset augmented with clusters learnt from random paths representation. The *rank delta* column diplays the absolute difference in rank between the two models, while the *predicate* column displays the predicate involved. Only top 30 results are shown.

s	**r**	**o**	**L/R**	**base rank**	**model rank**	**rank delta**	**predicate**
v_common	rare	common	s	0.000086	1.000000	0.999914	_member_of_domain_region
common	common	v_common	o	0.000212	1.000000	0.999788	_synset_domain_topic_of
v_rare	v_common	v_common	o	0.001639	1.000000	0.998361	_hypernym
v_common	rare	v_rare	s	0.002278	1.000000	0.997722	_member_of_domain_usage
v_common	rare	v_rare	s	0.003436	1.000000	0.996564	_member_of_domain_region
common	common	v_common	o	0.004000	1.000000	0.996000	_synset_domain_topic_of
v_common	rare	v_rare	s	0.004405	1.000000	0.995595	_member_of_domain_region
v_rare	common	v_common	o	0.004831	1.000000	0.995169	_synset_domain_topic_of
v_common	rare	v_rare	s	0.008547	1.000000	0.991453	_member_of_domain_usage
rare	common	v_common	o	0.009709	1.000000	0.990291	_synset_domain_topic_of
common	common	v_common	o	0.009804	1.000000	0.990196	_instance_hypernym
rare	v_common	v_common	o	0.012987	1.000000	0.987013	_hypernym
common	common	v_common	s	1.000000	0.018182	-0.981818	_synset_domain_topic_of
v_rare	common	v_common	o	0.022222	1.000000	0.977778	_instance_hypernym
rare	common	v_common	o	0.031250	1.000000	0.968750	_synset_domain_topic_of
v_common	rare	v_rare	s	0.034483	1.000000	0.965517	_member_of_domain_region
v_common	rare	v_rare	s	0.040000	1.000000	0.960000	_member_of_domain_region
v_common	rare	v_rare	s	0.047619	1.000000	0.952381	_member_of_domain_region
common	v_common	common	o	0.052632	1.000000	0.947368	_hypernym
v_common	rare	common	s	0.052632	1.000000	0.947368	_member_of_domain_region
common	common	v_common	o	0.066667	1.000000	0.933333	_synset_domain_topic_of
common	v_common	common	o	0.071429	1.000000	0.928571	_hypernym
v_common	rare	v_rare	s	0.083333	1.000000	0.916667	_member_of_domain_usage
common	common	common	o	0.083333	1.000000	0.916667	_instance_hypernym
v_common	common	rare	s	1.000000	0.090909	-0.909091	_member_meronym
common	v_common	v_common	o	0.090909	1.000000	0.909091	_hypernym
v_rare	v_common	v_common	o	0.090909	1.000000	0.909091	_hypernym
v_common	rare	v_rare	s	0.090909	1.000000	0.909091	_member_of_domain_region

Table 9.7: Performance on a default reasoning task for WN18RR test set, comparing random and ComplEx concept representations, for k=100.

# c	Rep.	MRR	H@1	H@3	H@5	H@10	H@50	H@100
1e3	Random	0.0003±0.0001	3e-05±7e-05	0.0001±9e-05	0.0002±0.0002	0.0002±0.0002	0.0012±0.0006	0.0023±0.0008
	ComplEx	**0.005±0.0004**	**0.002±0.0004**	**0.004±0.0006**	**0.006±0.0004**	**0.011±0.0019**	**0.028±0.0004**	**0.038±0.0003**
4e3	Random	0.0002±8e-05	0.0±0.0	0.0±7e-05	0.0001±9e-05	0.0002±0.0002	0.0011±0.0007	0.002±0.001
	ComplEx	**0.026±0.0003**	**0.018±0.0004**	**0.028±0.0002**	**0.034±0.0004**	**0.041±0.0002**	**0.06±0.0001**	**0.071±0.0001**
1e4	Random	0.0002±4e-05	0.0±0.0	0.0001±9e-05	0.0001±9e-05	0.0002±0.0001	0.001±0.0005	0.0026±0.0009
	ComplEx	**0.085±0.0003**	**0.072±0.0003**	**0.091±0.0005**	**0.099±0.0003**	**0.109±0.0001**	**0.126±0.0001**	**0.136±0.0001**

Table 9.8: Summary of results obtained for ComplEx models of rank 100.

	Representation	**MRR**	**H@1**	**H@3**	**H@10**
WN18RR	Baseline	0.462	0.428	0.475	0.528
	ComplEx	0.467	0.430	0.480	0.539
	Random Clusters	0.471	0.433	0.485	0.549
	2P	0.469	0.434	0.484	0.540
	Random Paths	**0.476**	**0.439**	**0.489**	**0.552**
FB15K-237	Baseline	0.3455	0.2535	0.379	0.5322
	ComplEx	0.3470	0.2547	0.3814	0.5316
	Random Clusters	0.3465	0.2543	0.3802	0.5324
	WL Kernel	0.3478	0.2552	0.3837	0.5341
	2P	**0.3492**	**0.2565**	0.3834	**0.5373**
	Random Paths	0.3486	0.2547	**0.3849**	0.5353
YAGO3-10	Baseline	0.559	0.489	0.601	0.685
	Random Clusters	0.560	0.487	0.606	0.690
	ComplEx	**0.567**	**0.499**	**0.608**	0.688
	2P	0.565	0.493	0.607	**0.693**
	Random Paths	0.566	0.495	0.610	0.692

Chapter 10

Conclusions

10.1 Summary

Research suggests that concept formation constitutes an essential aspect of human cognition [Love et al., 2004, Lake et al., 2016, Zeithamova et al., 2019, Bowman and Zeithamova, 2018]. In this work we have presented a case for the task of concept learning in KGs. Drawing inspiration from plausible models of unsupervised concept learning in humans, we propose two models for learning concepts in KGs. The **Similarity-based Concept Learning** framework (Section 6) uses propositionalisation to learn entity vector representations, makes concept assignments by clustering the said representations and introduces the new concept triples into the original KG. A KGE model is then trained for the link prediction task on the augmented dataset. The resulting approach **scales easily** even to large KGs, allows for concepts to be learned in a **fully unsupervised** manner, and is **model-agnostic** thus can be easily integrated with any existing KGE model. The second approach, **Concept Formation with EM** (ConFormE) can be seen as a generalisation of the initial framework by allowing for the concepts and model parameters to be learned *jointly*, using a probabilistic EM-like algorithm. As such, it allows for the concept formation task to be driven by optimising performance on link prediction tasks, allowing us to learn the optimal number of clusters and concept assignments. While both, the approach itself and the experiments, are largely preliminary, the initial results show a lot of promise.

In fact, the similarity-based concept learning can be thought of as a special case of ConFormE, where we only perform a series of M-steps while holding the concept assignments constant. Hence, ConFormE can perform **at least** as well as the results achieved in Sections 9.1-9.10 for the static similarity-based approach.

Across all of the experiments, a key finding is that explicitly augmenting KGs by introducing concept entities can indeed lead to an improvement on link prediction tasks. This finding is supported by our approach outperforming a vanilla ComplEx model on all 3 large benchmark datasets, with the biggest improvement being seen on W18RR. We note that in WN18RR each entity is connected to, on average, only four other entities. As such, the representation learned for each entity is derived from sparse data. One interpretation of the results suggests that explicitly assigning entities to concept clusters can potentially improve generalisation for rare entities. More generally, we hypothesise that a concept entity can act as an *attractor*, pulling the participating entity representations closer together in the embedding space. The significant gains seen upon introducing concepts learned from random-paths representation (for WN8RR and FB15K-237) and 2P representation (for YAGO3-10) suggest that this framework can be seen as a way of implicitly introducing rule-based information derived from observable graph patterns into latent feature models.

Aside from allowing for better performance on the standard link prediction task we also show that the concept representation learned in this way can be used in an inductive reasoning setting via default reasoning — representing the unseen entities using the embeddings of their respective concept clusters.

Moreover, we show that the clusters learned, especially when creating a more fine-grained clustering, are easily semantically-interpretable. As such, concept learning could be used as a way to qualitatively analyse the embedding representation. This application could be of particular interesting in domains like the biomedical field, where explainability is of paramount importance.

Lastly, in our analysis we demonstrate the value of looking beyond a single MRR metric and considering performance on different entity and predicate sub-populations.

10.2 Future Work

Firstly, due to time constraints we have only been able to repeat some of the experiments. Ideally, we would have liked to have repeated all of them and extended them to include different embedding sizes. Nonetheless, looking at results shown in e.g. Figure 9-13 and Figure 9-12, we note that almost the same trends would be observed even if one were to sample randomly any one out of the observations for each experimental configuration. The rank analysis has only been performed for WN18RR but we would like to extend it to FB15K-237 and YAGO3-10. Similarly, for some of the experiments only WN18RR and FB15K-237 were considered — it would be instructive to also perform them on YAGO3-10 and compare whether the same trends are observed. Preliminary experiments on UMLS [McCray, 2003] show that it possible to learn concepts for medical data — we would like to verify this finding by applying framework to larger medical datasets, such as OpenBioLink [Breit et al., 2020] or Hetionet [Himmelstein et al., 2017] and observe the effect on link prediction performance.

Thus far we have verified that the concepts learned are semantically-meaningful in a *qualitative* way, but ideally we would also like to prove this in a *quantitative* way. One experiment which could be performed to test this is using a dataset with a pre-existing conceptual structure, such as SNOMED CT [1]. Then, one could simply compute the point-wise mutual information between the pre-existing and learned concept assignments.

All of the experiments performed here were implemented using ComplEx [Trouillon et al., 2016]. However, as the method we propose is based on data augmentation it could be easily incorporated into any existing KGE model. Given more time, it would be instructive to test this hypothesis by repeating the experiments using another state-of-the-art KGE model. As ComplEx is a tensor factorisation model, ideally a model from a different family should be used, such as RotatE [Sun et al., 2019] which is a state-of-the-art translational model.

[1] https://www.nlm.nih.gov/healthit/snomedct/index.html

The current framework allows for each entity to be assigned to a single cluster - this can be somewhat restrictive. Instead, it would be interesting to consider extending the framework to allow for multiple, cross-cutting clusters as in the approach proposed by [Kok and Domingos, 2007].

Moreover, as noted in Section 1.1.1, humans tend to learn a concept *hierarchy*, such as `mammals` → `animals` → `living beings`. The concept learning framework presented here can be thought of as learning just the first layer of the hierarchy. However, in principle it could easily be extended to learning higher levels by e.g. implementing a hierarchical clustering algorithm, as in [Nickel et al., 2011]. Learning more hierarchy levels is interesting for a number of reasons. Firstly, we would like to see whether the higher-order concepts will be semantically meaningful. Moreover, we noticed consistently that better results on link prediction are observed when learning a small number of concepts, as opposed to learning a large number of more fine-grained clusters. This observation was true of both, meaningful and random clusters. We believe that this is because assigning entities to few clusters pushes them closer together in embedding space, whereas learning more fine-grained clusters, even if they are meaningful, might be pushing them too far away. If that is in fact the case, we hypothesise that inducing a higher-order hierarchy over the first-layer concepts will prevent them from spreading too far apart in the embedding space.

Throughout this work we represent entities using embeddings in the Euclidean space. Nonetheless, as observed by [Nickel and Kiela, 2017], Euclidean space is inherently ill-suited for modelling hierarchies. It could be of interest to i) learn the hierarchies using a hierarchical clustering algorithm, as explained above, and ii) represent the entities in non-Euclidean spaces, such as the hyperbolic [Nickel and Kiela, 2017] or polar [Zhang et al., 2019] coordinate systems.

While the results on link prediction using ConFormE show a lot of promise — we have been able to demonstrate that learning with ConFormE is able to outperform learning with vanilla ComplEx — the work presented here is still largely foundational. There remains large scope for i) further developing the ConFormE framework, especially the E-step and ii) exploring different parameter settings. For example, one could in principle learn an *optimal* number of clusters by using an initialisation where the number of clusters is equal to the number of entities i.e. $N_c = N_e$. Moreover, while the assignments made in the E-step are currently entirely deterministic, it could be interesting to introduce a small degree of stochasticity, especially when taking into account the parallels drawn earlier with K-Means. This could be achieved using a meta-stochastic sampling regime, akin to that of the epsilon-greedy policy in reinforcement learning [Sutton and Barto, 2018]. With a certain probability $p = \epsilon$ the assignment is sampled, and with probability $1 - \epsilon$ it is deterministic, where ϵ can be decayed over time. Moreover, while the regularisation weights used here are determined by using a held-our validation set, perhaps they could be further optimised using a meta-learning, framing it as a bi-level optimisation approach.

Appendix

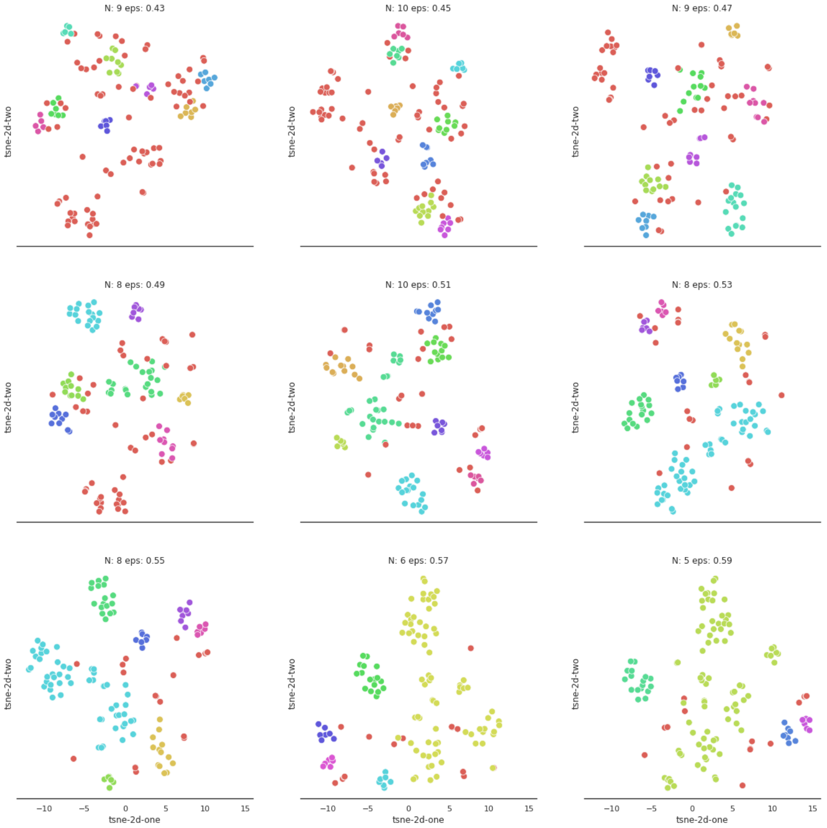


Figure 10-1: t-SNE visualisation of UMLS embeddings clustered using DBSCAN for a range of epsilon values, illustrating the sensitivity of DBSCAN to small changes in ϵ values. The entities coloured in red have been labelled by DBSCAN as outliers. We observe that the cluster assignments change quite drastically, with most entities being labelled as outliers for $\epsilon = 0.43$, while most entities for $\epsilon = 0.59$ have been assigned to a single huge cluster. Visually, ideal clusters seemed to be formed for $\epsilon = 0.51$.

Bibliography

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40:57–84, 03 2014. doi: 10.1162/COLI_a_00164.

Gabor Angeli and Christopher Manning. Philosophers are mortal: Inferring the truth of unseen facts. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 133–142, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W13-3515`.

Stephan Baier, Yunpu Ma, and Volker Tresp. Improving visual relationship detection using semantic modeling of scene descriptions. *CoRR*, abs/1809.00204, 2018. URL `http://arxiv.org/abs/1809.00204`.

François Belleau, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. Bio2rdf: Towards a mashup to build bioinformatics knowledge system. *Journal of biomedical informatics*, 41:706–16, 04 2008. doi: 10.1016/j.jbi.2008.03.004.

Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.463. URL `https://www.aclweb.org/anthology/2020.acl-main.463`.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1160`.

Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge graph embeddings and explainable AI. In Ilaria Tiddi, Freddy Lécué, and Pascal Hitzler, editors, *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, volume 47 of *Studies on the Semantic Web*, pages 49–72. IOS Press, 2020. doi: 10.3233/SSW200011. URL `https://doi.org/10.3233/SSW200011`.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/p10008. URL http://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

Caitlin Bowman and Dagmar Zeithamova. Abstract memory representations in the ventromedial prefrontal cortex and hippocampus support concept generalization. *The Journal of Neuroscience*, 38:2811–17, 02 2018. doi: 10.1523/JNEUROSCI.2811-17.2018.

Caitlin Bowman and Dagmar Zeithamova. Training typicality and set size effects on concept generalization and recognition. 05 2019. doi: 10.31234/osf.io/7g2q5.

Anna Breit, Simon Ott, Asan Agibetov, and Matthias Samwald. OpenBioLink: a benchmarking framework for large-scale biomedical link prediction. *Bioinformatics*, 36(13):4097–4098, 04 2020. ISSN 1367-4803. doi: 10.1093/bioinformatics/btaa274. URL https://doi.org/10.1093/bioinformatics/btaa274.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1306–1313. AAAI Press, 2010.

Nick Chater. The search for simplicity: A fundamental cognitive principle? *The Quarterly Journal of Experimental Psychology Section A*, 52(2):273–302, 1999. doi: 10.1080/713755819. URL https://doi.org/10.1080/713755819.

Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 76–83, Vancouver, British Columbia, Canada, June 1989. Association for Computational Linguistics. doi: 10.3115/981623.981633. URL https://www.aclweb.org/anthology/P89-1010.

Tri Dao, Albert Gu, Alexander Ratner, Virginia Smith, Christopher De Sa, and Christopher Ré. A kernel theory of modern data augmentation. *Proceedings of machine learning research*, 97, 03 2018.

Rajarshi Das, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Non-parametric reasoning in knowledge bases. In *Automated Knowledge Base Construction*, 2020. URL `https://openreview.net/forum?id=AEY9tRqlU7`.

Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17, Mar. 1993. doi: 10.1609/aimag.v14i1.1029. URL `https://www.aaai.org/ojs/index.php/aimagazine/article/view/1029`.

Gerben K. D. de Vries. A fast approximation of the weisfeiler-lehman graph kernel for rdf data. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 606–621, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: 10.1111/j.2517-6161.1977.tb01600.x. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x`.

Woodrow Denham. *The Detection of Patterns in Alyawara Nonverbal Behaviour*. PhD thesis, 05 1973.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL `https://doi.org/10.1007/BF01386390`.

Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 601–610, 2014. URL `http://www.cs.cmu.edu/~nlao/publication/2014.kdd.pdf`. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz.

Sourav Dutta and Gerhard Weikum. Cross-document co-reference resolution using sample-based clustering with knowledge enrichment. *Transactions of the Association for Computational Linguistics*, 3:15–28, 2015. doi: 10.1162/tacl_a_00119. URL `https://www.aclweb.org/anthology/Q15-1002`.

Jeffrey Scott Eder. Knowledge graph based search system, 2012.

Lance Eliot and Michael Eliot. *Autonomous Vehicle Driverless Self-Driving Cars and Artificial Intelligence: Practical Advances in AI and Machine Learning*. LBE Press Publishing, 1st edition, 2017. ISBN 0692051023.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open question answering over curated and extracted knowledge bases. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2014. doi: 10.1145/2623330.2623677.

D.A. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56:1:1–1:15, 05 2012. doi: 10.1147/JRD.2012.2184356.

Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007. ISSN 0036-8075. doi: 10.1126/science.1136800. URL `https://science.sciencemag.org/content/315/5814/972`.

K. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127–138, 2010.

Alberto Garcia-Duran and Mathias Niepert. Kblrn : End-to-end learning of knowledge base representations with latent, relational, and numerical features, 2017.

Lise Getoor and Ben Taskar. Introduction to statistical relational learning. 01 2007.

Noah Goodman, Joshua Tenenbaum, Jacob Feldman, and Thomas Griffiths. A rational analysis of rule-based concept learning. *Cognitive science*, 32:108–54, 01 2008. doi: 10.1080/03640210701802071.

Noah D. Goodman, J. Tenenbaum, and Tobias Gerstenberg. Concepts in a probabilistic language of thought. 2014.

David Haussler. Convolution kernels on discrete structures, 1999.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

Daniel Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6:e26726, 09 2017. doi: 10.7554/elife.26726.

Frank L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927. doi: 10.1002/sapm192761164. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164`.

Charles Kemp, Joshua Tenenbaum, Thomas Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. *Cognitive Science*, 21, 01 2006.

Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 433–440, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273551. URL `https://doi.org/10.1145/1273496.1273551`.

Stefan Kramer, Nada Lavrac, and Peter Flach. Propositionalization approaches to relational data mining. 01 2001. doi: 10.1007/978-3-662-04599-2_11.

Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. *CoRR*, abs/1508.02593, 2015. URL `http://arxiv.org/abs/1508.02593`.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion, 2018.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350 (6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL `https://science.sciencemag.org/content/350/6266/1332`.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people, 2016.

George Lakoff and Mark Johnson. The metaphorical structure of the human conceptual system. *Cognitive Science*, 4(2):195 – 208, 1980. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(80)80017-6. URL `http://www.sciencedirect.com/science/article/pii/S0364021380800176`.

Niels Landwehr, Andrea Passerini, Luc De Raedt, and Paolo Frasconi. Kfoil: Learning simple relational kernels. volume 1, 01 2006.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6, 01 2014. doi: 10.3233/SW-140134.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2181–2187. AAAI Press, 2015. ISBN 0262511290.

Bradley Love. Comparing supervised and unsupervised category learning. *Psychonomic bulletin & review*, 9:829–35, 01 2003. doi: 10.3758/BF03196342.

Bradley Love, Doug Medin, and Todd Gureckis. Sustain: a network model of category learning. *Psychological review*, 111:309–32, 05 2004. doi: 10.1037/0033-295X.111.2.309.

Anderson LW, Krathwohl DR, Airasian PW, Cruikshank KA, Richard Mayer, Pintrich PR, J. Raths, and Wittrock MC. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. 01 2001. ISBN ISBN: 080131903X.

Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, Mar 2011. ISSN 0378-4371. doi: 10.1016/j.physa.2010.11.027. URL `http://dx.doi.org/10.1016/j.physa.2010.11.027`.

Michael Mack, Alison Preston, and Bradley Love. Decoding the brain's algorithm for categorization from its neural implementation. *Current biology : CB*, 23, 10 2013. doi: 10.1016/j.cub.2013.08.035.

Michael L. Mack, Bradley C. Love, and Alison R. Preston. Dynamic updating of hippocampal object representations reflects new conceptual knowledge. *Proceedings of the National Academy of Sciences*, 113(46):13203–13208, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1614048113. URL `https://www.pnas.org/content/113/46/13203`.

Michael L. Mack, Alison R. Preston, and Bradley C. Love. Ventromedial prefrontal cortex compression during concept learning. *bioRxiv*, 2019. doi: 10.1101/178145. URL `https://www.biorxiv.org/content/early/2019/06/10/178145`.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press. URL `https://projecteuclid.org/euclid.bsmsp/1200512992`.

F. Mahdisoltani, J. Biega, and Fabian M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2015.

Alexa McCray. An upper-level ontology for the biomedical domain. *Comparative and functional genomics*, 4:80–4, 01 2003. doi: 10.1002/cfg.255.

Scott McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg Corrado, Ara Darzi, Mozziyar Etemadi, Florencia Garcia-Vicente, Fiona Gilbert, Mark Halling-Brown, Demis Hassabis, Sunny Jansen, Alan Karthikesalingam, Christopher Kelly, Dominic King, and Shravya Shetty. International evaluation of an ai system for breast cancer screening. *Nature*, 577:89–94, 01 2020. doi: 10.1038/s41586-019-1799-6.

George Miller, R. Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to wordnet: An on-line lexical database*. 3, 01 1991. doi: 10.1093/ijl/3.4.235.

Robert M. Mok and Bradley C. Love. A non-spatial account of place and grid cells based on clustering models of concept learning. *bioRxiv*, 2019. doi: 10.1101/421842. URL https://www.biorxiv.org/content/early/2019/05/22/421842.

Vassil Momtchev, Deyan Peychev, Todor Primov, and Georgi Georgiev. Expanding the pathway and interaction knowledge in linked life data. *Semantic Web Challenge: 2009; Amsterdam*, 01 2009.

Hans Moravec. *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press, 1990. ISBN 0674576187.

Igor Mordatch. Concept learning with energy-based models. *CoRR*, abs/1811.02486, 2018. URL http://arxiv.org/abs/1811.02486.

Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629 – 679, 1994. ISSN 0743-1066. doi: https://doi.org/10.1016/0743-1066(94)90035-3. URL http://www.sciencedirect.com/science/article/pii/0743106694900353. Special Issue: Ten Years of Logic Programming.

R. Navigli and P. Velardi. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1075–1086, 2005.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 849–856. MIT Press, 2001.

Maximilian Nickel. Learning taxonomies from multi-relational data via hierarchical link-based clustering. 2011.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104 (1):11–33, Jan 2016. ISSN 1558-2256. doi: 10.1109/jproc.2015.2483592. URL http://dx.doi.org/10.1109/JPROC.2015.2483592.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7213-poincare-embeddings-for-learning-hierarchical-representations.pdf.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43, July 2019. ISSN 0001-0782. doi: 10.1145/3331166. URL https://doi.org/10.1145/3331166.

Peter Orbanz and Daniel M. Roy. Bayesian models of graphs, arrays and other exchangeable random structures, 2013.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014. doi: 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732.

Plato. *The Allegory of the Cave*. P & L Publication, 2010. ISBN 9781452800882. URL https://books.google.co.uk/books?id=e7nRRwAACAAJ.

John Platt. Probabilities for sv machines. pages 61–74, 01 1999.

Emmanuel M. Pothos and Nick Chater. A simplicity principle in unsupervised human categorization. *Cognitive Science*, 26(3):303 – 343, 2002. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(02)00064-2. URL http://www.sciencedirect.com/science/article/pii/S0364021302000642.

Luc De Raedt and Kristian Kersting. *Statistical Relational Learning*, pages 916–924. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_786. URL https://doi.org/10.1007/978-0-387-30164-8_786.

Nadia Rahmah and Imas Sukaesih Sitanggang. Determination of optimal epsilon (eps) value on DBSCAN algorithm to clustering data on peatland hotspots in sumatra. *IOP Conference Series: Earth and Environmental Science*, 31:012012, jan 2016. doi: 10.1088/1755-1315/31/1/012012. URL https://doi.org/10.1088%2F1755-1315%2F31%2F1%2F012012.

R. Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1–2):81–132, April 1980. ISSN 0004-3702. doi: 10.1016/0004-3702(80)90014-4. URL https://doi.org/10.1016/0004-3702(80)90014-4.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62 (1–2):107–136, February 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-5833-1. URL https://doi.org/10.1007/s10994-006-5833-1.

Brett D. Roads and Bradley C. Love. Learning as the unsupervised alignment of conceptual systems. *Nature Machine Intelligence*, 2:76–82, 2020.

E. Rosch, C. Mervis, Wayne D. Gray, D. M. Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive Psychology*, 8:382–439, 1976.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkxSmlBFvr.

R. Rummel. Dimensionality of nations project: attributes of nations and behavior of nation dyads. 1999.

Alan Ruttenberg, Jonathan Rees, Matthias Samwald, and M Marshall. Life sciences on the semantic web: The neurocommons and beyond. *Briefings in bioinformatics*, 10:193–204, 04 2009. doi: 10.1093/bib/bbp004.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011. URL http://jmlr.org/papers/v12/shervashidze11a.html.

D. Silver, Aja Huang, Chris J. Maddison, A. Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, S. Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

Ryan Smith, Philipp Schwartenbeck, Thomas Parr, and Karl J. Friston. An active inference approach to modeling structure learning: Concept learning as an example case. *Frontiers in Computational Neuroscience*, 14:41, 2020. ISSN 1662-5188. doi: 10.3389/fncom.2020.00041. URL https://www.frontiersin.org/article/10.3389/fncom.2020.00041.

Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. pages 697–706, 01 2007. doi: 10.1145/1242572.1242667.

Daniel J. Sullivan. *An Introduction to Philosophy.* Milwaukee, Bruce Pub. Co., 1957.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. 07 2015. doi: 10.18653/v1/W15-4007.

Théo Trouillon, Éric Gaussier, Christopher R. Dance, and Guillaume Bouchard. On inductive abilities of latent factor models for relational learning. *CoRR*, abs/1709.05666, 2017. URL `http://arxiv.org/abs/1709.05666`.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction, 2016.

Alex Waibel and Kai-Fu Lee, editors. *Readings in Speech Recognition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1558601244.

Qingyun Wang, Manling Li, Xuan Wang, Nikolaus Parulian, Guangxing Han, Jiawei Ma, Jingxuan Tu, Ying Lin, Haoran Zhang, Weili Liu, Aabhas Chauhan, Yingjun Guan, Bangzheng Li, Ruisong Li, Xiangchen Song, Heng Ji, Jiawei Han, Shih-Fu Chang, James Pustejovsky, Jasmine Rah, David Liem, Ahmed Elsayed, Martha Palmer, Clare Voss, Cynthia Schneider, and Boyan Onyshkevych. Covid-19 literature knowledge graph construction and drug repurposing report generation, 2020a.

Z. Wang, L. Li, Q. Li, and D. Zeng. Multimodal data enhanced representation learning for knowledge graphs. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zhigang Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

Ziyu Wang, Nanqing Luo, and Pan Zhou. Guardhealth: Blockchain empowered secure data management and graph convolutional network enabled anomaly detection in smart healthcare. *J. Parallel Distributed Comput.*, 142:1–12, 2020b.

William Wattenmaker. The influence of prior knowledge in intentional versus incidental concept learning. *Memory & cognition*, 27:685–98, 08 1999. doi: 10.3758/BF03211562.

Gerhard Weikum and Martin Theobald. From information to knowledge: Harvesting entities and relationships from web sources. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, page 65–76, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300339. doi: 10.1145/1807085.1807097. URL `https://doi.org/10.1145/1807085.1807097`.

Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1301–1316, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450347037. doi: 10.1145/3183713.3183729. URL `https://doi.org/10.1145/3183713.3183729`.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2659–2665. AAAI Press, 2016.

Fei Xu and Joshua Tenenbaum. Word learning as bayesian inference: Evidence from preschoolers. 01 2005.

Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel. Infinite hidden relational models. *CoRR*, abs/1206.6864, 2012. URL `http://arxiv.org/abs/1206.6864`.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases, 2014.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4805–4815, Red Hook, NY, USA, 2018. Curran Associates Inc.

Dagmar Zeithamova, Michael L. Mack, Kurt Braunlich, Tyler Davis, Carol A. Seger, Marlieke T.R. van Kesteren, and Andreas Wutz. Brain mechanisms of concept learning. *Journal of Neuroscience*, 39(42):8259–8266, 2019. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.1166-19.2019. URL `https://www.jneurosci.org/content/39/42/8259`.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction, 2019.

Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. Knowledge graph embedding with hierarchical relation structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3198–3207, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1358. URL `https://www.aclweb.org/anthology/D18-1358`.