Statistical Relational Learning for Knowledge Graphs

Link Prediction with Latent Feature Models

Luyolo Magangane

Department of Mathamatics
Stellenbosch University

This dissertation is submitted for the degree of
Master of Sciences

January 2020

I would like to dedicate this thesis to my loving parents . . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Luyolo Magangane
January 2020

Abstract

This is where you write your abstract ...

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Statistical Relational Learning

1.1.1 Artificial General Intelligence

The ultimate ambition of artificial intelligence (A.I.) research is to produce human-level intelligence. Such intelligence can take form the form of conversational artificial general intelligence (AGI). The problem can be posed as a question and answer game [Alan Turing]. The premise - a human asks a series of questions to an unobservable agent, where the task is for the agent to provide answers indistinguishable from human responses. The challenge of course is the breadth of topics that can be queried, as well as the permutations of possible coherent responses.

1.1.2 General Question Answering

This task is also known as general question answering (GQA) [references]. Typical A.I. implementations operate within well bounded, narrow domains. For example audio to text synthesisers [references]; object detection within images and videos [references]; financial, meteorological and operational forecasting [references]. All of these modelling applications focus more generally on finite perceptive domains [references]. In order to achieve AGI, reasoning capability is also required [requirements for AGI reference].

1.1.3 Applications of General Question Answering

Intelligent virtual assistants (IVA) such as Alexa, Cortana, Google Assistant and Siri are mobile application attempts at AGI; Watson, Wolfram Alpha and Knowledge Engine are other GQA computer systems. So often a user will ask one of these systems a question which results in a search engine lookup, due to failure in comprehending the question and the capability of providing a coherent response.

1.1.4 General Question Answering Approaches

Reinforcement learning. Will not be covered in this dissertation.... Knowledge representation and reasoning (KRR) is an active area of research being used as an approach to solve this problem [references]. The field aims to use mathematical logic to construct knowledge representations about a domain using knowledge graphs. Knowledge graphs (KG) model information in the form of facts presented as subject-predicate-object triples. These triples are expressed as entity-relational sets (e_1, r, e_2) where e_1 is the subject entity and e_2 is the object entity, and the relationships, r, between them. KGs have been used for information extraction, search query augmentation and general question answering. Google Knowledge Graph, DBPedia and Yago are some of the largest graph-structured database knowledge graphs implemented as knowledge bases.
KGs can be used to generate new knowledge from existing facts. These KGs can then be queried to provide logically consistent answers, more formally ontological reasoning is used to query ontological domains.

1.1.5 Uncertainty in Knowledge Representations

The problem with KRR is it contains no mechanism of capturing the uncertainty in known facts, as well as the uncertainty when new facts are generated, statistical relation learning (SRL) is an approach to solve this problem and active research area. SRL comprises three primary tecnhiques: Latent Feature Modelling - this comprises building entity-rational representations for relationship classification, Graph Feature modelling - Using graphical structures to model entity-relational properties within a domain, and Inductive Probabilistic Logic programing - Building probabilistic models of facts to directly model uncertainty in knowledge bases. Tasks in SRL include link prediction, entity-resolution and link-based clustering. Link prediction .... Entity-resolution ..., Link-based clustering ... Link prediction with Latent Feature Models is the focus of this dissertation.

A *LaTeX class file* is a file, which holds style information for a particular LaTeX.

$$CIF: \quad F_0^j(a) = \frac{1}{2\pi\imath} \oint_\gamma \frac{F_0^j(z)}{z-a} dz \tag{1.1}$$

1.2 Modelling Techniques

1.2.1 Latent Feature Modelling

Entities and relations are words that can be represented as real-valued vectors [references]. These real-valued vectors form part of a euclidean embedding space that represents a knowledge domain [references]. The entity and relational vectors can be randomly generated, or be pre-computed to capture semantic meaning [references]. A classification model can then be constructed that generates a probability distribution over probable facts within the knowledge domain. In order to compute the probability distribution, a number of latent feature modelling techniques are used, including tensor factorisation [references], circular correlations [reference] and convolutional feature maps [reference]. These methods can broadly be defined as linear and nonlinear. Attractive attributes of linear latent models are their simplicity, ease of implementation and computational efficiency. Linear latent models however suffer from a lack of expressiveness and struggle to model complex, contradictory or incoherent relationships between entities. Nonlinear latent feature models are able to produce more expressive latent feature sets, and so more adept at capturing complex relationships. Nonlinear models however suffer from computational inefficiency and poorly generalise concepts.

1.2.2 Graph Feature Modelling

In Graph Feature Modelling, Knowledge Graphs (KG) are used to model domains. KGs are composed of nodes and edges, where nodes represent entities and edges represent relations. The graphical structure then captures local, quasi-local and global domain properties. This global structure exhibits particular properties about relations within the domain, characteristics of the entities of the domain, and local entity-relational sub-structures. These graph structure properties are used in supervised [reference] and unsupervised [Graph Infomax] settings for SRL tasks such as link prediction and entity-resolution. The directional nature of edges in graph structures (uni-relational and bi-relational) is also exploited to further enhance the fulfillment of SRL tasks [reference]. The assumption in general in KGs is that similar entities will be collocated within a local and quasi-local regions, and that global similiarty patterns between entities will be captured by the ensemble of all paths between entities. Link-based clustering [reference] is thus used at all these structural scopes, and supports link prediction and entity-resolution tasks.

1.2.3 Inductive Probabilistic Logic Programming

Inductive logic programming uses ontological facts to discover new facts within a knowledge domain [reference]. logical rules check for things such as consistency. coherence and contradiction. Knowledge domains are implemented as knowledge bases (KB) that follow the resource description framework [reference]. KBs initialised in two steps: fact recording and materialisation - the discovery of new facts by running logical queries over the entire KB. KBs are extremely computationally demanding [reference], they also suffer from an inflexibility in modelling complex relationships due to their exactness, a fact is either true or false with no measure of ambiguity. Probabilistic logic programming languages have recently gained a lot of attention as flexible alternatives to logic programming languages as they are able to capture uncertainty in logical assertions through by modelling probability distributions over KB facts using stochastic variational inference. These models are thus more flexible in modelling complex relationships, and are also more computationally efficient [reference]. Inductive probabilistic logic programming has recently gained a lot of research attention due to it's capability of extending probabilistic logic programming languages with the capability of knowledge discovery.

1.3 Link Prediction with Latent Feature Models

1.3.1 Knowledge Graph Latent Feature Models

Link prediction with latent feature models involves building entity-relational representations from the nodes and edges of knowledge graphs expressed as subject-predicate object triples. These triples explicitly model facts within a knowledge domain. Entity and relational representations are commonly implemented as real-valued vectors. The vectors are then combined using compositional models, such as neural networks, to produce latent relational representations that can be used to compute the likelihood of plausible relationships between entities. The domain can be said to represent a multidimensiona embedding space into which the entities and relations are projected. Knowledge graph based latent feature modelling approaches are similar to semantic embedding representations [rerferenc]. They differ in that knowledge graph approaches explicitly model entity-relational interactions, and semantic embedding approaches rely on the distributional word representation techniques [reference], relying on Skip-Gram [reference] and Contious Bag of Words [reference] to generate word representations. This is an implicit modelling of relationships between word vectors.

1.3.2 Factorisation of Latent Feature Models

Factorisation attempts to model concepts between words, these facts are discovered using unsupervised techniques such as singular value decomposition. In the case of knowledge graphs, we obtain explicit representations of these concepts and can use them for entity-relational transformations that represent intermediate relational concepts that can then be used to determine plausible relationships when tested against subject entities. Tensor factorisation is an approach used for link prediction with latent feature models. It involves modelling entity relationships as matrix slices that comprise a relational tensor. The entity between entities is then computed using a bilinlear tensor product [reference], where the inner product of the object entity is taken with the matrix relational representation before an inner product of the resultant representation is taken with the subject entity. Bilinear tensor factorisation models are efficient in their number of parameters but lack expressiveness. Multilayer perceptrons have been used to overcome the lack of expressiveness however often suffer from overfitting. Recently convolutional neural networks have been proposed to allow expressive factorisation [references], do not suffer from overfitting and remain computationally efficient.

1.3.3 Other of Latent Feature Modelling Approaches

A number of alternative approaches to latent feature model factorisation have been proposed for link prediction, including circular correlation [reference], holographic entity-relational transformations [reference], toroidal representations [reference]. The rest of this dissertation focuses on factorisation of latent feature models, with the explicit representation of relational concepts.

Chapter 2

Literature Review

Statistical relational learning for knowledge graphs
Latent feature models
Link prediction

2.1 Knowledge Graphs

Knowledge graphs (KG) encode the relational information of a domain [reference]. KGs are composed of nodes and edges where nodes represent the entities and edges represent relations between entities. Two entities linked by a relation in a KG represents a fact, a KG is a collection of such facts represented as nodes and edges. Furthermore, the graphical structure of the representation models local, quasi-local and global properties about entities and relations. We can use direct entity relationships, or implicit structural properties to perform inference on the domain being modelled [reference]. For example
KGs are modelled under closed world or open world assumptions [reference]. Under a closed world assumption, the KG is considered complete, and the graph completely captures all facts within a domain. Under an open world assumption, it is possible to infer new facts about a domain from existing facts [reference]. It is also possible to infer new facts about a domain from the structural properties of the graph. Under an open world assumption, link prediction is used to generate new facts from existing ones. Graphs can thus be queried for existing facts, and can be 'reason' new facts. KGs have been applied to the field of information extraction [reference], search query augmentation [reference] and general question answering [reference].

Knowledge Bases
Ontologies

RDF

I'm going to randomly include a picture Figure 2.1.

If you have trouble viewing this document contact Krishna at: kks32@cam.ac.uk or raise an issue at https://github.com/kks32/phd-thesis-template/

Fig. 2.1 This is just a long figure caption for the minion in Despicable Me from Pixar

2.2 Matrix Factorization

Singular Value Decomposition Matrix factorisation is an approach used to compute latent features that capture relations between entities within a dataset [reference]. More formally, independent and identically distributed (IID) interactions between entities can be aggregated and represent in matrix format. The factorisation of the matrix will produce two real-valued unitary matrices that form a basis [reference], which can be thought of as a basis for a domain from where the data was sampled. Factorisation also produces a real-valued diagonal mastrix. The entries along the diagonal represent concepts captured by the dataset. A popular application of matrix factorisation is singular value decomposition [reference]. This is an implicit nodelling of relational domain concepts.

1. The first topic is dull
2. The second topic is duller
   (a) The first subtopic is silly
   (b) The second subtopic is stupid
3. The third topic is the dullest

Itemize

- The first topic is dull
- The second topic is duller
  - The first subtopic is silly

  - The second subtopic is stupid
- The third topic is the dullest

Description

The first topic is dull

The second topic is duller

> **The first subtopic** is silly
>
> **The second subtopic** is stupid

The third topic is the dullest

2.3 Tensor Factorisation

Tensor factorisation is an explicit modelling of entity relations using matrix slices of relational tensors [reference]. It involves modelling interactions of subject and object entities using relation-specific latent representations. These interactions have been modelled using the bilinlear tensor product, where the inner product of the subject entity is taken with a matrix relational representation, and the dot product of the generated embedding is then taken with the object entity. Bilinear tensor factorisation models are efficient in their number of parameters but lack expressiveness. Multilayer perceptrons were used to try overcome the lack of expressiveness however often suffer from overfitting. Recently convolutional neural networks have been proposed to allow expressive factorisation, do not suffer from overfitting and remain computationally efficient.
Tensor factorisation is a technique used in latent feature modelling to build representations of entities modified by their relationships. These entity-relational representations are then combined with other entities to score facts within the KG as well as the possibility of a new fact. Tensor factorisation models aim to be linearly scalable with datasets and so attempt to balance expressiveness with parameter efficiency.
RESCAL is a bilinear tensor model that combines latent entity-relational features of different entities using a matrix for each relation to compute a link score.
Distmult simplifies RESCAL by limiting the full rank relational matrix to a diagonal, with zeros everywhere else. Relational transformation of the entity is thus limited only to a stretch, limiting the expressiveness of the model whilst improving scalability.
TransE projects object entities e2 via relation-specific offsets and then aggregates the generated embedding features with the subject entity e1 and uses the new representation as input to a scoring function.

2.4 Nonlinear Factorisation

Parameterisation problem from tensor factorisation approach [nickel].
E-MLP ...
ER-MLP ...
Neural Tensor Networks extend the bilinear tensor product by concatenating the two entities and applying a weight operation on the concatenation to generate a latent representation which is then applied to a fully-connected neural layer to generation a score. HolE uses holographic embeddings described as circular correlations to compute triple scores. This

involves taking the circular product of the interactions between each entity, and then taking the product of the generated representation along with a transposed relational vector.
ComplEx ...
HolE ...
ConvE provides a more expressive model by taking the convolution of triple entities with a 2-dimensional convolutional relational model. This allows more expressive latent representations to be computed, whilst using a CNN architecture for parameter tying and parameter efficiency.
HypER simplifies ConvE by using 1-dimensional convolutional relational filters. The convolution of the relational filter is then taken with the subject entity e1, before a dot product with the object entity e2 generates a triple score.

(a) Tom and Jerry (b) Wall-E (c) Minions

Fig. 2.2 Best Animations

Subplots

I can cite Wall-E (see Fig. 2.2b) and Minions in despicable me (Fig. 2.2c) or I can cite the whole figure as Fig. 2.2

Chapter 3

Link Prediction With Latent Feature Modelling

3.1 Link Prediction

And now I begin my third chapter here . . . And now to cite some more people **? ?**]

Link prediction aims to rank plausible entity relationships. In latent feature modelling, it is posed as a classification task. These ranks are computed by scoring entity-relational pairs against all entities within the knowledge base. Relationships that rank highly become potential new facts that have been discovered and generate new knowledge within the knowledge base. Similar to inductive logic **? ?**], these facts need to be coherent, consistent and are not contradictory. Latent feature models contain statistical properties such as the capability to model a likelihood distribution over candidate entity relationships. This is.a more flexible modelling approach that provides a measure "confidence" in a fact, given the magnitude of the computed relationship score.

3.1.1 Entity-Relational Classification

. . . and some more
Entity-relational classification is a subsection of the more general ranking problem in machine learning (ML) **? ?**]. Standard classification involves the determining of the most appropriate categories in which entity belongs. If a logistic approach is used, a logit distribution across candidate classes is generated. A logit is referred to as an inverse probability **?**], a magnitude with is the passed thought a thresholding logarithmic sigmoid function to generate a likelihood, or a probability in a finite number of mutually exclusive classes..

The category with the highest logit associated with is then determined to be the most likely category in which the entity belongs. In the ranking.

3.2 Nonlinear Factorisation Models

and here I write more ... Tensor facorisation makes use of explicit relational representations, in contrast to implicit relational representations such as SVD. The relational representations are expressed as matrices, which are used to generate subject entity relational (ER) transformations by computing the inner product between them respectively. An inner product of the ER representation and object entity is then taken to produce a relational plausibility magnitude. The tensor factorisation algorithm thus makes use of linear representations as well as linear operations to compute relational scores. Deep learning modes have the capability to build useful latent feature representations, and given previous approaches to build relational representations using matrices, research was conducted to explore replacing relational matrices, with relational neural networks. E-MLP and ER-MLP were early attempts at relational factorisation using neural networks. Computing relational scores using these approaches is computationally expensive and prone to overfitting [**?**]. Recursive neural tensor networks were proposed as an alternative, however suffer from limited expressive power [**?**]. Convolutional neural networks were then used to model relational representations. In this approach, CNN filters explicitly model relations, and subject and object entities are modeled using real-valued feature vectors.. A convolutional operation is then performed using the the relational filters and subject entity. The generated ER representation is then flattened and an inner product is taken with the object entity to compute a relational score. ConvE and HypER are CNN approaches to relational factorisation.

3.2.1 Models

... and some more what they are, and how we got here, trial and error

Model	**Scoring Function**
RESCAL (Nickel, Tresp, and Kriegel 2011)	$e_1^T W_r e_2$
TransE (Bordes et al. 2013)	$\|\|e_1 + w_r - e_2\|\|$
NTN (Socher et al. 2013)	$u_r^T f(e_1 W_r^{[1..k]} e_2 + V_r \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_r)$
HolE (Nickel, Rosasco, and Poggio 2016)	$r_p^T (e_s * e_o)$
HypER (Balazevicl, Allen, and Hospedales 2018)	$f(vec(e_1 * vec^{-1}(w_r H))W)e_2$
HypER+ (Magangane and Brink 2019)	$f(vec(e_1 * vec^{-1}(w_r H))W) f(vec(e_2 * vec^{-1}(w_r H))W$

Table 3.1 Scoring functions of link prediction models. $*$ is the convolutional operator $F_r = vec^{-1}(w_r H)$ the matrix of relation specific convolutional filters, f is a non-linear function

3.2.2 Training Algorithm

... and some more ...

```
Result: Write here the result
initialization;
while While condition do
    instructions;
    if condition then
        instructions1;
        instructions2;
    else
        instructions3;
    end
end
```

Algorithm 1: How to write algorithms

First subsub section in the second subsection

... and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it ...

3.3 Model Analysis

... and some more ... Latent feature models determine factual plausibility by ranking all possible entity relationships within a knowledge graph. This is done by scoring all entities against against an ER representation, and then ranking those scores in descending order. To evaluate the performance of the latent feature model, a number of benchmark ranking metrics are used [?]. These metrics include the Hit@10 accuracy, Hit@3 accuracy, Hit@1 accuracy, Mean Rank, and Mean Reciprocal Rank. Definitions for each of the metrics are as follows:

- **Hit@10 accuracy**: Occurs when the target subject entity appears within the top ten ranked subject entities when scored against an ER representation
- **Hit@3 accuracy**: Occurs when the target subject entity appears within the top three ranked subject entities when scored against an ER representation
- **Hit@1 accuracy**: Occurs when the target subject entity appears as the top top ranked subject entity when scored against an ER representation
- **Mean Rank**: For all questions asked within the test dataset, what is the average target subject entity rank. Where a smaller value is better and a desired value of close to 1, which is the minimum.
- **Mean Reciprocal Rank**: The inverse of the mean rank. Where a larger value is better with a desired metric of close to 1, which is the maximum.

First subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more . . .

Second subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

3.4 Nonlinear Factorisation Hypothesis

and here I write more . . . Progress in ER classification using latent feature models has tracked progress made in deep learning model development. The current state-of-the art in image classification architecture makes use of fundamental convolutional architectures [?]. The current state-of-the in natural language modelling makes use of contextual word embeddings [?]. A potential approach to further improve the performance of latent feature models is to take context into account when selecting entities. A practical implementation of this approach is using distinct subject and object representations for scoring facts. This has the effect of taking into consideration the directionality of the the entity similar to the approach considered in [**?**]. A new model using this approach can be developed using the current state-of-the-art latent feature model, HypER, as a basis, and simply incorporate the use use of distinct subject and object entity representations. This new model, Hyper+, can then be evaluated using the model analysis metrics discussed above.

3.4.1 Models

...and some more what they are, and how we got here, trial and error

3.4.2 Training Algorithm

...and some more ...

First subsub section in the second subsection

...and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it ...

3.4.3 Model Analysis

...and some more ...

First subsub section in the third subsection

...and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more ...

Second subsub section in the third subsection

...and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it ...

Chapter 4

Deep Learning Models

4.1 Image Classification

. . . and some more . . . The state-of-the-art in deep learning classification models is convolutional neural networks (CNN) **?**]. CNNs are typically applied to object classification within images, and take advantage of the properties of object translation invariance and object location invariance [**?**]. These properties allow CNNs to perform robust classifications that generalise well across datasets. Translation invariance allows the CNN to build an object representation that is consistent under transformation, for example an image rotation would lead to a consistent final object representation being generated prior to logit computation for classification. Locality invariance allows the model to correctly identify an object no matter where it may reside with the boundaries of an image.

These properties are possible because of how the final object representation is generated. CNNs perform a convolutional operation on an image, using a trainable image filter [**?**]. The operation generates feature map of the image that constructs latent object representations of the image. In practise these representations are chained together to produce more complex latent representations the deeper the CNN.

It is possible to decompose convolution operations into spatial and depth-wise convolutions [**?**]. A spatial convolution operates on different regions of an image, producing distinct representations for each region, for example and image can be divided into four regions, where a spatial convolution operates independently on each of these regions using region-specific filters.This operation produces four distinct feature maps which are later flattened into a single hidden layer representation, before begin run through a linear layer to generate the final logits. In a depth-wise convolution, the distinct convolutional feature maps are generated using the depth dimensions of the input image, typically three dimensions with images, the red, blue and green image channels in a colour image. Channel-specific filters

are used to generate these feature maps and once again the future maps are flattened prior to computing logits for the model classes.
Multilayer perceptrons (MLP) are the precursors to CNNs and suffer from two problems when used in deep model architectures: an explosion of parameters and poor generalisation. In addition to the above mentioned properties, CNNs also address both these problems. Convolutional filters reduce the parameter space with subsequent convolutional operations in deep layers of the network. There are a number of techniques employed that further reduce the parameter space, these techniques include striding, dilation and pooling. CNNs also generalise better than MLPs by preventing the construction of redudant of correlated latent features. Without the capability of multi latent feature reliance in subsequent, of form a principle component generation is achieved by CNNs. Because these principle components are independent, the model is able to achieve good generalisation.

4.2 Natural Language Models

. . . and some more . . . The state-of-the-art in deep learning natural language models is contextual word representations. **?**]. Language models are trained using the distributional representation of words, a concept where a word sequence is used to classify the next word in the sequence, continuous bag of words [**?**], or a context word either side of a centre word, skip gram [**?**]. Language models are trained using a corpus of words such as news articles, Wikipedia and customer reviews. A text corpus is organised as a collection of documents, and the text within that document is broken up into sequential word tokens. Language models are build using euclidean embedding spaces [**?**]. Every word is represented as a real-valued feature vector of fixed dimension, and the teaks of natural language modelling, is to train word vectors such that accurate classifications can be made using either the CBOW or skip-gram techniques. Typically a deep learning model classification model is trained to make classification predictions [**?**]. The input word vectors are adjusted until the model achieves good performance, and a word to vector dictionary forms the complete language mode. Common language models in use today are Elmo [**?**], Bert [**?**], Fasttext [**?**], Glove [**?**] and Word2Vec [**?**]. ,

First subsub section in the second subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

4.3 Model Development Framework

. . . and some more . . . Deep learning classification models are trained using a supervised learning framework - input samples are fed into the network with corresponding target labels. The computed class, represented as a logit magnitude, is compared against the target class and an objective is defined that is to be minimised. The logit magnitude is then typically passed through a thresholding function, a softmax function [**?**] if a proper probability is desired, or a sigmoid [**?**] if only a likelihood is required. Two techniques are used to simultaneously improve both variance and bias respectively: batch normalisation [**?**] and dropout [**?**].

Batch normalisation attempts to account for internal covariance shift [**?**]. Batch normalisation is inspired from population based feature mean normalisation, when the entire population is taken into consideration when computing the mean and standard deviation of the values which the feature can take on within the dataset. When training a deep learning model, it is common to perform a uniform random shuffle of the training set, then subdivide that the set into mini-batches [**?**]. This is the same process of taking a uniform random sample batch from the training data. The size of the sample batch determines whether the normalised features remain within distributional alignment of the population. In order to compensate for the resultant loss computed after a foward pass, it is important to re-align feature vector parameters to the sample distribution mean and standard deviation estimators. The generated loss surface is thus a closer approximation to the true loss surface of the population, and subsequent parameter updates do not suffer from sample distributional distortion. This results in improved test accuracy as the model is able to more closely approximate the true distribution of the data.

Model over parameterisation can lead to overfitting. This because the degrees of freedom available to estimate a function allow very complex nonlinear functions to be discovered, where these nonlinear functions or not representative of the generative process of the data at all [**?**]. These overfitting makes the model very sensitive to the training data, and results in poor generalisation across training, validation and testing data. This problem forms part of the broader bias/variance trade off, specifically high model variance due to over parameterisation. Deep learning models are particularly sensitive to overfitting given the high number of parameters present within the model. In order to compensate for this overfitting, it is common zero out a sample of neurons within a deep learning network. This has the effect of removing partial dependence between nodes deeper within the network, a form of principal component analysis generating independent latent features. These latent features result in a simpler model representation of the generative process of the data, and therefore allow it to generalise better across datasets.

First subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more and some more . . .

Second subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

4.4 Loss Surface Analysis

and here I write more . . . Defining an objective function is best informed by analysing the surface it generates. If the objective functions aims to minimise an error, it is a loss function, and if an objective function maximises an expected return, it is a reward function. Nonlinear factorisation models are commonly trained using loss functions presented in the following table:

Name	**Expression**
Log (Nickel, Tresp, and Kriegel 2011)	$e_1^T W_r e_2$
Constrastive Max Margin (Bordes et al. 2013)	$\|\|e_1 + w_r - e_2\|\|$
Cross Entropy (Socher et al. 2013)	$u_r^T f(e_1 W_r^{[1..k]} e_2 + V_r \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_r)$
Binary Cross Entropy (Nickel, Rosasco, and Poggio 2016)	$r_p^T (e_s * e_o)$
Softmax Cross Entropy (Balazevicl, Allen, and Hospedales 2018)	$f(vec(e_1 * vec^{-1}(w_r H))W)e_2$
Sparse Categorical Cross Entropy (Magangane and Brink 2019)	$f(vec(e_1 * vec^{-1}(w_r H))W) f(vec(e_2 * ve$

Table 4.1 Scoring functions of link prediction models. $*$ is the convolutional operator $F_r = vec^{-1}(w_r H)$ the matrix of relation specific convolutional filters, f is a non-linear function

The choice of loss function determines the training loss surface which in turn determines the expected model accuracy and convergence rate.

4.4.1 Models

. . . and some more what they are, and how we got here, trial and error

4.4.2 Training Algorithm

. . . and some more . . .

First subsub section in the second subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

4.4.3 Model Analysis

. . . and some more . . .

First subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more and some more and some more and some more . . .

Second subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

4.5 Deep Learning Best Practice

This section has been modified from "Publication quality tables in LaTeX*" by Simon Fear.
Xavier initilisation [**?**] is commonly used to initialise model parameters. This initialisation technique has the benefit of non-bias. It also influences the starting position on the loss surface increasing the likelihood of convergence. An other common parameter initialisation strategy is truncated normal initialisation [**?**], where model paramaters are sampled from a univariate Gaussian distribution with mean zero and variance one.
The Adam [**?**] optimiser is commonly used for model training. This optimiser is an adaptive moment optimiser that scales the learning rate depending on the scale of parameter update changes. Other optimisers used for parameter updates include stochastic gradient descent [**?**], ADAG-Grad [**?**] and RMS-Prop [**?**].
Learning rate scheduling [**?**], the practice of reducing the learning rate after successive training cycles is now also common. It is implemented either as a continual process, where

the learning rate is reduced by some proportion after every iteration, or on a step basis, where the learning rate is reduced by a proportion after every epoch.
One of the most challenging aspects of model training is hyper parameter optimistaion [**?**]. Bayesian optimisation techniques have been used recently to overcome this problem [**?**]. Another hyper parameter search technique commonly used is random grid search [**?**]. This technique relies on iterations of hyper parameter sets that are uniformly sampled from range of available values.

The layout of a table has been established over centuries of experience and should only be altered in extraordinary circumstances.

When formatting a table, remember two simple guidelines at all times:

1. Never, ever use vertical rules (lines).
2. Never use double rules.

These guidelines may seem extreme but I have never found a good argument in favour of breaking them. For example, if you feel that the information in the left half of a table is so different from that on the right that it needs to be separated by a vertical line, then you should use two tables instead. Not everyone follows the second guideline:

There are three further guidelines worth mentioning here as they are generally not known outside the circle of professional typesetters and subeditors:

3. Put the units in the column heading (not in the body of the table).
4. Always precede a decimal point by a digit; thus 0.1 *not* just .1.
5. Do not use 'ditto' signs or any other such convention to repeat a previous value. In many circumstances a blank will serve just as well. If it won't, then repeat the value.

A frequently seen mistake is to use '\begin{center}' … '\end{center}' inside a figure or table environment. This center environment can cause additional vertical space. If you want to avoid that just use '\centering'

Table 4.2 A badly formatted table

	Species I		Species II	
Dental measurement	mean	SD	mean	SD
I1MD	6.23	0.91	5.2	0.7
I1LL	7.48	0.56	8.7	0.71
I2MD	3.99	0.63	4.22	0.54
I2LL	6.81	0.02	6.66	0.01
CMD	13.47	0.09	10.55	0.05
CBL	11.88	0.05	13.11	0.04

Table 4.3 A nice looking table

Dental measurement	Species I		Species II	
	mean	SD	mean	SD
I1MD	6.23	0.91	5.2	0.7
I1LL	7.48	0.56	8.7	0.71
I2MD	3.99	0.63	4.22	0.54
I2LL	6.81	0.02	6.66	0.01
CMD	13.47	0.09	10.55	0.05
CBL	11.88	0.05	13.11	0.04

Table 4.4 Even better looking table using booktabs

Dental measurement	Species I		Species II	
	mean	SD	mean	SD
I1MD	6.23	0.91	5.2	0.7
I1LL	7.48	0.56	8.7	0.71
I2MD	3.99	0.63	4.22	0.54
I2LL	6.81	0.02	6.66	0.01
CMD	13.47	0.09	10.55	0.05
CBL	11.88	0.05	13.11	0.04

Chapter 5

Results and Analysis

5.1 Nonlinear Tensor Factorisation

Hypothesis Overview

In this chapter we evaluate three hypothesis explored in nonlinear tensor factroisation for link prediction. Latent feature modelling has traditionally relied on linear modelling techniques in order to constrain parameterisation for computational efficiency at the expense of model expressiveness i.e. models that suffer from high bias. Recently nonlinear approaches have been proposed to overcome the bias problem and each propose hypothesis that aim to take advantage of their distinct architecture. The first nonlinear model to gain surpass state-of-the-art tensor factorisation models, was the recursive neural tensor network [**?**]. There are two hypotheses proposed that were proposed in using this model architecture for link prediction, namely that recursive architectures are useful in part of speech tagging [**?**], as the perform a combinatorial analysis of word sequences in sentences, instead of the conventional linear sequence used in part of speech tagging [**?**]. The second hypothesis is that pre-trained word embeddings offer superior performance to randomly generated entity vectors which were commonly used at the time for tensor factorisation [**?**].
The second hypothesis we explored is the use of convolutional hyper networks with relational factorisation for link prediction [**?**]. This research is the current state-of-that-art latent feature modelling technique for link prediction [**?**]. There are also two hypotheses explored in this research - the first is that convolutional filters can be used explicitly for relational factorisation [**?**], the second is that the hyper network architecture is useful in generating the relational filters due to increased parameterisation using a hyper linear layer [**?**].
The final hypothesis we explored is the use of contextual entity representations as input to convolutional hyper networks. State-of-the-art natural language modelling techniques makes

use contextual word representations [? ?] instead of static word embeddings [? ? ?]. Contextual word embeddings are represented as distinct subject entity representations, and distinct object entity representations.
The following chapter presents the results and analysis from the experiments performed to validate these hypotheses.

5.2 Recursive Neural Tensor Networks

Model Summary

Tensor factorisation has been a popular approach applied to statistical relational learning (SRL) [? ? ?]. In order to improve model expressiveness, a Recursive Neural Tensor Network architecture was proposed [?]. This model is inspired by recursive neural networks [?]. Recursive neural networks have been applied to natural language processing tasks, particularly part of speech tagging [?]. Traditionally, recurrent neural networks have been applied to this task [?]. Recurrent neural networks are based on the Markov assumption [?], where the future states of the process depends only on the present state, not on the sequence of events that preceded it [?]. The state is constructed using sequential input within a time step window, for example, a sentence will contain a sequence of words, and the window will be the last word in the sentence and the four previous words before an anchor word. The state consists of the four previous words before the anchor word, and along with the anchor word, can be used to predict the next word in the sequence.
Instead of producing a state using a sequence of previous words, recursive neural networks generate the state using a combinatorial tree of the word sequence. The number of potential combinations are dependent on the length of the sequence, for example a three word sequence has three combinations, a four word sequence has six combinations and a five word sequence has ten combinations. The combinations which produce the smallest magnitudes are then filtered out from the network, and remaining combinations are then recombined as new combinations, This filtering and recombination process is then continued until a next word prediction is made.
Recursive neural networks are combined with tensor factorisation to produce a more expressive model for link prediction. The subject and object entities are concatenated and then an matrix multiplication is taken with the resultant vector before a bias is added.

Contrastive Max Margin Loss

The contrastive max-magrin loss [**?**] is used to train the RTNT model. The input consists of an entity-relational pair, where the entity is a subject entity. An object entity is presented as a target to complete the triple. A non-related object entity is presented is then presented as a corrupt object entity. Fact score is then computed for the target triple, as well as the corrupt triple as logits, and the difference between the two logits is the contrast between the true and false facts. The task is for the model is then to produce a higher fact score for the true triple than the false triple. If the model gets it wrong, then loss is generated and back propagated through the network to update model parameters.

Experimental Setup

We use the following benchmark knowledge graphs: Wordnet - a lexical database for English, it is a taxonomy with hypernyms (is-a) relationships, and synonym sets.
Freebase - a large collaborative knowledge base consisting of data about the world composed mainly by its community members. It was an online collection of structured data harvested from many sources, including user-submitted wiki contributions.
Visualisations of the respective knowledge graphs are presented below:

Fig. 5.1 Wordnet Entities and Relations Graphplot

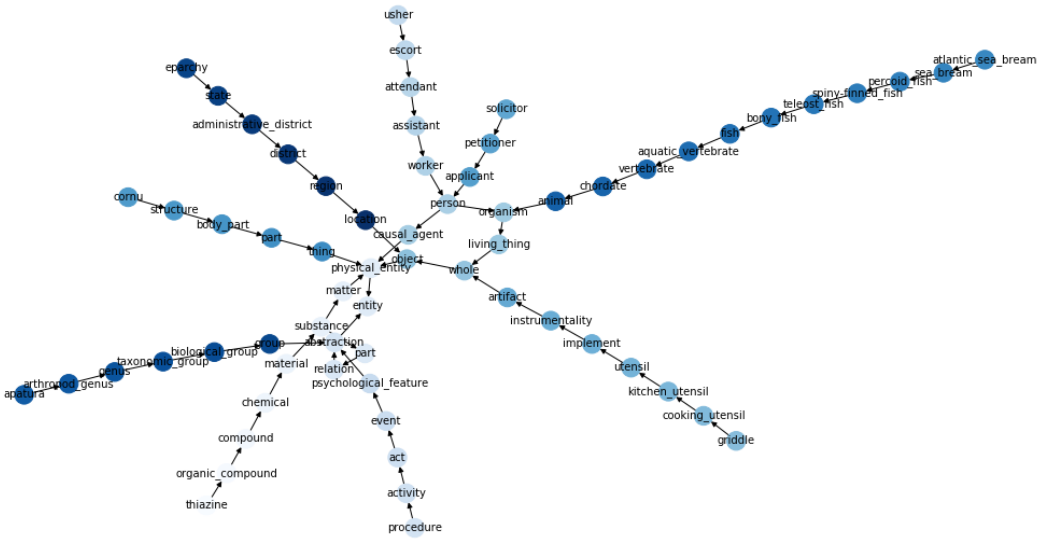
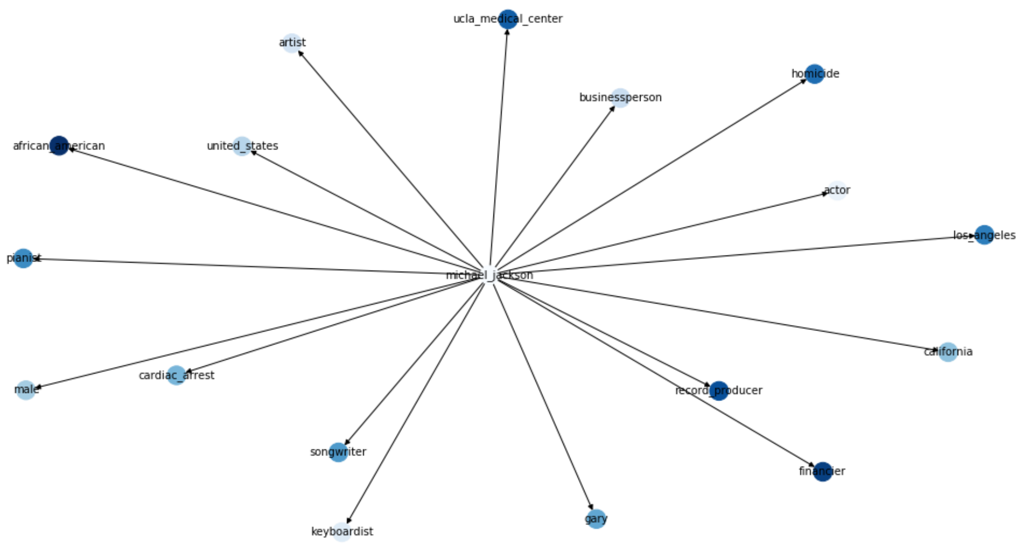


Fig. 5.2 Freebase Entity and Relations Graphplot



We used the Tensorflow framework to develop our model. Randomly initialised entity and relational embeddings are used to initialise model training. These embeddings are dynamically adjusted during the training process to generate latent representations specific to the knowledge domain. Property counts for the respective knowledge graphs are presented below:

Fig. 5.3 Wordnet Property Barplot

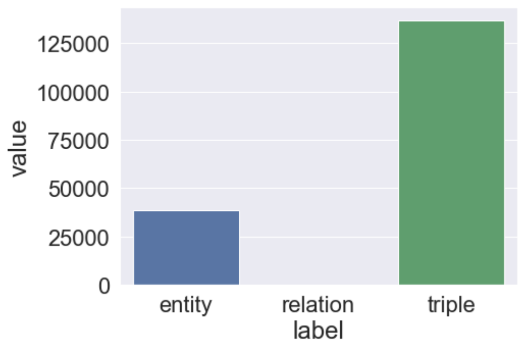


Fig. 5.4 Freebase Property Barplot

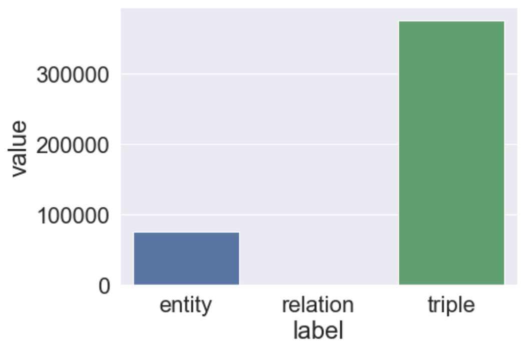


Table 5.1 Wordnet Property Counts

Property	Count
Entities	38,696
Relations	11
Triples	136,611

Table 5.2 Freebase Property Counts

Property	Count
Entities	75,043
Relations	13
Triples	375,499

Summary statistics of the respective knowledge graphs Resource Description Framework (RDF) decompision - subject, predicate, object - are presented below:

Fig. 5.5 Wordnet Predicate Barplot

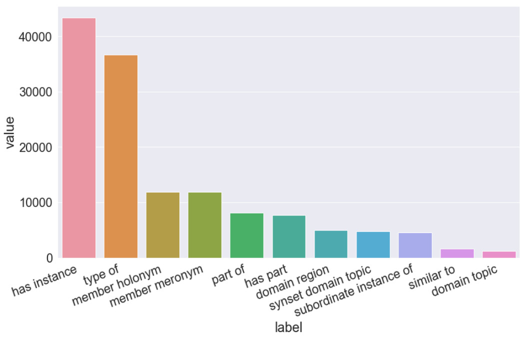


Fig. 5.6 Freebase Predicate Barplot

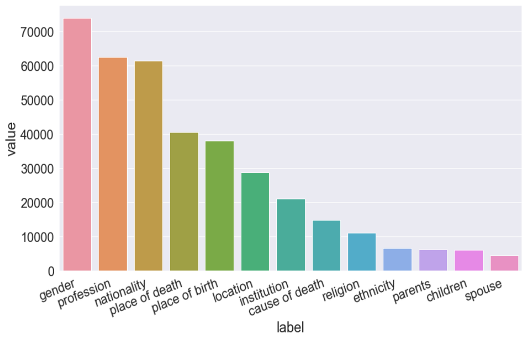


Table 5.3 Wordnet Predicate Statistics

Statistic	Value
Count	11
Max	43,312
Min	1, 229
Median	7,705
IQR	7,257.5

Table 5.4 Freebase Predicate Statistics

Statistic	Value
Count	13
Max	73,897
Min	4, 464
Median	21,149
IQR	34, 033

Fig. 5.7 Wordnet Subject Barplot

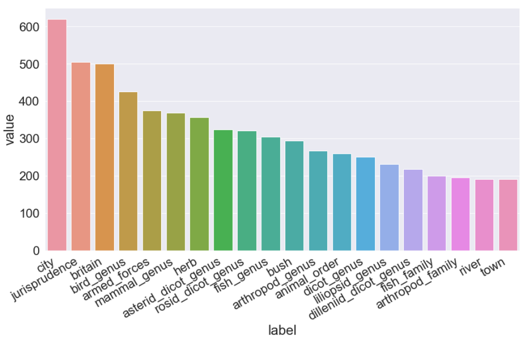


Fig. 5.8 Freebase Subject Barplot

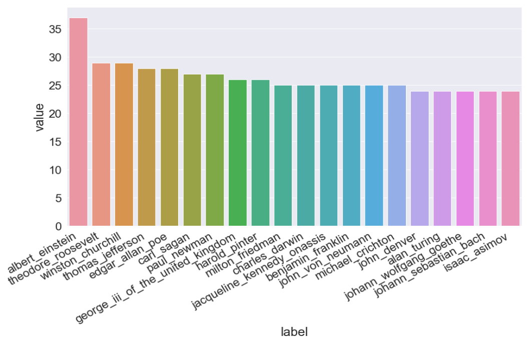


Table 5.5 Wordnet Subject Statistics

Statistic	**Value**
Count	32,720
Max	619
Min	1
Median	2
IQR	2

Table 5.6 Freebase Subject Statistics

Statistic	**Value**
Count	67,393
Max	37
Min	1
Median	5
IQR	4

Fig. 5.9 Wordnet Object Barplot

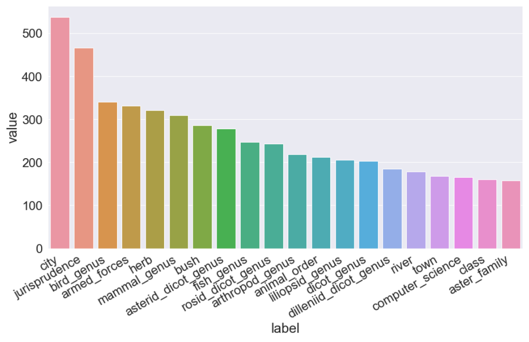


Fig. 5.10 Freebase Object Barplot

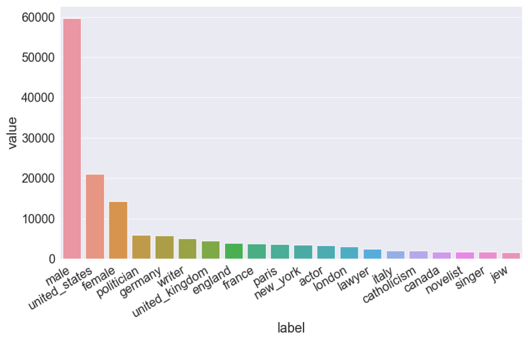


Table 5.7 Wordnet Object Statistics

Statistic	**Value**
Count	33,011
Max	537
Min	1
Median	3
IQR	2

Table 5.8 Freebase Object Statistics

Statistic	**Value**
Count	15,342
Max	59,663
Min	1
Median	3
IQR	6

For Wordnet, it can be seen that relations are skewed toward "has instance" 43,312 occurrences, and "type of" 36,659 occurrences. We would expect poor performance from the model for out-of-sample data containing containing relations that are not "has instance" or "type of". Freebase relations are somewhat more uniform, however four relations have occurrences under 10,000. We would expect reasonable performance across all relations for this knowledge graph.

Wordnet and Freebae subjects are somewhat uniform, although the median number of occurrences is 2 and 5 respectively, with an IQR of 2 and 4 respectively. This are thus sparse distributions of data points, and the model will need to rely on similarity of subject features

in order to produce sensible inference.
Wordnet object occurrences are somewhat uniform. Freebase object occurrences are skewed, with a single object, "male" occurring 59,663 times, representing 15,88% of facts. This is in comparison to a median object occurrence of 3 and an interquartile range of 6. We would expect poor performance from a Freebase link prediction model given this distribution of objects.

Hypothesis 1: Hyper Parameter Optimisation

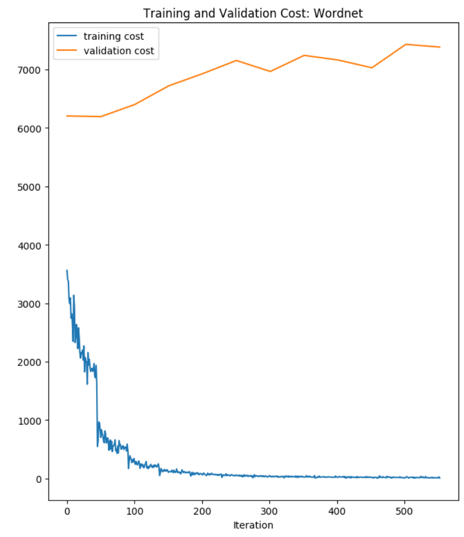
Model hyperparameter optimisation is implemented using random grid search. The the model was trained on a MacBook Pro 2015 with 8 cores, 16GB RAM, and 512GB SSD.
We evaluate the model by ranking the accuracy scores of the predicted triples for the respective datasets. Code to reproduce: https://github.com/xhosaBoy/deep-knowledge-modelling

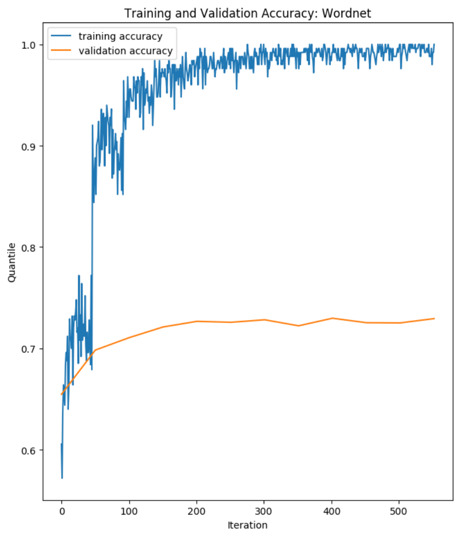
Bias Variance Trade Off

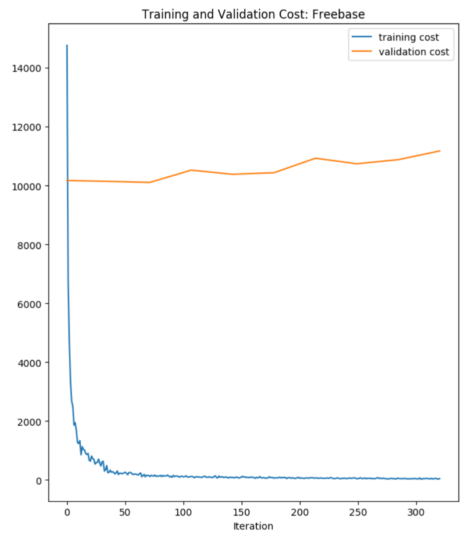
Machine learning models can suffer from overfitting [**?**] - when training has progressed too long, and the model parameters have been optimised for prediction on the training set. This leads to poor generalisation and reveals itself in the form of increasing training accuracy, and decreasing validation accuracy. In order to overcome this problem, early stopping is often used during training. This is when triaining is terminated should decreasing validation accuracy be detected.

Link Prediction Results

The link prediction accuracy results of the RNTN model, a nonlinear factorisation model compared against linear factorisation models, are presented in table 5.1:


Training and Validation Cost: Wordnet
training cost
validation cost
Iteration


Training and Validation Accuracy: Wordnet
training accuracy
validation accuracy
Quantile
Iteration


Training and Validation Cost: Freebase
training cost
validation cost
Iteration

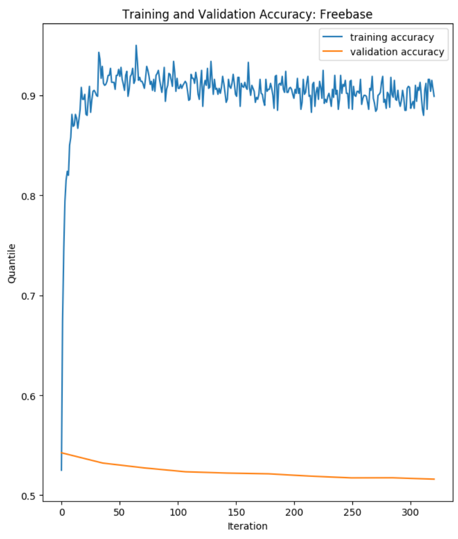

Training and Validation Accuracy: Freebase
training accuracy
validation accuracy
Quantile
Iteration

Table 5.9 Link prediction accuracy on Wordnet and Freebase

Model	Wordnet	Freebbase	Avg
Distance Model	.683	.610	.647
Hadamard Model	.800	.688	.744
Single Layer Model	.760	.853	.807
Bilinear Model	**.841**	**.877**	**.859**
Recursive Neural Tensor Network	.732	.518	.625
Recursive Neural Tensor Network	.732	.518	.625

5.3 HypER Convolutional Neural Networks

Model Summary

In order to overcome the expressiveness problems evident in the Freebase test dataset, more nonlinear relational factorisation approaches have been proposed [**?** **?** **?**].
HypER is a model that uses convolutional relational filters r that are convolved with the subject entity e1, producing an intermediate relational representation. The dot product of the relational representation is then taken with the object entity e2 to produce relation-specific score between the two entities. The HypER model only produces a latent relational representation for the subject entity e1, we extend this model to also produce a relational representation for the object entity e2, and call this model HpyER+. HypER+ thus produces entity-relational representations that approximate KG spacial locality.

Binary Cross Entropy Loss

The binary cross entropy loss [**?**] is used to train the HypER Convolutional model. Like the RNTN model, the input consists of an entity-relational pair, where the entity is a subject entity and an object entity is presented as a target to complete the triple. A logit is generated for each sample and passed through through a logarithmic sigmoid or softmax function. Loss is generated by comparing the produced likelihood with the expected likelihood, 0 or 1. The sum of all losses is aggregated and back propagated through the network for parameter update.

Experimental Setup

We use the following benchmark datasets: WN18 (Bordes et el. 2013): WN18 is a subset of Wordent, a lexical database for English, it is a taxonomy with hypernyms (is-a) relationships, and synonym sets. The dataset contains 40,943 entities and 18 relations. WN18RR (Detmers etl el. 2018): WN18RR is a subset of WN18, created by removing the inverse relations. WN18RR contains 40,943 entities and 11 relations.
FB15k (Bordes et el. 2013): FB15k is a subset of Freebase, a large collaborative knowledge base consisting of data about the world composed mainly by its community members. It was an online collection of structured data harvested from many sources, including user-submitted wiki contributions. FB15k contains 14,951 entities and 1,345 relations. FB15k-237 (Toutanova et el. 2018): FB15k-237 is a subset of FB15k, created by removing the inverse relations. WN18RR contains 40,943 entities and 11 relations.
We used the PyTorch framework to develop our model. The code to reproduce results can be found on GitHub 1. Randomly initialised entity and relational embeddings are used to initialise model training. These embeddings are dynamically adjusted during the training process to generate latent representations specific to the knowledge domain. Model hyperparameter optimisation is implemented using random grid search. The the model was trained on a high-memory Google Compute Engine instance with 8 cores, 16GB RAM, 250GB SSD with a NVIDIA Tesla V100.
We evaluate the model by ranking the scores of the predicted triples. Three ranking metrics are computed: Hit@10 - The percentage of predictions where the target object entity appeared within the top 10 results. Hit@3 - The percentage of predictions where the target object entity appeared within the top 3 results. Hit@1 - The percentage of predictions where the target object entity appeared as the topped ranked result. The mean rank (MR) of target the target entity object, as well as the reciprocal of the mean rank (MRR) are also computed as model evaluation metrics. Code to reproduce: https://github.com/xhosaBoy/hypernetwork-factorisation

Link Prediction Results

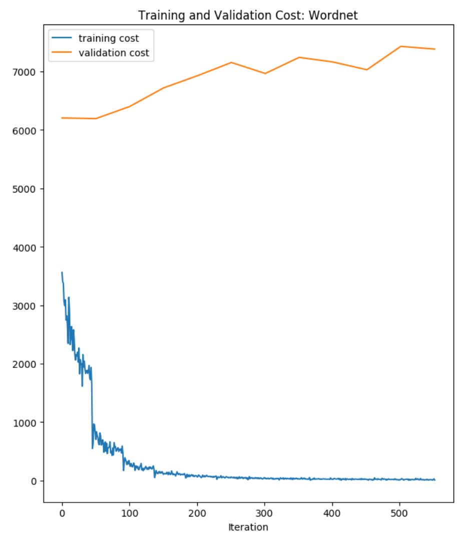
The link prediction accuracy results of the HypER Convolutional model, compared against other nonlinear factorisation models, are presented in table 5.2 and 5.3:

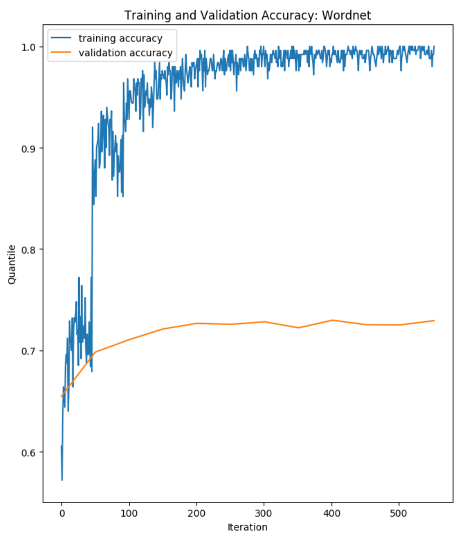
Table 5.10 Link prediction results on WN18RR and FB15k-237.

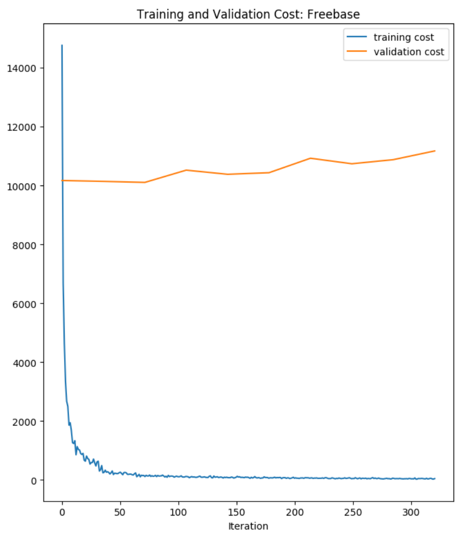
Dataset	**WN18RR**					**FB15k-237**				
	MR	MRR	H@10	H@3	H@1	MR	MRR	H@10	H@3	H@1
DistMult	5110	.430	.490	.440	.390	254	.241	.419	.263	.155
ComplEx	5261	.440	.510	.460	.410	339	.247	.428	.275	.158
Neural LP	-	-	-	-	-	-	.250	.408	-	-
R-GCN	-	-	-	-	-	-	.248	.417	.264	.151
MINERVA	-	-	-	-	-	-	-	.456	-	-
ConvE	**4187**	.430	.520	.440	.400	**244**	.325	.501	.356	.237
HypER	5798	**.465**	**.522**	**.477**	**.436**	250	**.341**	**.520**	**.376**	**.252**
HypER 300d	5772	.461	.515	.475	.432	487	.316	.488	.346	.230
HypER+ 300d	**2692**	**.474**	**.546**	**.484**	**.440**	**450**	**.317**	**.492**	.346	.230

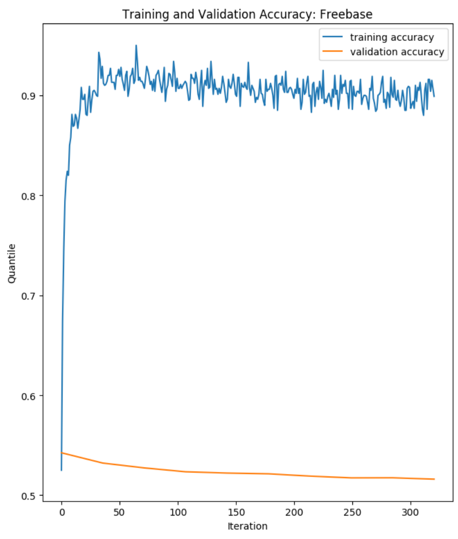
Table 5.11 Link prediction results on WN18 and FB15k.

Dataset	**WN18**					**FB15k**				
	MR	MRR	H@10	H@3	H@1	MR	MRR	H@10	H@3	H@1
TransE	**251**	-	.892	-	-	125	-	.471	-	-
DistMult	902	.822	.936	.914	.728	97	.654	.824	.733	.546
ComplEx	-	.941	.947	.936	.936	-	.692	.840	.759	.599
ANALOGY	-	.942	.947	.944	.939	-	.725	.854	.785	.646
Neural LP	-	.940	.945	-	-	-	.760	.837	-	-
R-GCN	-	.819	**.964.**	.929	.697	-	.696	.842	.760	.601
TorusE	-	.947	.954	.950	.943	-	.733	.832	.771	.674
ConvE	374	.943	.956	.946	.935	51	.657	.831	.723	.558
HypER	431	**.951**	.958	**.955**	**.947**	**44**	**.790**	**.885**	**.829**	**.734**
HypER 300d	518	.950	.958	.953	.946	75	.826	.893	.852	**.786**
HypER+ 300d	**248**	**.951**	**.959**	**.954**	.946	**70**	**.828**	**.901**	**.859**	.783









5.4 HypER Convolutional Neural Networks with Pretrained Entity Embeddings

5.5 Leader Board Research Versus Hypothesis Research

Appendix A

How to install $\LaTeX$

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from
   https://www.tug.org/texlive/
2. Download WinCDEmu (if you don't have a virtual drive) from
   http://wincdemu.sysprogs.org/download/
3. To install Windows CD Emulator follow the instructions at
   http://wincdemu.sysprogs.org/tutorials/install/
4. Right click the iso and mount it using the WinCDEmu as shown in
   http://wincdemu.sysprogs.org/tutorials/mount/
5. Open your virtual drive and run setup.pl

or

Basic MikTeX - $\TeX$ distribution

1. Download Basic-MiK$\TeX$(32bit or 64bit) from
   http://miktex.org/download
2. Run the installer
3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin) and choose Package Manager

4. Select or search for packages to install

TexStudio - TeX editor

1. Download TexStudio from http://texstudio.sourceforge.net/#downloads
2. Run the installer

Mac OS X

MacTeX - TeX distribution

1. Download the file from https://www.tug.org/mactex/
2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - TeX editor

1. Download TexStudio from http://texstudio.sourceforge.net/#downloads
2. Extract and Start

Unix/Linux

TeXLive - TeX distribution

Getting the distribution:

1. TexLive can be downloaded from http://www.tug.org/texlive/acquire-netinstall.html.
2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

   ```
   mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
   ```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

   ```
   cd /your/download/directory
   ./install-tl
   ```

4. Enter command ‘i’ for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64bit OS

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
```

```
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CentOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.