University College London

MSc Data Science and Machine Learning

Neural Variational Inference For Embedding Knowledge Graphs

Author:
Alexander Cowen-Rivers

Supervisor:
Prof. Sebastian Riedel

Disclaimer *This report is submitted as part requirement for the MSc Degree in Data Science and Machine Learning at University College London. 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.The report may be freely copied and distributed provided the source is explicitly acknowledged.*

Machine Reading Group

October 23, 2018

"*Probability is expectation founded upon partial knowledge. A perfect acquaintance with all the circumstances affecting the occurrence of an event would change expectation into certainty, and leave nether room nor demand for a theory of probabilities.*"

George Boole

Acknowledgements

I would like to thank my supervisors Prof Sebastian Riedel, as well as the other Machine Reading group members, particularly Dr Minervini, who helped through comments and discussions during the writing of this thesis. I would also like to thank my father who nurtured my passion for mathematics. Lastly, I would like to thank Thomas Kipf from the University of Amsterdam, who cleared up some queries I had regarding some of his recent work.

Abstract

Alexander Cowen-Rivers

Neural Variational Inference For Embedding Knowledge Graphs

Statistical relational learning investigates the development of tools to study graph-structured data. In this thesis, we provide an introduction on how models of the world are learnt using knowledge graphs of known facts of information, later applied to infer new facts about the world. Current state-of-the-art methods are unable to quantify and control their inherent uncertainties. This inability is a significant problem: within specific domains inferred relations could entail enormous consequences. There has been a shortage of literature around the topic of modelling knowledge graphs through probabilistic embeddings, and even less exploration into embedding knowledge graphs using variational inference. One of the hindering factors is creating methods that are highly scalable, due to the number of possible connections in a knowledge graph growing exponentially with each additional node or relation. Traditional knowledge graph embeddings have used static vector representations, whereas we would like access to more expressive generative models. However, this expressiveness comes at the cost of the complexity in performing fast and accurate inference over the parameters. This paper proposes a novel framework, which can be used to create several generative models: first to approach the difficulty of objects and relationships having multiple meanings and second to aid with the development of tools to quantify uncertainty in facts. The new framework can create models able to discover underlying probabilistic semantics for entities or relations; this is achieved by utilising parameterisable distributions over entities and relations which permit training by back-propagation in the context of neural variational inference, resulting in a highly-scalable method. Our generative framework is flexible enough to allow training under any prior distribution that permits a re-parametrisation trick, as well as under any scoring function that permits maximum likelihood estimation of the parameters. Experiment results

display the potential and efficiency of this framework by improving upon multi-relational generative and non-generative benchmarks, whereby some datasets are so challenging that even state-of-the-art models do not exceed 50% performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Chapter gives an introduction to the project. Section 1.1 justifies the relevance of this study to the broader field of statistical relational learning. Section 1.2 states the research questions we attempted to answer in this project. Section 1.3 details the computational resources available to investigate these research questions. Section 1.4 details the code used to aid the project's progress, as well as the public repository of the projects work. Furthermore, Section 1.5 outlines the content of this report.

1.1 Motivation

Traditional machine learning methods take a feature matrix as input, which represents the examples as categorical (binary one hot encoded) or numeric features. Recent machine learning methods can take the raw examples as input. Both traditional and recent methods share the same primary goal of learning to infer the correct output/prediction from their input. These can fall into a variety of tasks such as; regression, classification, unsupervised representation learning. For a regression task, the output would be a score. For a classification task, the output would be a class label. For an unsupervised learning problem, the output would be a cluster assignment of similar representations, or a latent (hidden) representation of the input, such as word embeddings. Statistical Relational Learning involves the input being a representation of the relations between objects. This form of input can be directly visualised as a graph, where the nodes of information, referred to as entities, and the labelled directed/undirected edges are relationships between entities. The primary tasks in statistical relational learning include graph completion (predicting unknown edges), prediction of properties of nodes, as well as grouping nodes together based on their inter-connectivity. These tasks are used in domains such as

the analysis of communication networks, automatically extracting interpretable information from scientific articles [Muzaffar, Azam, and Qamar, 2015,Plake et al., 2006] and drug discovery [Sellwood et al., 2018,Fujiwara, Kamada, and Okuno, 2018]. The methods developed by statistical relation learning researchers can be applied to large-scale knowledge graphs, which store factual information through relationships (edges) between entities (nodes). Several knowledge graphs have been created, such as UMLS [Bodenreider, 2004], Kinship [Quinlan, 1989], WN18RR [Dettmers et al., 2017] and FB15k-237 [Toutanova and Chen, 2015a]. Modern knowledge graphs can contain millions of nodes and billions of connections,[1] which forces us to develop highly scalable methods, which, at worst, have time/space complexity linear in the number of nodes in the graph.

During link prediction, a model attempts to infer the relationship between entities in the system, which is a crucial challenge. Many artificial intelligence researchers strive towards creating systems capable of inferring novel and useful information, as this is important to any self-improving system, as well as to a system with the goal of aiding human progression. How, then, can we create systems capable of inferring novel and useful information? The previous question is the cardinal focus of attention for many researchers working on machine learning. This direction motivates experimenting with knowledge graph representations. However, these tools commonly lack the language of uncertainty, i.e. when the system is uncertain regarding its output. [Ghahramani, 2015,Gal, 2016] argue that uncertainty is of vital importance to be utilised by a system, and be informed of by an interpreter of the system.

Knowledge graph researchers have debated the challenges of multiple relation semantics, while none have discussed the problems of multiple relation **and** multiple entity semantics. By modelling a knowledge graph, we typically try to create clusters around similar relationships and underlying semantics. For example; there exist two latent semantics of the 'has part' relation. The relation 'has part' contains both location and composition relationships as "France is a part of Europe" and "lead is a part of a pencil". Entities can also have multiple latent meanings such as the verb 'set'. We may want the representation for 'set' to occupy the latent semantic space associated with mathematics and set theory. However, we would also desire that the object 'set' share semantic space with tangible objects, e.g. "set the flowers on the table", or configurable objects, e.g. "set the watch". A model capable of representing these various semantic meanings should, in theory, be able to better cope with the ambiguity induced

[1] https://medium.com/@Pinterest_Engineering/pinsage-a-new-graph-convolutional[..]

in language and knowledge, where ambiguity can be defined as a mixture over semantic relationships. The freedom to occupy numerous semantic spaces can be represented well with a probabilistic model of objects and relationships.

We now further motivate the desire to develop a probabilistic perspective on these representations. Measuring uncertainty is essential for the following example. If funding were available for only one research group to create a cure to a rare disease, we would want to invest in the correct research direction. To correctly spend, we may wish to simulate the disease to try and predict the likelihood of the various methods leading to a successful cure. To simulate this, we may create a knowledge graph model of known protein interactions, with the aim of predicting a reaction relationship between the proteins observed in the disease and potential cures. But what if the system had never observed the particle formations seen in this disease, how would the model react? When maintaining a probabilistic model, we can identify during the output stage that this is a new example and thus the model knows it will have relatively low confidence in the outcome. If we did not have this, it could lead to the wrong investment decision. This case is formally referred to as an out-of-distribution test example. Thus the ideal model would still attempt to predict the relationship between moving particles.

However, it would quantify its uncertainty in the information and display this alongside the output. The two main types of uncertainty are *aleatoric* and *epistemic* uncertainty. Aleatoric uncertainty is usually found due to noisy data, possibly caused by the imprecision of the measurement apparatus. Epistemic uncertainty usually arises from parameter uncertainty, i.e. which combination of parameters provides the more robust selection for a well-generalising model, as well as structure uncertainty, arising from choices of various model structures, such a depth of layers, latent embedding dimensions.

When we combine both aleatoric and epistemic uncertainty, we get *predictive* uncertainty. It is predictive uncertainty we use to quantify the confidence of the output from our model predictions. Through training our model, if we observe that predictive uncertainty is either over confident or under confident, this can guide us towards useful remedies. In the under-confident case, this could be due to a lack of diverse data as discussed previously.

Bayesian machine learning researchers choose to investigate through the lens of probabilistic models, with an emphasis on quantifying uncertainty. They tend to use models that express the frequent and less frequent behaviour of the observations. Thus allowing the modeller to view the confidence bounds when

an automated system is about to take a series of decisions, especially useful when these have low confidence. More generally, using a Bayesian modelling framework allows the modeller to ask significant questions such as "Was that output nearly a random decision, or a well-supported estimate?" and "Is my data lacking the variety required for a confident model?". We believe these questions motivate our desire to encourage knowledge graph researchers to create robust probabilistic systems which reflect these desirable characteristics.

The main argument of this thesis is that there is a lack of methods for quantifying predictive uncertainty in a knowledge graph embedding representation, which can only be utilised using probabilistic modelling, as well as a lack of expressiveness under fixed-point representations. This constitutes a significant contribution to the existing literature because we introduce a framework for creating a family of highly scalable probabilistic models for knowledge graph representation, in a field where there has been a lack of this. We do this in the context of recent advances in variational inference, allowing the use of any prior distribution that permits a re-parametrisation trick, as well as any scoring function which permits maximum likelihood estimation of the parameters.

1.2 Problem Statement

Amongst the research activity of knowledge graph construction and link prediction, the fundamental questions we wish to address in this thesis are:

1. *Firstly, can we propose an alternative neural approach to Knowledge Graph representation and link prediction that allows us to identify better and measure predictive uncertainty?*

2. *Secondly, can we incorporate previous work within stochastic variational inference to scale these robust representations to work on large graph structures?*

3. *Lastly, can we better address the challenge of entities and relations that require multiple representations?*

We will aim to answer these critical questions in parallel, and hope it will lead us towards a method which is highly scalable, efficient and robust. During this dissertation, we also answer other research questions such as;

1. *What can we learn from analysing the variances?*

2. *What are the trade-offs for representing Knowledge Bases under this alternative framework?*

1.3 Computational Resources

Fortunately, we had access to the University College London's computer science cluster which has CPU and GPU nodes. We also had access to an Apple MacBook Air 13 A1369 Core I5 4GB Ram 128GB SSD laptop. The MacBook was used for simple experiments and analysis but quickly become redundant with the introduction of more extensive datasets and the implementation of more sophisticated models.

1.4 Code

There were four external sources of code we found useful. These were: (i) code from the UCL MR groups public GitHub[2] used to evaluate the models; (ii) code from [Miao, Yu, and Blunsom, 2016], available also on Github[3] as a template for neural variational inference in tensorflow; and (iii) Dr Pasquale Minervini from the UCL Machine Reading lab had shared some private code on Github to load and process the datasets. However, Model A and Model B as introduced later were completely and individually implemented by me. We have released the code for this project on a public repository.[4] The majority of the code is written in Python 3.6 using the Tensorflow distributions framework [Abadi et al., 2015,Dillon et al., 2017]. We also used code from a guide for using TF distributions [Hafner, 2018].[5]

1.5 Thesis Outline

In the exploration of the challenges detailed previously, we have chosen to take an experimental approach on large publicly available datasets.

[2] https://github.com/uclmr/inferbeddings
[3] https://github.com/carpedm20/variational-text-tensorflow
[4] https://github.com/acr42/Neural-Variational-Knowledge-Graphs
[5] https://danijar.com/building-variational-auto-encoders-in-tensorflow/

Background

Chapter 2 introduces relevant background information into neural methods and the motivation for using variational inference, accompanied with an example of variational matrix factorisation. Chapter 3 introduces relevant background information on constructing datasets from knowledge graphs under various assumptions, as well as the different categories of models used to complete knowledge graph tasks. Chapter 4 identifies typical approaches to learning a model of knowledge graphs. Chapter 5 discusses related work and identifying how they differ from our approach.

Method

Chapter 6 outlines the theoretical justification for the two proposed generative models from this thesis, as well as providing implementation details of each model and the approaches used to scale the models to large datasets.

Experiment Analysis

Chapter 7 discusses the results from experiments over the model architectures introduced in Chapter 6. Chapter 8 analyses the quality of the learnt embedding distributions across a range of experiments as well as assessing two methods for estimating the confidence of our generative models, finishing with an ablation study on our best performing model.

Conclusion

Lastly, Chapter 9 explored the conjectures of our work and outlined future research directions we believe to be the most fruitful.

1.6 Thesis Contributions

Through this thesis, we would like to encourage knowledge graph researchers to step away from static vector representations in knowledge graphs and focus research further on more expressive probabilistic generative representations. As

a step towards this, we have proposed two novel generative models for multi-relational knowledge graph completion, a domain where few generative models have never been implemented. These generative models are also extremely flexible and enable the changing of scoring functions to any which allows maximum likelihood estimation of the parameters, as well as any prior distributions which would enable a re-parametrisation trick. Definite improvements on WN18 for Variational ComplEX are seen compared with the initially published results. Secondly, we have shown how to utilise the generative model to utilise uncertainty estimates. Thirdly, we were able to improve with re-implementations of previously published ComplEX results; Filtered Hits@1, and Filtered Hits@3 results on Kinship and UMLS, as well as improving upon ComplEX Filtered Hits@1 results on Nations. Fourth, we provide an alternative justification for a commonly used technique to estimate the Evidence Lower Bound using Bernoulli sampling. We aim to submit this work to the International Conference on Learning Representations 2019.

Chapter 2

Neural Variational Methods

2.1 Outline

This chapter will assume only knowledge of basic algebra, probability and optimisation theory, but not of the specific areas the thesis builds upon. We will give a brief snapshot of progress within neural methods, as well as introduce the theory upon which the proposed models are reliant. Section 2.2 is a formal introduction to the basics of Neural modelling and Section 2.3 a formal introduction to Bayesian neural modelling. Section 2.4 details the difficulties of Bayesian methods, leading to a popular alternative technique: variational inference. Section 2.5.2 introduces the key components to variational neural modelling, such as the re-parametrisation trick. Section 2.7 discusses active learning, a technique that can be applied to aid the learning process of a probabilistic system. Lastly, Section 2.8 will walk through an example of variational matrix factorisation.

2.2 Neural Modelling

We will now introduce the reader to neural modelling, as this is key to understanding the complexities of a method fundamental to the contributions of this thesis: neural variational methods. Machine learning has undergone a rebirth since 2013 as Deep Learning [Goodfellow, Bengio, and Courville, 2016]. The name "deep learning" was chosen due to the deep architectures (stacked layers of computations) used by many of the most successful machine learning models. One of the most prominent models within the field of deep learning (DL) is the neural network, categorically referred to as neural models. One of many areas within computer science that have exponentially improved as a result of DL is Machine Translation (MT); where MT entails the task of translating text from

one language to another. All of the top performing model choices for the most recent Machine Translation 2017 Shared Task [O, 2017], were DL models.

First, we will introduce the reader to a one layer neural network [Rumelhart, Hinton, and Williams, 1988], arguably the simplest neural model. This model has many similarities to logistic regression, and is commonly used to model the probability of a random variable Y being assigned to two classes, e.g $\{-1, 1\}$. Given a dataset of n examples $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Where x_i is the observed information and y_i^{true} is the true target label "1". We usually partition $\mathcal{D}$ into $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$. We further partition this into $\mathbf{X}_{train} = \{x : x \in \mathcal{D}_{train}\}$ and $\mathbf{Y}_{train} = \{y : y \in \mathcal{D}_{train}\}$, similarly for $\mathbf{X}_{test}$ and $\mathbf{Y}_{test}$, with the property that $\mathbf{Y}_{train} \cap \mathbf{Y}_{test} = \emptyset$ and $\mathbf{Y} = \mathbf{Y}_{train} \cup \mathbf{Y}_{test}$, similarly for $\mathbf{X}$. We would then train on $\mathbf{X}_{train}, \mathbf{Y}_{train}$ and use $\mathbf{X}_{test}, \mathbf{Y}_{test}$ to measure our model performance. Inference on our neural model is performed using the below equation;

$$y_i^{prediction} = P(x_i = 1) = f^{\mathbf{w}}(x_i) = \Theta(\mathbf{b}_1 + \mathbf{W}_1 x_i) \tag{2.1}$$

and,

$$y_i^{prediction} = P(x_i = -1) = (1 - P(x_i = 1)) = 1 - f^{\mathbf{w}}(x_i) = 1 - \Theta(\mathbf{b}_1 + \mathbf{W}_1 x_i) \tag{2.2}$$

Where $\Theta(x) = \frac{1}{1+e^{-x}}$.

We can think about training logistic regression simply as finding the best parameters $\mathbf{b}_1$, our bias, and $\mathbf{W}_1$, our weights, such that we minimise our prediction loss $\mathcal{L}$ w.r.t $\mathbf{W}_1$ and $\mathbf{b}_1$, calculated through the negative log likelihood (NLL) $\mathcal{L}(\mathbf{W}_1, \mathbf{b}_1) = \sum_{i \in \mathcal{D}_{train}} -y_i^{true} \log(y_i^{prediction})$.
It is quite common to add a regularisation term into our loss function for better generalisation onto $\mathcal{D}_{test}$. Regularisation can be added by including a scaled L_2 norm of the parameter weights into the loss.

$$\mathcal{L}(\mathbf{W}_1, \mathbf{b}_1) = (\sum_{i \in \mathcal{D}_{train}} -y_i^{true} \log(y_i^{prediction})) + \lambda_1 ||\mathbf{W}_1||_2 \tag{2.3}$$

Where the weight decay λ_1 is a hyperparameter that is tuned to maximise performance. This neural model is typically trained through maximum likelihood estimation, using optimisation techniques such as stochastic gradient descent.

2.3 Bayesian Neural Modelling

We will now introduce Bayesian neural modelling, as this is the field which inspired the creation of variational methods. A Bayesian network is formally a directed graphical model, in which the joint distribution factorises into the below, where is η the parameters for the posterior distribution, which generates the variable X.

$$P(X) = P(X|\eta) \tag{2.4}$$

Continuing with our neural model as described in Section 2.2, we will now convert this into a Bayesian framework. The goal is now to find the parameters $\mathbf{w}$, which are *likely* to have generated the outputs y.

$$P(y|x, \mathbf{w}) = \mathcal{G}(f^{\mathbf{w}}(\cdot), \sigma^2) \tag{2.5}$$

where σ^2 is a parameter used to induce model precision, through corrupting the output with variance σ^2. Providing a dataset X, Y. We try to acquire an accurate posterior distribution $P(\mathbf{w}|X, Y) = \frac{P(Y|\mathbf{w},X)P(\mathbf{w})}{P(Y|X)}$.

Notice that to get $P(Y|X)$, we have marginalised over $\mathbf{w}$, hence,

$$P(Y|X) = \int P(Y|X, \mathbf{w})P(\mathbf{w})d\mathbf{w}. \tag{2.6}$$

Eq (2.6) is a critical component of the posterior distribution, as it normalises all values to maintain fundamental properties. Eq (2.6) is often referred to as *model evidence*.

The distribution $P(\mathbf{w}|X, Y)$ contains the most *likely* parameter values $\mathbf{w}$ w.r.t. the data X, Y. We can then use this distribution in order to perform *inference* on a new dataset $\mathbf{X}^*$, through the below equation.

$$P(\mathbf{Y}^*|\mathbf{w}, \mathbf{X}^*, X, Y) = \int P(\mathbf{Y}^*|\mathbf{w}, \mathbf{X}^*)P(\mathbf{w}|X, Y)d\mathbf{w}. \tag{2.7}$$

For complicated models, such as deep neural models (with a considerable number of parameters), the marginalisation calculation $P(Y|X)$ within the posterior distribution Eq (2.7) is computationally in-feasible (intractable).

2.4 Neural Variational Inference

We are now able to introduce neural variational methods. This section will provide the theoretical background required to understand the contributions of this thesis. Following on from the major obstacle identified in Bayesian neural modelling where the true posterior $P(\mathbf{w}|X, Y)$ in Eq (2.7) is intractable, we define an approximation; a *variational* distribution $q^{\phi}(\mathbf{w})$. This method is referred to as Vartiational Bayes (VB). VB comes at the cost of confidence we obtain from our predictions when using exact inference over the posterior. Hence we can turn this into a minimisation problem w.r.t. θ using Kullback–Leibler (KL) divergence [Lindley and Kullback, 1959], which is a measure of similarity between any two distributions. The closer our approximating distribution is the better. This method is typically referred to as variational inference (VI). We refer the reader to [Ghahramani, 2015,Gal, 2016,Challis and Barber, 2013] for additional information.

2.4.1 Inference as Optimisation

In this subsection, we will give a formal introduction to the variational lower bound (aka variational objective). The variational lower bound is the objective our model learns to maximise, and this incorporates optimising the likelihood of the data. The variational objective is obtained by first starting with the formula for marginal likelihood.

$$\begin{aligned} P(x) = \int P(x,z)dz = \\ \int \tfrac{q^{\phi}(z)}{q^{\phi}(z)} P(x,z)dz \\ = \mathbb{E}_{q^{\phi}}[\tfrac{P(x,z)}{q^{\phi}(z)}] \end{aligned} \tag{2.8}$$

Using Jensen's inequality due to the logarithm's concavity, $\log(\mathbb{E}_{q^{\phi}}[(P(x))]) \geq \mathbb{E}_{q^{\phi}}[\log(P(x))]$ [Jensen, 1906].

We now have:

$$\begin{aligned} \log(P(x)) &\geq \mathbb{E}_{q^\phi}[\log(\tfrac{P(x,z)}{q^\phi(z)})] \\ &= \mathbb{E}_{q^\phi}[\log(P(x|z)) + \log(P(z)) - \log(q^\phi(z))] \\ &= \mathbb{E}_{q^\phi}[\log(P(x|z))] - KL(q^\phi(z)||P(z)) \end{aligned} \tag{2.9}$$

Where q is our variational approximation (e.g., a one layer neural network) to the true data generating probability distribution p and KL is the Kullback-–Leibler divergence [Kullback and Leibler, 1951]. It is this objective function, otherwise known as the variational lower bound $\mathcal{L}$, which we attempt to optimise. This procedure is holistically known as *variational inference*, a popular technique in Bayesian learning [Jordan et al., 1998].

We then have the Evidence Lower Bound (ELBO) $\mathcal{L}$ for all training examples, which we want to maximise.

$$\mathcal{L} \approx= \mathbb{E}_{q^\phi}[\log(P(x^n|z))] - D_{KL}(q(z)||P(z)) \tag{2.10}$$

2.4.2 Computing the Kullback–Leibler Divergence

We will now derive the explicit calculations required for the $D_{KL}(q(z)||P(z))$ divergence term in Eq 2.10, as this is required to implement the proposed models in this thesis. We start with the definition of the KL divergence,

$$D_{KL}(q^\phi(z)||P(z|x)) = \int q^\phi(z) \log \frac{q^\phi(z)}{P(z|x)} dz \tag{2.11}$$

When both the prior $p^\theta(z) = \mathcal{G}(0, \mathcal{I})$ and posterior approximation $q^\phi(z|x) = \mathcal{G}(\mu, \sigma^2\mathcal{I})$ are assumed Gaussian we can produce a closed form solution to the KL term.

$$\begin{aligned} &\int q^\phi(z) \log p^\theta(z) dz \\ &= \int \mathcal{G}(z|\mu, \sigma^2) \log \mathcal{G}(z|0, \mathcal{I}) dz \\ &= -\frac{J}{2} \log 2\pi - \frac{1}{2}(\mu^2 + \sigma^2) \end{aligned} \tag{2.12}$$

And:

$$\begin{aligned}
&\int_z q^\phi(z) \log q^\phi(z) dz \\
&= \int \mathcal{G}(z|\mu, \sigma^2) \log \mathcal{G}(z|\mu, \sigma^2) dz \\
&= -\frac{J}{2} \log 2\pi - \frac{1}{2}(1 + \log \sigma^2)
\end{aligned} \tag{2.13}$$

So,

$$\begin{aligned}
-D_{KL}(q^\phi(z)||P(z|x)) &= \int q^\phi(z) \log(p^\theta(z)) dz - \int q^\phi(z) \log q^\phi(z) dz \\
&= \tfrac{1}{2} \sum_{d=1}^{D} (1 + \log(\sigma_d^2) - (\mu_d^2) - (\sigma_d^2))
\end{aligned} \tag{2.14}$$

Where D is the dimension of the vectors σ^2 and μ.

2.4.3 Approximate Inference

Once we have approximated the posterior distribution with $q^\phi(\mathbf{w}) \approx P(\mathbf{w}|X, Y)$, we can then use this to perform approximate inference of the true posterior,

$$\begin{aligned}
P(\mathbf{Y}^*|\mathbf{X}^*, X, Y) &= \int P(\mathbf{Y}^*|\mathbf{w}, \mathbf{X}^*, X, Y) P(\mathbf{w}|X, Y)\ d\mathbf{w} \\
&\approx \int P(\mathbf{Y}^*|\mathbf{w}, \mathbf{X}^*, X, Y) q^\phi(\mathbf{w})\ d\mathbf{w} \\
&= \mathbb{E}_{q^\phi}[(P(Y^*|\mathbf{w}, \mathbf{X}^*, X, Y))]
\end{aligned} \tag{2.15}$$

In general, this can still be quite computationally expensive to compute.

2.4.4 Mean Field Variational Inference

We will now introduce a simplification in variational inference, applied later during model derivations. There is a balance when we determine the function q^ϕ we would like to use to approximation the posterior. We need one which is expressive, to approximate the posterior to a high standard, and simple enough so that we still have a tractable approximation [Bishop, 2006b]. A favourite function q^ϕ is one that is a fully factorised approximation to the posterior, otherwise referred to as a mean field approximation to the posterior. This choice of function makes the strong assumption that all latent variables are independent, which significantly simplifies derivations and speeds up the training process, which comes at a cost. It is trivial to see that this independence assumption

when made on a model with dependent latent variables, would perform worse. With the independence assumption, the model will be unable to capture the interactions between the latent variables. This idea of assuming a fully factorised approximation originates from the mean field theory of physics [Opper and Saad, 2001]. An example of a fully factorised approximation to a posterior over N latent variables is shown in Eq (2.16), where each factor over a latent variable z_i has its variational parameters θ_i.

$$q^{\phi}(z) = \prod_{i=1}^{N} q^{\theta_i}(z_i) \tag{2.16}$$

The benefits of using mean field variational inference (MFVI) is that it permits optimisation of the ELBO through an iterative updating procedure. We will now focus on methods for calculating the ELBO efficiently.

2.4.5 Stochastic Variational Inference

We will now briefly overview a technique used to scale variational inference to large datasets, as new datasets for modelling knowledge graphs can be extremely large (discussed later in detail Section 3.4). Large datasets raise computational difficulties with Bayesian methods, thus, research into scalable inference algorithms essential. We have shown VI can be used to re-frame Bayesian inference as an optimisation problem. For most models we deal with, when we use MFVI, our variational objective breaks down into a sum over variational objectives across all M individual training examples. Problems of independent objectives are known to be solved efficiently using stochastic optimisation techniques [Badrinarayanan, Kendall, and Cipolla, 2017, Robbins and Monro, 1985]. Stochastic VI equates to using a stochastic optimisation algorithm to maximise the ELBO (the variational objective function) [Hoffman, Bach, and Blei, 2010,Hoffman et al., 2013,Ghahramani and Attias, 2000,Wang, Paisley, and Blei, 2011]. The motivator for applying stochastic optimisation algorithms is that they allow VI to scale to extensive datasets. Using stochastic optimisation methods for variational objectives was proposed in [Hoffman et al., 2013,Ghahramani and Attias, 2000,Sato, 2001. Returning to the ELBO, when using MFVI, we have a decomposable loss over our M data points.

$$\hat{\mathcal{L}} = \sum_{i=1}^{M} \mathbb{E}_{q}^{\theta_i}[\log(P(x_i|z_i)) + \log(P(z_i)) - \log(q^{\theta_i}(z_i))] \tag{2.17}$$

For Eq (2.17), each iteration of our stochastic optimisation algorithm scales with M, which is extremely expensive for large M. Stochastic VI solves this scalability issue using the same techniques employed in stochastic gradient descent - mini-batches. For each iteration, we randomly sample mini-batches of size S which can then be used to estimate the ELBO, as shown in Eq (2.18).

$$\hat{\mathcal{L}} \approx \frac{M}{S} \sum_{i=1}^{S} \mathbb{E}_q^{\theta_i}[\log(P(x_i|z_i)) + \log(P(z_i)) - \log(q^{\theta_i}(z_i))] \tag{2.18}$$

2.4.6 Amortised Variational Inference

We will now introduce a family of variational models, named amortised variational models, as they help with scalability by simplifying the learning process. The simple idea of amortised VI is to use a function $f(x_i)$ to predict the optimal latent variables z_i. By doing this, we replace the local variational parameters θ_i by a function of the observations x, typically with the parameters shared across all observations. Therefore we could say inference is amortised. An example where amortised VI has been successfully applied is to Deep Gaussian Processes (DGPs) [Damianou and Lawrence, 2013]. To allow DGPs to scale to large datasets, the authors of [Dai et al., 2015] estimate the latent variables as functions of deep neural networks, known as the inference network in VI. The use of the inference network to estimate the parameters of the distribution over the latent variables, which also significantly improved convergence rate in [Dai et al., 2015]. Amortised VI became a commonly used tool in DGPs, which naturally led to the concept of the Variational Auto-Encoder [Kingma and Welling, 2013,Rezende, Mohamed, and Wierstra, 2014].

2.5 Variational Auto-encoder

We will now explore one of the commonly used frameworks for VI — amortised VI, later applied by the models proposed in this thesis. The graphical model for the VAE is shown in Fig 2.1. The VAE is a probabilistic extension of the Auto-encoder model [Vincent et al., 2008]. Variational Deep Learning can be classified into a subcategory called Amortised VI which uses a deterministic mapping function onto the latent variables distribution parameters. Thus using a neural network (deterministic) as an encoder can be classified as Amortised

VI. We will discuss some of the key components that contribute to the success of the VAE; the inference network and the re-parametrisation trick.

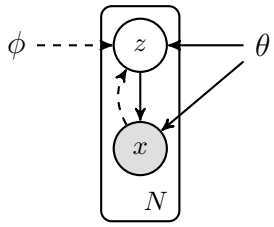

FIGURE 2.1: The original variational auto-encoder directed the graphical model. Solid lines denote the generative model $p^\theta(z)p^\theta(x|z)$, dashed lines denote the variational approximation $q^\phi(z|x)$ to the intractable posterior $p^\theta(z|x)$. The variational parameters θ are learned jointly with the generative model parameters ϕ.

Returning to our variational objective: Evidence Lower Bound $\mathbb{E}_{q^\phi}[\log(P(x|z))] - D_{KL}(q^\phi(z)||P(z))$. Using the VAE framework, we can define $P(x|z)$ to be the reconstruction loss (decoder), and define $q(z|x)$ to be the encoder from our data to the distribution parameters. The first VAE [Kingma and Welling, 2013] assumes the data was generated conditioned on a set of latent variables, i.e. the data is generated by a Gaussian distribution, with unknown parameters. So how exactly do we then estimate these unknown parameters?

2.5.1 Estimating Distribution Parameters

The first key concept in variational deep learning is that we parametrise a probability distribution with a neural network. We will denote a neural network, parametrised by $\mathbf{w}$ as $f^{\mathbf{w}}(\cdot)$. Hence we can approximate the parameters for a normal distribution $\mathcal{G}(\mu, \sigma^2)$, through $\mu \approx f^{\mathbf{w}1}(\cdot)$ and $\sigma^2 \approx \exp(f^{\mathbf{w}2}(\cdot))$.

2.5.2 Re-parametrisation trick

The second key concept in variational deep learning is the application of the *Stochastic Variational Bayes Estimator* (SVBE). SVBE allows us to train the parameters for a probability distribution through stochastic gradient descent, using a *re-parametrisation trick*. If our variational distribution $q^\phi = \mathcal{G}(\mu, \sigma^2)$ is assumed to be Gaussian, then $\theta = \mu + \hat{\epsilon} \cdot \sigma$, otherwise known as the re-parametrisation trick, where $\hat{\epsilon} \sim \mathcal{G}(0, 1)$. With this re-parametrisation trick, we can then compute the gradients required for our distribution using Eq (2.19) and Eq (2.20).

$$\frac{\partial \hat{\mathcal{L}}}{\partial \mu} \approx \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \tag{2.19}$$

$$\frac{\partial \hat{\mathcal{L}}}{\partial \sigma} \approx \hat{\epsilon} \cdot \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \tag{2.20}$$

This is known to permit unbiased and low-variance approximations of the model parameters' gradients. The current framework is most suited for training continuous latent variables. However, if the latent variables are discrete, other options exist such as using a modified REINFORCE algorithm [Vilnis and McCallum, 2014].

VAE Improvements

The typical VAE makes firm assumptions regarding inference on the posterior distribution. The assumptions made are that the posterior is approximately factorial, as well as the assumption that we can approximate these parameters through logistic regression (in the classification case). Thus several improvements have been made to VAE's, notably the Importance Weighted VAE (IWAE) [Burda, Grosse, and Salakhutdinov, 2015] was proposed to avoid these strong assumptions. In the IWAE's recognition network (encoder), the model uses multiple samples to approximate the conditional distribution parameters for the generation network. This method was proven to reproduce MNIST [LeCun and Cortes, 2010] images to a much higher accuracy than the original VAE when using Maximum Mean Discrepancy (MMD) as the loss measure.

2.6 Latent Regularisation

We will now introduce a regularisation method that can be applied to latent models [Xie, Deng, and Xing, 2015], we believe could benefit the work of this thesis. Latent regularisation is used to try and force a latent variable, say $\mathbf{A}$, to be as independent as possible, this often leads to better generalisation, as with normal regularisation methods on neural network weights, as described in Section 2.2. One recent technique to regularise latent variables is the Mutual Angular Regularisation [Xie, Deng, and Xing, 2015] (Ξ), as shown in Eq (2.21). This technique has shown to improve performance in neural variational models on a range of natural language tasks [Miao, Grefenstette, and Blunsom, 2017].

First, we define the cosine distance between two relation vectors $a(t_i, t_j) = arccos(\frac{|t_i \cdot t_j|}{||t_i|| \cdot ||t_j||})$. The mean angle of all pairs of N relations is $\xi = \frac{1}{N^2} \sum_i \sum_j a(t_i, t_j)$ with the variance defined as $V = \frac{1}{N^2} \sum_i \sum_j (a(t_i, t_j) - m)^2$. We can then optimise a hyper-parameter λ which controls how varying the angles need to be for each latent variable vector.

$$\Xi(\mathbf{A}) = \lambda(\xi - V) \tag{2.21}$$

2.7 Active Learning

One of the potential benefits that can be gained from representing a knowledge graph in a probabilistic setting is the ability to use active learning [Tong, 2001,Kapoor et al., 2007] framework; this serves the purpose of identifying and reducing model uncertainty throughout the training procedure. Modelling a knowledge graph using probability distributions could enable us to implement Bayesian active learning methods similar to [Gal, Islam, and Ghahramani, 2017]. We first introduce importance sampling, so that it can be used in the description of active learning.

Importance Sampling

Importance sampling requires that you draw a list of indices from an alternative distribution q(c). This produces a list of indices with potential overlap as you sample with replacement according to your sampling distribution q(C). With our list of indices $idx = i_1, \cdot, i_N$ we can form an approximation to Z,

$$Z \approx \frac{1}{N} \sum_{n \in idx} \frac{z_s}{q(s)} \tag{2.22}$$

We have obtained an unbiased estimator of Z.

Active Learning

We first define an activation function for active learning, which is the enabler in active learning. Let M denote the model in which we desire to enhance with active learning and $\mathcal{D}$ our Dataset, with inputs defined as $x \in \mathcal{D}$. We define the

acquisition function $a(M, \mathcal{D})$ as a function aids the active learning system with deciding the input $x*$ to acquire, to reduce model uncertainty maximally after $x*$ has been processed through the learning system of model M. This can be formulated as 2.7;

$$x* = argmax_{x \in \mathcal{D}} \; a(M, x) \tag{2.23}$$

The acquisition function that seemed most successful in high dimensional active learning from [Gal, Islam, and Ghahramani, 2017], was the Variation Ratios [Freeman, 1965] function 2.7.

$$\text{Variation Ratio}(x) = 1 - max_y P(y|x, \mathcal{D}_{train}) \tag{2.24}$$

During situations where the uncertainty estimates around points are unavailable, on can tend to other methods for importance sampling, which usually involve calculating the gradient norm of each sample [Zhao and Zhang, 2015], a computationally expensive process. The gradient norm is computed to try to minimise the variance of the gradient norms, for a more robust learning procedure. Recently it has been proposed to approximate the sample probabilities via an LSTM, using a detailed history of loss's [Katharopoulos and Fleuret, 2017].

2.8 Example: Variational Matrix Factorisation

We will now discuss how MFVI (Section 2.4.4) is applied to matrix factorisation [Lim, 2007], to derive the variational lower bound, as this shares similarity with derivations in Chapter 6. Overall we would like $X \approx ER$. In [Lim, 2007] this is used in a collaborative filtering setting, where we wish to factorise a matrix of user & movie ratings. This factorisation is then applied to better recommend new movies to users, based on similarity patterns. $E \in \mathcal{R}^{IxN}$ represents the users latent space, $R \in \mathcal{R}^{JxN}$ is a matrix defining the interactions of the latent components. We denote N as the latent dimension, I the number of users and J the number of movies. Using the squared loss, Eq (2.25), we can phrase the problem as one minimising the cost function.

$$f(E_i, R_j) = \sum_{d=1}^{N} (E_{id} R_{jd} - X_{i,j})^2 \tag{2.25}$$

We then have our density,

$$X_{i,j} \sim \mathcal{G}(< E_i, R_j >, \tau^2) \tag{2.26}$$

Where τ^2 is the variance around the observation noise of mean $< E_i, R_j >$. Here the probability of an obervation $\Theta(X_{i,j})$ is given by a probability density function (PDF).

$$\begin{aligned} \Theta(X_{i,j}) = p^\theta(X_{i,j} = 1 | E, R) &= \mathcal{G}(X_{i,j}; < E_i R_j >, \tau^2) \\ &= \frac{1}{\sqrt{2\pi\tau^2}} exp(-\frac{1}{2} \frac{(X_{i,j} - < E_i R_j >)^2}{\tau^2}) \end{aligned} \tag{2.27}$$

We place independent priors on each latent dimension in E and R, e.g $e_{il} \sim \mathcal{G}(0, \sigma_l^2)$ and $p_{jl} \sim \mathcal{G}(0, \rho_l^2)$. This gives us the PDF for E and R below.

$$\begin{aligned} p^\theta(E) &= \prod_{i=1}^{I} \prod_{l=1}^{N} \frac{1}{\sqrt{2\pi\sigma_l^2}} exp(-\frac{1}{2} \frac{\mu_{il}^2}{\sigma_l^2}) \\ p^\theta(R) &= \prod_{j=1}^{J} \prod_{l=1}^{N} \frac{1}{\sqrt{2\pi\rho_l^2}} exp(-\frac{1}{2} \frac{v_{jl}^2}{\rho_l^2}) \end{aligned} \tag{2.28}$$

Where the prior variances ρ_l^2 and σ_l^2 are column dependent. To approximate the posterior, we can use Bayes rule. However, this is computationally expensive.

$$p^\theta(E, R | X) = \frac{p^\theta(X | E, R) p^\theta(E) p^\theta(R)}{p^\theta(X)} \tag{2.29}$$

We can then derive the Evidence Lower Bound (ELBO) Eq (2.30).

$$\begin{aligned} &\mathbb{E}_{q^\phi}[\log p^\theta(X|E,R) + \log p^\theta(E) + \log p^\theta(R) - \log q^\phi(E) - \log q^\phi(R)] \\ &= \mathcal{F}(q^\phi(E) q^\phi(R)) \end{aligned} \tag{2.30}$$

In order to maximise $\mathcal{F}(q^\phi(E)q^\phi(R))$, we need to optimise one, keeping the other fixed. We continue to do this until we have converged to a solution. To maximise $q^\phi(E)$ with respect to $q^\phi(R)$ being fixed, we must differentiate $\frac{\partial \mathcal{F}(q^\phi(E)q^\phi(R))}{\partial q^\phi(E)}$ and solve for $q^\phi(E)$. Similarly for $q^\phi(R)$.

Pros: This method, unlike the MAP point estimates, accounts for empirical uncertainty observed in the data. **Cons:** Computationally expensive for large matrices, as we need to calculate the inverse of a large matrix which has time complexity $\mathcal{O}(N^3)$ in the number of dimensions N. Computing the inverse of a matrix also requires we store the full users/ rating matrix in memory, thus we have space complexity $\mathcal{O}(N^2)$ for a square $N \times N$ matrix.

2.9 Chapter Conclusion

Section 2.4, Section 2.4.4 and Section 2.4.6 have described one of the core methods used in this thesis such as neural VI, mean field VI and amortised VI. Section 2.5 highlighted recent advances within the area of neural VI and Section 2.4.5 describes stochastic VI techniques used to combat the scalability problems encountered when applying VI to large datasets. Section 2.6 and Section 2.7 highlight general methods which could be used to improve the performance of a probabilistic system. Section 2.8 details an application of VI to matrix factorisation, a problem we will later observe shares many similarities with knowledge graph completion.

Chapter 3

Knowledge Graphs

3.1 Outline

Section 3.2 provides a formal introduction to knowledge graphs. Section 3.3 defines the different assumptions used to create negative examples in knowledge graphs. Lastly, Section 3.4 presents data exploration of knowledge graph datasets experimented with in this thesis.

3.2 Knowledge Representation

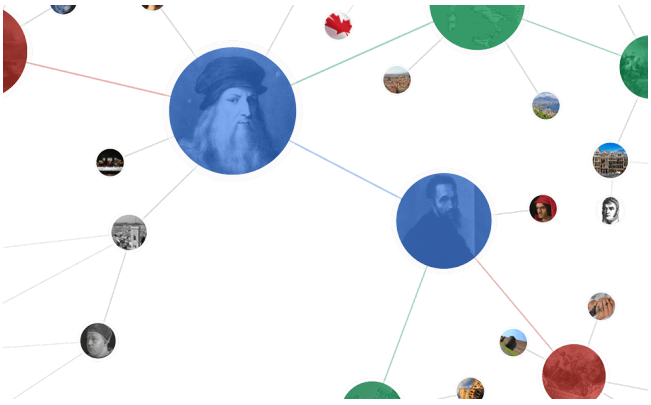

FIGURE 3.1: Google Knowledge Graph

This section will give a brief history as to what knowledge graphs have been used for, leading to their formal definition. For an extended period in logic and artificial intelligence, knowledge graphs have been a core model for information and the relationships between various types of information [Davis, Shrobe, and Szolovits, 1993], such as modelling semantics [Sowa, 1984]. In a study on the definition of a knowledge graph [Ehrlinger and Wöß, 2016], the description was identified "A knowledge graph acquires and integrates information into an

ontology and applies a reasoner to derive new knowledge." Knowledge graphs are used by Google to represent the internet, known as "Google Knowledge Graph"[1]; shown in Fig 3.1, Facebook has a knowledge graph based on social networks named the "Entity Graph"[2] and BeneloventAI have a healthcare knowledge graph to represent protein interactions[3], for a more sophisticated drug discovery search method. One of the main benefits of representing your information as a knowledge graph is the ability to perform link prediction, i.e., an attempt to conclude the entities are connected or unconnected.

Taking the example from [Nickel et al., 2015], when representing the information;

"Leonard Nimoy was an actor who played the character Spock in the science-fiction movie Star Trek."

We have multiple relationships that can be extracted, as shown in Table 3.1.

subject	predicate	object
(LeonardNimoy,	profession,	Actor)
(LeonardNimoy,	starredIn,	StarTrek)
(LeonardNimoy,	played,	Spock)
(Spock,	characterIn,	StarTrek)
(StarTrek,	genre,	ScienceFiction)

TABLE 3.1: Example: Knowledge Graph Extraction (Knowledge Base)

We can use all the subject, object, and predicate triples to represent a knowledge graph of the known information relationships from this small extract of text. We represent each of the entities as a node and a directed edge indicating a fact, with the edge pointing towards the object from the subject. For different relationships we have different types of edges in our graph, e.g., one way of representing different relationships visually would be a distinct coloured edge between entities. This construction is what is referred to as a (KG), as well as a heterogeneous information network [Sun and Han, 2012]. Depending on the KG, we may also have a hierarchy of relationships within it, i.e. Donald Trump is a professional politician, a professional politician is a type of full-time job, and a full-time job is a type of contract. The KG can also provide constraints, such as a professional politician can only have a full-time contract with the USA, so long as the politician does not have a full-time deal with Russia.

[1] https://developers.google.com/knowledge-graph/

[2] https://www.adweek.com/digital/facebook-builds-knowledge-graph-with-info-modules-on-community-pages/

[3] https://benevolent.ai/news/announcements/medical-ai-award-sets-its-sights-on-treating-parkinsons/

We will formally describe the core modules of knowledge graphs, using similar definitions to those found in [Minervini et al., 2017]. Facts are represented in the form of a binary value for truth concerning a given statement. We describe a knowledge graph as a tuple (E, R, T) of;

- A set of *entities* E, which form the nodes.
- A set of *predicates/ relations* P/R, which form the edge labels and represent the node relationships.
- A set of facts/ triples T, representing all the entity pairs known to have predicates. We show these as a tuple (subject, relation, object)/ (subject, predicate, object), otherwise stated (s,r,o) and referred to as a *fact*.

The existence of a tuple implies the existence of a fact in the knowledge base.

3.3 Open vs Closed World Assumption

This section will discuss a few of the available assumptions available to create negative examples. Observed triples will always encode existing facts, and there are many ways to interpret unknown/ non-existing facts.

- Closed World Assumption (CWA): Everything that is not known, is false. This powerful claim leads to the largest number of negatives generated under a given KG, as each non-existing triple is assigned the false binary value. For example, if we had no 'hasmarried' relationships for Donald Trump, under CWA, we would assume he is not 'hasmarried' to anyone, which we know in reality is a false assumption.
- Open World Assumption (OWA): Everything that is not known, remains unknown. Each non-existing triple is left unassigned. OWA's more cautious approach is justified in the previous example, as we do not know anything about Donald Trumps 'hasmarried' relationships, we will neither assume he is/ is not married. This assumption ties in well with the sparsity of knowledge graphs, as we can be working with some whereby less than 1% of the facts are known/ have been discovered.
- Local Closed World Assumption (LCWA): Everything that is not known remains unknown unless we have a positive permutation of that unknown, then it is assigned the false label. I.e. If we know Donald Trump is married, then we assign not 'hasmarried' between Donald Trump and

# Training Examples Per Epoch			Dataset Statistics	
Dataset	positive triples	negative triples	# Entitys	# Relations
Nations	10,686	0	55	14
Kinship	1,992	0	104	25
UMLS	6,529	0	135	46
FB15K-237	310,116	0	14,541	237
WN18RR	93,003	0	40,943	11
WN18	151,442	0	40,943	18

TABLE 3.2: Dataset Statistics

all other entities who are not Melania Trump. However, we can see the pitfalls of this assumption as this assumes the relationship 'hasmarried' is one-to-one, however, for certain entities, there is, in fact, a one-to-many relationship. In practice, this assumption works well and can generate a reasonable number of negatives.

3.4 Datasets

This section will describe in detail the datasets used, as well as complete a basic data exploration over one dataset. We evaluate our models across six public datasets to assess performance. The datasets were chosen so as to cover a range of domains of small knowledge graphs; Kindship[4], UMLS[5], Nations, and three large datasets; WN18[Bordes et al., 2013a], FB15K-237[Toutanova and Chen, 2015b], WN18RR [Dettmers et al., 2017]. Where FB15K-237 is an modified version of FB15K without the symmetric relations i.e if (e1,r,e2) is true, then (e2,r,e1) is also true. Removing symmetric relations makes the dataset significantly harder, as shown by [Dettmers et al., 2017]. Similarly, WN18RR is based on WN18 with the symmetric relationships removed. Some fundamental analysis of the size of each knowledge graph is shown in Table 3.2. Notice how we are not provided with negative examples; thus we will later create negatives using an assumption discussed in Section 6.5.

Table 3.2 displays the number of positive triples, entities and predicates in the original dataset. It will be useful to explore the models proposed in Section 6 on a variety of dataset sizes to reveal scalability issues.

[4] http://archive.ics.uci.edu/ml/datasets/Kinship
[5] https://www.nlm.nih.gov/research/umls/

jordan	indonesia	netherlands	china	brazil	usa	uk	ussr	burma	poland	india	israel	cuba	egypt
260	146	249	232	284	302	215	243	146	313	287	462	514	331

TABLE 3.3: Nations: Entity Frequency

Fig 3.3 shows each of the entities in the Nations dataset has a similar frequency, apart from Cuba and Israel, which have the highest frequency counts in the dataset, in contrast to Indonesia and Burma, with the lowest frequency counts.

Similarly, in Table 3.4 most predicates are within a range of 10–60 observations. However, NGOs (non-governmental organisations) have the highest frequency count of 95, in contrast with Commonbloc1, which has a frequency count of two.

3.5 Chapter Conclusion

Section 3.2 and Section 3.3 have introduced the fundamental components of a knowledge graph as well as how to create negative examples when they are not provided, as is typically seen in many real-world knowledge graph datasets. Section 3.4 provided analysis of two knowledge graphs; such as the frequency of entities and relations, as well as the various types of nodes and edges within a knowledge graph.

Frequency	Predicate
23	relemigrants
2	exportbooks
1	economicaid
56	accusation
28	reltreaties
8	relbooktranslations
31	treaties
97	lostterritory
54	relintergovorgs
66	officialvisits
7	relngo
9	independence
12	militaryactions
10	relstudents
141	relexports
8	reldiplomacy
5	unoffialacts
14	severdiplomatic
34	timesincewar
78	weightedunvote
84	intergovorgs3
93	attackembassy
3	intergovorgs
9	conferences
21	booktranslations
34	militaryalliance
24	exports3
68	aidenemy
92	embassy

Frequency	Predicate
16	unweightedunvote
31	tourism
19	commonbloc0
27	tourism3
87	expeldiplomats
12	warning
5	eemigrants
13	commonbloc2
42	pprotests
94	nonviolentbehavior
91	blockpositionindex
12	dependent
23	negativecomm
52	timesinceally
2	commonbloc1
14	releconomicaid
95	ngo
21	emigrants3
23	reltourism
18	students
55	duration
5	boycottembargo
46	negativebehavior
9	relexportbooks
4	violentactions
64	ngoorgs3

TABLE 3.4: Nations: Predicate Frequency

Chapter 4

Learning In Knowledge Graphs

4.1 Outline

These newer models applied to Knowledge Graphs fall into two model categories; *Latent Feature Models* (Section 4.2), and *Graph Feature Models* (Section 4.3). This chapter is centred around *learning* models under these frameworks. Section 4.4 we will discuss methods which are a *hybrid* of both Latent Feature Models and Graph Feature Models. Lastly, we will consider two cases of learning a knowledge graph model using VI; the Variational Path Ranking method (Section 4.4.2), as well as the Variational Graph Auto-encoder (Section 4.4.3). Learning to model a knowledge graph is typically referred to as link prediction, as you want to learn the links (relationships) between entities.

4.2 Latent Feature Models

In this section, we assume the binary label of an edge (relation) can be predicted solely based on latent representations of the fact, which in essence leads to a sparse adjacency matrix factorisation problem. These latent feature models models are typically popular due to their empirical success [Nickel, Tresp, and Kriegel, 2011, Bordes et al., 2013b, Socher et al., 2013, Yang et al., 2014, Nickel, Rosasco, and Poggio, 2015, Trouillon et al., 2016, Dettmers et al., 2017]. We will also introduce latent feature models which learn the parameters of distributions over latent features [He et al., 2015,Xiao, Huang, and Zhu, 2016]. We will utilise a few of these models. Recent work shows that representing the subject, relation, and object independently as vectors yields state-of-the-art results for link prediction. These methods are referred to as Latent Feature Models, and the 'vectors' are typically referred to as *embeddings*.

The task of link prediction can be rephrased into a problem where the aim is to learn a function score(s, r, o), where this is of significant value when we observe a fact, conversely this function ideally takes a small amount when we find a false fact.

We can use a neural link prediction system, where each entity and relation has an associated (unique) learned embedding. The specific task of link prediction is then to complete either the triple $(s, r, ?)$ or $(?, r, o)$. We learn these unique embeddings via a scoring function Θ, applied to leverage the similarity between these embeddings. Previous methods took a rule-based approach to the matter (manually engineered features/rules), whereas newer methods take what can be interpreted as a neural approach (learnt features /rules).

Given an observation matrix $X_j \in \mathbb{R}^{||\mathcal{E}||\times||\mathcal{E}||}$, for the j'th predicate, where each element $x_j \in X_j$ is a Bernoulli distributed random variable , with probability p_j, i.e. we have x_j is a binary random variable indicating whether a connection exists.

Ideally, we want to decompose X_j to discover latent factors, i,e. $X_j \approx ER_jE$ such that; $E \in \mathbb{R}^{\kappa\times||\mathcal{E}||}$ represents the entity latent space, $R \in \mathbb{R}^{\kappa\times||P||}$ is an asymmetric matrix defining the interactions of the latent components, $||P||$ denotes the number of predicates, κ represents the latent dimension and $||\mathcal{E}||$ the number of entities and R_j refers to the j'th column of R.

Using the above factorisation and given a triple $(i, j, k) \in \mathcal{D}ataset$, otherwise known as a fact, we can approximate the binary value of subject i being connected through predicate j to object k by $x_{i,j,k} \approx \Theta(\text{score}(E_iR_jE_k^T))$.

Where Θ is the logistic inverse link function, given by

$$\Theta(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

4.2.1 Scoring Functions

ConvE function [Dettmers et al., 2017].

$$ConvE(e_s, e_r, e_o) = f(vec(f([\hat{e}_s, \hat{e}_r] * w))W)e_o \tag{4.2}$$

Where $e_r \in \mathbb{R}^K$, $\hat{e}_s, \hat{e}_r$ are the 2D reshaping of e_s, e_r, $*$ denotes the convolution operator and f denotes a MLP layer. This has space complexity $\mathcal{O}(K)$.

Pros: Convolution operation can summarise multiple interactions, including those between nearby dimensions well. **Cons:** Less interpretable embeddings when performing cluster analysis.

DistMult function [Yang et al., 2014].

$$DistMult(e_s, e_r, e_o) =< e_s, e_r, e_o > \tag{4.3}$$

Where $e_r \in \mathbb{R}^K$, with time complexity $\mathcal{O}(K)$ and space complexity $\mathcal{O}(K)$.

Pros: Generally well performing across tasks, and fast to compute. **Cons:** Struggles with modelling asymmetric relationships.

ComplEx function [Trouillon et al., 2016].

$$\begin{aligned} ComplEx(e_s, e_r, e_o) &= Re(< e_s, e_r, e_o >) \\ =< Re(e_s), Re(e_r), Re(e_o) > &+ < Im(e_s), Re(e_r), Im(e_o) > \\ + < Re(e_s), Im(e_r), Im(e_o) > &- < Im(e_s), Im(e_r), Re(e_o) > \end{aligned} \tag{4.4}$$

Where $e_r \in \mathcal{C}^K$ is a complex vector, with time complexity $\mathcal{O}(K)$ and space complexity $\mathcal{O}(K)$. This function was motivated by the lack of DisMults ability to model anti-symmetric predicate relationships.

Pros: Great performance, especially when recovering non-symmetric adjacency matrices. **Cons:** Requires twice the number of embeddings as the DistMult.

RESCAL function [Nickel, Tresp, and Kriegel, 2011].

$$RESCAL(e_s, e_r, e_o) = e_s^T e_r e_o \tag{4.5}$$

$e_r \in \mathbb{R}^{K^2}$. With time complexity $\mathcal{O}(K^2)$ and space complexity $\mathcal{O}(K^2)$.

Pros: Symmetric similarity measure between relation and object. Can capture multiplicative interactions between two entity vectors. **Cons:** Slow inference.

HolE function [Nickel, Rosasco, and Poggio, 2015].

$$HolE(e_s, e_r, e_o) = e_r^T(\mathcal{F}^{-1}[\mathcal{F}[e_s] \odot \mathcal{F}[e_o]]) \tag{4.6}$$

Where $\odot$ is the element-wise product between two vectors, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier and Inverse Fourier transform. $e_r \in \mathbb{R}^K$, time complexity $\mathcal{O}(K \log K)$ and space complexity $\mathcal{O}(K)$.

Pros: Using correlation as the main operator it can capture complex relationships. **Cons:** Time complexity can be a problem for extremely large KG's.

NTN function [Socher et al., 2013].

$$NTN(e_s, e_r, e_o) = u_r^T f(e_s W_r^{[1...D]} e_o + V_r \begin{bmatrix} e_s \\ e_0 \end{bmatrix} + b_r) \tag{4.7}$$

$W_r \in \mathbb{R}^{K^2 D}$, $b_R \in \mathbb{R}^K$, $V_R \in \mathbb{R}^{2KD}$, $u_r \in \mathbb{R}^K$. Where D is an additional latent dimension of the NTN model. With time complexity $\mathcal{O}(K^2 D)$ and space complexity $\mathcal{O}(K^2 D)$.

Pros: A generalisation of the RESCAL approach. **Cons:** Many parameters leading to slow inference.

TransE function [Bordes et al., 2013b].

$$TransE(e_s, e_r, e_o) = ||(e_s + e_r) - e_o||_r \tag{4.8}$$

$e_r \in \mathbb{R}^K$. With time complexity $\mathcal{O}(K)$ and space complexity $\mathcal{O}(K)$.

Pros: Fast to compute, simplifies the embedding space decomposition. **Cons:** Asymmetric similarity measure between object and relation.

The above scoring functions are not designed to measure the similarity between distributions. However, there has been work creating score functions that take into consideration the full distribution, such as the energy function proposed in [Vilnis and McCallum, 2014], more generally known as potential functions [Aizerman, Braverman, and Rozonoer, 1964]. Potential functions measure the energy difference between two multivariate Gaussian distributions. A potential function used between two Gaussian distributions is given in Eq (4.9).

Potential between two Gaussian distributions(used for word embedding) [Vilnis and McCallum, 2014]

$$KL(\mathcal{N}_|||\mathcal{N}_\rangle) = \int_{x\in\mathbb{R}^n} \mathcal{G}(x;\mu_i \sum_i w) \log \frac{\mathcal{G}(x;\mu_j \sum_j)}{\mathcal{G}(x;\mu_i \sum_i)} dx \tag{4.9}$$

The authors of He et al., 2015 introduce two knowledge graph specific potential functions measuring the similarity between graph embeddings for link prediction.

KG2E_KL function [He et al., 2015].

$$\begin{gathered} KG2E\_KL(\Sigma_s, \Sigma_r, \Sigma_o, \mu_s, \mu_r, \mu_o) \\ = \frac{1}{2} tr\{(\Sigma_r^{-1}(\Sigma_s + \Sigma_o)) + \mu^T \Sigma_r^{-1} \mu - \log \frac{det(\Sigma_s+\Sigma_o)}{det(\Sigma_r)}\} \end{gathered} \tag{4.10}$$

$\mu = \mu_s - \mu_o - \mu_r$. With time complexity $\mathcal{O}(K)$ and space complexity $\mathcal{O}(K)$.

Pros: Produces a cheap computation for comparison between three distributions. **Cons:** Uses the KL divergence as a measure of similarity between the entity and the relation distribution, which is asymmetric.

KG2E_EL function [He et al., 2015].

$$KG2E\_EL(\Sigma_s, \Sigma_r, \Sigma_o, \mu_s, \mu_r, \mu_o) = \frac{1}{2}\{\mu^T \Sigma^{-1} \mu + \log det\Sigma\} \tag{4.11}$$

$\Sigma = \Sigma_s + \Sigma_r + \Sigma_o,\ \mu = \mu_s - \mu_o - \mu_r$. With time complexity $\mathcal{O}(K)$ and space complexity $\mathcal{O}(K)$.

Pros: Symmetric divergence metric. **Cons:** Performance strictly worse than KG2E_KL.

4.3 Graph Feature Models

This section introduces the alternative method used in modelling knowledge graphs: graph feature models. Graph feature models assume that the binary label of an edge can be predicted through a function of existing edge relationships in the graph. This assumption leads to a new class of link prediction models and is justified by the following example: the parents of a person are often married, so we could produce a strong estimate of (Donald,marriedTo, Ivana)

from the observation of the path $Donald \xrightarrow{ParentOf} IvankaTrump \xleftarrow{ParentOf} IvanaTrump$, as this represents a common child between the parents. In contrast to matrix factorisation models, this rule-based decision is completely interpretable, making graph feature models an attractive architecture choice for certain problem domains. The problem of graph feature models is rephrased as relational inference $(e_1, ?, e_2)$, whereas link prediction in the adjacency matrix factorisation model is working on entity inference $(e_1, r, ?)$.

Uni-relational

For uni-relational data, we can look at local similarity indices, such as common neighbours. The notion of common neighbours is a computationally low intensive index, thus scale desirably for large-scale KGs. Uni-relational graphs tend to utilise global similarity features such as Katz index [Katz, 1953], which derive the similarity of entities from the combination of all paths between nodes, or local features which measure similarity based on random probabilistic walks. Global similarities usually provide improved results over local similarities. However, this comes with an increased computational cost.

Multi-relational

For multi-relational graph feature models, we can choose to use rule mining and inductive logic programming (ILP) methods. These methods using mining techniques to discover new rules, which are then applied to the knowledge graph to extract a further new selection of rules. For example, ALEPH is an ILP algorithm that attempts to learn logical rules from relational data by inverting the known entailment statements [Muggleton, 1995]. ALEPH is also able to cope with the OWA on knowledge graphs, and it also scales desirably compared to other rule-based systems. The distinct advantage of these multi-relational rule-based systems is the interpretability, however, usually rules over entities only explain a small proportion of the natural patterns within the system.

4.3.1 Path Ranking Algorithm

Path ranking algorithms employ the idea of using constrained maximal length random walks for predicting relational links in knowledge graphs [Lao and Cohen, 2010, Lao, Mitchell, and Cohen, 2011]. If we define a path of length L

as $\pi_L(i,k,j,t)$, which denotes the sequence $e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} e_3 \cdots \xrightarrow{r_L} e_j$, where t is a sequence of relations /edges traversed $t = (r_1, r_2, \cdots, r_L)$. We set a constraint such that there needs to exist an edge of relation k joining entity i, j $e_i \xleftarrow{r_k} e_k$. For small datasets, we can construct the set of all possible paths over all path types $\prod_L(i,j,k)$. When the datasets are extremely large, we can combat this through sampling methods. From enumerating the paths, we can compute a probability density function (PDF) over paths. We are able to compute this PDF under the assumption that we follow an outgoing link uniformly at random, per each step of traversing the knowledge graph. Let $P(\pi_L(i,j,k,t))$ be the probability of a path, we can then use these these probabilities as features, for example in a logistic regression model, for inferring new relationships between entities.

$$\begin{aligned} \phi^{PRA}_{i,j,k} &= [P(\pi) : \pi \in \textstyle\prod_L(i,j,k)] \\ f^{PRA}_{i,j,k} &= w_k^T \phi^{PRA}_{i,j,k} \end{aligned} \tag{4.12}$$

Based on the path we choose to follow with the highest likelihood, we can then easily interpret this. An example of a relational path that could be learned is $isFriends, livedWith$. The relation path implies that it is likely that person A has spoken with person B if person A is friends with person C, and person C has lived with person B. This statement can be expressed in logical form as a Horn clause:

$$(pA, hasSpokenTo, pB) \leftarrow (pA, isFriends, pC) \wedge (pC, livedWith, pB) \tag{4.13}$$

We can then use L1 regularisation to promote sparsity on values of w; this is now equivalent to rule learning.

4.4 Graph Feature and Latent Methods

Recent work has used hybrid ideas from both Graph Feature and Latent Feature models. We will introduce two of these models, the Neural Theorem Prover [Rocktäschel and Riedel, 2017] and the Graph Convolution Network.

Neural Theorem Prover

The Neural Theorem Prover (NTP) [Rocktäschel and Riedel, 2017] is both a graph feature and latent model as it learns features regarding graph relations, using latent embeddings. The NTP, inspired by Prolog's backward chaining algorithm, recursively transforms a query into subqueries via learnt rules (graph features). As NTP does this for all rules in the knowledge base, it effectively undergoes a depth-first search. There have been recent improvements to speed up this depth-first search, which only consider the most promising proofs based on an approximate nearest embedding neighbour search [Minervini et al., 2018]. **Pros:** Interpretable rules learnt. **Cons:** The search over deep proofs is very time-consuming.

4.4.1 Graph Convolution Network

We will now cover in more detail the Graph Convolution Network (GCN), as it lays the foundations for models discussed in the related work, Chapter 5. As many important real-world problems are found in the natural form of graphs, there has been research focus on generalising neural networks for graph-structured data [Duvenaud et al., 2015,Henaff, Bruna, and LeCun, 2015,Kipf and Welling, 2016,Defferrard, Bresson, and Vandergheynst, 2016]. A more recent and principled approach is using spectral graph convolutions [Defferrard, Bresson, and Vandergheynst, 2016], however these spectral operations are extremely time consuming. The authors of [Kipf and Welling, 2016] use this spectral framework while introducing simplifications enabling greatly reduced training time and improved classification performance. The GCN achieved these performance gains in part by sharing filter parameters across the whole graph structure, thus allowing the model to learning structural information across the full graph.

We will now define the typical problem setting. Given a graph $G = (\{Vertices\}, \{Edges\})$, when using a graph convolution network on G, we desire to learn a function over the adjacency and/or feature matrix $f(A, X)$. Where $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix and $X \in \mathbb{R}^{N \times D}$ is the feature matrix, N being the number of nodes and D being the number of features. When there are no available features for a given graph, we can set X equal to the diagonal matrix and allow the system to learn representations based purely on the graph structure, represented in the adjacency matrix. The output of G is a matrix Z, which represents each node $Z \in \mathbb{R}^{N \times F}$, where F is the specified number of features per node. Using

the same notation as Tipf[1], we can define the $l'th$ neural network layer as a non-linear function of the previous $l-1$ layer, as shown in Eq (4.14).

$$H^{(l)} = f(H^{(l-1)}, A)\,, \tag{4.14}$$

Where $H^0 = X$ and $H^(L) = Z$, with L being the total depth of the network. We then have much freedom in choosing the function f, which is the different component between models in the Graph Convolution Network family. We will take a simple function f to highlight the simplifications introduced which have contributed to the model's scalability and success. We define the simple function in Eq (4.15), where Θ is an arbitrary activation function.

$$f(H^{l-1}, A) = \Theta(AH^{l-1}W^{l-1}) \tag{4.15}$$

Limitations

There were two initial problems identified with the original Graph Convolution networks; (1) When a feature matrix multiplies the adjacency matrix, it accounts for all local features to a node, apart from the nodes features. (1) Was solved by simply adding the diagonal matrix to the adjacency matrix at the start, $\hat{A} = A + I$, allowing the additional property of self-loops. (2) The matrix multiplications are un-normalised, which could lead to issues such as exploding gradients and different feature scales. To solve (2), we can normalise A to enforces that we can only take a convex combination, at each layer, of each node's feature matrix H^lW^l. We can normalise $\hat{A}$ by $\hat{D}^{-1}\hat{A}$, where $\hat{D}$ is the diagonal node degree matrix of the modified adjacency matrix $\hat{A}$. In [Kipf and Welling, 2016] the authors identified that in practise, using a symmetric normalisation ($\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$) yields improved results. Having dealt with issues (1) and (2), we have produced the fundamental forward propagation rule used by the Graph Convolutional Networks in Eq (4.16).

$$f(H^l, A) = \Theta(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^lW^l) \tag{4.16}$$

The corresponding backpropagation rule is trivial to calculate through automatic differentiation, enabled through frameworks such as Tensorflow [Abadi et al.,

[1] https://tkipf.github.io/graph-convolutional-networks/

2015]. This method although quite simple has shown to be extremely powerful in featureless and feature-rich graph data [Kipf and Welling, 2016].

Pros: Easy to implement and excellent performance on graph data. **Cons:** Requires using the full observation matrix, which can be expensive in practice for large knowledge bases.

There has been recent work by researchers at an online blogging site Pinterest, on scaling these GCN's to extremely large-scale datasets with billions of connections [Ying et al., 2018]. Pinterest researchers accomplished this application using simplifications such as;

1. Working on sub-graph structures identified using random walks.
2. Enhanced parallelism in the computations and dynamic computation graphs to apply graph convolutions to the dynamically generated sub-graph structures

This proved beneficial to the PinInterest by resulting in a 30% improvement in customer engagement rates[2].

4.4.2 Variational Path Ranking Algorithm

Recent work by the authors of [Chen et al., 2018] has created a variational path ranking algorithm (VPR). VPR performed competitively to previous path ranking methods on small datasets. However, it significantly outperformed all path ranking competitors when scaled to greater datasets, such as FB15k. This method split the task into two sub-tasks: (1) path-finding and (2) path reasoning. The authors also assume there is a latent representation per entity pair, such as the set of connected paths or shared properties between the entities. These latent representations subsequently contain the underlying semantics, crucial to the relational classification of the entity pairs.

Path Finder (Prior)

They frame the pathfinder $P(L|e_s, e_o)$ as a Markov Decision Process to recursively predict actions. Where actions are an entity relation pair (e,r), this is obtained through an MLP with a softmax over the total number of outgoing edges. The

[2] https://medium.com/@Pinterest_Engineering/pinsage-a-new-graph-convolutional-[...]

authors then use Amortised Variational Inference to approximate the posterior $q(L|(e_s, e_o), r)$. The posterior has the same architecture as the prior. However, it is now aware of the relation so can adjust the decisions accordingly.

Path Reasoner (Likelihood)

The path reasoner is based on a Convolutional Neural Network. They first take a path sequence $L = \{a_1, e_1, \cdots, a_i, e_i, \cdots, a_n, e_n\}$, with a_i is the i-th immediate relation and e_i is the i-th immediate entity. They then project the entities and relations onto the embedding space, whereby they then concatenate each entity, relation pair to form a matrix of height n. The matrix is then passed through a Convolutional and Multi-Layer Perceptron network, where at the output there is a softmax distribution over all relations R.

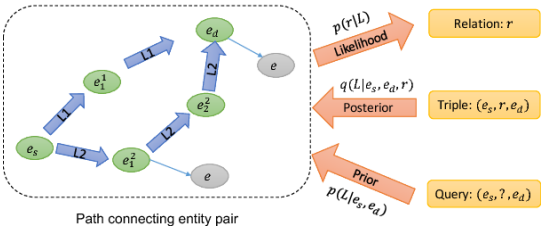

FIGURE 4.1: Variational Knowledge Graph Reasoner Graphical Model

4.4.3 Variational Graph Auto-encoder

The Variational Graph Auto-encoder [N. Kipf and Welling, 2016] is a follow up from the Convolution Graph Auto-encoder mentioned in Section 4.3. This modified version of the GCN is able to utilise a latent representation of a uni-relation undirected graph.

Given a graph $G = (\{Vertices\}, \{Edges\})$, we define $N = |Vertices|$, adjacency matrix $A \in \mathbb{R}^{N \times N}$ and diagonal node degree matrix $D \in \mathbb{R}^{N \times N}$ and a node feature matrix $X \in \mathbb{R}^{N \times D}$. The previous setup is a typical GCN. Lastly, we introduce a matrix of stochastic latent variables $Z \in \mathbb{R}^{N \times F}$, with F being the latent feature size; the new component.

Inference Model

The inference model in [N. Kipf and Welling, 2016] is parametrised by a two-layer GCN, which we will refer to by the function $GCN(\cdot)$ for simplicity.

$$q(Z|X,A) = \prod_{i=1}^{N} q(z_i|X,A), \text{ with } q(z_i|X,A) = \mathcal{G}(z_i|\mu_i, diag(\sigma_i^2)) \quad (4.17)$$

where $\mu = GCN_{mu}(X,A)$ is the matrix of mean vectors μ_i, per each example i; similarly $\log\sigma = GCN_{\sigma}(X,A)$ is the matrix of log standard deviations σ_i, for each example i. The authors then define a two, two-layer GCN by $GCN^{\mu}(X,A) = \hat{A}ReLU(\hat{A}XW_0)W_1^{\mu}$ and $GCN^{\sigma}(X,A) = \hat{A}ReLU(\hat{A}XW_0)W_1^{\sigma}$, with weight matrix W_0 being shared between GCN_{σ} and GCN_{mu}. GCN_{μ}. Where $ReLU(\cdot) = max(0,\cdot)$ and $\hat{A}$ is as previously defined $\hat{A} = \hat{D}^{-\frac{1}{2}}(A+I)\hat{D}^{-\frac{1}{2}}$ is the symmetrically normalised adjacency matrix with self loops accounted for by $A+I$.

Generative Model

The generative model in [N. Kipf and Welling, 2016] is given by the inner product between latent variables,

$$p(A,Z) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{i,j}|z_i,z_j), \text{ with } p(A_{i,j}=1|z_i,z_j) = \sigma(z_i^T, z_j) \quad (4.18)$$

Where $A_{i,j}$ are the elements of A and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

Learning

Tipf et al then optimises the variation lower bound $\mathcal{L}$ with respect to the variational parameters $W_0, W_1^{\mu}, W_1^{\sigma}$

$$\mathbb{E}_{q(Z|X,A)}[\log p(A|Z)] - KL[q(Z|X,A)||p(Z)], \quad (4.19)$$

Where $KL[q()||p(\cdot)]$ is the Killback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. The authors also take a Gaussian prior for $p(Z) = \prod_i p(z_i) = \prod_i \mathbb{N}(z_i|0,I)$

4.5 Chapter Conclusion

Section 4.2 and Section 4.3 have discussed the main methods of graph and latent feature methods, used to model a knowledge graph and perform link prediction. Section 4.4 discusses recent methods which can be seen as a hybrid between a graph and latent feature models. Section 4.4.2 and Section 4.4.3 discuss the application of variational inference in learning a few of the hybrid models.

Chapter 5

Related Work

5.1 Outline

We will first discuss related work to Variational Knowledge Graphs in Section 5.2, then summarise how our model differs from all the related approaches.

5.2 Previous Work

Motivated by the fact that previous procedures for matrix factorisation do not take into account the full posterior distribution, the authors of [Lim, 2007] decided to investigate variational matrix factorisation method, focused on collaborative filtering. In later years, authors improved this process to larger [Salakhutdinov and Mnih, 2008] and more complicated temporal tensors[Xiong et al., 2010]. Then these techniques were extended to factorise word semantic representations [Zhang et al., 2014a]. However, these techniques are not typically applied to knowledge graphs, due to the size of the tensors.

Variational Deep Learning has seen great success in areas such as parametric/non-parametric document modelling [Miao, Grefenstette, and Blunsom, 2017,Miao, Yu, and Blunsom, 2016] and image generation [Kingma and Welling, 2013]. Stochastic variational inference has been used to learn probability distributions over model weights [Blundell et al., 2015], which the authors named "Bayes By Backprop", as well as proven powerful enough to train deep belief networks [Vilnis and McCallum, 2014], by improving upon the stochastic variational bayes estimator [Kingma and Welling, 2013], using general variance reduction techniques.

Previous work has been done to re-frame word embeddings in a Bayesian framework [Zhang et al., 2014b,Vilnis and McCallum, 2014], as well as re-frane graph embeddings in a Bayesian framework [He et al., 2015]. However, these methods are expensive to train due to the evaluation of complex tensor inversions. Recent work by the authors of [Barkan, 2016,Bražinskas, Havrylov, and Titov, 2017] show that it is possible to train word embeddings through a VB [Bishop, 2006a] framework.

KG2E [He et al., 2015] proposed a probabilistic embedding method for modelling the uncertainties in KGs. However, this was not a generative model. The authors of [Xiao, Huang, and Zhu, 2016] argue they created the first generative model for knowledge graph embeddings. Firstly, this work is empirically worse than a few of the generative models built under our proposed framework. Secondly, their method is restricted to a Gaussian distribution prior, whereas we can use this, as well as any other prior that permits a re-parameterisation trick — such as the von-Mises distribution.

Later, the authors of [N. Kipf and Welling, 2016] propose a generative model for graph embeddings. However, their method lacks scalability as it requires the use of the full adjacency tensor of the graph as input. Secondly, our work differs from [N. Kipf and Welling, 2016] as they work with uni-relational data, whereas we create a framework for many variational generative models over multi-relational data.

Recent work by the authors of [Chen et al., 2018] led to successfully constructing a variational path ranking algorithm, a graph feature model. This work differs from ours for two reasons. Firstly, it does not produce a generative model for knowledge graph embeddings. Secondly, their work is a graph feature model, with the constraint of at most one relation per entity pair, whereas our model is a latent feature model with a theoretical unconstrained limit on the number of existing relationships between a given pair of entities the model can process.

In summary, our work differs from all previously proposed generative knowledge graph methods, as we propose a framework for creating a highly scalable multi-relational generative model for embeddings knowledge graphs. These models are flexible enough to learn any prior distribution over the embeddings which permits a re-parametrisation trick, as well as use any scoring function that allows maximum likelihood estimation of the parameter. Our contribution can also be viewed as creating a family potential function (see Section 4.2), which measures the approximate similarity between three distributions, using a sampling technique.

Chapter 6

Method

6.1 Outline

In this chapter, we introduce the theory behind the two proposed probabilistic models. Section 6.3 & Section 6.4 define both models and then derive their Evidence Lower Bound. Section 6.5 discusses the methods used to scale the models and reduce computation cost. Section 6.6 introduces an alternative method to linearly distributing the KL divergence across mini-batches. All methods introduced in this chapter later experimented with in Chapter 7.

6.2 Link Prediction: Revisited

Given an observation matrix $X_j \in \mathbb{R}^{||\mathcal{E}||\times||\mathcal{E}||}$, for the j'th predicate, where each element $x_j \in X_j$ is a Bernoulli distributed random variable , with probability p_j, i.e. we have x_j is a binary random variable indicating whether a connection exists.

Ideally, we want to decompose X_j to discover latent factors, i,e. $X_j \approx ER_jE$ such that; $E \in \mathbb{R}^{\kappa\times||\mathcal{E}||}$ represents the entity latent space, $R \in \mathbb{R}^{\kappa\times||P||}$ is an asymmetric matrix defining the interactions of the latent components, $||P||$ denotes the number of predicates, κ represents the latent dimension and $||\mathcal{E}||$ the number of entities and R_j refers to the j'th column of R.

Given a triple $(i, j, k) \in \mathcal{D}ataset$, otherwise known as a fact, we can approximate the binary value of subject i being connected through predicate j to object k using DistMult [Yang et al., 2014] by $x_{i,j,k} \approx< E_iR_jE_k >= E_iR_jE_k^T$.

6.3 Model A

Algorithm 1 Variational Inference for Embedding Knowledge Graphs

Model A

for $ent = 1, \ldots, ||\mathcal{E}||$:
$e_{ent} \sim \mathcal{G}(\mu_{ent}, \sigma_{ent})$
end for
for $pred = 1, \ldots, ||\mathcal{P}||$:
$r_{pred} \sim \mathcal{G}(\Omega_{pred}, \rho_{pred})$
end for
for $\{i, j, k\} \in \mathcal{D}ataset$:
$x_{i,j,k} \sim \mathcal{B}ern(\Theta(\text{score}(e_i, r_j, e_k)))$.
end for

Moving into a probabilistic setting, in this section we will derive our first generative model: Model A. First assume that the knowledge graphs are generated according to the following generative model. We place independent priors on each row vector p in E and R, e.g., $p^{\theta}(e_p) = \mathcal{G}(e_p; 0\mathcal{I})$ and $p^{\theta}(r_p) = \mathcal{G}(r_p; 0\mathcal{I})$.

Define a variational posterior distribution over each column vector p by $q^{\phi}(e_p) = \mathcal{G}(e_p; \mu_p, \sigma_p^2\mathcal{I})$ and $q^{\phi}(r_p) = \mathcal{G}(r_p; \Omega_p, \rho_p^2\mathcal{I})$. Where $\mu_p, \sigma_p, \Omega_p, \rho_p \in \mathbb{R}^n$ and $\mathcal{I} \in \mathbb{R}^{nxn}$ is the Identity matrix. Note $\mathcal{I}$ enforces the assumption that vectors have **isotropic** co-variance priors.

$$\begin{aligned} e_i &\sim \mathcal{G}(\mu_i,\, \sigma_i^2\mathcal{I}) \\ r_j &\sim \mathcal{G}(\Omega_j,\, \rho_j^2\mathcal{I}) \\ e_k &\sim \mathcal{G}(\mu_k,\, \sigma_k^2\mathcal{I}) \end{aligned} \tag{6.1}$$

Where $e_i \in \mathbb{R}^n$ as the latent subject vector, $r_j \in \mathbb{R}^n$ as the latent predicate vector and $e_k \in \mathbb{R}^n$ as the latent object vector. We can define our variational parameters ϕ by;

$$\phi = \begin{cases} (\mu_i,\, \sigma_i^2\mathcal{I}) & \in \omega_i, \\ (\Omega_j,\, \rho_j^2\mathcal{I}) & \in \omega_j, \\ (\mu_k,\, \sigma_k^2\mathcal{I}) & \in \omega_k, \end{cases} \tag{6.2}$$

$$\Theta(x) = \frac{1}{1 + e^{-x}} \tag{6.3}$$

The probability of a triple is given by the logistic inverse link function $\Theta(x)$, applied to the score acquired from the chosen score function. In this example, we will use the score function DistMult. We will then refer to this as **Variational DistMult** (Model A).

$$p^{\theta}(x_{i,j,k} = 1) = \Theta(score(e_i, r_j, < e_j)) = \Theta(< e_i, r_j, < e_j >) \tag{6.4}$$

and

$$\begin{aligned} p^{\theta}(x_{i,j,k} = 0) = \Theta(-score(e_i, r_j, e_k)) &= (1 - \Theta(score(e_i, r_j, e_k))) \\ &= (1 - \Theta(< e_i, r_j, e_k >)) \end{aligned} \tag{6.5}$$

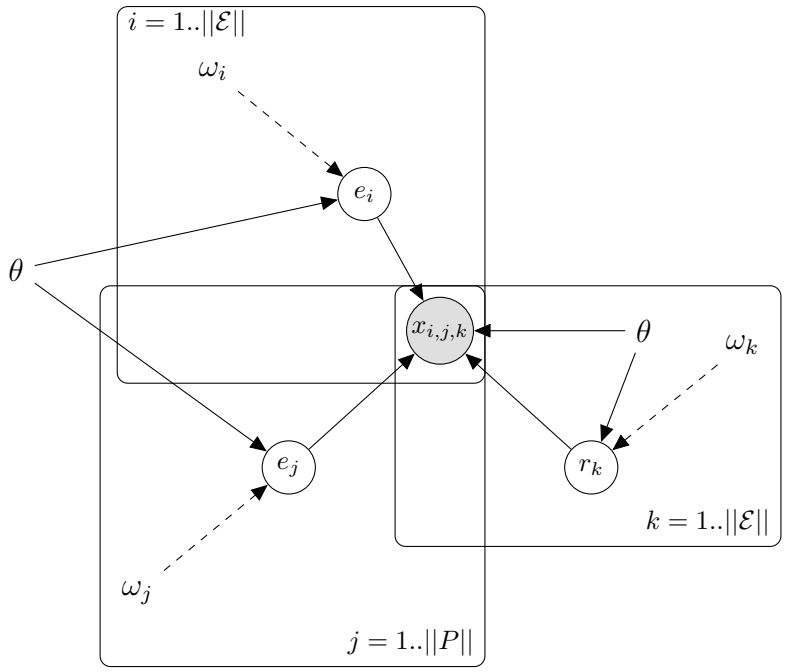

FIGURE 6.1: Model A: Graphical model for the proposed factorisation scheme. Solid lines denote the generative model $p^{\theta}(e_i)p^{\theta}(r_j)p^{\theta}(e_k)p^{\theta}(x_{i,j,k}|e_i, r_j, e_k)$, dashed lines denote the variational approximation $q^{\phi}(e_i), q^{\phi}(r_j), q^{\phi}(e_k)$ to the intractable posterior $p^{\theta}(e_i), p^{\theta}(r_j), p^{\theta}(e_k)$. Where the variational parameters $\phi = \{\omega_{subject}, \omega_{predicate}, \omega_{object}\}$ are learned.

Evidence Lower Bound Derivation

We would now like to derive the variational objective to optimise directly. To do this, we start by maximising the joint probability for Model A (Fig 6.1).

$$p^{\theta}(X, E, R) = \prod_{i,j,k} p^{\theta}(x_{i,j,k}|e_i, r_j, e_k) \prod_{i'=1}^{||\mathcal{E}||} p^{\theta}(e_{i'}) \prod_{j'=1}^{||P||} p^{\theta}(r_{j'}) \tag{6.6}$$

So we can express the marginal as

$$p^{\theta}(X) = \int \prod_{i,j,k} p^{\theta}(x_{i,j,k}|e_i, r_j, e_k) \prod_{i'=1}^{||\mathcal{E}||} p^{\theta}(e_{i'}) \prod_{j'=1}^{||P||} p^{\theta}(r_{j'}) \tag{6.7}$$

Which is the same as maximising the log-likelihood,

$$\begin{aligned} \log p^{\theta}(X) &= \log \int \prod_{i,j,k} p^{\theta}(x_{i,j,k}|e_i, r_j, e_k) \prod_{i'=1}^{||\mathcal{E}||} p^{\theta}(e_{i'}) \prod_{j'=1}^{||P||} p^{\theta}(r_{j'}) \\ &= \log \int p^{\theta}(X|E, R) p^{\theta}(E) p^{\theta}(R) dE, R \end{aligned} \tag{6.8}$$

Now multiply by our mean field (Section 2.4.4) variational distribution $1 = \frac{q^{\phi}(E,R)}{q^{\phi}(E,R)} = \frac{q^{\phi}(E)q^{\phi}(R)}{q^{\phi}(E)q^{\phi}(R)}$.

$$= \log \int p^{\theta}(X|E, R)) p^{\theta}(E) \frac{q^{\phi}(E)}{q^{\phi}(E)}) p^{\theta}(R) \frac{q^{\phi}(R)}{q^{\phi}(R)} \; dE, R \tag{6.9}$$

Using the definition of the expectation,

$$= \log \mathbb{E}_{q^{\phi}}[p^{\theta}(X|E, R)) \frac{p^{\theta}(E)}{q^{\phi}(E)} \frac{p^{\theta}(R)}{q^{\phi}(R)}] \tag{6.10}$$

Using Jensen's inequality due to the logarithm's concavity $\log(\mathbb{E}_{q^{\phi}}[\mathbf{P}(x)]) \geq \mathbb{E}_{q^{\phi}}[\log(\mathbf{P}(x))]$ [Jensen, 1906],

$$\geq \mathbb{E}_{q^{\phi}}[\log(p^{\theta}(X|E, R) \frac{p^{\theta}(E)}{q^{\phi}(E)} \frac{p^{\theta}(R)}{q^{\phi}(R)})] \tag{6.11}$$

$$= \mathbb{E}_{q^{\phi}}[\log(p^{\theta}(X|E, R)) + \log(\frac{p^{\theta}(E)}{q^{\phi}(E)}) + \log(\frac{p^{\theta}(R)}{q^{\phi}(R)})] \tag{6.12}$$

$$= \mathbb{E}_{q^{\phi}}[\log(p^{\theta}(X|E, R))] - \mathbb{E}_{q^{\phi}}[\log \frac{q^{\phi}(E)}{p^{\theta}(E)})] - \mathbb{E}_{q^{\phi}}[\log \frac{q^{\phi}(R)}{p^{\theta}(R)})] \tag{6.13}$$

$$= \mathbb{E}_{q^\phi}[\log(p^\theta(X|E,R))] - \mathbb{E}_{q^\phi}[\log(\frac{q^\phi(E)}{p^\theta(E)})] - \mathbb{E}_{q^\phi}[\log(\frac{q^\phi(R)}{p^\theta(R)})] \tag{6.14}$$

We can obtain an estimate for $\mathbb{E}_{q^\phi}[\log(p^\theta(\cdot))]$, using the reparametrisation trick allows us to estimate the expectation over our variational distribution q^ϕ via an unbiased estimator (as detailed in Section 2.5.2).

$$= \log(p^\theta(X|E,R)) - \mathbb{E}_{q^\phi}[\log(\frac{q^\phi(E)}{p^\theta(E)})] - \mathbb{E}_{q^\phi}[\log(\frac{q^\phi(R)}{p^\theta(R)})] \tag{6.15}$$

However, this induces two restrictions, as stated by Kingma et al. [Kingma and Welling, 2013].

1. Using a differentiable encoding function on the input, such as the embedding look-up function we use.
2. We are justified in only taking one sample to approximate the posterior, dependent on using a large batch size. However, if in a situation where the graph batch sizes are required to be small, an average gradient from multiple posterior samples may be required to obtain better posterior estimates.

Using the definition of KL-divergence (Eq 6.16) we obtain the loss function ($\mathcal{L}^A$) for Model A.

$$D_{KL}(q^\phi(e)\|p^\theta(e)) = \int_e q^\phi(e) \log \frac{q^\phi(e)}{p^\theta(e)} \tag{6.16}$$

$$\mathcal{L}^A = (\log p^\theta(X|E,R) - D^e_{KL}(q^\phi(E)\|p^\theta(E)) - D^r_{KL}(q^\phi(R)\|p^\theta(R)) \tag{6.17}$$

Based on our prior assumptions and the results from Section 2.4.2, we can calculate the D_{KL} term explicitly, for the entity (D^e_{KL}) and relation (D^r_{KL}) posterior distributions.

$$-D^e_{KL}(q^\phi(e_p)||p^\theta(e_p)) = \frac{1}{2}\sum_{d=1}^{D}(1 + (\log(\sigma^2_{p,d})) - (\mu_{p,d})^2 - (\sigma^2_{p,d})) \tag{6.18}$$

Similarly,

$$-D^{r}_{KL}(q^{\phi}(r_p)||p^{\theta}(r_p)) = \frac{1}{2}\sum_{d=1}^{D}(1 + \log(\rho^2_{p,d}) - (\Omega_{p,d})^2 - (\rho^2_{p,d})) \tag{6.19}$$

Returning to our loss function in Eq (6.17), we now have

$$\begin{aligned}\mathcal{L}^A = &\sum_{i,j,k}[(\log p^{\theta}(x_{i,j,k}|e_i, r_j, e_k) - \frac{1}{2}\sum_{i}\sum_{d=1}^{D}(1 + (\log(\sigma_{i,d})^2) - (\mu_{i,d})^2 - (\sigma^2_{i,d})) \\ &- \frac{1}{2}\sum_{j}\sum_{d=1}^{D}(1 + (\log(\rho_{j,d})^2) - (\Omega_{j,d})^2 - (\rho_{j,d})^2)]\end{aligned} \tag{6.20}$$

Substituting in our generative model $\log p^{\theta}(x_{i,j,k}|e_i, r_j, e_k)$=$(-\log(1 + exp(-\cdot < e_i, r_j, e_k >))$, we obtain the regularised log likelihood term.

$$\begin{aligned}\mathcal{L}^A = \sum_{i,j,k}\mathcal{ELBO}^A_{i,j,k} = &\sum_{i,j,k} x_{i,j,k}(-\log(1 + exp(-\cdot < e_i, r_j, e_k >)) \\ &(1 - x_{i,j,k})(1 - (-\log(1 + exp(-\cdot < e_i, r_j, e_k >))))+ \\ &- \frac{1}{2}\sum_{i}\sum_{d=1}^{D}(1 + (\log(\sigma^2_{i,d})) - (\mu_{i,d})^2 - (\sigma^2_{i,d})) \\ &- \frac{1}{2}\sum_{j}\sum_{d=1}^{D}(1 + (\log(\rho^2_{j,d})) - (\Omega_{j,d})^2 - (\rho^2_{j,d})\end{aligned} \tag{6.21}$$

6.4 Model B

Algorithm 2 Variational Inference for Embedding Knowledge Graphs

Model B
for $\{i, j, k\} \in \mathcal{D}ataset$:
$e_i \sim \mathcal{G}(\mu_i, \sigma_i)$
$r_j \sim \mathcal{G}(\Omega_j, \rho_j)$
$e_k \sim \mathcal{G}(\mu_k, \sigma_k)$
$h_{i,j,k} \leftarrow [e_i, r_j, e_k]$
$x_{i,j,k} \sim \mathcal{B}ern(\Theta(\text{score}(h_{i,j,k})))$.
end for

We will now purpose a second generative knowledge graph model: Model B. To derive Model B, we assume that the knowledge graphs are generated according to the following generative model. In a similar process to Section 6.3, we first place independent priors on each row vector p in E and R, e.g $p^{\theta}(e_p) = \mathcal{G}(e_p; 0\mathcal{I})$ and $p^{\theta}(r_p) = \mathcal{G}(r_p; 0\mathcal{I})$.

We then define our variational posterior distribution over each column vector p by $q^{\phi}(e_p) = \mathcal{G}(e_p; \mu_p, \sigma_p^2\mathcal{I})$ and $q^{\phi}(r_p) = \mathcal{G}(r_p; \Omega_p, \rho_p^2\mathcal{I})$. Where $\mu_p, \sigma_p, \Omega_p, \rho_p \in \mathbb{R}^n$ and $\mathcal{I} \in \mathbb{R}^{nxn}$ is the Identity matrix.

We then view each triple as a latent variable, as shown in Fig 6.2. This leads us to having one latent variable per triple, named $h_{i,j,k}$. Similarly, we can place a Gaussian prior $p^{\theta}(h_{i,j,k}) = \mathcal{G}(h_{i,j,k}; 0\mathcal{I})$ and corresponding variational posterior distribution

$$q^{\phi}(h_{i,j,k}) = \mathcal{G}(h_{i,j,k}; \begin{bmatrix} \mu_i \\ \Omega_j \\ \mu_k \end{bmatrix}, \begin{bmatrix} \sigma_i & 0 & 0 \\ 0 & \rho_j & 0 \\ 0 & 0 & \sigma_k \end{bmatrix}) \tag{6.22}$$

$$h_{i,j,k} = \begin{cases} e_i \sim \mathcal{G}(\mu_i,\, \sigma_i^2\mathcal{I}) \\ r_j \sim \mathcal{G}(\Omega_j,\, \rho_j^2\mathcal{I}) \\ e_k \sim \mathcal{G}(\mu_k,\, \sigma_k^2\mathcal{I}) \end{cases} \tag{6.23}$$

$$h_{i,j,k} \sim \mathcal{G}\left(\begin{bmatrix} \mu_i \\ \Omega_j \\ \mu_k \end{bmatrix}, \begin{bmatrix} \sigma_i & 0 & 0 \\ 0 & \rho_j & 0 \\ 0 & 0 & \sigma_k \end{bmatrix} \right) \tag{6.24}$$

The probability of a triple is similarly given by the logistic inverse link function $\Theta(x)$, of the score achieved through computing the score function of the vectors e_i, r_j, e_k, which are recovered from corresponding vector $h_{i,j,k}$. Below we show the likelihood calculation explicitly using the DistMult scoring function, for further details we suggest revisiting Section 4.2. Similarly, we will refer to this as **Variational DistMult** (Model B).

$$\begin{aligned} p^{\theta}(x_{i,j,k} = 1) = \Theta((score(h_{i,j,k})) &= \Theta(score(e_i, r_j, e_k)) \\ &= \Theta(< e_i, r_j, e_k >) \end{aligned} \tag{6.25}$$

Similarly,

$$
\begin{aligned}
p^{\theta}(x_{i,j,k} = 0) &= \Theta(-score(h_{i,j,k})) \\
&= (1 - \Theta(-score(e_i, r_j, e_k))) = (1 - \Theta(- < e_i, r_j, e_k >))
\end{aligned}
\tag{6.26}
$$

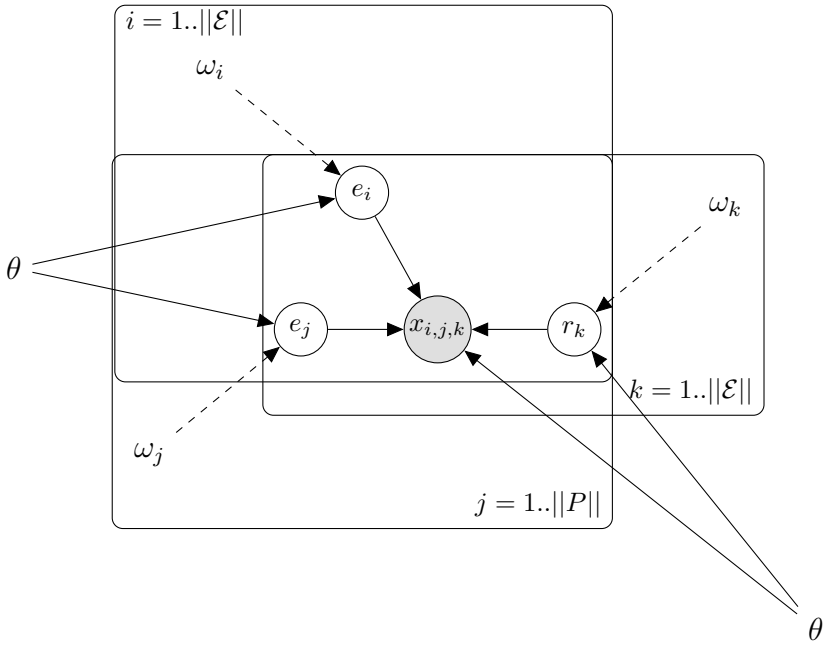

FIGURE 6.2: Model B: Graphical model for proposed factorization scheme. Solid lines denote the generative model $p^{\theta}(e_i)p^{\theta}(r_j)p^{\theta}(e_k)p^{\theta}(x_{i,j,k}|e_i, r_j, e_k)$, dashed lines denote the variational approximation $q^{\phi}(e_i), q^{\phi}(r_j), q^{\phi}(e_k)$ to the intractable posterior $p^{\theta}(e_i), p^{\theta}(r_j), p^{\theta}(e_k)$. Where the variational parameters $\phi = \{\omega_{subject}, \omega_{predicate}, \omega_{object}\}$ are learned.

Evidence Lower Bound Derivation

We will now derive the ELBO for Model B 6.2. We want to maximise the joint distribution of our model. We can express the joint distribution from the graphical model shown in Fig 6.2 as:

$$
p^{\theta}(X, H) = \prod_{i=1}^{||\mathcal{E}||} \prod_{j=1}^{||P||} \prod_{k=1}^{||\mathcal{E}||} p^{\theta}(x_{i,j,k}|h_{i,j,k}) p^{\theta}(h_{i,j,k}) \tag{6.27}
$$

Then taking the marginal we get,

$$
\begin{aligned}
p^{\theta}(X) &= \int \prod_{i=1}^{||\mathcal{E}||} \prod_{j=1}^{||P||} \prod_{k=1}^{||\mathcal{E}||} p^{\theta}(x_{i,j,k}|h_{i,j,k}) p^{\theta}(h_{i,j,k}) \\
&= \int p^{\theta}(X|H) p^{\theta}(H)\ dH
\end{aligned}
\tag{6.28}
$$

This is the same as maximising the log marginal, hence we apply the log function to both sides of the equation,

$$
\log p^{\theta}(X) = \log \int p^{\theta}(X|H) p^{\theta}(H)\ dH \tag{6.29}
$$

Now multiply by our mean field (Section 2.4.4) variational distribution $1 = \frac{q^{\phi}(H)}{q^{\phi}(H)}$, approximating the true posterior $p^{\theta}(H, X)$

$$
= \log \int p^{\theta}(X|H) p^{\theta}(H) \frac{q^{\phi}(H)}{q^{\phi}(H)}\ dH \tag{6.30}
$$

Using the definition of the expectation, with $\mathbb{E}_{q^{\phi}(H)} = \mathbb{E}_{q^{\phi}}$ for simplicity on the expectation.

$$
= \log \mathbb{E}_{q^{\phi}}[p^{\theta}(X|H) \frac{p^{\theta}(H)}{q^{\phi}(H)}] \tag{6.31}
$$

Using Jensen's inequality due to the logarithm's concavity $\log(\mathbb{E}_{q^{\phi}}[\mathbf{P}(x)]) \geq \mathbb{E}_{q^{\phi}}[\log(\mathbf{P}(x))]$ [Jensen, 1906],

$$
\geq \mathbb{E}_{q^{\phi}}[\log p^{\theta}(X|H) \frac{p^{\theta}(H)}{q^{\phi}(H)}] \tag{6.32}
$$

$$
= \mathbb{E}_{q^{\phi}}[\log p^{\theta}(X|H) + \log p^{\theta}(H) - \log q^{\phi}(H)] \tag{6.33}
$$

$$
= \mathbb{E}_{q^{\phi}}[\log p^{\theta}(X|H)] - \mathbb{E}_{q^{\phi}}[\log \frac{q^{\phi}(H)}{p^{\theta}(H)}] = \mathcal{ELBO}^{B} \tag{6.34}
$$

Which we can obtain an estimate for, using the reparametrisation trick, which allows us to estimate the expectation over our variational distribution q^{ϕ}.

$$
\approx \log p^{\theta}(X|H) - \mathbb{E}_{q^{\phi}}[\log \frac{q^{\phi}(H)}{p^{\theta}(H)}] \tag{6.35}
$$

This induces the two restrictions on the inference network and sampling scheme as previously stated in Model A's derivation (Section 6.3). Using the definition of KL-divergence we now have an alternative variational lower bound for Model B ($\mathcal{L}^B$).

$$\mathcal{L}^B = \log p^\theta(X|H) - D^B_{KL}(q^\phi(|H)\|p^\theta(H)) \tag{6.36}$$

Based on our prior assumptions, when both the prior $p^\theta(h_{i,j,k}) = \mathcal{G}(0\mathcal{I})$ and posterior approximation $q^\phi(h_{i,j,k}|x_{i,j,k})$ are assumed Gaussian, we can produce a closed form solution to the KL term (as proved earlier in Section 2.4.2).

$$\begin{aligned}
&- D^B_{KL}(q^\phi(h_{i,j,k})||p^\theta(h_{i,j,k})) \\
&= \frac{1}{2}\sum(1 + \sum(\begin{bmatrix}\log\sigma^2_{i,d}\\ \log\rho^2_{j,d}\\ \log\sigma^2_{k,d}\end{bmatrix}) - \sum\begin{bmatrix}\mu^2_{i,d}\\ \Omega^2_{j,d}\\ \mu^2_{k,d}\end{bmatrix} - \sum(\begin{bmatrix}\sigma^2_{i,d}\\ \rho^2_{j,d}\\ \sigma^2_{k,d}\end{bmatrix}) \\
&= \frac{1}{2}\sum_{d=1}^{D}(1 + \log(\sigma^2_{i,d}) + \log(\rho^2_{j,d}) + \log(\sigma^2_{k,d}) \\
&- (\mu_{i,d})^2 - (\Omega_{j,d})^2 - (\mu_{k,d})^2 - (\sigma^2_{i,d}) - (\rho^2_{j,d}) - (\sigma^2_{k,d}))
\end{aligned} \tag{6.37}$$

Returning to our loss function in Eq (6.36),

$$\mathcal{L}^B = \sum_{i,j,k}(\log p^\theta(x_{i,j,k}|h_{i,j,k}) - D^B_{KL}(q^\phi(h_{i,j,k})||p^\theta(h_{i,j,k})) \tag{6.38}$$

We now have

$$\begin{aligned}
\mathcal{L}^B = \sum_{i,j,k}[(\log p^\theta(x_{i,j,k}|h_{i,j,k}) - \frac{1}{2}\sum_{d=1}^{D}(1 + \log(\sigma^2_{i,}) + \log(\rho^2_{j,d}) + \log(\sigma^2_{k,d}) \\
- (\mu_{i,d})^2 - (\Omega_{j,d})^2 - (\mu_{k,d})^2 - (\sigma^2_{i,d}) - (\rho^2_{j,d})^2 - (\sigma^2_{k,d}))]
\end{aligned} \tag{6.39}$$

Substituting in our generative model $\log p^\theta(x_{i,j,k}|h_{i,j,k})$=$(-\log(1 + exp(-\cdot <$ $e_i, r_j, e_k >))$, we obtain the regularised log likelihood term.

$$\begin{aligned}\mathcal{L}^B = \sum_{i,j,k} \mathcal{ELBO}^B_{i,j,k} = \sum_{i,j,k} x_{i,j,k}(-\log(1+exp(-\cdot < e_i, r_j, e_k >)) \\ (1-x_{i,j,k})(1-(-\log(1+exp(-\cdot < e_i, r_j, e_k >))))+ \\ -\frac{1}{2}\sum_{d=1}^{D}(1+\log(\sigma_{i,}^2)+\log(\rho_{j,d}^2)+\log(\sigma_{k,d}^2) \\ -(\mu_{i,d})^2-(\Omega_{j,d})^2-(\mu_{k,d})^2-(\sigma_{i,d}^2)-(\rho_{j,d}^2)-(\sigma_{k,d}^2))]\end{aligned} \tag{6.40}$$

6.5 Large Scale Variational Inference

Now that we have defined both the generative models, we will now discuss two methods available to scale these graphs to large-scale problems.

The three smaller datasets contain under two million trainable triples under LCWA, as shown in Table 3.2, while the three larger datasets contain billions of triples. We attempted to train our model on the smaller datasets, and after almost a week of unfinished training, we were forced to explore faster approximate methods.

We will discuss two methods that we employed for large-scale variational inference in this chapter. A technique used to approximate the evidence lower bound across all examples, by Bernoulli sampling, and a negative sampling method.

6.5.1 Negative Sampling

It has been shown that negative sampling is an approximation to noise-contrastive estimation [Mikolov et al., 2013], which justifies the application of this method. For the classic approach of negative sampling, we will maximise our log-likelihood over all positive facts, as well minimise our log-likelihood over a subset of uniformly selected negative facts. The authors of [Mikolov et al., 2013] also explain that the optimal parameters learnt from negative sampling will generally differ from the optimal parameters learnt from using the full log-likelihood.

6.5.2 Bernoulli Sampling

We can rephrase the problem of calculating the ELBO as approximating a sum by sampling, which we can do due to the compositional nature of the

ELBO. As shown previously, computing the full ELBO overall positive and negative examples is computationally expensive; even for the smaller datasets, and unfeasible for the bigger datasets. Consider the general problem of summing over a series of elements.

$$Z = \sum_{c=1}^{C} z_c \tag{6.41}$$

We can estimate this summation using either importance sampling; as discussed in Chapter 2, or Bernoulli sampling. Bernoulli Sampling is the preferred alternative for large summations[Botev, Zheng, and Barber, 2017], as you sample without replacement, whereas importance sampling samples with replacement.

We rephrase our example sum from Eq (6.41) as;

$$Z = \sum_{c=1}^{C} z_c = \mathbb{E}_{s\sim b}[\sum_{c=1}^{C} \frac{s_c}{b_c} z_c] \tag{6.42}$$

with $s_c \sim Bernoulli(b_c)$. The positive of this sampling method is no samples are repeated. Using a single joint sample, we get an alternative approximation to our summation.

$$Z \approx \sum_{c:s_c=1} \frac{z_c}{b_c} \tag{6.43}$$

6.5.3 Estimating The Evidence Lower Bound By Bernoulli Sampling

We introduce Bernoulli ELBO sampling, an alternative justification to the typically used equation for stochastic variational inference (Section 2.4.5). The ELBO for Model A (Section 6.3) and Model B (Section 6.4) can be decomposed into two summations over the set of positive (y^+) and negative (y^-) facts.

$$\begin{aligned} p(D) &\geq \sum_{\tau\in y} \mathcal{ELBO}_\tau \\ &= \sum_{\tau^+\in y^+} \mathcal{ELBO}_{\tau^+} + \sum_{\tau^-\in y^-} \mathcal{ELBO}_{\tau^-} \\ &= \mathbb{E}_{s_\tau^+\sim b^+}[\sum_{\tau^+\in y^+} \frac{s_\tau^+}{b^+} \mathcal{ELBO}_{\tau^+}] + \mathbb{E}_{s_\tau^-\sim b^-}[\sum_{\tau^-\in y^-} \frac{s_\tau^-}{b^-} \mathcal{ELBO}_{\tau^-}] \end{aligned} \tag{6.44}$$

If we let the full dataset count be denoted as usual by $|\mathcal{D}|$, we can split this up into positive and negative facts $|\mathcal{D}| = |\mathcal{D}|^+(2(|\mathcal{E}| - 1)) + |\mathcal{D}^+| = 2|\mathcal{D}|^+|\mathcal{E}|$, Where $|\mathcal{D}^-| = |\mathcal{D}^+|(2(|\mathcal{E}| - 1))$ is the maximum number of negatives generated under LCWA. We define $|\mathcal{N}^-|$ to be the number of negatives facts sampled per positive. Now if we set $b^+ = \frac{|\mathcal{D}^+|}{|\mathcal{D}^+|} = 1$, where $|\mathcal{D}^+|$ is total count of positive facts. We also set $b^- = \frac{|\mathcal{D}^+||\mathcal{N}^-|}{|\mathcal{D}^-|}$, we get

$$\begin{aligned} p(D) &\geq \sum_{\tau \in y} \mathcal{ELBO}_\tau \\ &= \sum_{\tau^+ : s_\tau^+ = 1} \frac{\mathcal{ELBO}_{\tau^+}}{b^+} + \sum_{\tau^- : s_\tau^- = 1} \frac{\mathcal{ELBO}_{\tau^-}}{b^-} \\ &= \sum_{\tau^+ : s_\tau^+ = 1} \mathcal{ELBO}_{\tau^+} + \sum_{\tau^- : s_\tau^- = 1} \frac{|\mathcal{D}^-|}{|\mathcal{D}^+||\mathcal{N}^-|} \mathcal{ELBO}_{\tau^-} \\ &= \sum_{\tau^+} \mathcal{ELBO}_{\tau^+} + \sum_{\tau^- : s_\tau^- = 1} \frac{|\mathcal{D}^-|}{|\mathcal{D}^+||\mathcal{N}^-|} \mathcal{ELBO}_{\tau^-} \\ &= \sum_{\tau^+} \mathcal{ELBO}_{\tau^+} + \sum_{\tau^- : s_\tau^- = 1} \frac{2(|\mathcal{E}| - 1)}{|\mathcal{N}^-|} \mathcal{ELBO}_{\tau^-} \end{aligned} \tag{6.45}$$

Now if we set $|\mathcal{N}^-| = 1$, so we only generate one negative fact per positive, we get

$$p(D) \geq \sum_{\tau^+} \mathcal{ELBO}_{\tau^+} + \sum_{\tau^- : s_\tau^- = 1} 2(|\mathcal{E}| - 1) \mathcal{ELBO}_{\tau^-} \tag{6.46}$$

Eq (6.46) is an intuitive result, and this claims that if we take the same number of negatives in a dataset as the observed positive triples, we scale the negative ELBO to the expected total loss overall negative examples that we could have used under LCWA. As a result, this is a tremendous computational save, as it reduces the complexity to calculate the ELBO from the previous $\mathcal{O}(|\mathcal{E}||\mathcal{D}^+|)$ to $\mathcal{O}(|\mathcal{D}^+|)$. This yields the same result Section 2.4.5 as is typically seen in stochastic variational inference literature [Zhang et al., 2017] for estimating the log-likelihood of a model.

6.6 Mini-batch Kullback–Leibler Divergence

We will now discuss an alternative method for distributing the KL divergence across mini-batches. Returning to the KL divergence term for Model A Section 6.17, when using mini-batches we can linearly distribute the KL regularisation term. As typically seen, with the KL term over M batches equal to Eq (6.47) per each batch. Or we can non-linearly distribute the KL regularisation term across each mini-batch.

$$D_{KL} = \frac{1}{M}(D^e_{KL} + D^r_{KL}) \tag{6.47}$$

To non-linearly distribute the KL term (D_{KL}), we can adjust the KL divergence with a compression cost (π_i) at each iteration i, as recommended by the authors of [Blundell et al., 2015]. Using a non-linear KL weight $\hat{D_{KL}} = D_{KL}\hat{\pi}_i$, rather than a linear weight $\hat{D_{KL}} = \frac{D_{KL}}{\text{Batch No.}}$, is theoretically justified as the KL term is only required to be distributed across the whole batch such that $\sum \hat{\pi}_i D_{KL} = D_{KL}$. Hence we can rewrite $\hat{D^i_{KL}}$

$$\hat{D^i_{KL}} = D_{KL} \cdot \hat{\pi}_i \tag{6.48}$$

The critical feature of the compression cost proposed by [Blundell et al., 2015] is an exponentially decreasing KL term across mini-batches.

$$\pi_i = \frac{2^{M-i}}{2^M} \tag{6.49}$$

We propose an alternative whilst maintaining exponentially decreasing KL term across mini-batches. We had to create this alternative due to the 2^M causing numerical issues with large M. We first compute M exponentially decreasing numerically stable values.

$$\pi_i = \exp\left((M - 1 - i) \cdot \log(2 - M) \cdot \log 2\right) \tag{6.50}$$

Then applying normalisation so that $\sum \hat{\pi}_i = 1$

$$\hat{\pi}_i = \frac{\pi_i}{\sum \pi_i} \tag{6.51}$$

So $\pi_i = 1$ for i=0 then exponentially decays at each iteration towards 0.

Implementation Details

We refer the reader to the Appendix B for further implementation details of the models.

6.7 Chapter Conclusion

Section 6.3 and Section 6.4 introduce the two proposed models of this thesis. The reason we introduce and implement both Model A and Model B, regardless of the many similarities, was due to the difficulty in concluding which framework would yield the highest performing models. We argue that both models are trained by amortised variational inference, as described in Section 2.4.6, as having an embedding look-up function to the distribution parameters is a deterministic operation.

Notice the look-up function in this application is equivalent to having two (amortised) linear inference networks. One entity linear inference network mapping a one hot encoded subject or object vector to the distribution parameters over the entities latent embedding. The second a relation linear inference network mapping a one hot encoded relation vector to the distribution parameters over the relations latent embedding. This alternative and equivalent interpretation of the models is more apparent when looking at the architecture (Fig6.3) of Model A and Model B.

Section 6.5 introduces an alternative justification for using stochastic variational inference (Section 2.4.5) which, to the best of our knowledge, is the first SVI justification under a Bernoulli sampling framework. Lastly, Section 6.6 describes a numerically stable alternative KL divergence mini-batches weighting scheme.

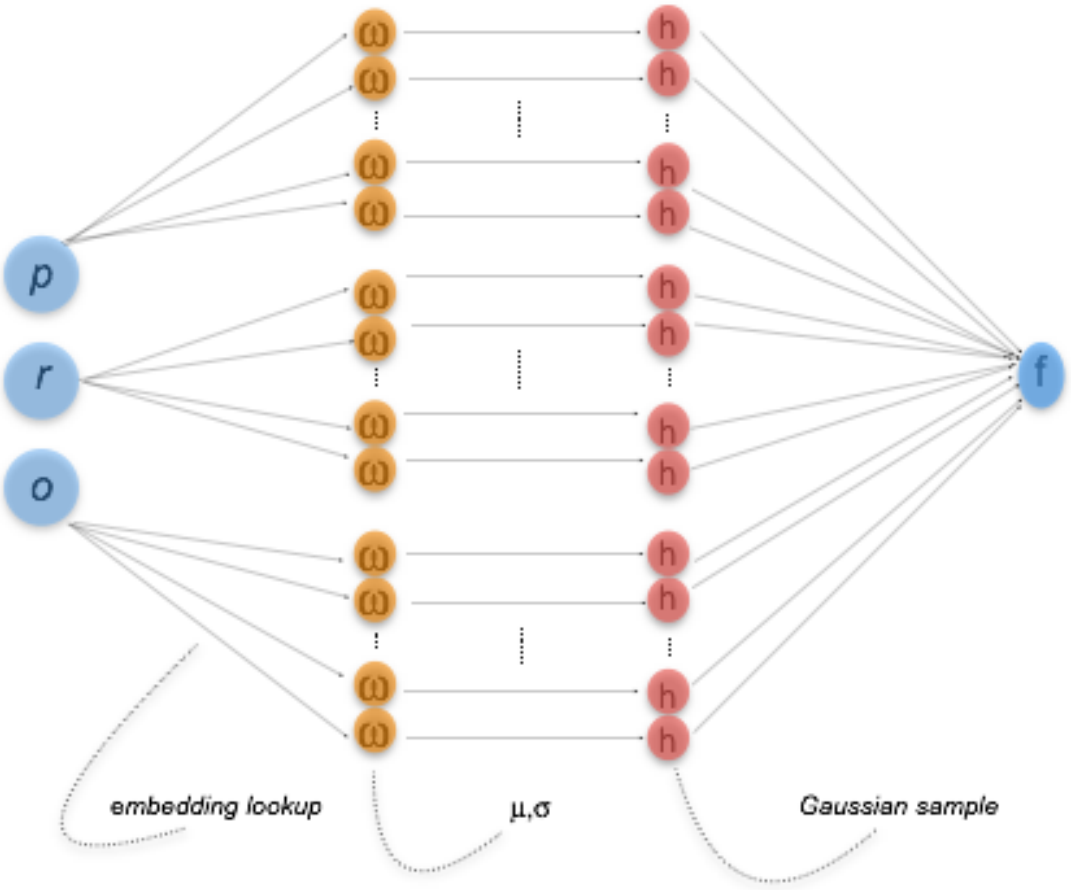

FIGURE 6.3: Model A&B: Variational Inference for Embedding Knowledge Graphs

Chapter 7

Experiments & Results

7.1 Outline

This chapter presents results experiments on Model A and Model B. Section 7.2 provides an introduction to the general experiment set up. Section 7.3 presents the results from latent feature scoring functions. Section 7.6 and Section 7.7 discuss methods used for large-scale stochastic variational inference. Section 7.8 experiments with the compression cost (re-weighting the KL regularisation term across mini-batches). Section 7.6 presents the benefits of using linear warm-up (zero out the KL term for x epochs), which leads to improved results compared with existing literature.

7.2 Experimental Set Up

Each KB dataset is separated into 80 % training facts, 10% development facts, and 10% test facts. During the evaluation, for each fact, we include every possible corrupted version of the fact under the local closed world assumption, such that the corrupted facts do not exist in the KB. Subsequently, we make a ranking prediction of every fact and its corruptions, summarised by mean rank and filtered hits@m. **Mean rank** Eq (7.1) measures the average rank correct observations (facts) have been assigned. A value of 1.0 being the top mean rank a model could theoretically achieve, and $\mathcal{E}$ (number of entities) being the worst mean rank a model could achieve. **Hits@m** measures for the amount of true subject/ objects predicted in the top m ranked predictions for each evaluated fact. **Filtered Hits@m** measures the hits@m only for the positive facts and ignores the hits@m for the negative facts. E.g. Filtered Hits@1 would be the

proportion of true test facts which were assigned the highest ranking compared with either the subject or object corrupted by all other entities.

For all experiments in Chapter 7, we have selected the best performing model on a validation set and then report the results from evaluating that model on the test set.

$$\text{mean rank} = \frac{1}{N} \sum \text{rank} \tag{7.1}$$

7.3 Non-generative Models

We first produced the non-probabilistic model results, to ensure our implementations of the scoring functions aligned with recent literature. We train using ADAM [Kingma and Ba, 2014] and initialise embedding mean vectors using Glorot's initialiser [Glorot and Bengio, 2010]. We run each experiment over 500 epochs and validate every 50 epochs. We evaluated all combinations from the following architecture options;

- **Scoring functions**: TransE, DistMult and ComplEx.
- **Datasets**: 'FB15K-237', 'Kinship', 'Nations', 'UMLS', 'WN18', 'WN18RR'.
- **Latent embedding dimension**: 200,250,300.
- **Adam epsilon decay rate** of: 10^{-1},10^{-3},10^{-5},10^{-7},10^{-9}.
- **Adam learning rate**: 10^{-1},10^{-2},10^{-3},10^{-4}.
- **Batch Sizes** : 10,100.
- **Negative Samples**: 1,5,20.

The reason we selected these scoring functions is two-fold; (i) as they are used in the later variational framework, (ii) as these include the highest performing scoring function ComplEx, as well as previous state-of-the-art scoring functions DistMult & TransE.

We looked at four different baseline architectures during the baseline optimisation stage.

1. Models trained with maximum likelihood estimation and regularised by an L2 weight decay term with lambda weight decay parameter set to 0.001, as successfully used in [Trouillon et al., 2016].

Dataset	Non-generative Models - Experiment 7.3	Mean Rank		Filtered Hits @		
		Filter	Raw	1	3	10
Kinship	TransE	18	23	0.181	0.344	0.556
	DistMult	5	12	0.379	0.596	0.869
	ComplEx	2	10	**0.776**	**0.939**	**0.988**
	ComplEx (Trouillon et al., 2016)	-	-	0.70	0.89	**0.99**
	NTPλ (*RocktäschelAndRiedel*, 2017)	**-**	-	0.76	0.82	0.89
Nations	TransE	5	8	0.246	0.458	0.970
	DistMult	2	6	0.597	0.808	0.980
	ComplEx	2	7	0.580	0.826	**0.990**
	ComplEx (Trouillon et al., 2016)	**-**	-	**0.62**	0.84	**0.99**
	NTPλ (*RocktäschelAndRiedel*, 2017)	-	-	0.59	**0.89**	**0.99**
UMLS	TransE	7	17	0.351	0.631	0.840
	DistMult	5	19	0.606	0.746	0.872
	ComplEx	1	18	**0.897**	**0.979**	**0.995**
	ComplEx (Trouillon et al., 2016)	-	-	0.82	0.96	**1.00**
	NTPλ (*RocktäschelAndRiedel*, 2017)	**-**	-	0.87	0.98	**1.00**
WN18	TransE	2088	2101	0.268	0.445	0.565
	DistMult	813	827	0.754	0.911	0.939
	ComplEx	708	725	0.936	0.944	0.946
	ComplEx (Trouillon et al., 2016)	-	-	**0.939**	0.944	**0.947**
	ConvE (Dettmers et al., 2017)	**504**	-	0.935	**0.947**	0.955
WN18 RR	TransE	3441	3454	0.139	0.251	0.413
	DistMult	8595	8595	0.367	0.390	0.412
	ComplEx	7744	7758	**0.408**	**0.459**	0.474
	ComplEx (Trouillon et al., 2016)	**5261**	-	**0.41**	**0.46**	**0.51**
	ConvE (Dettmers et al., 2017)	5277	-	0.39	0.43	0.48
FB15K -257	TransE	1386	1510	0.129	0.203	0.295
	DistMult	355	501	0.187	0.282	0.400
	ComplEx	324	483	0.195	0.294	0.420
	ComplEx (Trouillon et al., 2016)	339	-	0.159	0.258	0.417
	ConvE (Dettmers et al., 2017)	**246**	-	**0.239**	**0.350**	**0.491**

TABLE 7.1: Non-generative Model Results

2. Models trained with maximum likelihood estimation and regularised by an embedding projection onto the unit sphere or unit cube.

3. Models trained with a hinge loss, of varying margin sizes such as $1, 2, 3, 4, 5$ and regularised by a lambda weight decay term with default parameter set to 0.001.

4. Models trained with a hinge loss, of varying margin sizes such as $1, 2, 3, 4, 5$ and regularised by an embedding projection onto the unit sphere or unit cube.

In general, the results are shown in Table 7.1 highlight the dominance of ComplEX across all datasets. We were also able to improve upon previously published ComplEX; Filtered Hits@1, and Filtered Hits@3 results on Kinship and UMLS, as well as improving upon the ComplEX Filtered Hits@1 results on Nations.

7.4 Generative Model Hyper-parameters

For the purposed generative models, we searched over the same subset of hyper-parameters, scoring functions and datasets as Section 7.3. We train using ADAM [Kingma and Ba, 2014] and initialise our mean values using Glorot's initialiser [Glorot and Bengio, 2010] with our variance values initialised to $\frac{1}{\text{embedding size}}$ and We also set a random seed equal to zero for all numpy and tensorflow random operations, for reproducibility purposes. We run each experiment over 500 epochs and validate every 50 epochs.

- **Scoring functions**: Variational TransE, Variational DistMult and Variational ComplEx.
- **Datasets**: 'FB15K-237', 'Kinship', 'Nations', 'UMLS', 'WN18', 'WN18RR'.
- **Latent embedding dimension**: 200,250,300.
- **Adam epsilon decay rate** of: 10^{-3},10^{-5},10^{-7}.
- **Adam learning rate**: 10^{-1},10^{-2},10^{-3}.
- **Batch Sizes** : 1, 100 and 1000.
- **Gradient clipping by norm**: True or False.

Gradient clipping was used to limit the total variance of our gradients for more stable learning. The constraint of enforcing variance vectors to sum to one was also explored.

7.5 Maximising The Evidence Lower Bound: Full Batch

We attempted to train both Models A&B on the full batch of negative and positive examples. This lead to four days of training on the smaller datasets with no further progress than 30 epochs. This failed attempt motivated the use of large-scale variational inference methods (Section 6.5).

7.6 Estimating The Evidence Lower Bound By Bernoulli Sampling

In this experiment, we investigate the use of Bernoulli sampling for estimating the evidence lower bound for Model A&B.

7.6.1 Model A

Dataset	Model A - Experiment 7.7.1 (BS)	Mean Rank		Filtered Hits @		
		Filter	Raw	1	3	10
Kinship	Variational TransE	38	43	0.032	0.074	0.200
	Variational DistMult	5	11	0.335	0.568	0.881
	Variational ComplEx	3	9	0.576	0.808	0.961
Nations	Variational TransE	5	8	0	0.488	0.922
	Variational DistMult	2	7	0.597	0.808	0.995
	Variational ComplEx	2	7	0.52	0.776	0.993
UMLS	Variational TransE	-	-	-	-	-
	Variational DistMult	-	-	-	-	-
	Variational ComplEx	-	-	-	-	-
WN18	Variational TransE	785	786	0.415	0.866	0.877
	Variational DistMult	761	774	0.694	0.891	0.922
	Variational ComplEx	911	925	0.929	0.937	0.942
WN18 RR	Variational TransE	-	-	-	-	-
	Variational DistMult	7483	7497	0.387	0.412	0.437
	Variational ComplEx	7788	7801	0.397	0.417	0.440
FB15K -257	Variational TransE	940	1062	0.088	0.138	0.212
	Variational DistMult	1224	1367	0.160	0.247	0.355
	Variational ComplEx	1315	1455	0.170	0.259	0.371

TABLE 7.2: Model A Bernoulli Sampling ELBO Estimation

Table 7.2 shows that the Bernoulli sampling method for estimating the evidence lower bound using only one negative sample was effective. We see improvement in the DistMult function when used in conjunction with our variational framework on WN18RR, leading to improved Hits@1, Hits@3 and Hits@10 and competitive results on other datasets.

7.6.2 Model B

Table 7.3 shows poor performance across almost all datasets apart from Nations when using Bernoulli sampling to estimate the evidence lower bound. We believe this to be down to Model A's latent variables being regularised less intensely than Model B's, however it is inconclusive.

Dataset	Model B - Experiment 7.6.2 (BS)	Mean Rank		Filtered Hits @		
		Filter	Raw	1	3	10
Kinship	Variational TransE (GC + Proj)	48	53	0	0.022	0.110
	Variational DistMult (GC + Proj)	48	53	0.0178	0.044	0.103
	Variational ComplEx (GC + Proj)	47	52	0.006	0.023	0.095
Nations	Variational TransE(GC + Proj)	5	7	0	0.371	0.955
	Variational DistMult (GC + Proj)	4	7	0.153	0.503	0.950
	Variational ComplEx (GC + Proj)	5	8	0.111	0.387	0.935
UMLS	Variational TransE(GC + Proj)	60	70	0	0.021	0.049
	Variational DistMult (GC + Proj)	43	56	0.030	0.074	0.236
	Variational ComplEx (GC + Proj)	53	63	0.007	0.039	0.097
WN18	Variational TransE	15524	15528	0	0	0
	Variational DistMult	19919	19928	0	0	0.003
	Variational ComplEx	20744	20752	0	0	0.001
WN18 RR	Variational TransE	15524	15528	-	-	-
	Variational DistMult	19919	19928	0	0.001	0.003
	Variational ComplEx	20933	20941	0	0	0
FB15K -257	Variational TransE	940	1062	0.088	0.138	0.212
	Variational DistMult	6094	6215	0.007	0.010	0.019
	Variational ComplEx	7287	7405	0.003	0.003	0.004

TABLE 7.3: Model B Bernoulli Sampling ELBO Estimation

7.7 Negative Sampling

In this experiment, we investigate the use of negative sampling for Model B. We do not duplicate this test for Model A, as it is included in later experiments.

7.7.1 Model A

See Section 7.8 for the analysis of negative sampling with and without the compression cost introduced in Section 6.6.

7.7.2 Model B

Similarly to Section 7.6.2, Table 7.4 shows poor performance across almost all datasets apart from Nations when using negative sampling. We believe this to be down to Model B's latent variables being regularised heavily, which was confirmed with a simple experiment of re-running Model B on WN18 with a KL re-weighting of 1/3. This enabled Model B to learn better than it currently does, but this is still significantly worse than Model A.

Dataset	Model B - Experiment 7.7.2 (NS)	Mean Rank		Filtered Hits @		
		Filter	Raw	1	3	10
Kinship	Variational TransE	45	49	0.004	0.269	0.283
	Variational DistMult	49	54	0.001	0.003	0.104
	Variational ComplEx	48	53	0.002	0.028	0.102
Nations	Variational TransE	4	7	0.104	0.567	0.933
	Variational DistMult	4	7	0.157	0.522	0.960
	Variational ComplEx	4	7	0.048	0.450	0.970
UMLS	Variational TransE	49	57	0.129	0.390	0.406
	Variational DistMult	41	50	0.045	0.127	0.325
	Variational ComplEx	51	61	0.003	0.037	0.105
WN18	Variational TransE	16983	16991	0	0	0.001
	Variational DistMult	19919	19692	0	0	0.003
	Variational ComplEx	20300	20307	0	0	0.001
WN18 RR	Variational TransE	16983	16991	0	0	0.001
	Variational DistMult	19919	19928	0	0.001	0.003
	Variational ComplEx	20849	20857	0	0	0
FB15K -257	Variational TransE	6812	6858	0.006	0.018	0.068
	Variational DistMult	5853	5974	0.006	0.012	0.022
	Variational ComplEx	7287	7405	0.003	0.003	0.004

TABLE 7.4: Model B Negative Sampling

7.8 Mini-batch Kullback–Leibler Divergence

In this experiment the effects of negative sampling is investigated with and without the use of a compression cost (see Section 6.6 for more information regarding the compression cost). We perform architecture search over the Variational DistMult, Variational ComplEx and Variational TransE scoring functions.

7.8.1 Model A

Table 7.5 displays Model A's promising results across all datasets and all scoring functions. Highest results in this experiment were obtained, for the *smaller* datasets, when using compression cost **with** negative sampling (without Bernoulli sampling to estimate the ELBO). Generally, we find that Bernoulli sampling for estimating the ELBO (Section 7.6) is the better performing method across all scoring functions on the *larger* datasets.

Dataset	Model A Experiment 7.8	Mean Rank		Filtered Hits @		
		Filter	Raw	1	3	10
Kinship	Variational TransE	45	50	0	0.032	0.123
	Variational DistMult	5	10	0.335	0.588	0.896
	Variational ComplEx	2	9	0.668	0.876	0.975
Nations	Variational TransE	5.23	8	0	0.393	0.910
	Variational DistMult	2	6	0.600	0.833	0.990
	Variational ComplEx	2	7	0.530	0.823	0.998
UMLS	Variational TransE	15	24	0.116	0.318	0.557
	Variational DistMult	5	15	0.473	0.697	0.874
	Variational ComplEx	3	16	0.555	0.811	0.950
WN18	Variational TransE	6205	6215	0.015	0.036	0.008
	Variational DistMult	546	559	0.465	0.711	0.859
	Variational ComplEx	619	631	0.644	0.831	0.875
WN18 RR	Variational TransE	13760	13770	0.001	0.005	0.010
	Variational DistMult	6095	6108	0.267	0.361	0.421
	Variational ComplEx	5775	5786	0.322	0.387	0.427
FB15K -257	Variational TransE	2893	3016	0.067	0.121	0.196
	Variational DistMult	761	889	0.096	0.143	0.358
	Variational ComplEx	986	1110	0.151	0.244	0.362

TABLE 7.5: Model A Compression Cost

7.8.2 Model B

Model B is unable to use the compression cost as the Kullback—Leibler Divergence term is not independent of the latent variables; as shown in the derivation of Model B in Eq (6.17), thus isn't distributed across mini-batches.

7.9 Linear warm-up

7.9.1 Model A

Motivated by the success of linear warm-up in [Bowman et al., 2016,Davidson et al., 2018], we investigate the effects of linear warm-up applied to Model A (a reminder we were unable to apply linear warm-up to Model B as the KL term was not independent of the log-likelihood term, and is generated on a per example basis).

A reminder that linear warm-up allows the model to learn without the KL term (weight set to 0) for X epochs, then after X epochs the model instantly bring it

in (setting the weight to 1). For this experiment, used linear warm-up to bring the KL term into the loss function after *ten epochs*.

We performed architecture search over the top two performing scoring functions previously, thus do not conduct further experiments on Variational TransE.

Dataset	Model A Experiment 7.9.1	Mean Rank		Filtered Hits @		
		Filter	Raw	1	3	10
WN18	Variational DistMult	786	798	0.671	0.931	0.947
	Variational ComplEx	753	765	0.934	**0.945**	**0.952**
	ComplEx (Trouillon et al., 2016)	-	-	**0.939**	0.944	0.947
WN18 RR	Variational DistMult	6095	6109	0.357	0.440	0.423
	Variational ComplEx	6500	6514	0.385	0.446	0.489
	ComplEx (Trouillon et al., 2016)	**5261**	-	**0.41**	**0.46**	**0.51**
FB15K -257	Variational DistMult	679	813	0.171	0.271	0.397
	Variational ComplEx	1221	1347	0.168	0.260	0.369
	ComplEx (Trouillon et al., 2016)	339	-	**0.159**	**0.258**	**0.417**

TABLE 7.6: Model A Linear warm-up

Table 7.6 shows improved performance across all models and datasets from using a linear warm-up of ten epochs before introducing the KL divergence regularisation term, compared with just using Bernoulli sampling to estimate the ELBO for Model A as discussed in Section 7.6. We can see definite improvements on WN18 for Variational ComplEx compared with the initially published ComplEX. We believe this due to the well-balanced model regularisation induced by the zero mean unit variance Gaussian prior.

We ran this experiment over 500 epochs and observed that most models were still steadily increasing in the validation Hits@1, Hits@3, Hits@5, and Hits@10. The training also appeared more stable using Bernoulli sampling to estimate the ELBO (with linear warm-up), than using negative sampling.

7.9.2 Model B

Due to time issues, we were unable to run this experiment on Model B. It is likely that Model B would have performed considerably worse than Model A, given the previous track record across all tests conducted.

7.10 Chapter Conclusion

This chapter has shown the dominance of Model A over Model B experimentally, across multiple datasets and scoring functions. We have also improved previously published results of ComplEX [Trouillon et al., 2016] through further hyper-parameter optimisation (Section 7.3), as well as providing additional performance gains when using a Variational ComplEX model (Section 7.6), compared with published results from the ComplEX model on WN18 [Trouillon et al., 2016]. This chapter has also shown Model A & Model B's competitive performance to state-of-the-art baselines across multiple datasets, highlighting future potential.

Chapter 8

Analysis

8.1 Outline

Section 8.2 analyses two available methods for approximate inference in Model A&B. Section 8.3 analyses the predictions from Model A in further detail. Section 8.4 investigates the use of measuring output variance as a proxy for predictive uncertainty. Section 8.6 analyses the mini-batch Kullback–Leibler Divergence re-weighting. Section 8.7 visual analysis of test facts on the Nations dataset. Section 8.8 compares the purposed models to previous state-of-the-art multi-relational generative models. Lastly, as an ablation study is fundamental to understanding the significance of each component in a model, Section 8.9 conducts ablation analysis on the highest performing model (Model A), on one of the most challenging datasets (WN18RR).

8.2 Inference Methods

During inference in Model A&B, we have at least two methods that are applicable: (i) forward sampling, (ii) use the distributional centres (means) as the scoring estimates. The former is significantly more time-consuming, scaling in time complexity with the number of entities, thus less attractive for large knowledge graphs. For example, applying forward sampling for inference in the WN18 dataset: we would need to predict 40,000 entities' score per test fact, using 100 samples would then require over four million computations per test fact, whereas only 40,000 computations per test fact using method (ii). We select a model previously evaluated under the first inference scheme (with results printed on the x-axis), and plot the forward sampling predictions (using 300 samples).

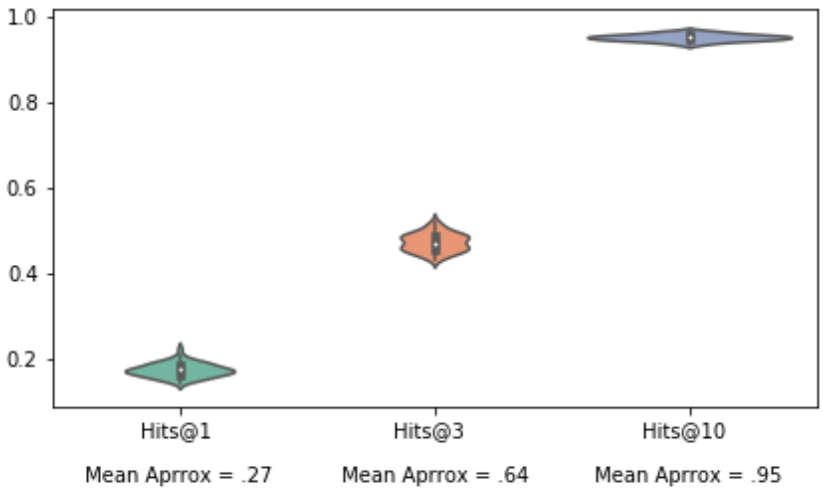

FIGURE 8.1: Forward Sampling Approximation

Figure 8.1 displays the second inference schemes reliable estimates to the first inference schemes (forward sampling). This is conjectured due to the predictions from forward sampling centring around method the second inference schemes approximate prediction for the performance metrics; Hits@3, Hits@10.

However, Hits@1 increase in performance when using the second inference scheme compared to forward sampling. The improvement in performance is most likely linked with the sensitivity of ranking. However, this is inconclusive.

As a result of the improved performance from the second inference scheme, we always chose to evaluate at test time using the mean embeddings as input into the selected scoring function.

8.3 Link Prediction Analysis

We will now explore the predictions made by Model A; with ComplEx, trained with Bernoulli sampling to estimate the ELBO, compression cost and linear warm-up on the WN18RR dataset. We split the analysis into the predictions of subject $((?, r, o))$ or object $((s, r, ?))$ for each test fact.

Note all results are filtered predictions, i.e., not including the predictions made on negative examples generated under LCWA. For both subject and object prediction results, we sort the relations in descending order from the most substantial contribution (proportion of test set) to the least.

8.3.1 Subject Prediction

	Proportion	Hits@1	Hits@3	Hits@10
_hypernym	0.399170	0.091926	0.123102	0.162270
_derivationally_related_form	0.342693	0.947858	0.956238	0.959032
_member_meronym	0.080728	0.007905	0.019763	0.035573
_has_part	0.054882	0.011628	0.058140	0.122093
_instance_hypernym	0.038928	0.393443	0.508197	0.713115
_synset_domain_topic_of	0.036375	0.219298	0.315789	0.464912
_also_see	0.017869	0.589286	0.625000	0.625000
_verb_group	0.012444	0.743590	0.974359	0.974359
_member_of_domain_region	0.008296	0.000000	0.038462	0.115385
_member_of_domain_usage	0.007658	0.000000	0.000000	0.000000
_similar_to	0.000957	1.000000	1.000000	1.000000

TABLE 8.1: WN18RR Subject Prediction

Table 8.1 shows that the relation "_derivationally_related_form", comprising 34% of test subject predictions, was the most accurate relation to predict for Hits@1 when removing the subject from the tested fact. Contrarily, "_member_of_domain_region" with zero Hits@1 subject prediction, making up less than 1% of subject test predictions. However, "_member_meronym " was the least accurate and prominent (8% of the test subject predictions) for subject Hits@1.

8.3.2 Object Prediction

	Proportion	Hits@1	Hits@3	Hits@10
_hypernym	0.399170	0.000000	0.014388	0.046363
_derivationally_related_form	0.342693	0.945996	0.957169	0.959032
_member_meronym	0.080728	0.031621	0.047431	0.086957
_has_part	0.054882	0.034884	0.081395	0.139535
_instance_hypernym	0.038928	0.024590	0.081967	0.131148
_synset_domain_topic_of	0.036375	0.035088	0.043860	0.078947
_also_see	0.017869	0.607143	0.625000	0.625000
_verb_group	0.012444	0.897436	0.974359	0.974359
_member_of_domain_region	0.008296	0.038462	0.076923	0.076923
_member_of_domain_usage	0.007658	0.000000	0.000000	0.000000
_similar_to	0.000957	1.000000	1.000000	1.000000

TABLE 8.2: WN18RR Object Prediction

Table 8.2 displays similar results to Table 8.1, as before the relation "_derivationally_related_form" was the most accurate relation to predict Hits@1. Table 8.2 differs from Table 8.1

as it highlights Model A's its inability to achieve a high Hits@1 performance predicting objects for the "_hypernym" relation, which is significantly hindering model performance as it is the most seen relation in the test set— its involvement in 40% of object test predictions.

Overall, the results suggest there is still a large margin for improvement, which we conjecture to be caused by convergence to a relatively poor local minima. However, this is inconclusive. If this were a local minima issue, attempts to combat this could be achieved through; multiple initiations of the experiment, or possibly an alternative initialisation scheme. The "_similar_to" relation performs desirably across both subject and object prediction. However, as there are only three test facts involving this relation, it is almost insignificant in contribution to the overall model performance.

8.4 Predictive Uncertainty Estimation

Motivated by the primary goals of this thesis (Section 1.1): desiring to make confident estimations, we explore two methods for confidence estimation by; taking the magnitude of the prediction as confidence, attempting to measuring the models' predictive uncertainty through output variance. This experiment was carried out on Model A; Variational DistMult, on Nations dataset, with the learning rate set to 10^{-7}, an embedding size of 200, projection True and learning rate 10^{-2}.

8.4.1 Typical Approach

For the first confidence estimation method, we interpret the magnitude of the prediction as the confidence, similarly interpreted in [Culotta and McCallum, 2004]. For this, we search over 1,000 coverage values between (0,1]. At each coverage value, we implement a threshold in which predictions outside this confidence range are discarded. We then plot these and fit a regression line of order two, to estimate the trend. The results from the typical approach are plotted on the same graph as the sampling approach, in Fig 8.2.

Algorithm 3 Model A Confidence Estimation via Output Variance

for $\{i, j, k\} \in \mathcal{D}ataset$:
$\mathrm{x}_{\mathrm{all}} = [\,]$
for n= 1 to number of samples N :
$\mathrm{e}_i \sim \mathcal{G}(\mu_i, \sigma_i)$
$r_j \sim \mathcal{G}(\Omega_j, \rho_j)$
$e_k \sim \mathcal{G}(\mu_k, \sigma_k)$
$h_{i,j,k} \leftarrow [e_i, r_j, e_k]$
$x^n_{i,j,k} \sim \mathcal{B}ern(\Theta(\mathrm{score}(h_{i,j,k})))$.
$\mathrm{x}_{\mathrm{all}}.append(x^n_{i,j,k})$
end for
$\mathrm{confidence} = \frac{\sum x_{\mathrm{all}}}{N}$
$\mathrm{Score} = \mathrm{score}(\mu_i, \Omega_j, \mu_k)$
Evaluate Score rank **if** confidence $\geq$ 1 - coverage
end for

8.4.2 Confidence Estimation via Output Variance

Then we propose a confidence estimation method that relies on measuring the variance in the output predictions. We effectively measure the decision predictions sensitivity from multiple predictions. We begin with 400 test (full test set) examples evaluated at test time when the coverage is one, reducing to zero examples evaluated when the coverage is zero. Specifically, we notice that the precision only changes once we reach coverage of around 0.5 and is at its maximum when the coverage is reduced to 0.4, and we received the results shown in Fig 8.2.

The same results across both experiments are likely due to small variance around predictions, as observed in earlier experiments on forward sampling vs mean approximation. However, we believe it is still useful to highlight this method, as it could be applicable for future variational knowledge graphs, whereby the embedding uncertainty is more significant than this example.

Based on Fig 8.2, we can see a general trend of increased accuracy with a decrease in coverage, exactly what we would desire from a model to estimate its confidence in a prediction. We also provide the relationship between the number of test examples evaluated at test time and the coverage, as well as mean rank.

We expect a similar story across other score functions and other datasets, due to similar performances across scoring functions in Chapter 7. However, this conjecture is inconclusive.

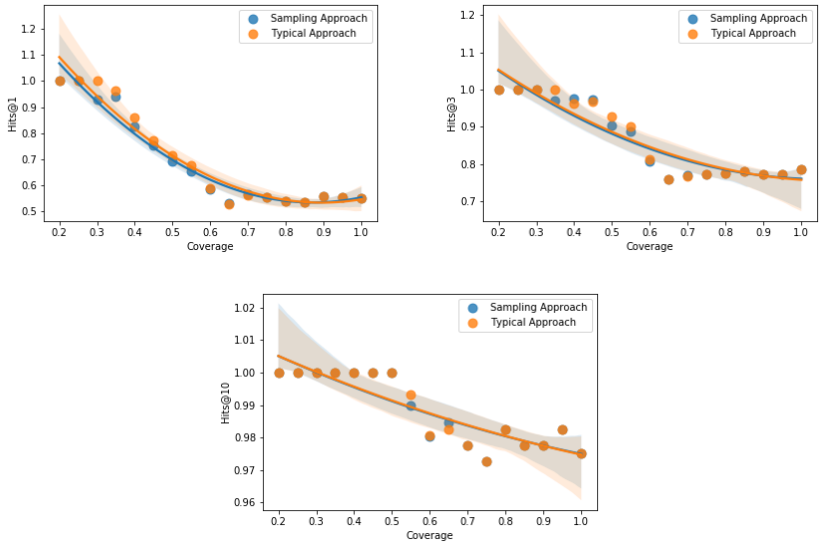

FIGURE 8.2: Precision - Coverage Relationship

8.5 Training

We will now explore the training process in further detail for the training graph of the Model A that was used in Section 8.3. It is no surprise that we observe an extremely training noisy process, as the training process is based off sampling. Fig 8.4 displays the negative log-likelihood across the positive facts over 20 epochs and 100 mini-batches.

FIGURE 8.3: Positive Fact Negative Log Likelihood

We are also able to visualise the distribution of weights learnt in the variance embeddings. It is satisfying to see that although the variance values are initialised to a constant, the system quickly alters and discovers a range of variance embeddings suitable for the task.

The histogram of learned standard deviation values for the predicate embeddings is of a similar trend to Fig 8.4, hence we choose not to include it.

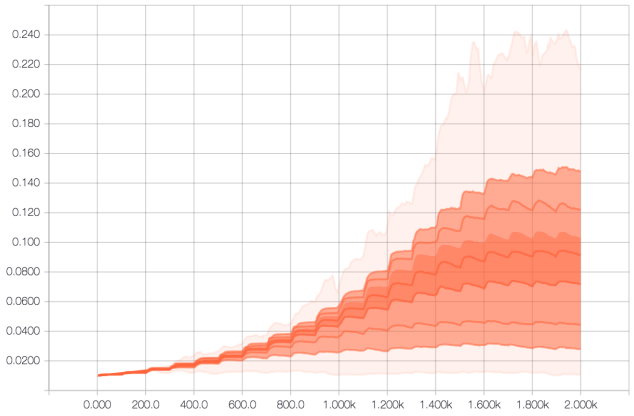

FIGURE 8.4: Smoothed Histogram of Entity Std Deviation Values

8.6 Mini-batch Kullback—Leibler Divergence Weight

We now investigate the coefficient values of the created compression cost function over batch sizes of; 5, 10, 50, and 100. We display both the KL coefficient due to the compression cost, as well as the cumulative sum of all previous KL coefficients.

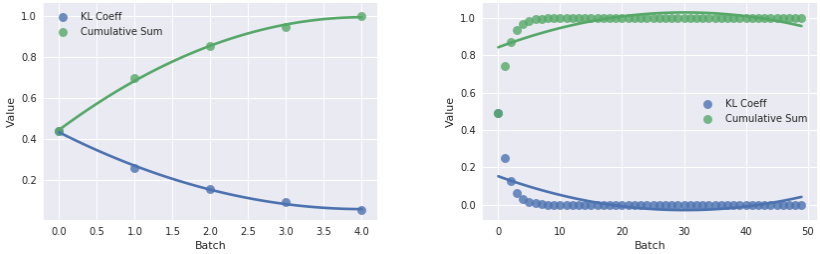

FIGURE 8.5: Compression Cost Analysis

Figure 8.5 shows that the compression cost behaves similarly across most of the batches. The behaviour begins with a substantial penalty followed by a sharp decline. However, it might be desirable to produce a dampened compression cost with an approximate exponential decrease at the beginning and an approximately exponential increase at the end of the batch: to spread information more smoothly, shown in Figure 8.6.

After analysis of the effect of the altered compression cost, we see the changes cause negligible impact in the training process. Thus we retain the original compression cost.

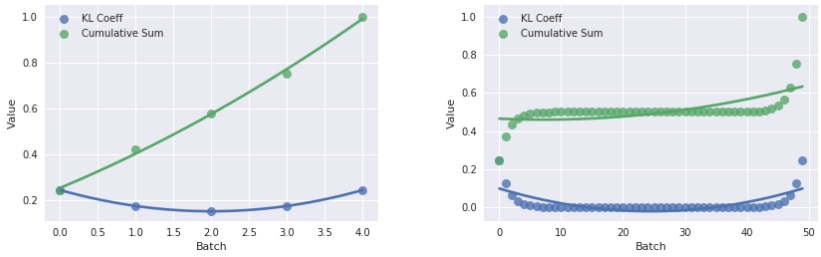

FIGURE 8.6: Modified Compression Cost

8.7 Visual Embedding Analysis

We first assess the relationship between the frequency of a node or connection within a knowledge graph, with its learnt latent representation. Second, we discuss a 2D toy experiment used during the experimental phases to visualise and perform an extrinsic evaluation on the learnt representation.

8.7.1 Variance Magnitude to Frequency Relationship

We would like to verify whether the isotropic co-variance matrices we learn are capturing information regarding the 'popularity' or frequency of a given node. We first compute the frequency of each node by doing a count across the training set whenever we see the node mentioned as either a subject or object, similarly for the relation types.

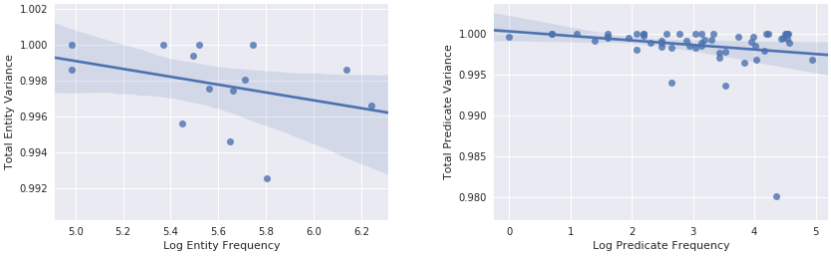

FIGURE 8.7: Diagonal Co-variance Sum vs. Log Frequency

Figure 8.7 and Table 8.3 show a weak correlation between the total weight (vector sum) of variance embedding and frequency. There is a gradual decrease in variance values as objects are seen more frequently in training, which shows how the model reduces its uncertainty over an object the more it is used: a desirable property.

Frequency	Entity	Sum Variance
249	netherlands	1
215	uk	1
146	burma	1
313	poland	1
243	ussr	0.999382
146	indonesia	0.99864
462	israel	0.998606
302	usa	0.998076
260	jordan	0.997578
287	india	0.997461
514	cuba	0.996625
232	china	0.995591
284	brazil	0.994613
331	egypt	0.992509

TABLE 8.3: Nations: Entity Variance Analysis

We now visualise the embeddings learnt from the previously used, and best performing model, Model A using Bernoulli sampling to estimate the ELBO with the compression cost and linear warm-up. We use the nations dataset, which we believe is the most trivial to interpret, lastly paired with DistMult score function.

8.7.2 2D Toy Experiment

From the preliminary 2D toy experiments (without constraining variance embeddings to unit variance) we discovered an explosion in the variance embedding values, with no constraint on their allowed values. However, we also observed that the mean embedding vectors behaved sensibly. Thus, there was no need to consider a constraint. This motivated the creation of the unit norm variance projection scheme. After some consideration, this comes as less of a surprise. For the nations dataset, we have only 25 relations and 1,992 positive triples, so the predicates obtain a more significant number of updates than the entities receive. We place images of these bloated embeddings in the appendix as additional information. We also observed from this experiment that the model could put three distributions on top of each other when they exhibit a fact, displaying the first sign that the system was learning correctly.

8.7.3 Model A Extrinsic Evaluation: Embedding Analysis

Finally, we chose to select the model which had the highest performance (Model A with Bernoulli sampling to estimate the ELBO, compression cost and linear warm-up) and then visualise the learned embeddings on the nations dataset. We take the first two dimensions of the embedding vector for each subject, object, and predicate. We also include a randomly sampled (corrupted) entity for comparison.

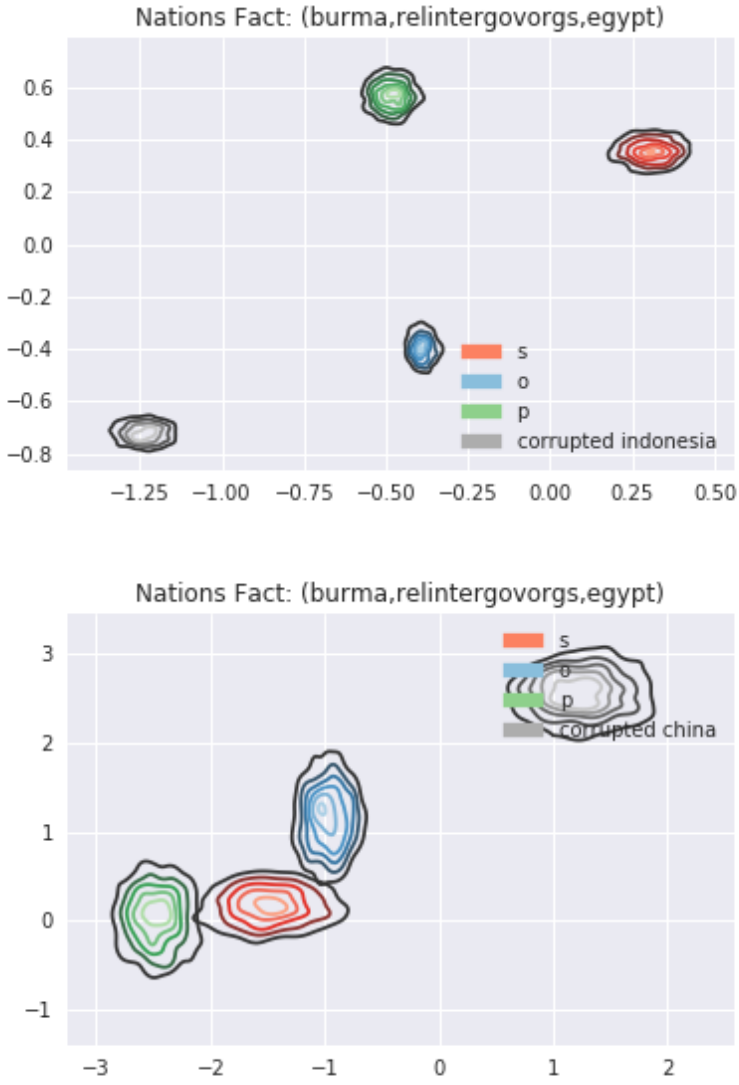

FIGURE 8.8: Two Visualisation Schemes For Embeddings. (Left) First two dimensions of each distributions embedding and (right) projected distribution using PPCA & NNMF

We encountered the problematic question during this stage of how to visualise

multiple high dimensional Gaussian distributions: We use two methods to acquire the parameters for a bi-variate distribution, which we can then easily plot. First, we arbitrarily select the parameters for two dimensions from each subject, object and predicate embeddings. Secondly, we project the high dimensional mean embedding vectors to two dimensions using Probabilistic Principal Component Analysis (PPCA) [Tipping and Bishop, 1999] to project the variance embedding vectors down to two dimensions using Non-negative Matrix Factorisation (NNMF) [Févotte and Idier, 2011].

Once we have the parameters for a bivariate normal distribution, we then sample from the bivariate normal distribution 1,000 times and then plot a bi-variate kernel density estimate of these samples. By visualising these two-dimensional samples, we can conceive the space in which the entity or relation occupies. We complete this process for the subject, object, relation, and a randomly sampled corrupted entity (under LCWA) to produce a visualisation of a fact, as shown in Figure 8.8.

The second visualisation scheme (Figure 8.9) of plotting samples from projecting the distribution parameters using PPCA & NNMF appears considerably more informative than the scheme of plotting samples from the first two dimensions. Figure 8.9 displays three true positives from test time predictions.

Figure 8.9 shows that the variational framework can learn high dimensional representations which when projected onto lower (more interpretable) dimensions. Figure 8.9 displays a clustering of the subject, object and predicate that create a positive (true) fact. We also observe a separation between the items which generate a fact and a randomly sampled (corrupted) entity which is likely to create a negative (false) fact. The first test fact "(USA, Commonbloc0, Netherlands)" shows clear irrationality similarity between all objects in the tested fact, i.e. the vectors are pointing towards a south-east direction. We can also see that the corrupted entity Jordan is quite a distance away from the items in the tested fact, which is good as Jordan does not share a common bloc either USA or Netherlands.

In the second test fact "(China, Embassy, Egypt)", of Figure 8.9, we can see the embeddings for China and Egypt are close together, indicating they share many relations together. We observe the predicate embedding for embassy is as separated from the subject and object embedding as the randomly sampled entity embedding Brazil. This would suggest that the scores acquired from "(China,Embassy,Egypt)", "(Egypt,Embassy,China)", "(Brazil,Embassy,China)" ,"(Brazil,Embassy,Egypt)","(China,Embassy,Brazil)" and "(Egypt,Embassy,Brazil)"

to be similarly high. From a simple investigation online [1] it is clear that all of these nations have at least one embassy in each of the other nations, so this visualisation is correctly extrapolating further truths around the similarities between these nations.

[1] https://www.embassypages.com/

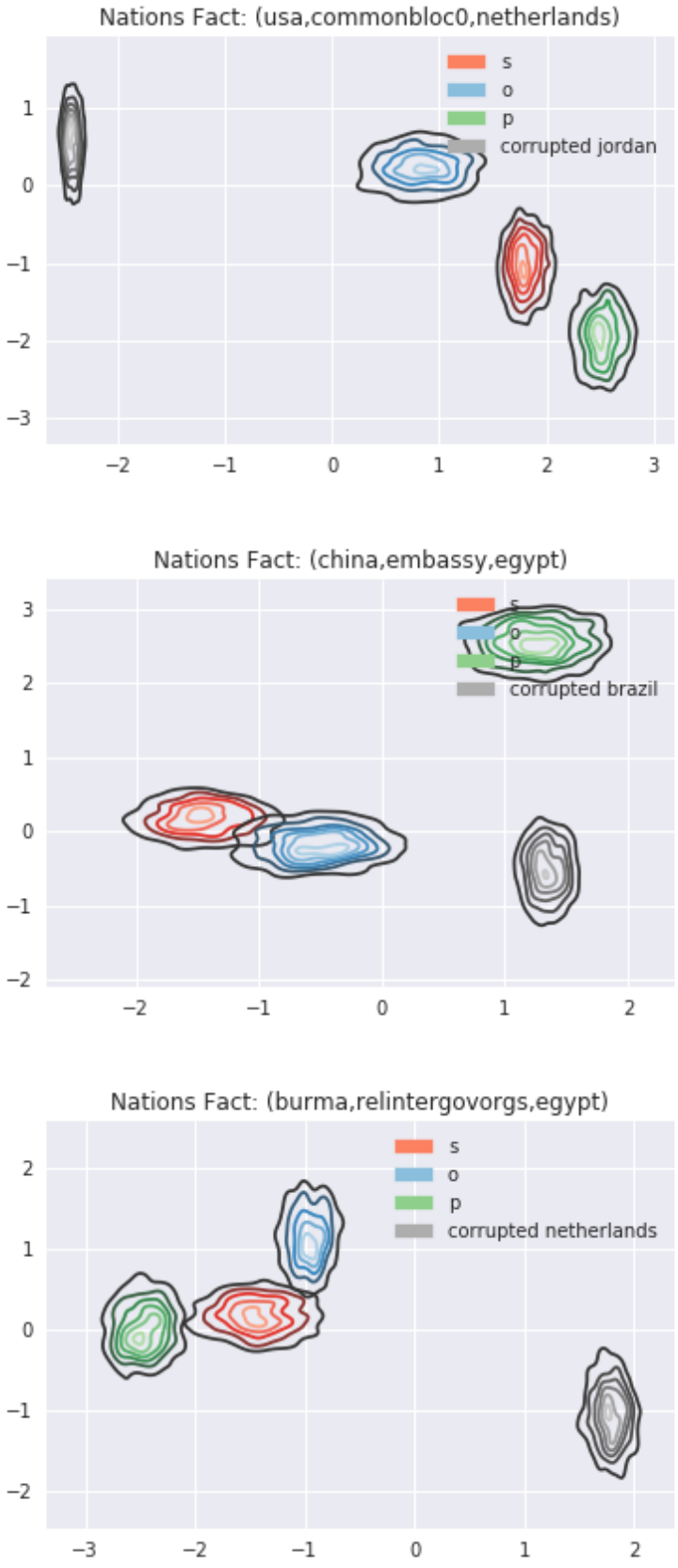

FIGURE 8.9: True Positives

8.8 Comparison to State-of-the-art

We now compare our model to the previous state-of-the-art multi-relational generative model TransG [Xiao, Huang, and Zhu, 2016], as well as to a previously published probabilistic embedding method KG2E [He et al., 2015] on the WN18 dataset. Unfortunately, we had to use the WN18 dataset for comparison as this was the only common dataset all three had been evaluated on, whereas we would have preferred to evaluate them based on the more challenging datasets WN18RR and FB15K-237.

Dataset	Model	Mean Rank		Raw Hits@	Filtered Hits @		
		Raw	Filter	10	1	3	10
WN18	KG2E He et al., 2015	362	345	0.805	-	-	0.932
	TransG Xiao, Huang, and Zhu, 2016	345	357	**0.845**	-	-	0.949
	Variational ComplEx (Model A)	753	765	83.58	**0.934**	**0.945**	**0.952**
	Variational ComplEx (Model B)	899	914	80.1	0.901	0.936	0.940

TABLE 8.4: Probabilistic Models

Table 8.4 makes clear the improvements in the performance of the previous state-of-the-art generative multi-relational knowledge graph model. Model A has marginally worse performance than the state-of-the-art model on raw Hits@10. We conjecture two reasons may cause this discrepancy. Firstly, the fact the authors of TransG use negative samples provided only (True negative examples), whereas we generated our negative samples using the LCWA. Secondly, we only use one negative sample per positive to estimate the Evidence Lower Bound using Bernoulli sampling, whereas it is likely they used significantly more negative samples. This conjecture was proved true in a follow-up experiment on Nations; increasing performance on raw Hits@10 when using 20 negative samples, with no change in filtered Hits@10.

8.9 Ablation Study

We begin the ablation study over the small and straightforward dataset Nations, in Section 8.9.1. Lastly, we explore the same model and parameters over one of the most challenging datasets, WN18RR, in Section 8.9.2.

The ablation study was conducted on Model A with; DistMult, KL Compression Cost, linear warm-up, Bernoulli sampling to estimate the ELBO, unit variance sum constraint, Xavier initialisation for the mean embeddings, and constant initialisation for the variance embeddings. We replace each component separately with a trivial alternative to attempt at quantifying overall contribution.

	Full Model	KL Compression Cost	Linear Warmup	Bernoulli Sampling	Unit Variance Init	Xavier Mean Init	Constrained Variance
Hits@10	0.995	0.995	0.995	0.983	0.993	0.985	0.995

TABLE 8.5: Nations Ablation Study Best Validation Hits@10

1. KL Compression Cost is replaced with linear $\frac{1}{\text{Number of Batchs}}$.
2. linear warm-up on the KL Compression Cost is replaced with only the KL Compression Cost term.
3. Bernoulli sampling to estimate the ELBO (using one negative sample) is replaced with negative sampling — again using one negative sample.
4. Variance embedding sum constraint in removed.
5. Xavier initialisation on the mean embeddings is replaced with random uniform initialisation between -0.001 and 0.001.
6. Constant variance initialisation of $\frac{1}{\text{embedding size}}$ is replaced with random uniform initialisation between zero and $\frac{1}{\text{embedding size}}$.

8.9.1 Small Dataset

Figure 8.10 displays the results of the ablation study on the small dataset, Nations. We run this study over 100 epochs, evaluating every ten epochs on a validation set. We set all random seeds to zero on tensorflow and numpy to enable a fairer comparison between model components to alleviate additional randomness factors.

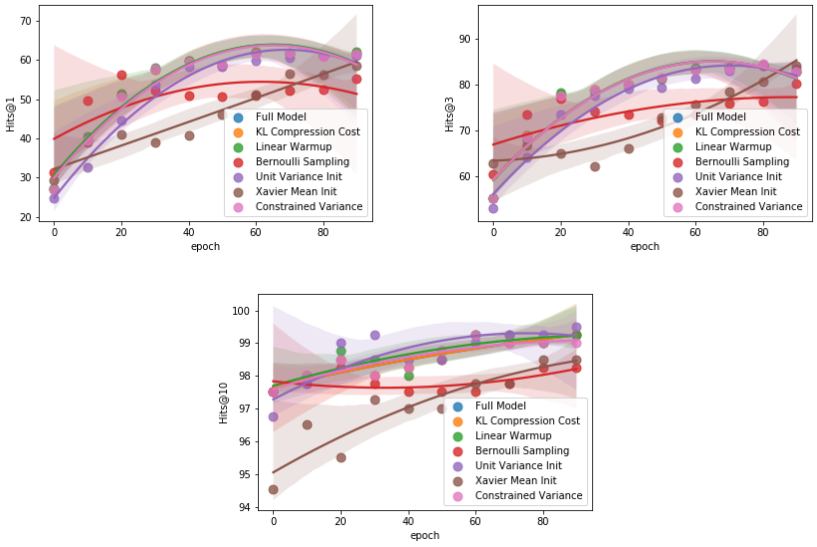

FIGURE 8.10: Nations Ablation Study Hits@m

Figure 8.10 displays the ablation results for small datasets. The most significant component is the mean embedding initialisation scheme. From using Xavier to random uniform, we have the largest damping of performance. Bernoulli sampling for ELBO estimation seems to be the second most significant component of this model, allowing the model to reach a higher accuracy. The negative impact of NS comes at a surprise, as typically negative sampling is used in knowledge graph construction.

Table 8.5 shows that all of the components seem to improve convergence collectively. The ablation study also shows not using Xavier Mean initialisation and using an initialisation between -0.001 and 0.001 seem to work marginally better with the correct model. We have low confidence claim due to only a single seed being used to support the claim, as well as the minuscule difference in results the claim is based off. Overall, it seems most the additional components aid the learning process in the early stages. However, it results in similar peaks of performance further down the training schedule.

8.9.2 Large Dataset

We then perform an ablation study in Figure 8.11 over 350 epochs, evaluating a validation set every five epochs on WN18RR. Similarly, we set all random seeds equal to zero on tensorflow and numpy to make a fairer comparison between model components without an additional randomness factor included.

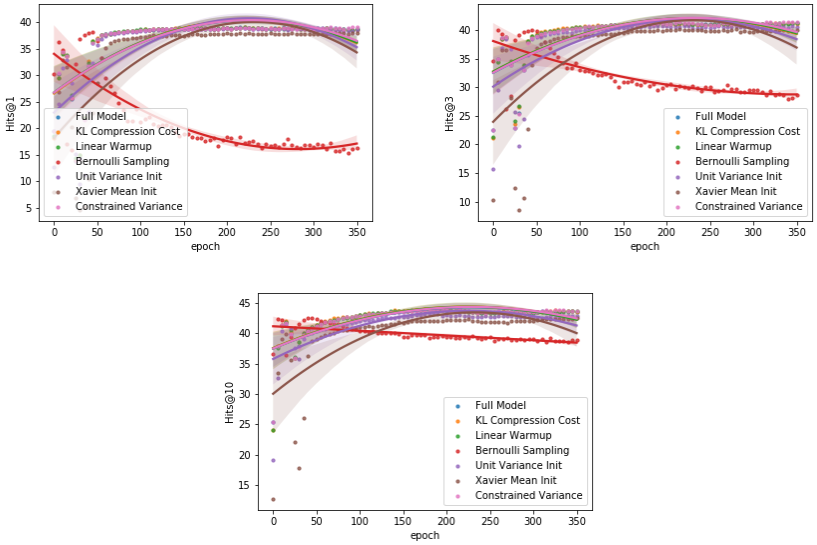

FIGURE 8.11: WN18RR Ablation Study Hits@m

	Full Model	KL Compression Cost	Linear Warmup	Bernoulli Sampling	Unit Variance Init	Xavier Mean Init	Constrained Variance
Hits@10	0.438	0.440	0.440	0.425	0.432	0.424	0.440

TABLE 8.6: WN18RR Ablation Study Best Validation Hits@10

Table 8.6 similarly shows for this dataset, all of the components collectively improve convergence. However, using the linear KL weight, un-constraining the variance and not using linear warm-up all lead to marginally better results than achieved with the current model, with a longer convergence rate. In contrast, Bernoulli Sampling to estimate the ELBO, using Xavier initialisation and unit variance initialisation all negatively impact the model's performance when taken out.

Chapter 9

Conclusion & Further Work

9.1 Conclusion

Overall we have provided a significant contribution to the field by unifying deep learning, graphical models, and knowledge graphs through the development of Model A and Model B. Returning to the thesis goals 1.2, we will discuss the progression we have made towards each primary goal.

- *Firstly, can we propose an alternative neural approach to Knowledge Graph representation and link prediction that allows us to identify better and measure predictive uncertainty?*

We have successfully created a framework allowing a model to learn embeddings of any prior distribution that permits a re-parametrisation trick Section 2.5 via any score function. We have shown, from preliminary experiments, that these display competitive results with current models. We have yet to discover a novel method for measuring predictive uncertainty within these systems. We trialled an approach which attempts at measuring predictive uncertainty in Section 8.4, which we hope will develop further theories as to measuring predictive uncertainty in generative knowledge graph models. Overall, we believe this work will enable knowledge graph researchers to work towards this goal with a new suite of tools. We would encourage further work on (1) investigating the use of sampling during training, as currently done, to learn the distribution parameters and then at test time evaluating the use of probability density functions over the learnt relation parameters to use as a score: or (2) for confidence estimation.

- *Secondly, can we incorporate literature within stochastic variational inference to scale these robust representations to work on large graph structures?*

This second primary goal has been achieved. In doing so, we have provided an alternative justification to the Stochastic Variational Inference Evidence Lower Bound estimator under the Bernoulli sampling framework, then shown its effectiveness in graph structure learning.

- *Lastly, can we better address the challenge of entities and relations that require multiple representations*

By using a generative model for the embeddings, we can address this problem. It is inconclusive if the proposed models have utilised this, so can be left for further work (3). Similarly for the secondary research questions:

- *What can we learn from analysing the variances?*

We observed a slight negative correlation between variance size and frequency—however, this was with almost no statistical significance. Thus, the uncertainty analysis remains an open question and one we would encourage for further work (4).

- *What are the trade-offs for representing Knowledge Bases under this alternative framework?*

The current method leads to a constant increase in the parameters learnt, which leads to a minor increase in training and inference due to the simplicity of the overall architecture.

9.2 Further Work

Continuing from the above, we present a further four ideas for future investigation:

5. Recent advances in variational auto-encoders permit the use of the Von Mises-Fisher distribution under a novel re-parameterisation trick [Davidson et al., 2018]. This distribution places a directional hyperspherical prior on the latent variables, which proved beneficial during experiments on knowledge graphs in [Davidson et al., 2018]. The bidirectionally of the distribution seems more natural for modelling a directed graph, especially when using a scoring function that takes into consideration the similarity between vectors, as shown in Fig 9.1 from Straub, 2017[1].

[1] http://people.csail.mit.edu/jstraub/download/straub2017vonMisesFisherInference.pdf

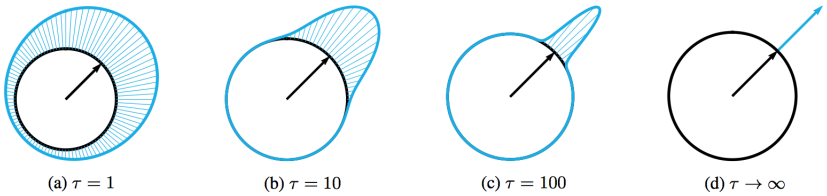

FIGURE 9.1: Depiction of 2D von-Mises-Fisher distributions with increasing concentration τ . As $\tau \to$ inf the von-Mises-Fisher distribution approaches a delta function on the sphere at its mode μ

6. Motivated by the authors of [Bražinskas, Havrylov, and Titov, 2017], who highlighted the weaknesses of using point estimates of distributions, as shown in the Fig 9.2, from their article. Where both distributions q have the same mean, however, one has a much larger variance, undesired when trying to grasp control of the uncertainties in knowledge graphs. We do not expect our model to suffer as dramatically as in Fig 9.2, as we sample our point estimates before computing the score. However, the score we acquire at test time even through forward sampling does not seem to differ much compared with the mean embeddings, thus using the learnt uncertainty to impact the results positively is a fruitful path.

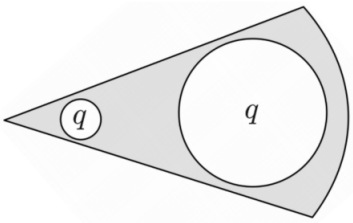

FIGURE 9.2: Shaded cone is a fixed angle, and ellipses are approximate posterior Gaussian distributions. The corner of the cone is at the origin.

7. We propose some alternative similarity functions in the Appendix A, however in their current form these distributional knowledge graph similarity functions can not be trained using optimisation methods.

8. It would also be interesting to investigate the use of multiple samples during training, to see the effect of this on the learnt embeddings.

9. It would be interesting investigating other forms of encoding functions, such as a multi-layer perceptron (MLP) on each input of one hot encoded entity/predicate vectors, or a graph convolution network.

10. Lastly, it would be interesting applying latent regularisation techniques, Section 2.6, to the predicate mean embeddings.

9.3 Critique

We have three main critiques from this thesis;

1. Although the metrics reported from Model A are a significant improvement compared to existing generative models, they remain slightly lower than the metrics obtained from the scoring functions ComplEx and DistMult etc. With a sufficient parameter sweep Model A should at least be on par with ComplEx and DistMult's recently published results.

2. We realised during intermediate stages of the experimental section that TransE did not theoretically permit maximum likelihood estimation of its parameters, as TransE's scoring function values are strictly negatives so computing $\Theta(score) \leq 0.5$. However, in some cases, Variational TransE seemed to produce reasonable results. It was inconclusive as to why this worked and can be left for further investigation.

3. There is an argument against using mean rank to compare performance across datasets, and instead use mean reciprocal rank which uses the mean rank relative to the total number of entities available to rank. We would have preferred to use this metric, however, had already conducted many experiments before this was seen an issue.

Appendix A

Similarity Functions

We now propose scoring functions which measure the similarity between three distributions of subject, object and predicate.

Product

Theorem 1. *For the special case of two Gaussian probability densities*

$$x_1(t) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(t-\mu_1)^2}{2\sigma_1^2}}$$
$$x_2(t) = \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(t-\mu_2)^2}{2\sigma_2^2}}$$

The product density has mean and variance given by

$$\mu = \frac{\frac{\mu_1}{2\sigma_1^2} + \frac{\mu_2}{2\sigma_2^2}}{\frac{1}{2\sigma_1^2} + \frac{1}{2\sigma_2^2}} = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_2^2 + \sigma_1^2}$$
$$\sigma^2 = \sigma_1^2 \| \sigma_2^2 = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}.$$

Proof

See Smith, 2018. [1].

Therefore the probability density function remains Gaussian, an extremely useful result.

[1] https://ccrma.stanford.edu/ jos/sasp/Product_Two_Gaussian_PDFs.html

GaussMult

Thus we define a new score function g_{θ_2}, which is based on the first moment of the inner product of three Gaussian distributions.

$$\begin{aligned} g_{\theta_2}(\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3) &= \mathbb{E}[< \mathcal{N}(x; \mu_1, \sigma_1)\mathcal{N}(x; \mu_2, \sigma_2)\mathcal{N}_(x; \mu_3, \sigma_3) >] \\ &= \mathbb{E}[\sum_{d=1}^{D} \mathcal{N}(x; \mu_1, \sigma_1)\mathcal{N}(x; \mu_2, \sigma_2)\mathcal{N}_(x; \mu_3, \sigma_3)] \\ &= \sum_{d=1}^{D} \int_{-\infty}^{\infty} x\mathcal{N}(x; (\frac{x; \mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_2^2 + \sigma_1^2}), (\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}))\mathcal{N}_(x; \mu_3, \sigma_3)dx \\ &= \sum_{d=1}^{D} \int_{-\infty}^{\infty} x\mathcal{N}(x; (\frac{\mu_{prod}\sigma_3 + \mu_3\sigma_{prod}}{\sigma_3 + \sigma_{prod}}), (\frac{\sigma_{prod}\sigma_3}{\sigma_{prod} + \sigma_3}))dx \\ &= \sum_{d=1}^{D} \frac{\mu_{prod,k}\sigma_{3,k} + \mu_{3,k}\sigma_{prod,k}}{\sigma_{3,k} + \sigma_{k,prod}} \end{aligned} \tag{A.1}$$

where K is the latent dimension size

$\sigma_{prod} = (\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2})$

$\mu_{prod} = (\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_2^2+\sigma_1^2})$
(A.2)

Then taking the sigmoid in order to estimate the likelihood gives

$$\begin{aligned} f_{\theta_2}(g_{\theta_2}) &= \frac{1}{1 + e^{-g_{\theta_2}}} \\ p^{\theta}(x_{s,r,o} = 1) &= f_{\theta_2} \end{aligned} \tag{A.3}$$

Note, it is clear that Eq A.3 approximated through the DistMUlt of Gaussian samples seen previously Eq 6.3, as shown in Eq A.4.

$$p^{\theta}(x_{s,r,o} = 1) = f_{\theta_2} \approx \frac{1}{1 + e^{-<e_s, e_p, e_o>}} = f_{\theta} \tag{A.4}$$

GaussXE

First, given three distributions $\mathcal{N}_s, \mathcal{N}_p, \mathcal{N}_o$. We first compute the probability density distribution of the product of subject and object embeddings.

$$\mathcal{N}_{mult} = \mathcal{N}_s \cdot \mathcal{N}_o = \mathcal{N}(\mu_s, \sigma_s) \cdot \mathcal{N}(\mu_o, \sigma_o) = \mathcal{N}(\frac{\mu_s\sigma_o^2 + \mu_o\sigma_s^2}{\sigma_o^2 + \sigma_s^2}, \frac{\sigma_s^2\sigma_o^2}{\sigma_s^2 + \sigma_o^2}) \quad \text{(A.5)}$$

Then calculating the probability density of the first moment of the difference distribution, over the predicate distribution.

$$\begin{aligned} p^\theta(x_{s,r,o} = 1) = f_{\theta_3}(\mathcal{N}_s, \mathcal{N}_p, \mathcal{N}_o) &= \mathcal{N}(\mathbb{E}[\mathcal{N}_{mult}]; \mu_p, \sigma_p) \\ &= \mathcal{N}(\frac{\mu_s\sigma_o^2 + \mu_o\sigma_s^2}{\sigma_o^2 + \sigma_s^2}; \mu_p, \sigma_p) \end{aligned} \quad \text{(A.6)}$$

GaussE

Given three distributions $\mathcal{N}_s, \mathcal{N}_p, \mathcal{N}_o$, we first compute the distribution of the difference of subject and object embeddings.

$$\mathcal{N}_{difference} = \mathcal{N}_s - \mathcal{N}_o = \mathcal{N}(\mu_s, \sigma_s) - \mathcal{N}(\mu_o, \sigma_o) = \mathcal{N}(\mu_s - \mu_o, \sigma_s^2 + \sigma_o^2) \quad \text{(A.7)}$$

Then calculating the probability density of the first moment of the difference distribution, over the predicate distribution.

$$\begin{aligned} p^\theta(x_{s,r,o} = 1) = f_{\theta_4}(\mathcal{N}_s, \mathcal{N}_p, \mathcal{N}_o) &= \mathcal{N}(\mathbb{E}[\mathcal{N}_{difference}]; \mu_p, \sigma_p) \\ &= \mathcal{N}(\mu_s - \mu_o; \mu_p, \sigma_p) \end{aligned} \quad \text{(A.8)}$$

Inner Product

We notice firstly, that the inner product for two discrete functions $f(x)$ & $q(x)$, is equivalent the dot product

$$\int_{x \in \mathcal{R}^n} f(x)q(x) = \sum_{d=1}^{D} f_d(x_d)q_d(x_d) =< f(x), q(x) > \quad \text{(A.9)}$$

This suggests that the inner product may be a more natural expression of similarity than the product operation between Gaussian probability density functions, as previously defined.

Hence using the below theorem we can derive additional scoring functions.

Theorem 2. *For the special case of the inner product of two Gaussians, we obtain the expected likelihood kernel*

$$K(P_i, P_j) = \int_{x \in \mathcal{R}^n} \mathcal{N}(x; \mu_i, \Sigma_i)\mathcal{N}(x; \mu_j, \Sigma_j)dx = \mathcal{N}(0; \mu_i - \mu_j, \Sigma_i + \Sigma_j)$$

Proof

See Jebara, Kondor, and Howard, 2004 .

InnerGaussE

Following the inspiration from TransE, we propose another novel score function.

Given three distributions $\mathcal{N}_s, \mathcal{N}_p, \mathcal{N}_o$, we first compute the distribution of the difference of subject and object embeddings.

$$\mathcal{N}_{difference} = \mathcal{N}_o - \mathcal{N}_s = \mathcal{N}(\mu_o, \sigma_o) - \mathcal{N}(\mu_s, \sigma_s) = \mathcal{N}(\mu_o - \mu_s, \sigma_o^2 + \sigma_s^2) \quad \text{(A.10)}$$

$$\begin{aligned} p^\theta(x_{s,r,o} = 1) = f_{\theta_5}(\mathcal{N}_s, \mathcal{N}_p, \mathcal{N}_o) &= K(\mathcal{N}(\mu_p, \sigma_p), \mathcal{N}_{difference}) \\ &= \mathcal{N}(0; \mu_p + \mu_s - \mu_o, \sigma_p^2 + \sigma_o^2 + \sigma_s^2) \end{aligned} \quad \text{(A.11)}$$

Appendix B

Implementation Details

We will briefly discuss the various methods we used for sampling via the re-parametrisation trick.

Sampling

For each input in the knowledge graph triple $z \in (O, P, S)$, we first perform a embedding look up into our second moment representation, producing the required vectors of parameters for the latent variables

$$\begin{aligned} \mu_i, \tau_i &= log(exp(\sigma_i) - 1) \\ \Omega_j, \tau j &= log(exp(\rho_j) - 1) \\ \mu_k, \tau_k &= log(exp(\sigma_k) - 1) \end{aligned} \tag{B.1}$$

with the $\sigma = log(1 + exp(\tau))$ representation [Blundell et al., 2015] enforcing σ to remain positive. Or the triple with the typical $\sigma = sqrt(exp(\tau))$ representation [Kingma and Welling, 2013].

$$\begin{aligned} \mu_i, \tau_i &= log(\sigma_i^2) \\ \Omega_j, \tau j &= log(\rho_j^2) \\ \mu_k, \tau_k &= log(\sigma_k^2) \end{aligned} \tag{B.2}$$

The reason we do not represent our variance directly, is to enforce our variance values to remain positive during gradient descent updates. We then sample as previously stated,

$$\begin{aligned} e_i &\sim \mathcal{N}(\mu_i, \sigma_i^2) \\ r_j &\sim \mathcal{N}(\Omega_j, \rho_j^2) \\ e_k &\sim \mathcal{N}(\mu_k, \sigma_k^2) \end{aligned} \tag{B.3}$$

However, we do this by utilising the re-parametrisation trick,

$$\begin{aligned} \hat{e}_i &\sim \mu_i + \hat{\epsilon}_i \cdot \sigma_i \\ \hat{r}_j &\sim \Omega_j + \hat{\epsilon}_j \cdot \rho_j \\ \hat{e}_k &\sim \mu_k + \hat{\epsilon}_k \cdot \sigma_k \end{aligned} \tag{B.4}$$

Where

$$\begin{aligned} \hat{\epsilon}_i &\sim \mathcal{N}(0, \mathcal{I}) \\ \hat{\epsilon}_j &\sim \mathcal{N}(0, \mathcal{I}) \\ \hat{\epsilon}_k &\sim \mathcal{N}(0, \mathcal{I}) \end{aligned} \tag{B.5}$$

B.0.1 Optimisation Problem

As $\mathcal{L}^A$ & $\mathcal{L}^B$ are compositional we can choose to either optimise it all in one go, or to optimise the g and e objective separately. This performed poorly in practise, so we chose to jointly optimise the ELBO. It is typical to constrain maximum variance values to unit variance, which produces the below optimisation problem

$$\min_{\mu_{1...||\mathcal{E}||} \quad \tau_{1...||\mathcal{E}||} \quad \Omega_{1...||P||} \quad \rho_{1...||P||}} -\mathcal{L} : ||\rho||_2 \leq 1 \;\; ||\sigma||_2 \leq 1 \tag{B.6}$$

B.0.2 Computing Variable Gradients

With gradients computed as normal at all nodes apart from the variation parameters which are calculated using the Stochastic Variational Bayes Estimator.

$$
\begin{aligned}
\frac{\partial \hat{\mathcal{L}}}{\partial \mu_i} &\approx \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \\
\frac{\partial \hat{\mathcal{L}}}{\partial \tau_i} &\approx \hat{\epsilon} \cdot \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \\
\frac{\partial \hat{\mathcal{L}}}{\partial \Omega_j} &\approx \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \\
\frac{\partial \hat{\mathcal{L}}}{\partial \tau_j} &\approx \hat{\epsilon} \cdot \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \\
\frac{\partial \hat{\mathcal{L}}}{\partial \mu_k} &\approx \frac{\partial \hat{\mathcal{L}}}{\partial \theta} \\
\frac{\partial \hat{\mathcal{L}}}{\partial \tau_k} &\approx \hat{\epsilon} \cdot \frac{\partial \hat{\mathcal{L}}}{\partial \theta}
\end{aligned}
\tag{B.7}
$$

Bibliography

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Aizerman, M. A., E. A. Braverman, and L. Rozonoer (1964). "Theoretical foundations of the potential function method in pattern recognition learning." In: *Automation and Remote Control,* Automation and Remote Control, 25, pp. 821–837.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12, pp. 2481–2495.

Barkan, Oren (2016). "Bayesian Neural Word Embedding". In: *CoRR* abs/1603.06571. arXiv: 1603.06571. URL: http://arxiv.org/abs/1603.06571.

Bishop, Christopher M. (2006a). *Pattern recognition and machine learning*. Springer.

– (2006b). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387310738.

Blundell, C. et al. (2015). "Weight Uncertainty in Neural Networks". In: *ArXiv e-prints*. arXiv: 1505.05424 [stat.ML].

Bodenreider, Olivier (2004). *The Unified Medical Language System (UMLS): Integrating Biomedical Terminology*.

Bordes, Antoine et al. (2013a). "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 2787–2795. URL: http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf.

Bordes, Antoine et al. (2013b). "Translating Embeddings for Modeling Multi-relational Data". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 2787–2795. URL: http://dl.acm.org/citation.cfm?id=2999792.2999923.

Botev, Aleksandar, Bowen Zheng, and David Barber (2017). "Complementary Sum Sampling for Likelihood Approximation in Large Scale Classification". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pp. 1030–1038. URL: http://proceedings.mlr.press/v54/botev17a.html.

Bowman, Samuel R. et al. (2016). "Generating Sentences from a Continuous Space". In: *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pp. 10–21. URL: http://aclweb.org/anthology/K/K16/K16-1002.pdf.

Bražinskas, A., S. Havrylov, and I. Titov (2017). "Embedding Words as Distributions with a Bayesian Skip-gram Model". In: *ArXiv e-prints*. arXiv: 1711.11027 [cs.CL].

Burda, Yuri, Roger B. Grosse, and Ruslan Salakhutdinov (2015). "Importance Weighted Autoencoders". In: *CoRR* abs/1509.00519. arXiv: 1509.00519. URL: http://arxiv.org/abs/1509.00519.

Challis, Edward and David Barber (2013). "Gaussian Kullback-Leibler Approximate Inference". In: *Journal of Machine Learning Research* 14, pp. 2239–2286. URL: http://jmlr.org/papers/v14/challis13a.html.

Chen, Wenhu et al. (2018). "Variational Knowledge Graph Reasoning". In: *NAACL-HLT*.

Culotta, Aron and Andrew McCallum (2004). "Confidence Estimation for Information Extraction". In: *Proceedings of HLT-NAACL 2004: Short Papers*. HLT-NAACL-Short '04. Boston, Massachusetts: Association for Computational Linguistics, pp. 109–112. ISBN: 1-932432-24-8. URL: http://dl.acm.org/citation.cfm?id=1613984.1614012.

Dai, Zhenwen et al. (2015). "Variational auto-encoded deep Gaussian processes". In: *arXiv preprint arXiv:1511.06455*.

Damianou, Andreas and Neil Lawrence (2013). "Deep gaussian processes". In: *Artificial Intelligence and Statistics*, pp. 207–215.

Davidson, T. R. et al. (2018). "Hyperspherical Variational Auto-Encoders". In: *ArXiv e-prints*. arXiv: 1804.00891 [stat.ML].

Davis, Randall, Howard E. Shrobe, and Peter Szolovits (1993). "What Is a Knowledge Representation?" In: *AI Magazine* 14.1, pp. 17–33. URL: http://citeseer.ist.psu.edu/davis93what.html.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee

et al. Curran Associates, Inc., pp. 3844–3852. URL: http://papers.nips.cc/paper/6081-convolutional-neural-networks-on-graphs-with-fast-localized-spectral-filtering.pdf.

Dettmers, Tim et al. (2017). "Convolutional 2D Knowledge Graph Embeddings". In:

Dillon, J. V. et al. (2017). "TensorFlow Distributions". In: *ArXiv e-prints*. arXiv: 1711.10604.

Duvenaud, David K et al. (2015). "Convolutional Networks on Graphs for Learning Molecular Fingerprints". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., pp. 2224–2232. URL: http://papers.nips.cc/paper/5954-convolutional-networks-on-graphs-for-learning-molecular-fingerprints.pdf.

Ehrlinger, Lisa and Wolfram Wöß (2016). "Towards a Definition of Knowledge Graphs". In: *SEMANTiCS*.

Freeman, L.C. (1965). *Elementary applied statistics: for students in behavioral science*. Wiley. URL: https://books.google.co.uk/books?id=r4VRAAAAMAAJ.

Fujiwara, T, M Kamada, and Y Okuno (2018). "Artificial Intelligence in Drug Discovery". In: *Gan to kagaku ryoho. Cancer & chemotherapy* 45.4, pp. 593–596.

Févotte, Cédric and Jérôme Idier (2011). "Algorithms for Nonnegative Matrix Factorization with the -Divergence". In: *Neural Computation* 23.9, pp. 2421–2456. DOI: 10.1162/NECO_a_00168. eprint: https://doi.org/10.1162/NECO_a_00168. URL: https://doi.org/10.1162/NECO_a_00168.

Gal, Yarin (2016). *Uncertainty in Deep Learning*. URL: http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani (2017). "Deep Bayesian Active Learning with Image Data". In: *CoRR* abs/1703.02910. arXiv: 1703.02910. URL: http://arxiv.org/abs/1703.02910.

Ghahramani, Zoubin (2015). "Probabilistic machine learning and artificial intelligence". In: *Nature* 521.7553, 452–459. DOI: 10.1038/nature14541.

Ghahramani, Zoubin and H Attias (2000). "Online variational Bayesian learning". In: *Slides from talk presented at NIPS workshop on Online Learning*.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep learning*. MIT Press.

Hafner, Danijar (2018). *Building Variational Auto-Encoders in TensorFlow*. Blog post. URL: https://danijar.com/building-variational-auto-encoders-in-tensorflow/.

He, Shizhu et al. (2015). "Learning to Represent Knowledge Graphs with Gaussian Embedding". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM '15. Melbourne, Australia: ACM, pp. 623–632. ISBN: 978-1-4503-3794-6. DOI: 10.1145/2806416.2806502. URL: http://doi.acm.org/10.1145/2806416.2806502.

Henaff, Mikael, Joan Bruna, and Yann LeCun (2015). "Deep Convolutional Networks on Graph-Structured Data". In: *CoRR* abs/1506.05163.

Hoffman, Matthew, Francis R Bach, and David M Blei (2010). "Online learning for latent dirichlet allocation". In: *advances in neural information processing systems*, pp. 856–864.

Hoffman, Matthew D et al. (2013). "Stochastic variational inference". In: *The Journal of Machine Learning Research* 14.1, pp. 1303–1347.

Jebara, Tony, Risi Kondor, and Andrew Howard (2004). "Probability Product Kernels." In: 5, pp. 819–844.

Jensen, J. L. W. V. (1906). "Sur les fonctions convexes et les inégalités entre les valeurs moyennes". In: *Acta Mathematica* 30, 175–193. DOI: 10.1007/bf02418571.

Jordan, Michael I. et al. (1998). "An Introduction to Variational Methods for Graphical Models". In: *Learning in Graphical Models*, 105–161.

Kapoor, A. et al. (2007). "Active Learning with Gaussian Processes for Object Categorization". In: *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8. DOI: 10.1109/ICCV.2007.4408844.

Katharopoulos, Angelos and François Fleuret (2017). "Biased Importance Sampling for Deep Neural Network Training". In: *CoRR* abs/1706.00043. arXiv: 1706.00043. URL: http://arxiv.org/abs/1706.00043.

Katz, Leo (1953). "A new status index derived from sociometric analysis". In: *Psychometrika* 18.1, pp. 39–43. URL: http://ideas.repec.org/a/spr/psycho/v18y1953i1p39-43.html.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

Kingma, Diederik P and Max Welling (2013). "Auto-Encoding Variational Bayes". In: *UvA*, pp. 1–14. arXiv: 1312.6114. URL: http://arxiv.org/abs/1312.6114.

Kipf, Thomas N. and Max Welling (2016). "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907.

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, 79–86.

Lao, Ni and William W. Cohen (2010). "Relational Retrieval Using a Combination of Path-constrained Random Walks". In: *Mach. Learn.* 81.1, pp. 53–67. ISSN: 0885-6125. DOI: 10.1007/s10994-010-5205-8. URL: http://dx.doi.org/10.1007/s10994-010-5205-8.

Lao, Ni, Tom Mitchell, and William W. Cohen (2011). "Random Walk Inference and Learning in a Large Scale Knowledge Base". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Edinburgh, United Kingdom: Association for Computational Linguistics, pp. 529–539. ISBN: 978-1-937284-11-4. URL: http://dl.acm.org/citation.cfm?id=2145432.2145494.

LeCun, Yann and Corinna Cortes (2010). "MNIST handwritten digit database". In: URL: http://yann.lecun.com/exdb/mnist/.

Lim, Yew Jin (2007). "Variational Bayesian Approach to Movie Rating Prediction". In:

Lindley, D. V. and Solomon Kullback (1959). "Information Theory and Statistics." In: *Journal of the American Statistical Association* 54.288, p. 825.

Miao, Y., E. Grefenstette, and P. Blunsom (2017). "Discovering Discrete Latent Topics with Neural Variational Inference". In: *ArXiv e-prints*. arXiv: 1706.00359 [cs.CL].

Miao, Yishu, Edward Grefenstette, and Phil Blunsom (2017). "Discovering Discrete Latent Topics with Neural Variational Inference". In: *CoRR* abs/1706.00359. arXiv: 1706.00359. URL: http://arxiv.org/abs/1706.00359.

Miao, Yishu, Lei Yu, and Phil Blunsom (2016). "Neural Variational Inference for Text Processing". In: *Proceedings of the 33rd International Conference on Machine Learning*.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Minervini, P. et al. (2018). "Towards Neural Theorem Proving at Scale". In: *ArXiv e-prints*. arXiv: 1807.08204 [cs.AI].

Minervini, Pasquale et al. (2017). "Adversarial Sets for Regularising Neural Link Predictors". In: *CoRR* abs/1707.07596. arXiv: 1707.07596. URL: http://arxiv.org/abs/1707.07596.

Muggleton, Stephen (1995). *Inverse entailment and Progol*.

Muzaffar, Abdul Wahab, Farooque Azam, and Usman Qamar (2015). "A Relation Extraction Framework for Biomedical Text Using Hybrid Feature Set". In: *Comp. Math. Methods in Medicine*.

N. Kipf, Thomas and Max Welling (2016). "Variational Graph Auto-Encoders". In:

Nickel, M., L. Rosasco, and T. Poggio (2015). "Holographic Embeddings of Knowledge Graphs". In: *ArXiv e-prints*. arXiv: 1510.04935 [cs.AI].

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel (2011). "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. Bellevue, Washington, USA: Omnipress, pp. 809–816. ISBN: 978-1-4503-0619-5. URL: http://dl.acm.org/citation.cfm?id=3104482.3104584.

Nickel, Maximilian et al. (2015). "A Review of Relational Machine Learning for Knowledge Graphs: From Multi-Relational Link Prediction to Automated Knowledge Graph Construction." In: *CoRR* abs/1503.00759. URL: http://dblp.uni-trier.de/db/journals/corr/corr1503.html#NickelMTG15.

O, Bojar et al. (2017). *Findings of the 2017 Conference on Machine Translation (WMT17)*. URL: http://www.aclweb.org/anthology/W17-4717.

Opper, M. and D. Saad (2001). *Advanced Mean Field Methods: Theory and Practice*. Neural information processing series. MIT Press. ISBN: 9780262150545. URL: https://books.google.co.uk/books?id=cuOX8sCDeNAC.

Plake, Conrad et al. (2006). "AliBaba: PubMed as a graph". In: *Bioinformatics* 22.19, pp. 2444–2445.

Quinlan, J R (1989). *Learning relations: comparison of a symbolic and a connectionist approach*. Tech. rep. University of Sydney.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082*.

Robbins, Herbert and Sutton Monro (1985). "A stochastic approximation method". In: *Herbert Robbins Selected Papers*. Springer, pp. 102–109.

Rocktäschel, Tim and Sebastian Riedel (2017). "End-to-end Differentiable Proving". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 3788–3800. URL: http://papers.nips.cc/paper/6969-end-to-end-differentiable-proving.pdf.

Rumelhart, D.e., G.e. Hinton, and R.j. Williams (1988). "Learning Internal Representations by Error Propagation". In: *Readings in Cognitive Science*, 399–421.

Salakhutdinov, Ruslan and Andriy Mnih (2008). "Probabilistic Matrix Factorization". In: *Advances in Neural Information Processing Systems*. Vol. 20.

Sato, Masa-Aki (2001). "Online model selection based on the variational Bayes". In: *Neural computation* 13.7, pp. 1649–1681.

Sellwood, Matthew A et al. (2018). *Artificial intelligence in drug discovery*.

Smith, Julius O. (2018). *Spectral Audio Signal Processing*. online book, 2011 edition. http:// ccrma.stanford.edu/ ~jos/ sasp/.

Socher, Richard et al. (2013). "Reasoning With Neural Tensor Networks For Knowledge Base Completion". In: *Advances in Neural Information Processing Systems 26*.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-14472-7.

Straub, Julian (2017). "Nonparametric Directional Perception". PhD thesis. 77 Massachusetts Avenue, Cambridge, MA 02139: Massachusetts Institute of Technology.

Sun, Yizhou and Jiawei Han (2012). *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers. ISBN: 1608458806, 9781608458806.

Tipping, Michael E. and Chris M. Bishop (1999). "Probabilistic Principal Component Analysis". In: *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 61.3, pp. 611–622.

Tong, Simon (2001). "Active Learning: Theory and Applications". In: *PhD thesis*. DOI: AAI3028187.

Toutanova, Kristina and Danqi Chen (2015a). "Observed versus latent features for knowledge base and text inference". In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66.

– (2015b). "Observed Versus Latent Features for Knowledge Base and Text Inference". In: *3rd Workshop on Continuous Vector Space Models and Their Compositionality*. ACL – Association for Computational Linguistics. URL:

https://www.microsoft.com/en-us/research/publication/observed-versus-latent-features-for-knowledge-base-and-text-inference/.

Trouillon, Théo et al. (2016). "Complex Embeddings for Simple Link Prediction". In: *CoRR* abs/1606.06357. arXiv: 1606.06357. URL: http://arxiv.org/abs/1606.06357.

Vilnis, Luke and Andrew McCallum (2014). "Word Representations via Gaussian Embedding". In: *CoRR* abs/1412.6623. arXiv: 1412.6623. URL: http://arxiv.org/abs/1412.6623.

Vincent, Pascal et al. (2008). "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning - ICML 08*.

Wang, Chong, John Paisley, and David Blei (2011). "Online variational inference for the hierarchical Dirichlet process". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 752–760.

Xiao, Han, Minlie Huang, and Xiaoyan Zhu (2016). "TransG : A Generative Model for Knowledge Graph Embedding". In: *ACL*.

Xie, Pengtao, Yuntian Deng, and Eric P. Xing (2015). "Latent Variable Modeling with Diversity-Inducing Mutual Angular Regularization". In: *CoRR* abs/1512.07336. arXiv: 1512.07336. URL: http://arxiv.org/abs/1512.07336.

Xiong, Liang et al. (2010). "Temporal collaborative filtering with bayesian probabilistic tensor factorization". In: *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, pp. 211–222.

Yang, Bishan et al. (2014). "Learning Multi-Relational Semantics Using Neural-Embedding Models". In: *CoRR* abs/1411.4072. arXiv: 1411.4072. URL: http://arxiv.org/abs/1411.4072.

Ying, Rex et al. (2018). "Graph Convolutional Neural Networks for Web-Scale Recommender Systems". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. London, United Kingdom: ACM, pp. 974–983. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219890. URL: http://doi.acm.org/10.1145/3219819.3219890.

Zhang, C. et al. (2017). "Advances in Variational Inference". In: *ArXiv e-prints*. arXiv: 1711.05597.

Zhang, Jingwei et al. (2014a). "Word Semantic Representations using Bayesian Probabilistic Tensor Factorization". In: *EMNLP*.

Zhang, Jingwei et al. (2014b). "Word Semantic Representations using Bayesian Probabilistic Tensor Factorization". In: *Proceedings of the 2014 Conference*

on Empirical Methods in Natural Language Processing (EMNLP). DOI: 10.3115/v1/d14-1161.

Zhao, Peilin and Tong Zhang (2015). "Stochastic Optimization with Importance Sampling for Regularized Loss Minimization". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1–9. URL: http://proceedings.mlr.press/v37/zhaoa15.html.