ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ - ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

τομέας: Ηλεκτρονικής & Υπολογιστών
εργαστήριο: Εργαστήριο Διαδραστικών Τεχνολογιών

Διπλωματική Εργασία

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του Πανεπιστημίου Πατρών

Αλκιβιάδη Παπαδόπουλου του Κωνσταντίνου

αριθμός μητρώου: 227355

Θέμα

Knowledge graph embedding for link prediction and entity classification

Επιβλέπων

Καθηγητής Νικόλαος Αβούρης

Αριθμός Διπλωματικής Εργασίας:

Πάτρα, Μάρτιος 2021

ΠΙΣΤΟΠΟΙΗΣΗ

Πιστοποιείται ότι η διπλωματική εργασία με θέμα

Knowledge graph embedding for link prediction and entity classification

του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

Αλκιβιάδη Παπαδόπουλου του Κωνσταντίνου

(Α.Μ.: 227355)

παρουσιάτηκε δημόσια και εξετάστηκε στο τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών στις

__/__/___

Ο Επιβλέπων

Νικόλαος Αβούρης
Καθηγητής

Ο Διευθυντής του Τομέα

Βασίλειος Παλιουράς
Καθηγητής

Στοιχεία διπλωματικής εργασίας

Θέμα: **Knowledge graph embedding for link prediction and entity classification**

Φοιτητής: **Αλκιβιάδης Παπαδόπουλος του Κωνσταντίνου**

Ομάδα επίβλεψης
Καθηγητής Νικόλαος Αβούρης

Γεώργιος Παλιούρας, Διυεθυντής Ερευνών, ΕΚΕΦΕ ΔΗΜΟΚΡΙΤΟΣ

Εργαστήρια

Εργαστήριο Διαδραστικών Τεχνολογιών

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Αλκιβιάδης Παπαδόπουλος

Abstract

Although machine learning has shown promising results and attracted great attention in the recent years, few methods can be applied to relational, or graph-structured, data. Knowledge Graph Embedding (KGE) is a learning methodology that attempts to bridge this gap by learning to represent nodes as vectors that can be used for various downstream tasks. A multi-relational, or knowledge, graph, i.e. a graph with multiple types of edges, is a general abstraction that can represent pairwise relationships between a set of entities such as friendships in a social network, citations and co-authorship in an academic network or even various types of atomic bonds in a molecule. In the recent years, a number of large knowledge bases have been assembled that contain up to several billions of entities and facts about them, represented as links between them. Since most large knowledge graphs are inherently incomplete, link prediction, which is the identification of missing links, is the standard task on which KGE models are evaluated and where they are able to achieve state-of-the-art performance. Entity classification, where the goal is to predict the class an entity belongs to, is another potential application of KGE models which has been less explored. For this thesis I implemented a KGE library which I used to evaluate various models and methodologies on link prediction for the benchmark datasets FB15K and WN18 as well as entity classification for spammer detection in a social network.

Keywords: Machine Learning, Knowledge Graphs, Knowledge Graph Embedding, Link Prediction, Entity Classification

Περίληψη

Παρόλο που η μηχανική μάθηση έχει δείξει πολλά υποσχόμενα αποτελέσματα και έχει προσελκύσει μεγάλη προσοχή τα τελευταία χρόνια, οι περισσότερες μέθοδοι δεν εφαρμόζονται σε σχεσιακά ή δομημένα σε γράφο δεδομένα. Η Ενσωμάτωση Γράφου Γνώσης (Knowledge Graph Embedding, KGE) είναι μια μεθοδολογία μάθησης που προσπαθεί να γεφυρώσει αυτό το κενό μαθαίνοντας να αντιπροσωπεύει κόμβους ως διανύσματα που μπορούν να χρησιμοποιηθούν για διάφορες εφαρμογές μηχανικής μάθησης. Ένας γράφος πολλαπλών σχέσεων ή γράφος γνώσεων, δηλαδή ένας γράφος με πολλούς τύπους άκρων, είναι μια γενική αφαίρεση που μπορεί να αντιπροσωπεύει δυαδικές σχέσεις μεταξύ οντοτήτων, όπως φιλίες σε ένα κοινωνικό δίκτυο, παραπομπές και συν-συγγραφή επιστημονικών δημοσιεύσεων σε ακαδημαϊκό δίκτυο ή ακόμα και διάφορους τύπους ατομικών δεσμών σε ένα μόριο. Τα τελευταία χρόνια, έχουν αναπτυχθεί πολλές μεγάλες βάσεις γνώσεων που περιέχουν έως και αρκετά δισεκατομμύρια οντότητες και γεγονότα για αυτές τα οποία αναπαριστώνται ως σύνδεσμοι μεταξύ τους. Δεδομένου ότι οι περισσότεροι μεγάλοι γράφοι γνώσης είναι εγγενώς ελλιπείς, η πρόβλεψη συνδέσμου, η οποία είναι ο προσδιορισμός των συνδέσμων που λείπουν, είναι η τυπική εφαρμογή στην οποία αξιολογούνται τα μοντέλα KGE και όπου μπορούν να επιτύχουν την καλύτερη δυνατή επίδοση. Η ταξινόμηση οντοτήτων, όπου ο στόχος είναι η πρόβλεψη της κατηγορίας στην οποία ανήκει η οντότητα, είναι μια άλλη πιθανή εφαρμογή μοντέλων KGE που έχει διερευνηθεί λιγότερο. Για αυτήν τη διατριβή, ανέπτυξα μια βιβλιοθήκη KGE την οποία χρησιμοποίησα για να αξιολογήσω διάφορα μοντέλα και μεθοδολογίες σχετικά με την πρόβλεψη συνδέσμων για τα σύνολα δεδομένων αναφοράς FB15K και WN18 καθώς και την ταξινόμηση οντοτήτων για ανίχνευση spammer σε ένα κοινωνικό δίκτυο.

Λέξεις Κλειδιά: Μηχανική Μάθηση, Γράφοι Γνώσης, Ενσωμάτωση Γράφου Γνώσης, Πρόβλεψη Συνδέσμου, Ταξινόμηση Οντοτήτων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους επιβλέποντες μου Γ. Παλιούρα και Γ. Κατσίμπρα για την βοήθεια και την υπομονή τους.

Επίσης θέλω να ευχαριστήσω τον υπεύθυνο καθηγητή Ν. Αβούρη που μου έδωσε την ευκαιρία να ασχοληθώ με αυτό το θέμα.

Τέλος θέλω να ευχαριστήσω την οικογένειά μου που με στίρηξε με όποιον τρόπο μπορούσε για όλη τη διάρκεια της φοιτητικής μου ζωής.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Introduction

Computer technology has been evolving at a great pace since the introduction of integrated circuits and by the late 2000s there was high availability in computer systems that were able to store, process and transmit large amounts of data each second. Since data collection was easier than ever before, the number of available datasets was by then large and rapidly growing. By 2012, it became apparent that the training of deep neural networks can be sped up by a factor of 50 or more by using graphics processing units (GPUs) and that those networks can significantly outperform all other methods in multiple computer vision and natural language processing tasks. Due to these factors, a growing number of researchers have shown interest in machine learning in the past decade.

Machine learning is the study of algorithms that learn through experience. It is commonly divided into three main categories: supervised learning, unsupervised learning and reinforcement learning. In supervised learning, a collection of input examples together with the corresponding desired output is provided and after training the algorithm is expected to predict the output values corresponding to new input examples. In unsupervised learning, the desired output is not known but a loss function that depends only on the input data is minimized. Reinforcement learning is quite distinct from the other two categories and it can be used to find optimal strategies for agents in environments that are modelled as Markov Decision Processes. This study focuses on supervised and unsupervised learning.

Since the early days of its inception in the 60s, machine learning has been traditionally applied to tabular data, i.e. collections of feature vectors of same size. Nevertheless, there exist various types of datasets that cannot be efficiently represented as tables of numbers. For example, the need to make predictions based on word sequences of variable size quickly arose in natural language processing and specialized methods have been developed to deal with this type of data such as recurrent neural networks, LSTMs and, more recently, attention based Transformers. Timeseries, including sounds, are another type of data that require specialized methods to process. A natural next step is to attempt to apply machine learning to graphs.

1.1 Learning from relational data

A graph represents a set of relationships (edges) between elements of a set of entities (nodes). It is a general concept, therefore a large amount of data can be found in this

form. However, traditional machine learning models cannot be directly used to extract useful information from such data since their input is usually restricted to Euclidean domains. Graph Embedding (GE) models are able to bridge this gap by learning low dimensional representations of nodes, edges or even whole graphs.

The graph concept is commonly refined by considering additional structure. For example, edges can be augmented with a weight or a direction leading to weighted and directed graphs respectively. Specialized models that take advantage of such additional information can often achieve greater performance. A multi-relational, or knowledge, graph is a graph with multiple types of relations. Multi-relational graphs can be used to model various situations but the most common categories are social networks, where entities are people who can have many different types of relationships, and knowledge bases, where the entities are heterogeneous and the edges represent facts about them. An example of a knowledge graph is shown in Figure 1.1.

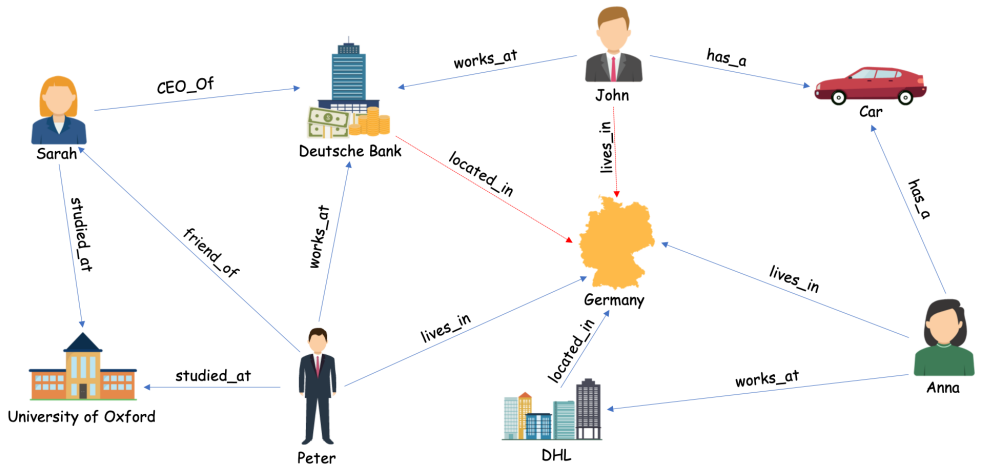

Figure 1.1: An illustration of a knowledge graph. (source: [1])

WordNet is one of the earliest knowledge graphs. It was created in the 1980s under the direction of George Armitage Miller, one of the founders of cognitive science, and it contains semantic relationships between words in more than 200 languages. In 2007 DBpedia and Freebase were independently founded. DBpedia was extracted mainly from Wikipedia and as of 2016 contained 6 million entities and 9.5 billion facts. Freebase was a collaborative knowledge base composed mainly by its community members but was discontinued in 2015, a few years after it was acquired by Google. In 2012, Google announced their Knowledge Graph project which was powered in part by Freebase, and has since been used to display information about google search terms in an infobox next to the results. Since then, many large multinational companies have announced their use of knowledge graphs, further popularizing the term.

Knowledge Graph Embedding methods learn to represent entities and relations in a knowledge graph as vectors in a latent space such that some notion of similarity between the entities is preserved. If first order proximity is preserved, linked entities

are expected to have similar embedding vectors. A more useful notion is the second order proximity, also referred to as structural or role-based similarity, which is the similarity between the neighbourhoods of two nodes. Figure 1.2 shows an embedding example that preserves structural similarity for the graph in Figure 1.

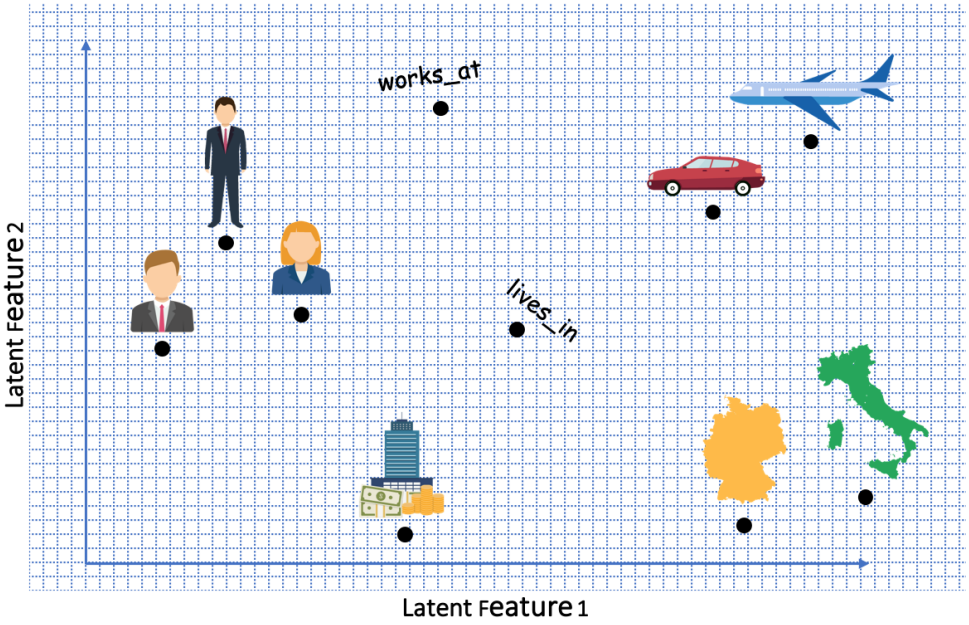

Figure 1.2: A possible embedding of the knowledge graph in Figure 1.1. (source: [1])

Many different techniques have been employed in graph embedding. Factorization based methods optimize an edge reconstruction based objective. Graph neural networks incorporate deep learning techniques and enable end-to-end semi-supervised learning. Random walk based methods generate node sequences from the graph and feed them to a sequence embedding method, eg. a word embedding model such as word2vec.

Graph embedding can be used to make predictions about the nodes, edges or whole graphs. Node level applications include node classification, node clustering and recommendation. Link prediction is a common edge level application, especially for knowledge graphs which are known to be highly incomplete. For example, in 2014 [38] found that out of all persons in Freebase 71% had no recorded place of birth and 99% had no ethnicity. Whole graph embedding can be used for tasks such as graph classification.

1.2 Motivation / Contribution

In the context of this thesis I will implement various knowledge graph embedding methods and I will study their performance on two tasks: link prediction and entity classification.

In link prediction, which is the most common application of knowledge graph embedding methods, the input graph is considered incomplete the goal is to identify the missing links that are most likely to be present in a (hypothetical) complete version of the graph. For the evaluation of a model on link prediction, a subset of the existing links are simply hidden from the model during training after which the model is expected to rank them higher than other missing links.

In entity classification, in addition to the knowledge graph, a set of labels corresponding to all or a subset subset of the entities is also provided. A subset of these labels is used for training and the goal is to predict the rest. Many distinct methodologies have been proposed for this task and in this thesis some of them will be compared.

1.3 Structure of the document

Section 2 presents an overview of the existing literature on knowledge graph embedding. In section 3 the knowledge graph embedding problem is defined and various related models and algorithms are described. Section 4 is a description of the implementations and methodologies that were used for the experiments. In section 5 the exact experimental setup used and the corresponding results are given for each task and each methodology. The conclusions of the study will be presented in the last section.

Related Work

Statistical Relational Learning (SRL) is a subdiscipline of Machine Learning that is concerned with the statistical analysis of relational data. SLR methods combine first-order logic formalisms and probabilistic graphical models in order to model statistical dependence between attributes of entities and/or relationships between entities. While SRL models have certain advantages such as explainability and the ability to extract complex patterns, their applicability is limited mainly because of their poor scalability. Knowledge Graph Embedding (KGE) can be motivated as a SRL approach where relationships are conditionally independent of each other given their latent representations of the entities and relations involved [23]. Nevertheless, KGE has been developped mostly independently of SRL and there is little interaction between the two fields. Graph Embedding (GE) is a related field that focuses on regular graphs, that is knowledge graphs with one type of relation. Several GE methods have been extended to knowledge graphs.

2.1 Knowledge graph embedding

Since 2011, the availability of large Knowledge Bases (KB), such as WordNet, DBPedia and Freebase, has motivated many researchers to search for efficient algorithms for knowledge base completion and other related tasks.

RESCAL [24] is one of the earliest KGE models which was originally proposed as a Tensor decomposition technique. A knowledge graph is equivalent to a set of graphs, one per relation, with common nodes and a tensor can be formed by stacking the adjecency matrices of these graphs. By decomposing this tensor into a product of low rank tensors, RESCAL is able to learn representations for entities and relations. It was introduced along with a specialized optimization method that could scale to large graphs.

Structured Embeddings [4] is an important early work that first introduced many of the techniques used to efficiently train KGE models today such as negative sampling and the margin based ranking loss. It also introduced the entity ranking protocol which is a standard procedure for evaluating KGE models on link prediction.

TransE [3] was proposed as a simple and scalable alternative to existing models which turned out to be surprisingly effective. Partially inspired by word embedding techniques such as word2vec [22], TransE models relations as translations in the entity embedding space. A number of related models have been subsequently developed

(TransH [37], TransR [21], etc.). DistMult [40] is simmilar to TransE in terms of simplicity, scalability and performance. Like RESCAL, it used a bilinear form to score triples but restricts the matrix of the form to a diagonal matrix which prevents it from being able to model asymmetric relations. ComplEx [35] solves this issue by considering complex embedding vectors and matrices.

While the models mentioned so far use simple scoring functions with no parameters, meaning that the only parameters are the embeddings themselves, others have opted for parametric scoring functions inspired by neural networks. Neural Tensor Networks (NTN) [33] employs a special neural network architecture to score triples but reqiures far too many parameters for practical scenarios. ER-MLP [7] is a simpler approach that uses a single hidden layer with a non-linear activation function and no bias.

DeepWalk [27] is a GE method that was inspired by word2vec which is a word embedding technique. It transforms the input graph into a set of sequences by performing random walks. These sequences can then be treated just like sentences in word embedding. This approach has been extended to knowledge graphs in RDF2vec [29].

A common characteristic of the above methods is that the embedding vectors are parameters of the optimization problem and they are directly optimized. These are called direct encoding methods. In contrast, Graph Neural Networks (GNNs) [39] is a family of methods that employ deep architectures that, starting with some initial representations which are simply constant and not considered as parameters, refine the node representations in each layer. The earliest GNNs were recursive networks that used the same neural component for each layer. Convolutional GNNs such as GCN [18], motivated by generalizing convolution to general graphs as opposed to grids, use different weights for each layer but tend to have fewer layers and are easier to compose with other components enabling a wide range of applications. While most GNNs are applicable to regular undirected graphs, some works, notably R-GCN [31], attempt to generalize them to multi-relational graphs.

Until recently, there were very few implementations of KGE models and most of them were not designed for GPUs, which is crucial for efficient training. Since 2019, several high-quality and efficient libraries have been released such as AmpliGraph [6], LibKGE [5] and PyKeen [2], that implement direct encoding methods, or DGL [36] and Pytorch Geometric [10], that implement GNNs.

2.2 Entity classification

Entity classification is the multi-relational counterpart of node classification. While Graph Embedding models are commonly evaluated on node classification, Knowledge graph embedding models are rarely evaluated on entity classification. Nevertheless, several methodologies have been proposed for entity classification and they can be divided into feature learning and end-to-end learning approaches.

In feature learning, entities are represented as feature vectors that can be fed to any conventional classifier. In manual feature engineering, a human selects a set of graph measures and analytics, such as node degree or centrality, to be used as features. KGE can remove the need for human intervention if the entity embeddings are used as

feature vectors. These approaches have been compared in [30].

In end-to-end entity classification, a classification model that depends on the input graph is directly optimized. CLASS-RESCAL [16] uses a loss function that penalizes not only the link prediction error of the RESCAL model but also the classification error of a classifier that uses the entity embedding vectors of that model as features. [24] proposed to augment the input KG with a new relation and to add the classes as entities connected to their members via this relation. Entity classification can be then cast as link prediction on the augmented graph. More recently, Relational Graph Convolutional Networks [32] were introduced to enable end-to-end supervised learning on knowledge graphs.

Knowledge graph embedding

3.1 Preliminaries

A multi-relational graph $G = (\mathcal{V}, \mathcal{R}, \mathcal{T}^+)$ consists of a set of nodes $\mathcal{V}$, a set of relations $\mathcal{R}$ and a set of positive triples $\mathcal{T}^+ \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ of the form (h, r, t) where $h \in \mathcal{V}$ is the head or subject entity and $t \in \mathcal{V}$ is the tail or object entity. A multi-relational graph is equivalent to a collection of graphs, one for each relation. As such, it can be represented as a set of $|\mathcal{V}| \times |\mathcal{V}|$ adjacency matrices that when stacked together form the adjacency tensor $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times |\mathcal{R}|}$.

In order to train a model for link prediction negative examples of triples are necessary. Under the closed-world assumption (CWA), the graph is considered complete and all missing triples are labeled as negative. In reality, however, knowledge graphs are known to be highly incomplete which means that many triples will be mislabeled under this assumption. Furthermore, the number of examples that are labeled negative is $|\mathcal{R}||\mathcal{V}|^2 - |\mathcal{T}^+|$ which becomes very large as the number of entities $|\mathcal{V}|$ grows since most graphs are extremely sparse. This leads to a disproportionately large number of negative examples which can hurt the predictive power of the model while also limiting the size of the input graph.

A more common approach is the locally closed world assumption (LCWA) which is to assume that a negative triple can only be the result of replacing the head or tail entity of a positive triple with any other entity, thus avoiding any assumptions about missing triples that don't share the same subject or object with any positive triple from the same relation. While the number of candidate negative triples under the LCWA is reduced to $\mathcal{O}(|\mathcal{T}^+||\mathcal{V}|)$, it's still large enough to pose the same issues as the CWA if all the candidates are labeled negative. Since there is no way of defining a preference between two candidate negative triples that differ only in one of the entities, most knowledge graph embedding methods generate a different random sample of negative triples for every epoch during the training of a model according to some heuristic. This way all candidate negative triples have a chance to be considered while the ratio of positive to negative triples within each epoch can be freely adjusted. The set of negative triples is denoted by $\mathcal{T}^-$, the set of all labeled triples is $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$ and

$$y_{hrt} = \begin{cases} 1 & \text{if } (h, r, t) \int \mathcal{T}^+ \\ -1 & \text{if } (h, r, t) \int \mathcal{T}^- \end{cases} \tag{3.1}$$

is the label of the triple $(h, r, t) \in \mathcal{T}$.

Link prediction is the task of ranking the unlabeled triples based on their probability of being positive. To this end, knowledge graph embedding methods based on link prediction learn a function $f\colon \mathcal{V} \times \mathcal{R} \times \mathcal{V} \to \mathbb{R}$ that assigns a score to every possible triple. The model is trained to assign high scores to positive triples and low scores to negative ones. By defining f as the composition of two functions, an encoder and a decoder, knowledge graph embedding methods are able to represent entities and relations as vectors in the intermediate domain which is called the embedding space.

The encoder ENC is the part of the model that generates the representations. A triple (h, r, t) is encoded by applying an entity encoder $ENC_{\mathcal{V}}\colon \mathcal{V} \to \mathbb{R}^{d_e}$ that maps the entities h and t to their d_e-dimensional embedding vectors $\mathbf{e}_h$ and $\mathbf{e}_t$ and a relation encoder $ENC_{\mathcal{R}}\colon \mathcal{R} \to \mathbb{R}^{d_r}$ that maps the relation r to a d_r-dimensional representation $\mathbf{w}_r$. The relation embedding dimension d_r is a function of the entity embedding dimension d_e which is a hyper-parameter.

The decoder DEC : $\mathbb{R}^{d_e} \times \mathbb{R}^{d_r} \times \mathbb{R}^{d_e} \to \mathbb{R}$, also called a scoring function, maps the representations generated by the encoder to a score. The score is usually a similarity measure between the embedding vectors of the subject and object entities under a transformation that depends on the representation of the relation. While some parametric decoders have been proposed, this study focuses on decoders without parameters that are simply functions of the embedding vectors.

The overall model is the composition of the encoder and decoder:

$$f(h, r, t) = \mathrm{DEC}(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t)$$

Most knowledge graph embedding methods employ direct encoders parameterized by an entity embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d_e}$ and a relation embedding matrix $W \in \mathbb{R}^{|\mathcal{R}| \times d_r}$ that simply output a different row of E and W for each entity and relation respectively. In this case, the embedding vectors of the entities and the relations are the only parameters of the optimization problem and are directly optimized.

3.2 Loss Functions

Various loss functions can be used to train a knowledge graph embedding model. They are divided into 3 categories according to the number of triples that they take as input: pointwise loss functions take a single triple which can be either positive or negative, pairwise loss functions take two triples, one positive and one negative while setwise loss functions take a set of triples containing one positive and a number of negative triples.

3.2.1 Pointwise Loss Functions

A pointwise loss function L assigns a cost to each labeled triple individually, as depicted in figure 3.1. The optimization problem takes the following form:

$$\min_{E,W} \sum_{(h,r,t) \in \mathcal{T}} L(y_{hrt}, f(h, r, t)) \tag{3.2}$$

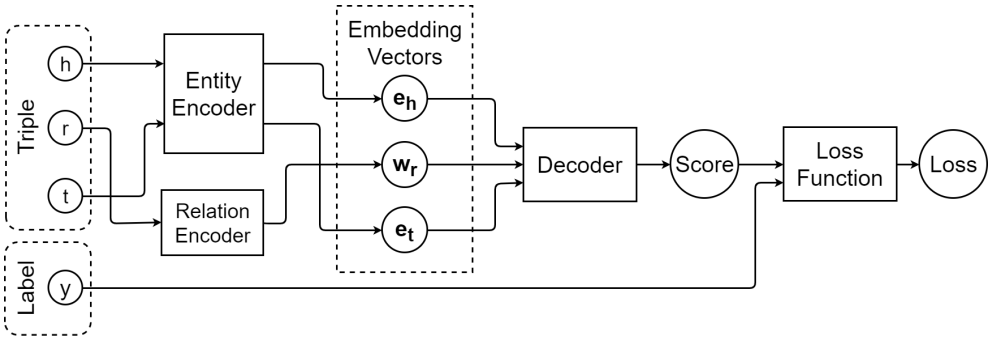

Figure 3.1: Model architecture for pointwise loss functions.

A common choice is the negative log likelihood of the logistic model, also known as logistic loss, that is given by the composition of the logistic function, which is a sigmoid function mapping $\mathbb{R}$ to the interval $(-1, 1)$, with the product $-y_{hrt}f(h, r, t)$:

$$L(y_{hrt}, f(h, r, t)) = \sum_{(h,r,t)\in\mathcal{T}} \log(1 + e^{-y_{hrt}\cdot f(h,r,t)}) \tag{3.3}$$

3.2.2 Pairwise Loss functions

A pairwise loss function assigns a cost to each pair of one positive and one negative example that only differ in one of the subject and object entities as a function of the difference of their scores. Figure 3.2 shows a diagram of the overall model. If L is a pairwise loss function, the following problem is solved:

$$\min_{E,W} \sum_{(h,r,t)\in\mathcal{T}^+} \left(\sum_{(h',r,t)\in\mathcal{T}^-} L\left(f(h, r, t) - f(h', r, t)\right) + \sum_{(h,r,t')\in\mathcal{T}^-} L\left((f(h, r, t) - f(h, r, t')\right) \right) \tag{3.4}$$

The margin based ranking loss (MRL) [4] is the most commonly employed pairwise loss function:

$$L\left(f(h, r, t) - f(h', r', t')\right) = \max(0, \gamma + f(h, r, t) - f(h', r', t')) \tag{3.5}$$

It assigns a cost to each pair where the score of positive triple is not greater than the score of the negative triple by at least some margin γ. Common values for the margin hyper-parameter are 0.5 and 1.

3.2.3 Setwise Loss functions

Set-wise loss functions assign a cost to each positive example by taking into account all negative examples generated by corrupting it. For example, [34] proposed a loss function based on an approximation of the Negative Log Likelihood (NLL) by defining:

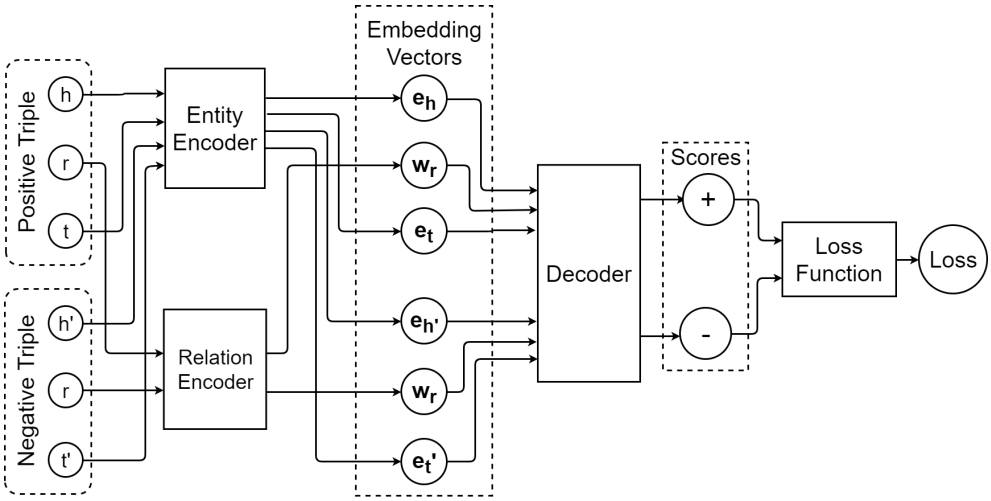

Figure 3.2: Model architecture for pairwise loss functions.

$$p(t|h,r) = \frac{e^{f(h,r,t)}}{\sum_{(h,r,t')\in\mathcal{T}^-} e^{f(h,r,t')}} \tag{3.6}$$

$$p(h|t,r) = \frac{e^{f(h,r,t)}}{\sum_{(h',r,t)\in\mathcal{T}^-} e^{f(h',r,t)}} \tag{3.7}$$

and then minimizing the following objective:

$$\min_{E,W} \sum_{(h,r,t)\in\mathcal{T}} (-\log(p(t|h,r)) - \log(p(h|t,r))) \tag{3.8}$$

3.2.4 Regularization

Overfitting is a common situation in machine learning where a model is trained to reproduce the labels labels of the training examples very accurately yet makes very poor predictions on test examples that are distinct enough from any training example. In knowledge graph embedding various regularization techniques have been proposed to reduce overfitting. One option is to constrain the entity embeddings to unit vectors by normalizing them after every update [4]. Another one is to add the absolute values ($p = 1$) or the squares ($p = 2$) of the parameters scaled by some constants λ_E and λ_W to the total loss $\mathcal{L}$, resulting in L_1 or L_2 regularization respectively:

$$\min_{E,W} \mathcal{L}(E,W) + \lambda_e \sum_{v\in\mathcal{V}} \|\mathbf{e}_v\|_p^p + \lambda_w \sum_{r\in\mathcal{R}} \|\mathbf{w}_r\|_p^p \tag{3.9}$$

3.3 Negative Sampling

For efficient training under the LCWA a heuristic for sampling negative triples is required. Common approaches generate negative triples by corrupting each positive triple $(h, r, t) \in \mathcal{T}^+$ by replacing either h or t with a random entity. Multiple negative triples can be generated for each positive one. Normally, all positive triples are excluded from being candidate negative triples. In practice however, checking whether a corrupted triple is positive is a relatively expensive operation and since $|\mathcal{V}^2| \gg |T^+|$, the probability of generating a positive triple by randomly corrupting another one is very low, therefore this check is often avoided.

In uniform negative sampling, there is an equal probability of replacing h and t. The authors of [37] noticed that some relations are one-to-many, meaning that more entities appear as objects than as subjects, while others are many-to-one. For the former, replacing the tail has higher probability of resulting in a positive triple, and vice versa for the latter. To address this issue, Bernouli negative sampling was proposed, which corrupts the head entity with probability $p_r = \frac{tph}{tph+hpt}$ and the tail entity with probability $1 - p_r$ for triples belonging to relation r, where tph is the ratio of unique tail entities to unique head entities and hpt is the ratio of unique head entities to unique tail entities in relation r

3.4 Decoders

The decoder, also referred to as the scoring function, is arguably the most important component, as it determines which elements of the embedding vectors are allowed to interact with each other. It also determines the relation embedding dimension d_r in terms of the chosen entity embedding dimension d_e. A summary of the decoders presented in this section is given in Table 3.1.

3.4.1 TransE

TransE [3] is one of the simplest scoring functions. It's called a translational model because it encodes relations as translations that move the embeddings of the subject entities as close as possible to those of the corresponding object entities. For example, if the input graph contains countries and cities as entities and a relation connecting countries with their capitals, the embedding vectors of Athens, Greece, Paris and France are expected to satisfy:

$$\mathbf{e}_{\text{athens}} - \mathbf{e}_{\text{greece}} \approx \mathbf{e}_{\text{paris}} - \mathbf{e}_{\text{france}} \approx \mathbf{w}_{\text{capital\_of}} \tag{3.10}$$

The relation embedding dimension is the same as the entity embedding dimension and the decoder is given by:

$$\text{DEC}(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = -\|\mathbf{e}_h + \mathbf{w}_r - \mathbf{e}_t\|_p \tag{3.11}$$

where $\|\mathbf{a}\|_p$ is the p-norm of $\mathbf{a}$ and p is either 1 or 2. The authors proposed the constraint $\|\mathbf{e}_v\| = 1, \forall v \in \mathcal{V}$ as a form of regularization. TransE is very efficient to train since the gradient can be computed in $\mathcal{O}(d_e)$ time.

3.4.2 RESCAL

RESCAL [24] represents each relation as a linear map under which the images of the embedding vectors of subject entities are similar to the embedding vectors of the corresponding object entities. The relation embedding dimension is the square of the entity embedding dimension and the decoder is a bilinear form:

$$\mathrm{DEC}(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = \mathbf{e}_h^\top W_r \mathbf{e}_t \tag{3.12}$$

where $\mathrm{vec}(W_r) = \mathbf{w}_r$. It is usually combined with L_2 regularization.

RESCAL is one of the most expressive models since all the elements of the embedding vectors of the subject and object entities are allowed to interact with each other, as shown in Figure 3.3, but it's also relatively expensive to train for the same reason.

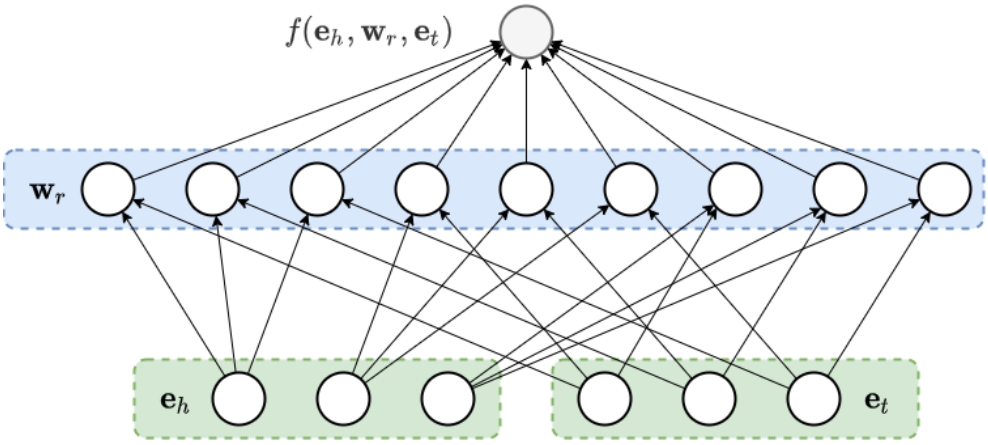

Figure 3.3: Interactions of the elements of the embedding vectors for RESCAL

3.4.3 DistMult

DistMult [40] uses a restricted bilinear form where the matrix is constrained to be diagonal:

$$\mathrm{DEC}(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = \mathbf{e}_h^\top W_r \mathbf{e}_t \tag{3.13}$$

where $W_r = \mathrm{diag}(\mathbf{w}_r)$ is the diagonal matrix formed by the elements of the vector $\mathbf{w}_r$. DistMult is very efficient to compute and has shown good performance on link prediction. One limitation is that interposing the subject and object entities does not change the score which means that DistMult ignores the direction of the links.

3.4.4 ComplEx

ComplEx [35] extends DistMult to complex representations:

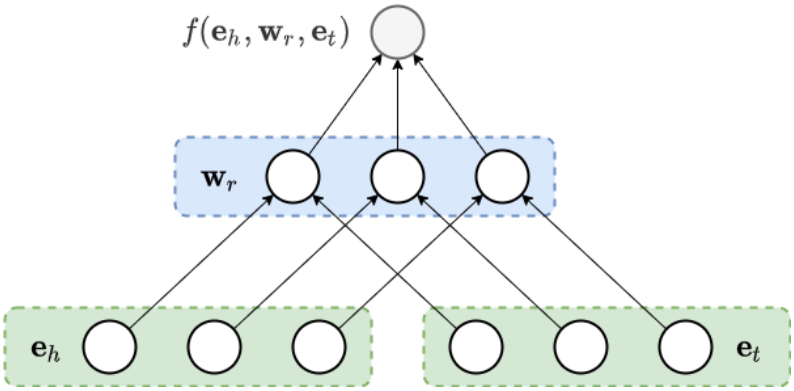

Figure 3.4: Interactions of the elements of the embedding vectors for DistMult

$$\begin{aligned}\text{DEC}(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = \text{Re}(\mathbf{e}_h^\top W_r \mathbf{e}_t^*) = &\langle \text{Re}(\mathbf{e}_h), \text{Re}(\mathbf{w}_r), \text{Re}(\mathbf{e}_t)\rangle \\ &+ \langle \text{Im}(\mathbf{e}_h), \text{Re}(\mathbf{w}_r), \text{Im}(\mathbf{e}_t)\rangle \\ &+ \langle \text{Re}(\mathbf{e}_h), \text{Re}(\mathbf{w}_r), \text{Im}(\mathbf{e}_t)\rangle \\ &- \langle \text{Im}(\mathbf{e}_h), \text{Im}(\mathbf{w}_r), \text{Im}(\mathbf{e}_t)\rangle\end{aligned} \tag{3.14}$$

where $\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t \in \mathbb{C}^{d_e}$, $W_r = \text{diag}(\mathbf{w}_r)$, x^* is the complex conjugate of $x \in \mathbb{C}$ and $\langle \mathbf{a}, \mathbf{b}, \mathbf{c}\rangle = \sum_i a_i b_i c_i$. Unlike DistMult, ComplEx is not symmetric with respect to the subject and object entities which allows it to model asymmetric relations.

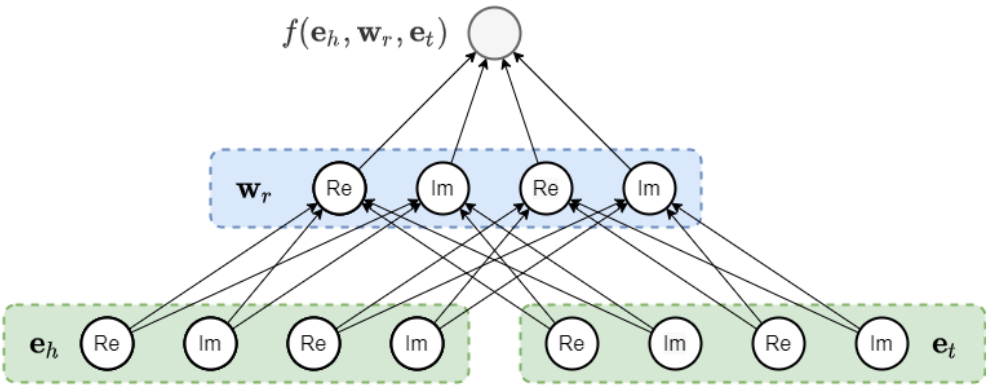

Figure 3.5: Interactions of the elements of the embedding vectors for ComplEx

Decoder	DEC($\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t$)	Number of parameters
TransE	$-\|\mathbf{e}_h + \mathbf{w}_r - \mathbf{e}_t\|_p$	$\|\mathcal{V}\|d_e + \|\mathcal{R}\|d_e$
RESCAL	$\mathbf{e}_h^\top W_r \mathbf{e}_t$	$\|\mathcal{V}\|d_e + \|\mathcal{R}\|d_e^2$
DistMult	$\mathbf{e}_h^\top W_r \mathbf{e}_t$	$\|\mathcal{V}\|d_e + \|\mathcal{R}\|d_e$
ComplEx	$\text{Re}(\mathbf{e}_h^\top W_r \mathbf{e}_t^*)$	$2\|\mathcal{V}\|d_e + 2\|\mathcal{R}\|d_e$

Table 3.1: Summary of decoders.

3.5 Optimization

3.5.1 Gradient Descent

The most common choice of optimization method in knowledge graph embedding and machine learning in general is gradient descent (GD) and its variants. GD is an iterative method which obtains a better estimate of the optimal parameters after each iteration, also called an epoch, starting from some initial values which are usually randomly sampled from some distribution. At each epoch, the parameter vector θ is updated by some amount proportional to the gradient of the loss L:

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta L(\theta_i) \tag{3.15}$$

The proportionality constant η is called the learning rate. If the learning rate is too low it will take too many iterations for GD to converge while if it's too high GD will overshoot and start oscillating. For this reason it's common to start with a relatively high learning rate and use a learning rate schedule, which is a strategy for decreasing the learning rate over time. Under a reasonable learning rate schedule, GD converges to the global minimum for convex loss functions and to a local minimum for non-convex ones. GD is favored when the objective is mostly smooth and convex and the problem is large enough for second order methods to be infeasible. GD has two main disadvantages:

- It can easily get trapped in the basin of attraction of a sub-optimal local minimum for highly non-convex objectives.
- When the loss is given as a sum over a set of examples, GD has linear time complexity with respect to the number of examples independently of any redundancy that is present. For example, duplicating every training example would double the time it takes for GD to converge to the same solution.

Stochastic Gradient Decsent (SGD) addresses both of these issues. Instead of following the exact gradient of the full loss, SGD follows a noisy approximation of it by performing each parameter update based on the gradient of the loss of one randomly selected example. This allows SGD to escape the basin of attraction of a local minimum and jump towards a potentially better one. Furthermore, the time required for each parameter update is independent of the number of examples therefore SGD can

converge faster than GD in the presence of redundancy.

While the stochastic nature of SGD can lead to better solutions for highly non-convex functions, it also complicates convergence for functions that are mostly convex. A balance between GD and SGD can be achieved with mini-batch stochastic gradient descent (MB-SGD) which uses more than one training example to approximate the gradient for each parameter update. This allows MB-SGD to retain some amount of stochasticity without sacrificing the good convergence characteristics of GD on smooth functions. In practice, the main disadvantage of SGD is that it is unable to take advantage of the parallel execution capabilities of modern hardware because the parameter updates must happen sequentially while the evaluation of the gradient for one training example is unlikely to saturate most modern hardware pipelines. With both GD and MB-SGD this issue can be eliminated by evaluating the gradient of multiple examples in parallel.

3.5.2 RESCAL-ALS

When training a knowledge graph embedding model under the CWA, the number of examples is $|\mathcal{V}|^2|\mathcal{R}|$ rendering GD and its variants prohibitively expensive for large graphs. For certain models, there exist algorithms that can find a solution in time that does not depend on $|\mathcal{V}|^2$ by taking advantage of the fact that most examples are negative which enables the use of sparse matrix operations. A notable example is RESCAL with direct encoding and a regularized squared error loss:

$$\min_{E,\mathbf{W}} f(\mathbf{X}, E, \mathbf{W})$$

where $\mathbf{W} \in \mathbb{R}^{d_e \times d_e \times |\mathcal{R}|}$ is a tensor with frontal slices $W_r \in \mathbb{R}^{d_e \times d_e}$ for $r = 1, 2, \ldots, |\mathcal{R}|$ and

$$f(\mathbf{X}, E, \mathbf{W}) = \left(\sum_{r=1}^{|\mathcal{R}|} \|X_r - E W_r E^\top\|_F^2 \right) + \lambda_E \|E\|_F^2 + \lambda_W \|W\|_F^2$$

RESCAL-ALS is an approximation of Alternating Least Squares (ALS) which is a cyclic block coordinate descent method for non-linear least squares problems. In ALS, the parameters are divided into a set of blocks and the algorithm cycles through each block optimizing it individually while holding the rest of the parameters constant. In traditional ALS, the optimization sub-problem for each block becomes a linear least squares problem which is solvable analytically.

RESCAL-ALS uses two parameter blocks E and $\mathbf{W}$. For $\mathbf{W}$ the optimization sub-problem can be solved analytically by setting $\frac{\partial f(\mathbf{X},E,\mathbf{W})}{\partial \mathbf{W}} = 0$, resulting in a formula for each slice W_r of the optimal $\mathbf{W}$:

$$\text{vec}(W_r) = ((E \otimes E)^\top (E \otimes E) + \lambda_W I)^{-1} (E \otimes E) \text{vec}(X_r) \tag{3.16}$$

where $A \otimes B$ is the Kronecker product of A and B. Naively evaluating this formula can be very expensive but the time complexity can be brought down to $\mathcal{O}(|\mathcal{V}|d_e^2 + |\mathcal{T}^+|d_e)$ by using the Singular Value Decomposition of E.

Unfortunately, the optimization sub-problem for E is still non-linear because E appears twice in the product $EW_rE^\top$. RESCAL-ALS settles for an approximation to the solution of the sub-problem by performing a single iteration of a fixed point iteration method. By setting $\frac{\partial f(\mathbf{X},E,\mathbf{W})}{\partial E} = 0$ and solving for one of the E factors that appears in the equation, the following update rule is obtained:

$$E \leftarrow \left[\sum_{r=1}^{|\mathcal{R}|}(X_rEW_r^\top + X_r^\top EW_r)\right]\left[\sum_{r=1}^{|\mathcal{R}|}(W_rE^\top EW_r^\top + W_r^\top E^\top EW_r) + \lambda_E I\right]^{-1} \tag{3.17}$$

While it's possible for a fixed point iteration to diverge, and no theoretical guarantees have been given for the convergence of RESCAL-ALS, in practice this update seems to almost always decrease the total loss. RESCAL-ALS tends to converge in a relatively small number of iterations and it is ideal for embedding large knowledge graphs under the CWA since the updates (13) and (14) can be computed in linear time with respect to the number of nodes $|\mathcal{V}|$ by using sparse representations for the slices of $\mathbf{X}$.

3.6 Evaluation

The performance of a KGE model on the link prediction task can be evaluated using the entity ranking protocol. The model is trained using only a subset of the positive triples while the rest are kept aside as a test set. During evaluation, for each positive triple (s, r, o) in the test set, the scores of all triples of the form $(?, r, o)$ are sorted and the rank of the test triple (s, r, o) is recorded. The same is done for all triples of the form $(s, r, ?)$. The recorded ranks can be aggregated in various ways. The most commonly employed metrics are:

- **Mean Rank (MR)**: The mean of the recorded ranks.
- **Mean Reciprocal Rank (MRR)**: The mean of the reciprocals of the recorded ranks.
- **Hits@k**: The percentage of the recorded ranks that were less than or equal to k.

By excluding positive triples that were seen during training when calculating the rank of each test triple, the filtered version of these metrics is obtained.

3.7 R-GCN

The graph embedding methods described so far are trained on a link prediction objective. While the representations that they produce can be used as features for down-

stream tasks, there is no guarantee that they will carry enough discriminative power since they were trained for a different objective. If the model f does not depend on the input graph in any way, an objective based on edge reconstruction is necessary for the input graph to enter the picture. Graph neural networks lift this restriction by employing an encoder that depends on the input graph allowing the labels to be used for any supervised or semi-supervised objective.

A popular graph neural network model for multi-relational graphs is the Relational Graph Convolutional Network (R-GCN) [32]. Given a source representation $\mathbf{h}_v^0$ for each node $v \in \mathcal{V}$, R-GCN can produce a series of new representations $\mathbf{h}_v^l$ by combining the representations of all nodes within the l-hop neighborhood of v. If numerical features for the entities are available, they can be used as the source representations, otherwise the original authors propose the use of a unique one-hot vector for each entity. For large graphs, one-hot encoding becomes inefficient and a direct encoder can be used to provide the source representations.

More specifically, if $\mathcal{N}_v^r$ is the set of neighbors of node $v \in \mathcal{V}$ in relation $r \in \mathcal{R}$, the representation of v in the l-th layer is given by:

$$\mathbf{h}_v^{l+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_v^r}\left(\frac{1}{c_{v,r}}W_r^l\mathbf{h}_u^l + W_0^l\mathbf{h}_v^l\right)\right) \tag{3.18}$$

where σ is an element-wise activation function such as the $\mathrm{ReLU}(x) = max(0, x)$, $c_{v,r}$ is either a normalization constant such as $|\mathcal{N}_v^r|$ or a parameter to be learned, $W_r^l, W_0^l \in \mathbb{R}^{d_{l+1}\times d_l}$ are weight matrices and d_l is the dimension of the l-th layer representations which is a hyper-parameter. Figure 3.6 shows the computational graph of a single layer.

By choosing an appropriate dimensionality d_L for the final layer L, the network can be trained on various supervised tasks. Regression can be performed by choosing $d_L = 1$ and using a loss function to measure the error between the output representation h_v^L and the label y_v of node v. For entity classification, if C is the set of classes, $d_L = |C|$ can be chosen and a classification loss function can be used such as the cross entropy loss. Link prediction is also possible by combining R-GCN with a decoder, in which case $d_L = d_e$ and the output of R-GCN are the node embedding vectors.

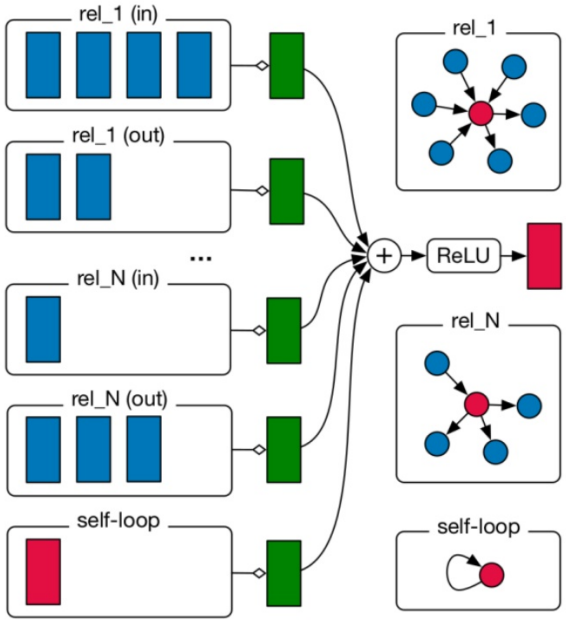

Figure 3.6: The computational graph of a layer of R-GCN. The representations of the previous layers are shown in blue. (source: [31])

Methodology

4.1 Knowledge Graph Embedding

Knowledge graph embedding is a relatively recent field. When I started this study, there were very few available implementations and most of them were focused on one particular model. It wasn't until 2019 that a number of high quality packages making use of GPU training were released. Nevertheless, since training KGE models on large graphs requires large amounts of memory and computation, many of them were designed for distributed training which meant that they were not very efficient for training on a single GPU. Furthermore, most of them were not structured as a library but rather as command line tools making them harder to compose and extend. PyKeen [2] is a notable exception which was released in 2020 but it was too late by then.

For these reasons, I decided to implement a modular and efficient python module[1] for knowledge graph embedding using PyTorch which is a generic optimization framework for machine learning models. This library is limited to direct encoding methods under the LCWA and was designed mainly for use on a single machine with a GPU with limited memory. It uses a simple edge list representation for the triples which are stored in CPU memory while model parameters reside in GPU memory. The embeddings are optimized with MB-SGD, optionally combined with any learning schedule or adaptive learning rate strategy provided by PyTorch. For each mini-batch, the corresponding triples are transferred to the GPU and the gradient of the loss is computed there and used to update the parameters.

One important feature of PyTorch is automatic differentiation. PyTorch provides special implementations for a wide range of functions and operators with known derivatives. In addition to the usual numerical values, these implementations can accept special objects called Tensors that keep a record of every operation performed on them. Given a model that is defined as a composition of these special implementations, PyTorch can automatically generate a function to compute the gradient of the model with respect to each parameter by evaluating the model using Tensors and applying the chain rule for differentiation on the resulting computational graph. This feature greatly simplifies the addition of new models and loss functions into the library.

The user of the library has to load a set of triples into a `KGraph` object. The data can

[1] `https://github.com/alkis-pap/kge`

be loaded from either a csv file or a NumPy array. They then need to initialize the following components:

- A **model**, which is a class that provides the decoder as well as a function that initializes the embedding matrices and optionally another one that is called after every mini-batch to normalize the embedding vectors, which is a form of regularization. The library provides 4 model classes: TransE, RESCAL, DistMult and ComplEx.
- A **loss function**. The library provides implementations for 2 loss functions, the pointwise logistic loss and pariwise margin based ranking loss, and a loss adaptor that can add L_1 or L_2 regularization to any loss function.
- An **optimizer**, which can be any optimizer from PyTorch such as SGD or Adam.

These components are passed to the `train` function together with the input graph and any additional hyper-parameters such as the batch size. Once the model is trained, it can be passed to the `evaluate` function together with a `KGraph` containing the test triples in order to calculate the MR, MRR and Hits@10 scores.

4.2 Hyper-parameter optimization

Graph embedding methods have many hyper-parameters and the chosen values can affect the performance of these methods significantly. In order to make fair comparisons between the various models, it is important to perform some form of Hyper-Parameter Optimization (HPO).

In HPO, the training examples are further split into a training and a validation set. The optimizer then performs a search in the hyper-parameter space, and for each configuration it trains the model on the training set and evaluates its performance on the validation set. Finally, the model with the highest validation score is selected and trained again on the union of the training and validation sets and evaluated on the test set to produce the final score. In contrast with most machine learning methods, the gradient of the loss is not available during HPO therefore a derivative-free optimizer is needed. In all HPO methods the user has to define the acceptable values for all hyper-parameters.

Grid search is the most basic HPO method which simply performs an exhaustive search over the parameter space. Unfortunately it's prohibitively expensive for large enough models with many hyper-parameters. Random search is a more effective strategy, where values are sampled according to some fixed distribution. In practice, this method tends to find better models in fewer iterations than grid search.

Bayesian optimization [11] is a family of derivative free optimization methods where the loss is modeled by a surrogate model, usually a Gaussian Process, which describes our prior beliefs and is updated after each observation. The resulting posterior distribution is used to determine the next configuration by maximizing the expected improvement under an acquisition function which trades off exploitation of collected observations and exploration of new configurations.

One shortcoming of the HPO methods mentioned so far is that they require a full evaluation of the function to be optimized for each selected configuration which can be very expensive. In the context of tuning KGE models, such an evaluation consists of training the model for a number of epochs and then evaluating it on the validation set. Nevertheless, for some configurations the model may converge relatively fast, rendering the subsequent training wasteful. Furthermore, good configurations tend to produce better results even with a small number of training epochs. The multi-armed bandit problem describes such a situation where a certain amount of resources is to be allocated to each candidate configuration and the goal is to maximize the gains. Successive Halving (SH) [15] proposes a solution to the multi-armed bandit problem. Given a minimum budget b_{min} and a maximum budget b_{max} and a halving factor $\eta \geq 2$ (usually either 2 or 3), SH randomly selects $\eta \frac{b_{\text{max}}}{b_{\text{min}}}$ configurations and evaluates them after b_{min} training epochs. Then the worst half of the configurations are discarded and the algorithm continues recursively with a new minimum budget of $2b_{\text{min}}$ until the maximum budget is reached at which point the last 2 configurations are evaluated and the best one is returned. An illustration of this procedure for $\eta = 2$ is shown in Figure 4.1.

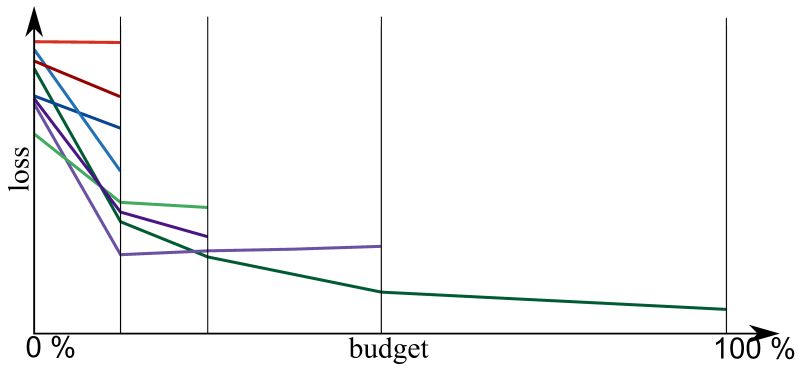

Figure 4.1: A run of Successive Halving. (source: `https://www.automl.org/blog_bohb/`)

A disadvantage of SH is that some configurations may require a budget larger than b_{min} to start showing good results, but SH will discard some of them in the first round. HyperBand [20] addresses this issue by running multiple instances of SH with different minimum budgets. Specifically, it runs for $\log_2(\frac{b_{\text{max}}}{b_{\text{min}}}) + 1$ iterations each of which performs SH with a minimum budget of $(\eta)^k b_{\text{max}}$ and returns the best configuration from all iterations. This is described in pseudo-code in Algorithm 1.

Input: b_{min}, b_{max}, η
$s_{\text{max}} = \left\lfloor \log_\eta(\frac{b_{\text{max}}}{b_{\text{min}}}) \right\rfloor$
for $s \in \{s_{\text{max}}, s_{\text{max}} - 1, \ldots, 0\}$ **do**
  Run SH with initial budget $\eta^s b_{\text{max}}$, max budget b_{max} and halving factor η
end

Algorithm 1: HyperBand

SH and, by extension, HyperBand use random search to select configurations. BOHB [9] extends HyperBand by employing a Bayesian optimization method, specifically a modified Tree-Structured Panzer Estimator (TPE), for selecting candidate configurations.

4.3 Entity Classification

Various methodologies have been proposed for entity classification but most of them fall into one of the following categories: feature extraction, and end-to-end learning.

4.3.1 Feature Extraction

This class of methods attempt to represent entities as feature vectors that can be fed to any traditional classifier. Feature extraction can be supervised or unsupervised depending on whether the classification labels are taken into account during the representation learning step or not. In this study following unsupervised feature extraction methodologies were used:

- **Handcrafted Features**: This methodology relies on a human to define a reasonable way of generating features for each node in the graph. Various graph analytics can be used including node degree and centrality measures such as PageRank. Features extracted this way cannot conserve all the information present in the graph therefore the human has to decide what to forget. This is not ideal since humans tend to be biased and error-prone, nevertheless this methodology can achieve good performance. In this study, a simple baseline is used where each node has two features per relation, indicating the number of incoming and outgoing links.
- **Graph Embedding**: The embedding vectors generated by a graph embedding method can be used as features in a downstream classification task. The hope is that the embeddings will contain most of the mutual information between the input graph and the labels and that a classifier will be able to extract this information.

The extracted representations were fed to the following classifiers as implemented in the Python module `scikit-learn` [26]:

- **k-Nearest Neighbors** (KNN): This classifier predicts the label of a test example as the most frequent class in the labels of the k training examples that are closest to the test example in terms of feature vectors.
- **Gaussian Naive Bayes** (NB): This classifier combines Baye's theorem with the 'naive' assumption that the features are samples of independent and random variables following a different normal distribution for each class.
- **Support Vector Machine** (SVM): Finds the hyperplane that separates two classes with the highest margin in a high (possibly infinite) dimensional space defined by a non-linear transformation of the feature vectors by making use of the kernel trick.

- **Decision Trees** (C4.5): This classifier builds a decision tree by recursively locating the feature with the highest information gain with respect to the class labels.

4.3.2 End-to-End learning

In end-to-end learning a single objective function is optimized. In this study two such methodologies were evaluated:

- Link prediction can be used for entity classification by including label information in the graph. For example, the classes can be added as nodes in the graph and each entity label can be represented as a link between the entity and the class. Knowledge graph embedding methods can be used to induce a ranking of the classes for each entity and the most probable class can be identified this way.
- Models that depend on the input graph such as GNNs can be used for end-to-end supervised learning, including entity classification. For this study a two-layer R-GCN with the cross entropy loss was implemented using DGL [36] and PyTorch [25].

Experiments

5.1 Link Prediction

5.1.1 Datasets

For the link prediction tasks the datasets FB15K and WN18 were used. They are widely used datasets for benchmarking knowledge graph embedding methods on link prediction.

FB15K
FB15K was extracted [3] from Freebase, a large collaborative knowledge base composed mainly by its community members. It is a sample of 592,213 triples involving 14,951 entities and 1,345 relations. At that time Freebase had around 1.2 billion triples and more than 80 million entities. Since then, Freebase has been discontinued in favor of Wikidata but FB15K is still used to compare embedding methods. In [34] the authors noticed that that for 81% of the test triples (h, r, t) in FB15K there was at least one triple of the form (h, r', t) or (t, r', h). Due to this unrealistic correlation between training and test data, extremely simple methods were able to outperform state-of-the-art models. FB15K-237 was created by removing rare, near-duplicate and inverse relations, resulting in a dataset with 237 relations and with no test triples sharing both entities with a training triple.

WN18
WordNet (WN) is a knowledge base containing word senses (two senses of the same word are represented as different entities) and lexical relations between them. Examples of relations are hyponymy, hypernymy and parthood. WN18 is a sample of WN extracted [13] by dropping infrequent relations and entities. WN18 suffers from the same issues as FB15K and WN18RR was created to address them using the same procedure as for FB15K-237.

5.1.2 Experimental Setup

For each of the above datasets (FB15K, FB15K-237, WN18 and WN18RR) the data was first augmented with inverse triples by adding a new relation r' for each relation r and a new triple (t, r', h) for each positive triple (h, r, t) and then the performance of 3 models - TransE, DistMult and ComplEx - was evaluated. Unfortunately RESCAL

Parameter	Distribution	Q	Candidate values
Embedding dimension	Uniform	10	$[50, 300]$
Negative triples per positive one	Uniform	2	$[2, 100]$
Loss function	Categorical	-	{ MRL, Logistic, NLL }
Regularization method	Categorical	-	{ Normalization, L_p-Regularization }
Learning Rate	LogUniform	-	$[0.01, 0.000001]$
Batch size	Categorical	-	{1000, 2000, 5000}
L_p-Regularization p	Categorical	-	{ 1, 2, 3 }
L_p-Regularization coefficient	LogUniform	-	$[0.1, 0.00001]$
TransE norm p	Categorical	-	{ 1, 2 }

Table 5.1: Hyper-parameter ranges for KGE models. Some parameters were quiantized with step Q.

required too much memory and time for the scope of this study. For each model, hyper-parameter tuning was performed to maximize the Mean Reciprocal Rank (MRR) on the validation set using an implementation of BOHB[1] with a halving factor of 2, a minimum budget of 100 epochs and a maximum budget of 800 epochs. Table 5.1 shows the hyper-parameters that were tuned and their respective ranges.

For all experiments, uniform negative sampling with equal corruption frequency for the subject and object entities was used without checking if the generated triples are actually positive. The models were optimized with Adam [17]. The best configuration for each model and dataset was subsequently trained on the combined training and validation sets for 800 epochs and evaluated on the test set to produce the final results.

5.1.3 Results

The results of BOHB for TransE, DistMult and ComplEx on FB15K-237 and WN18RR are presented in Table 1 and 2 respectively. They contain the final Mean Rank (MR), Mean Reciprocal Rank (MRR) and the proportion of hits in the top 10 (Hits@10) that each model achieved on the test set as well as the best found values for some important hyper-parameters.

Despite its inability to model asymmetric relations, DistMult achieved the best performance on both datasets. It is important to note that the available budget for hyper-parameter tunning was considerably low. KGE models can take several thousands of epochs to converge to the best solutions and there are many hyper-parameters to tune. Furthermore, BOHB performs a simple random search for the first few iterations until enough results are collected. It's likely that a higher maximum budget was needed for the Bayesian Optimization component of BOHB to have a significant effect. Therefore the relatively low scores of TransE and ComplEx may very well be

[1] `https://github.com/automl/HpBandSter`

	Scores			Best hyper-parameters			
Model	MR	MRR	Hits@10	Loss	d_e	# Neg.	Regularization
TransE (p=1)	1149.73	0.1865	0.3064	NLL	120	84	Norm.
DistMult	**398.29**	**0.2292**	**0.3915**	MRL	190	4	L_2 ($\lambda = 0.0075$)
ComplEx	782.66	0.1508	0.2802	MRL	280	28	Norm.

Table 5.2: Link prediction results for FB15K-237.

	Scores			Best hyper-parameters			
Model	MR	MRR	Hits@10	Loss	d_e	# Neg.	Regularization
TransE (p=2)	9243.86	0.2307	0.3698	MRL	150	6	Norm.
DistMult	**4320.64**	**0.4164**	**0.4836**	NLL	120	88	Norm.
ComplEx	10925.93	0.3557	0.3628	Logistic	120	70	L_2 ($\lambda = 0.001$)

Table 5.3: Link prediction results for WN18RR.

due to resource constraints.

5.2 Entity Classification

5.2.1 Dataset

For entity classification, the dataset published in [8] was used, which contains all user interactions from the social networking website tagged.com that happened within a sampling period of 10 days. There are 7 types of interactions including viewing another user's profile, sending friend requests and sending messages. For each user action, the sender, recipient and timestamp were recorded. Some data about each user is also available including 3 features and a label indicating whether this user was found to be malicious (spammer) by the administrative team of the website within some unspecified amount of time after the end of the 10-day sampling period.

A multi-relational graph can be extracted from this dataset by including users as nodes, action types as relations and the existence of at least one action between two users as a link in the corresponding relation. A downside of this method is that it ignores the timestamps as well as the multiplicity of the actions. Nevertheless the goal of this study is not necessarily to find the best method for spammer detection but to study the performance of various methods for entity classification.

The multi-relational graph extracted from this dataset contains 5,607,454 entities and more than 300 million links. The large number of entities leads to a high memory requirement for each dimension of the entity embeddings. Since most parameters are updated for every mini-batch so they must be present on the GPU at all times. Because of the limited capacity of GPU memory, not enough dimensions could fit to

Sampler	Entities	Edges	Spammer ratio
Random Walk (RW)	10000	100938	0.1292
Random Walk (RW)	50000	1000594	0.1153
Forest Fire (FFS)	10000	113246	0.1374
Forest Fire (FFS)	50000	1184603	0.1346
Frontier Sampling (FS)	10000	87088	0.1237
Frontier Sampling (FS)	50000	1009920	0.1197

Table 5.4: Statistics of the sampled datasets for spammer classification.

accurately represent the entities. Furthermore, because of the large number of triples, the optimization could take many hours to converge.

Due to these limitations and in order to avoid distributed training, I decided to extract smaller samples from the graph. In general, it's impossible to preserve all properties of a graph when extracting a sample. With uniformly random node sampling, the average node degree drops towards 0 for small enough sample sizes. Uniform edge sampling is also problematic since the number of included entities grows very large even for relatively small sample sizes. More advanced graph sampling methods [14] are able to better preserve the node degree distribution. Using the python module Little Ball of Fur[2], samples of various sizes were created with the following graph sampling methods:

- **ForestFire Sampling (FFS)** [19] generates a sequence of disjoint node sets by starting with a small subset $V_0 \subset \mathcal{V}$ of nodes and creating a new set V_i at every i-th iteration containing a geometrically distributed random number of the neighbors of V_{i-1} that haven't been visited already.
- **Random Walk Sampling (RWS)** [12] starts with a single node and randomly follows one neighbor at a time.
- **Frontier Sampling (FS)** [28], also called multi-dimensional random walk sampling, starts with a set of nodes V_0. At every i-th iteration, a node $v \in V_{i-1}$ is selected at random with a probability that is proportional to degree of v and a uniformly sampled neighbor u of v is added to the set, ie. $V_i = \{u\} \cup V_{i-1}$.

Detailed statistics about the sampled graphs are shown in Table 5.4.

5.2.2 Experimental Setup

For end-to-end binary entity classification, an implementation of R-GCN with the cross entropy loss included in DGL[3] was used, and its hyper-parameters were tuned by maximizing the Average Precision on the validation set using the Tree-Structured Panzer Estimator (TPE), which is a Bayesian Optimization approach, as implemented

[2] `https://github.com/benedekrozemberczki/littleballoffur`
[3] `https://github.com/dmlc/dgl`

Parameter	Type	Quantization	Range
Hidden dimension	Uniform	10	$[50, 300]$
Number of Bases	Uniform	1	$[1, \lvert\mathcal{R}\rvert]$
Dropout	Uniform	-	$[0, 0.5]$
Self loop	Categorical	-	$\{\text{True}, \text{False}\}$
Learning Rate	LogUniform	-	$[0.0001, 1]$

Table 5.5: Hyper-parameter ranges for RGCN.

in the Python module Hyperopt[4]. The hyper-parameters that were chosen to be tuned and their respective ranges are shown in Table 5.5. The models were optimized using Adam with 10% of the examples as the final test set, 10% as a validation set for HPO, another 10% as a validation set for early stopping and the remaining 70% for training. The final scores were produced after retraining the model using 80% of the examples while keeping 10% for early stopping.

5.2.3 Results

The performance of the best configurations was evaluated by measuring the following metrics:

- Accuracy: The ratio of the correctly predicted labels.
- Balanced Accuracy: The arithmetic mean of the sensitivity (true positive rate) and specificity (true negative rate). This metric is better suited for imbalanced datasets than the regular accuracy.
- Average Precision (AP): The average value of the precision as a function of recall over all possible thresholds. It is equal to the area under the Precision-Recall curve.
- Area under the Receiver Operating Characteristic curve (ROC-AUC): The ROC curve is a plot of the True Positive Rate, which is the ratio of the positive examples that were classified as such and is also known as the sensitivity or recall, versus the False Negative Rate, which the ratio of negative examples that were classified as positive.

The ROC curves for the best configurations on the various datasets are shown in Figure 5.1. While better than random, the classifier did not perform well on the smaller datasets. The Precision-Recall curves are shown in Figure 5.2.

[4] `https://github.com/hyperopt/hyperopt`

Dataset		Scores			
Sampler	Entities	Accuracy	Balanced acc.	AP	ROC-AUC
RW	10000	0.855	0.5645	0.2399	0.6659
RW	50000	0.8842	0.5813	0.3733	0.7801
FFS	10000	0.859	0.6204	0.3355	0.6629
FFS	50000	0.8736	0.6212	**0.4774**	**0.8008**
FS	10000	0.859	0.5365	0.2597	0.6352
FS	50000	**0.8922**	**0.6719**	0.4769	0.7861

Table 5.6: Entity classification results for R-GCN

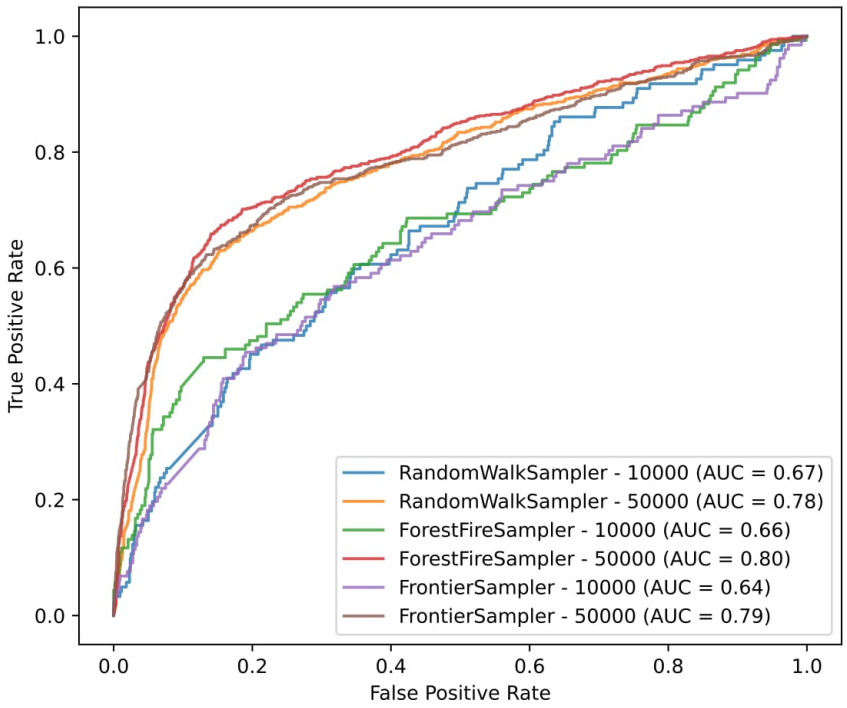

Figure 5.1: The ROC curves of R-GCN for different sampling strategies and sizes.

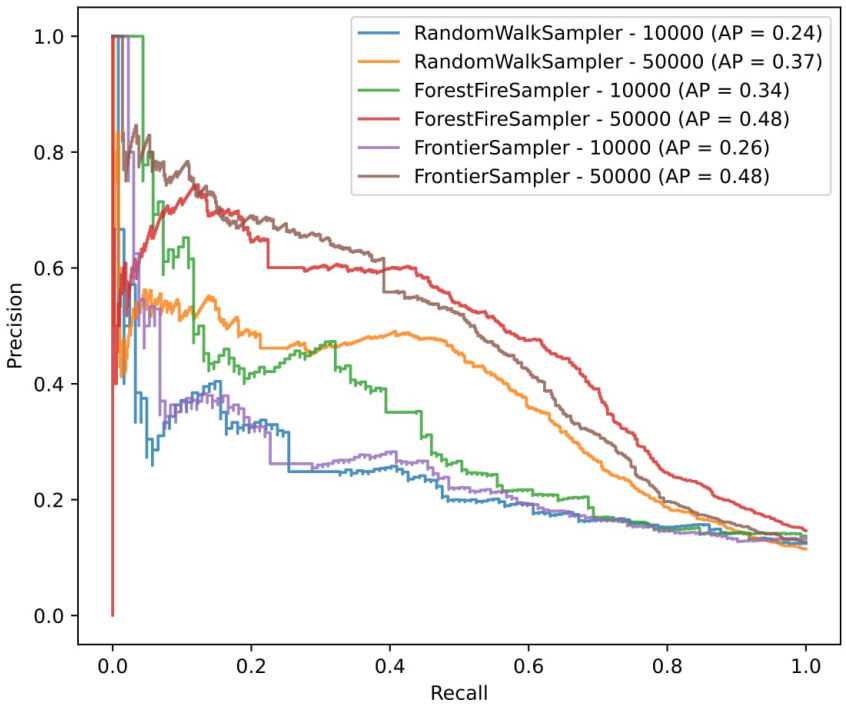

Figure 5.2: The Precision-Recall curves of R-GCN for different sampling strategies and sizes.

Conclusions

Graphs are ubiquitous and carry information that can be used to solve various tasks. Traditional machine learning methods are not directly applicable to graph-structured data but graph embedding methods provide a way to encode the information present in graphs in a low dimensional space, thus enabling various applications. In the context of this thesis I studied and implemented several knowledge graph embedding methodologies and evaluated them on multiple datasets and tasks in an effort to assess their ability to extract useful information from large multi-relational graphs.

The first set of experiments were on the link prediction task for which I implemented a python module based mainly on PyTorch. The models TransE, DistMult and ComplEx were compared on two datasets, FB15K-237 and WN18RR, after tuning up to 9 of their hyper-parameters using BOHB with a limited budget in terms of training epochs. Previous studies have shown that the performance of these models should be very similar after optimizing their hyper-parameters. The results were favorable for DistMult on both datasets indicating that it is less sensitive to the chosen values compared to the other models that required a higher budget for tuning.

The second set of experiments were on entity classification in a social network where users are linked with 7 types of relations indicating different interactions between them and where some of them have been labeled as spammers. The dataset was too large to process therefore 3 different sampling algorithms were used to create samples of two sizes. The performance of R-GCN as an end-to-end classifier was evaluated on these samples after tuning 5 of its hyper-parameters using TPE. The results show that R-GCN was able to perform singificantly better than a random classifier indicating that it was able to extract discriminative information from the graph.

Εκτενής Περίληψη

Εκτενής Περίληψη

8.1 Εισαγωγή

Η τεχνολογία των υπολογιστών εξελίσσεται με μεγάλο ρυθμό από την εισαγωγή ολοκληρωμένων κυκλωμάτων και στα τέλη της δεκαετίας του 2000 υπήρχε μεγάλη διαθεσιμότητα σε συστήματα υπολογιστών που ήταν σε θέση να αποθηκεύουν, να επεξεργάζονται και να μεταδίδουν μεγάλες ποσότητες δεδομένων κάθε δευτερόλεπτο. Δεδομένου ότι η συλλογή δεδομένων ήταν ευκολότερη από ποτέ, ο αριθμός των διαθέσιμων συνόλων δεδομένων ήταν τότε μεγάλος και ταχέως αναπτυσσόμενος. Μέχρι το 2012, κατέστη προφανές ότι η εκπαίδευση βαθέων νευρωνικών δικτύων μπορεί να επιταχυνθεί με συντελεστή 50 ή περισσότερων με τη χρήση μονάδων επεξεργασίας γραφικών (GPU) και ότι αυτά τα δίκτυα μπορούν να ξεπεράσουν σημαντικά όλες τις άλλες μεθόδους σε υπολογιστική όραση και επεξεργασία φυσικής γλώσσας. Λόγω αυτών των παραγόντων, ένας αυξανόμενος αριθμός ερευνητών έχει δείξει ενδιαφέρον για τη μηχανική μάθηση την τελευταία δεκαετία.

Η μηχανική μάθηση είναι η μελέτη αλγορίθμων που μαθαίνουν μέσω της εμπειρίας. Συνήθως χωρίζεται σε τρεις κύριες κατηγορίες: εποπτευόμενη μάθηση, μάθηση χωρίς επίβλεψη και ενισχυτική μάθηση . Στην εποπτευόμενη εκμάθηση, παρέχεται μια συλλογή παραδειγμάτων εισόδου μαζί με την αντίστοιχη επιθυμητή έξοδο και μετά την εκπαίδευση ο αλγόριθμος αναμένεται να προβλέψει τις τιμές εξόδου που αντιστοιχούν σε νέα παραδείγματα εισόδου. Στην μη επιτηρούμενη μάθηση, η επιθυμητή έξοδος δεν είναι γνωστή, αλλά ελαχιστοποιείται μια συνάρτηση κόστους που εξαρτάται μόνο από τα δεδομένα εισόδου. Η ενισχυτική μάθηση είναι αρκετά διαφορετική από τις άλλες δύο κατηγορίες και μπορεί να χρησιμοποιηθεί για την εύρεση βέλτιστων στρατηγικών για υποκείμενα (agents) σε περιβάλλοντα που διαμορφώνονται ως Διαδικασίες Απόφασης Markov. Σε αυτήν τη μελέτη θα επικεντρωθώ στην εποπτευόμενη και χωρίς επίβλεψη μάθηση.

Από τις πρώτες μέρες της έναρξής της στη δεκαετία του '60, η μηχανική μάθηση εφαρμόζεται παραδοσιακά σε πίνακες δεδομένων, δηλαδή. συλλογές διανυσμάτων χαρακτηριστικών ίδιου μεγέθους. Ωστόσο, υπάρχουν διάφοροι τύποι συνόλων δεδομένων που δεν μπορούν να αναπαρασταθούν αποτελεσματικά ως πίνακες αριθμών. Για παράδειγμα, η ανάγκη πρόβλεψης βασισμένων σε ακολουθίες λέξεων μεταβλητού μεγέθους προέκυψε γρήγορα στην επεξεργασία φυσικής γλώσσας και αναπτύχθηκαν εξειδικευμένες μέθοδοι για την αντιμετώπιση αυτού του τύπου δεδομένων, όπως επαναλαμβανόμενα νευρωνικά δίκτυα, LSTM και, πιο πρόσφατα, Transformers που

βασίζονται σε μηχανισμούς προσοχή (attention mechanism). Οι χρονοσειρές, συμπεριλαμβανομένων των ήχων, είναι ένας άλλος τύπος δεδομένων που απαιτούν εξειδικευμένες μεθόδους επεξεργασίας. Ένα φυσικό επόμενο βήμα είναι η προσπάθεια εφαρμογής μηχανικής μάθησης σε γράφους.

8.1.1 Μάθηση από σχεσιακά δεδομένα

Ένας γράφος αντιπροσωπεύει ένα σύνολο σχέσεων (ακμών) μεταξύ στοιχείων ενός συνόλου οντοτήτων (κόμβοι). Είναι μια γενική ιδέα, επομένως υπάρχει μεγάλη ποσότητα δεδομένων σε αυτήν τη μορφή. Ωστόσο, τα παραδοσιακά μοντέλα μηχανικής εκμάθησης δεν μπορούν να χρησιμοποιηθούν άμεσα για την εξαγωγή χρήσιμων πληροφοριών από τέτοια δεδομένα, δεδομένου ότι η εισαγωγή τους περιορίζεται συνήθως σε τομείς Ευκλείδειας. Τα μοντέλα Graph Embedding (GE) μπορούν να γεφυρώσουν αυτό το κενό μαθαίνοντας χαμηλές διαστάσεις αναπαραστάσεις κόμβων, ακμών ή ακόμη και ολόκληρων γράφων.

Η έννοια του γράφου είναι συνήθως εκλεπτυσμένη λαμβάνοντας υπόψη την πρόσθετη δομή. Για παράδειγμα, οι ακμές μπορούν να αυξηθούν με ένα βάρος ή μια διεύθυνση που οδηγεί σε σταθμισμένους και κατευθυνόμενους γράφους αντίστοιχα. Τα εξειδικευμένα μοντέλα που εκμεταλλεύονται αυτές τις πρόσθετες πληροφορίες μπορούν συχνά να επιτύχουν μεγαλύτερη απόδοση. Ένας γράφος πολλαπλών σχέσεων ή γνώσεων είναι ένας γράφος με πολλαπλούς τύπους σχέσεων. Οι πολυ-σχεσιακοί γράφοι μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση διαφόρων καταστάσεων, αλλά οι πιο συνηθισμένες κατηγορίες είναι τα κοινωνικά δίκτυα, όπου οι οντότητες είναι άτομα που μπορούν να έχουν πολλούς διαφορετικούς τύπους σχέσεων και βάσεις γνώσεων, όπου οι οντότητες είναι ετερογενείς και οι ακμές αντιπροσωπεύουν γεγονότα σχετικά με αυτές. Ένα παράδειγμα ενός γράφου γνώσης φαίνεται στο Σχήμα 8.1.

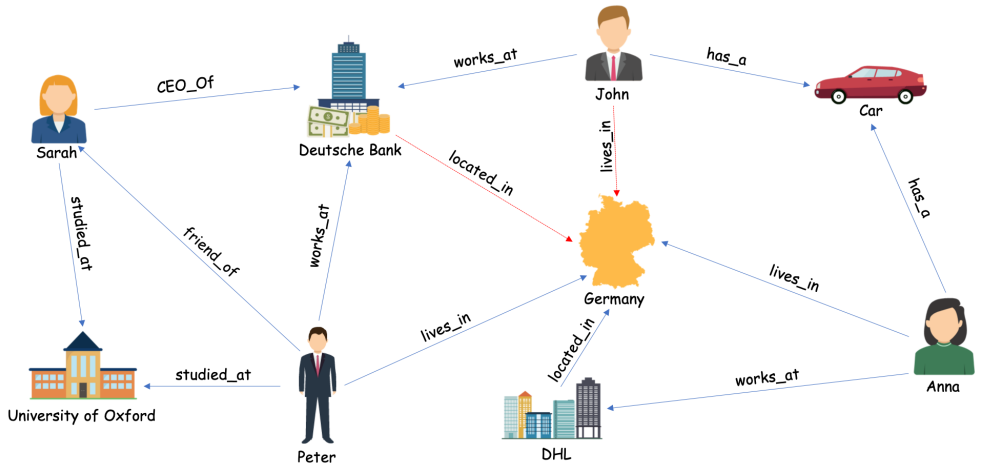

Figure 8.1: An illustration of a knowledge graph. (source: [1])

Το WordNet είναι ένας από τους πρώτους γράφους γνώσης. Δημιουργήθηκε τη δε-

καετία του 1980 υπό τη διεύθυνση του George Armitage Miller, ενός από τους ιδρυτές της γνωστικής επιστήμης και περιέχει σημασιολογικές σχέσεις μεταξύ λέξεων σε περισσότερες από 200 γλώσσες. Το 2007 ιδρύθηκαν ανεξάρτητα το DBpedia και το Freebase. Η DBpedia εξήχθη κυρίως από τη Wikipedia και από το 2016 περιείχε 6 εκατομμύρια οντότητες και 9,5 δισεκατομμύρια γεγονότα. Το Freebase ήταν μια συλλογική βάση γνώσεων που αποτελείται κυρίως από τα μέλη της κοινότητάς της, αλλά διακόπηκε το 2015, λίγα χρόνια μετά την απόκτησή της από την Google. Το 2012, η Google ανακοίνωσε το έργο Γράφο Γνώσης που τροφοδοτήθηκε εν μέρει από το Freebase και έκτοτε χρησιμοποιείται για την εμφάνιση πληροφοριών σχετικά με τους όρους αναζήτησης του Google σε ένα infobox δίπλα στα αποτελέσματα. Έκτοτε, πολλές μεγάλες πολυεθνικές εταιρείες έχουν ανακοινώσει τη χρήση γράφων γνώσης, διαδίδοντας περαιτέρω τον όρο.

Οι μέθοδοι Ενσωμάτωσης Γράφου Γνώσης μαθαίνουν να αντιπροσωπεύουν οντότητες και σχέσεις σε ένα γράφος γνώσης ως διανύσματα σε λανθάνωντα χώρο έτσι ώστε να διατηρείται κάποια έννοια ομοιότητας μεταξύ των οντοτήτων. Εάν διατηρηθεί η εγγύτητα πρώτης τάξης, οι συνδεδεμένες οντότητες αναμένεται να έχουν παρόμοια διανύσματα ενσωμάτωσης. Μια πιο χρήσιμη έννοια είναι η εγγύτητα δεύτερης τάξης, που αναφέρεται επίσης ως δομική ή ομοιότητα βάσει ρόλου, η οποία είναι η ομοιότητα μεταξύ των γειτονιών δύο κόμβων. Το Σχήμα 8.2 δείχνει ένα παράδειγμα ενσωμάτωσης που διατηρεί τη δομική ομοιότητα για το γράφο στο Σχήμα 8.1.

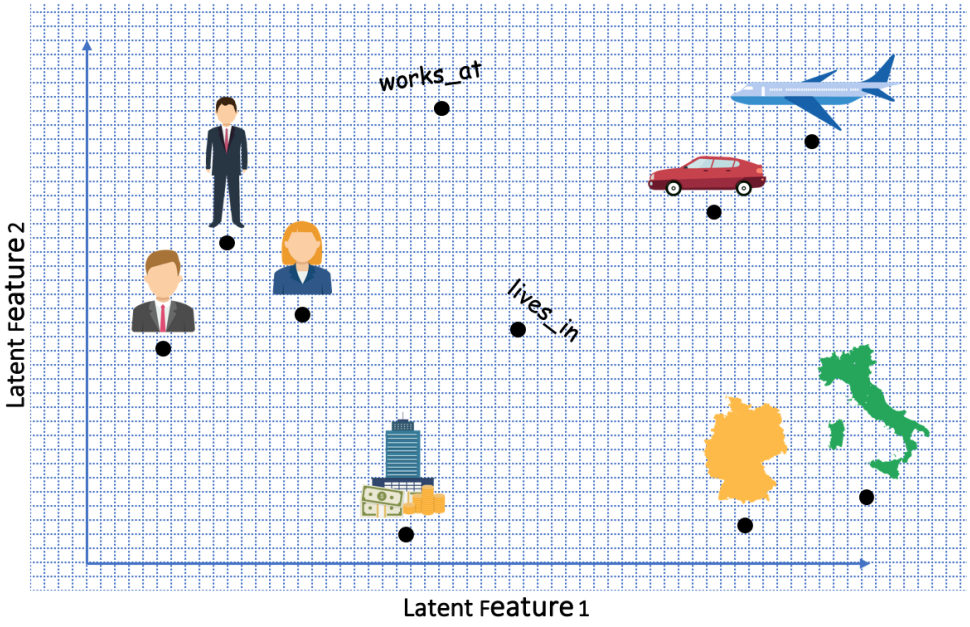

Figure 8.2: A possible embedding of the knowledge graph in Figure 1. (source: [1])

Πολλές διαφορετικές τεχνικές έχουν χρησιμοποιηθεί στην ενσωμάτωση γράφου. Οι μέθοδοι παραγοντοποίησης βελτιστοποιούν την ανακατασκευή ακμών. Τα νευρωνικά δίκτυα γράφων ενσωματώνουν τεχνικές βαθιάς μάθησης και επιτρέπουν την ημιεποπτευόμενη μάθηση από άκρο σε άκρο. Οι τυχαίες μέθοδοι βάδισης δημιουργούν

ακολουθίες κόμβων από το γράφο και τις τροφοδοτούν σε μια μέθοδο ενσωμάτωσης ακολουθίας, π.χ. ένα μοντέλο ενσωμάτωσης λέξεων όπως το word2vec.

Η ενσωμάτωση γράφου μπορεί να χρησιμοποιηθεί για την πραγματοποίηση προβλέψεων σχετικά με τους κόμβους, τις ακμές ή ολόκληρο τον γράφο. Οι εφαρμογές σε επίπεδο κόμβου περιλαμβάνουν ταξινόμηση κόμβων, ομαδοποίηση κόμβων και προτάσεις. Η πρόβλεψη συνδέσμου είναι μια σύνηθης εφαρμογή στο επιπέδο ακμών, ειδικά σε γράφους γνώσης που είναι γνωστό ότι είναι εξαιρετικά ελλιπείς. Για παράδειγμα, το 2014 οι συγγραφείς του [38] διαπίστωσαν ότι ανάμεσα σε όλα τα πρόσωπα στο Freebase, 71 % δεν είχαν καταγεγραμμένο τόπο γέννησης ενώ 99% δεν είχαν εθνικότητα. Ολόκληρη η ενσωμάτωση γράφου μπορεί να χρησιμοποιηθεί για εφαρμογές όπως η ταξινόμηση γράφων.

8.1.2 Συνεισφορά

Στο πλαίσιο αυτής της διατριβής θα εφαρμόσω διάφορες μεθόδους ενσωμάτωσης γράφου γνώσης και θα μελετήσω την απόδοσή τους σε δύο εφαρμογές: πρόβλεψη συνδέσμου και ταξινόμηση οντοτήτων.

Στην πρόβλεψη συνδέσμου, η οποία είναι η πιο συνηθισμένη εφαρμογή μεθόδων ενσωμάτωσης γράφου γνώσης, ο γράφος εισαγωγής θεωρείται ελλιπής. Ο στόχος είναι να εντοπιστούν οι ελλείποντες σύνδεσμοι που είναι πιθανότερο να υπάρχουν σε μια (υποθετική) πλήρη έκδοση του γράφου. Για την αξιολόγηση ενός μοντέλου σχετικά με την πρόβλεψη συνδέσμου, ένα υποσύνολο των υπαρχόντων συνδέσμων είναι απλώς κρυμμένο από το μοντέλο κατά τη διάρκεια της εκμάθησης, μετά το οποίο το μοντέλο αναμένεται να τους κατατάξει υψηλότερους από άλλους συνδέσμους που λείπουν.

Στην ταξινόμηση οντοτήτων, εκτός από το γράφο γνώσεων, παρέχεται επίσης ένα σύνολο ετικετών που αντιστοιχούν σε όλα ή σε ένα υποσύνολο των οντοτήτων. Ένα υποσύνολο αυτών των ετικετών χρησιμοποιείται για εκμάθηση και ο στόχος είναι να προβλέψουμε τα υπόλοιπα. Έχουν προταθεί πολλές διαφορετικές μεθοδολογίες για αυτό το έργο και σε αυτή τη διατριβή θα συγκρίνω μερικές από αυτές.

8.2 Σχετικές εργασίες

Η Στατιστική Σχεσιακή Μάθηση (SRL) είναι μια υποενότητα της Μηχανικής Μάθησης που ασχολείται με τη στατιστική ανάλυση των σχεσιακών δεδομένων. Οι μέθοδοι SLR συνδυάζουν λογικούς φορμαλισμούς πρώτης τάξης και πιθανοκρααατικά μοντέλα γράφων προκειμένου να μοντελοποιήσουν τη στατιστική εξάρτηση μεταξύ χαρακτηριστικών οντοτήτων ή / και σχέσεων μεταξύ οντοτήτων. Ενώ τα μοντέλα SRL έχουν ορισμένα πλεονεκτήματα όπως η επεξήγησιμότητα και η ικανότητα εξαγωγής σύνθετων προτύπων, η εφαρμογή τους περιορίζεται κυρίως λόγω της κακής επεκτασιμότητάς τους. Η Ενσωμάτωση Γράφου Γνώσης (KGE) μπορεί να νοηθεί ως μια προσέγγιση SRL όπου οι σχέσεις είναι υπό όρους ανεξάρτητες η μία από την άλλη, δεδομένης της λανθάνουσας αναπαράστασης των οντοτήτων και των σχετικών σχέσεων[23]. Ωστόσο, το KGE αναπτύχθηκε κυρίως ανεξάρτητα από το SRL και

υπάρχει μικρή αλληλεπίδραση μεταξύ των δύο τομέων. Το Graph Embedding (GE) είναι ένα σχετικό πεδίο που εστιάζει σε κανονικούς γράφους, δηλαδή γράφους γνώσης με έναν τύπο σχέσης. Αρκετές μέθοδοι GE επεκτάθηκαν και σε άλλους γράφους γνώσης.

8.2.1 Ενσωμάτωση Γράφου Γνώσης

Από το 2011, η διαθεσιμότητα μεγάλων Γνωσιακών Βάσεων (KB), όπως WordNet, DBPedia και Freebase, έχει παρακινήσει πολλούς ερευνητές να αναζητήσουν αποτελεσματικούς αλγόριθμους για την ολοκλήρωση της βάσης γνώσεων και άλλες σχετικές εφαρμογές.

Το RESCAL [24] είναι ένα από τα πρώτα μοντέλα KGE που προτάθηκε αρχικά ως τεχνική παραγοντοποίησης τανυστών. Ένας γράφος γνώσης είναι ισοδύναμος με ένα σύνολο γράφων, ένα ανά σχέση, με κοινούς κόμβους και ένας τανυστής μπορεί να σχηματιστεί με στοίβαγμα των πινάκων γειτνίασης αυτών των γράφων. Με την παραγοντοποίηση αυτού του τανυστή σε χαμηλότερης διάστασης τανυστές, το RESCAL είναι σε θέση να μάθει αναπαραστάσεις για οντότητες και σχέσεις. Παρουσιάστηκε μαζί με μια εξειδικευμένη μέθοδο βελτιστοποίησης που θα μπορούσε να κλιμακώθει σε μεγάλους γράφους.

Το Structured Embeddings [4] είναι ένα σημαντικό πρώιμο έργο που εισήγαγε για πρώτη φορά πολλές από τις τεχνικές που χρησιμοποιήθηκαν για την αποτελεσματική εκπαίδευση μοντέλων KGE σήμερα, όπως αρνητική δειγματοληψία και συνάρτηση κόστους βάσει περιθωρίου. Εισήγαγε επίσης το πρωτόκολλο κατάταξης οντοτήτων που είναι μια τυπική διαδικασία για την αξιολόγηση μοντέλων KGE σχετικά με την πρόβλεψη συνδέσμου.

Το TransE [3] προτάθηκε ως μια απλή και επεκτάσιμη εναλλακτική λύση στα υπάρχοντα μοντέλα που αποδείχτηκε εκπληκτικά αποτελεσματικά. Εν μέρει εμπνευσμένο από τεχνικές ενσωμάτωσης λέξεων, όπως το word2vec [22], το TransE μοντελοποιεί τις σχέσεις ως μεταφράσεις στον χώρο ενσωμάτωσης της οντότητας. Στη συνέχεια αναπτύχθηκαν ορισμένα σχετικά μοντέλα (TransH, TransR, κ.λπ.). Το DistMult [40] είναι παρόμοιο με το TransE όσον αφορά την απλότητα, την επεκτασιμότητα και την απόδοση. Όπως το RESCAL, χρησιμοποίησε μια διπλή μορφή για να βαθμολογήσει τριάδες αλλά περιορίζει τη μήτρα της φόρμας σε μια διαγώνια μήτρα που την εμποδίζει να είναι σε θέση να διαμορφώσει ασύμμετρες σχέσεις. Το ComplEx [35] λύνει αυτό το ζήτημα λαμβάνοντας υπόψη σύνθετα διανύσματα και πίνακες ενσωμάτωσης.

Ενώ τα μοντέλα που αναφέρονται μέχρι στιγμής χρησιμοποιούν απλές συναρτήσεις βαθμονόμησης χωρίς παραμέτρους, που σημαίνει ότι οι μόνες παράμετροι είναι οι ίδιες οι ενσωματώσεις, άλλοι έχουν επιλέξει παραμετρικές συναρτήσεις βαθμονόμησης εμπνευσμένες από νευρωνικά δίκτυα. Τα Neural Tensor Networks (NTN) [33] χρησιμοποιούν μια ειδική αρχιτεκτονική νευρωνικού δικτύου για να βαθμονομήσουν τριάδες αλλά απαιτούν πάρα πολλές παραμέτρους για πρακτικά σενάρια. Το ER-MLP [7] είναι μια απλούστερη προσέγγιση που χρησιμοποιεί ένα μόνο κρυφό επίπεδο με μια μη γραμμική συνάρτηση ενεργοποίησης χωρίς bias.

Το DeepWalk [27] είναι μια μέθοδος GE που εμπνεύστηκε από το word2vec που

είναι μια τεχνική ενσωμάτωσης λέξεων. Μετατρέπει το γράφο εισόδου σε ένα σύνολο ακολουθιών εκτελώντας τυχαίες διαδρομές. Αυτές οι ακολουθίες μπορούν στη συνέχεια να αντιμετωπιστούν όπως οι προτάσεις στην ενσωμάτωση λέξεων. Αυτή η προσέγγιση επεκτάθηκε σε γράφοι γνώσεων στο RDF2vec [29].

Ένα κοινό χαρακτηριστικό των παραπάνω μεθόδων είναι ότι τα διανύσματα ενσωμάτωσης είναι παράμετροι του προβλήματος βελτιστοποίησης και βελτιστοποιούνται άμεσα. Αυτές ονομάζονται μέθοδοι άμεσης κωδικοποίησης. Σε αντίθεση, το Graph Neural Networks (GNNs) [39] είναι μια οικογένεια μεθόδων που χρησιμοποιούν βαθιές αρχιτεκτονικές οι οποίες, ξεκινώντας με κάποιες αρχικές αναπαραστάσεις που είναι απλώς σταθερές και δεν θεωρούνται παράμετροι, βελτιώνουν τις αναπαραστάσεις κόμβων σε κάθε επίπεδο. Τα πρώτα GNN ήταν αναδρομικά δίκτυα που χρησιμοποιούσαν το ίδιο νευρωνικό στοιχείο για κάθε στρώμα. Τα συμπαγή GNN, όπως το GCN [18], υποκινούνται από τη γενίκευση της συνέλιξης σε γενικούς γράφους σε αντίθεση με τα πλέγματα, χρησιμοποιούν διαφορετικά βάρη για κάθε στρώμα, αλλά τείνουν να έχουν λιγότερα στρώματα και είναι ευκολότερα στη σύνθεση με άλλα στοιχεία που επιτρέπουν ένα ευρύ φάσμα εφαρμογών. Ενώ τα περισσότερα GNN ισχύουν για κανονικούς μη κατευθυνόμενους γράφους, ορισμένα προγράμματα, ιδίως το R-GCN [31], επιχειρούν να τα γενικεύσουν σε γράφους πολλαπλών σχέσεων.

Μέχρι πρόσφατα, υπήρχαν πολύ λίγες εφαρμογές μοντέλων KGE και τα περισσότερα από αυτά δεν είχαν σχεδιαστεί για GPU, κάτι που είναι ζωτικής σημασίας για την αποτελεσματική εκμάθηση. Από το 2019, έχουν κυκλοφορήσει αρκετές υψηλής ποιότητας και αποτελεσματικές βιβλιοθήκες, όπως AmpliGraph [6], LibKGE [5] και PyKeen [2], που εφαρμόζουν μεθόδους άμεσης κωδικοποίησης ή DGL [36] και Pytorch Geometric [10], που εφαρμόζουν GNN.

8.2.2 Ταξινόμηση οντοτήτων

Η ταξινόμηση οντοτήτων είναι το πολυ-σχεσιακό αντίστοιχο της ταξινόμησης κόμβων. Ενώ τα μοντέλα ενσωμάτωσης γράφου συνήθως αξιολογούνται στην ταξινόμηση κόμβων, τα μοντέλα ενσωμάτωσης γράφων γνώσης σπάνια αξιολογούνται κατά την ταξινόμηση οντοτήτων. Ωστόσο, έχουν προταθεί αρκετές μεθοδολογίες για την ταξινόμηση οντοτήτων και μπορούν να χωριστούν σε εκμάθηση χαρακτηριστικών (feature learning) και προσεγγίσεις από άκρο σε άκρο μάθησης.

Στη διαδικασία εκμάθησης, οι οντότητες αντιπροσωπεύονται ως διανύσματα χαρακτηριστικών που μπορούν να τροφοδοτησούν σε οποιονδήποτε συμβατικό ταξινομητή. Στη μη αυτόματη ταυτοποίηση χαρακτηριστικών, ένας άνθρωπος επιλέγει ένα σύνολο μετρήσεων γράφων και αναλυτικών στοιχείων, όπως βαθμός κόμβου ή κεντρικότητα, για χρήση ως χαρακτηριστικά (features). Το KGE μπορεί να αφαιρέσει την ανάγκη ανθρώπινης παρέμβασης εάν οι ενσωματώσεις οντοτήτων χρησιμοποιούνται ως διανύσματα χαρακτηριστικών. Αυτές οι προσεγγίσεις έχουν συγκριθεί στο[30].

Στην ταξινόμηση οντοτήτων από άκρο σε άκρο, ένα μοντέλο ταξινόμησης που εξαρτάται από το γράφο εισόδου βελτιστοποιείται άμεσα. Το CLASS-RESCAL [16] χρησιμοποιεί μια συνάρτηση κόστους που "τιμωρεί" όχι μόνο το σφάλμα πρόβλεψης συνδέσμου του μοντέλου RESCAL, αλλά και το σφάλμα ταξινόμησης ενός ταξινομητή που χρησιμοποιεί τα στοιχεία που ενσωματώνουν διανύσματα αυτού του μοντέλου

ως χαρακτηριστικά. Η εργασία [24] πρότεινε να αυξήσει την είσοδο KG με μια νέα σχέση και να προσθέσει τις τάξεις ως οντότητες που συνδέονται με τα μέλη τους μέσω αυτής της σχέσης. Η ταξινόμηση οντοτήτων μπορεί στη συνέχεια να μετατραπεί ως πρόβλεψη συνδέσμου στον αυξημένο γράφο. Πιο πρόσφατα, παρουσιάστηκαν Relational Graph Convolutional Networks[32] για να καταστεί δυνατή η επιτηρούμενη μάθηση από άκρο σε άκρο σε γράφους γνώσης.

8.3 Ενσωμάτωση Γράφου Γνώσης

8.3.1 Θεωρητικό υπόβαθρο - Σημειογραφία

Ένας γράφος πολλαπλών σχέσεων $G = (\mathcal{V}, \mathcal{R}, \mathcal{T}^+)$ αποτελείται από ένα σύνολο κόμβων $\mathcal{V}$, ένα σύνολο σχέσεων $\mathcal{R}$ και ένα σύνολο θετικών τριάδων $\mathcal{T}^+ \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ της φόρμας (h, r, t) όπου $h \in \mathcal{V}$ είναι η οντότητα κεφαλαίου ή θέματος και $t \in \mathcal{V}$ είναι η ουρά ή οντότητα αντικειμένου. Ένας γράφος πολλαπλών σχέσεων ισοδυναμεί με μια συλλογή γράφων, ένα για κάθε σχέση. Ως εκ τούτου, μπορεί να αναπαρασταθεί ως σύνολο $|\mathcal{V}| \times |\mathcal{V}|$ πίνακες αδράνειας που όταν στοιβάζονται μαζί σχηματίζουν τον τανυστή γειτονίας $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times |\mathcal{R}|}$.

Προκειμένου να εκπαιδευτεί ένα μοντέλο για την πρόβλεψη σύνδεσης, είναι απαραίτητα αρνητικά παραδείγματα τριάδων . Σύμφωνα με την υπόθεση κλειστού κόσμου (CWA), ο γράφος θεωρείται πλήρης και όλες οι τριάδες που λείπουν επισημαίνονται ως αρνητικές. Στην πραγματικότητα, ωστόσο, τα γράφοι γνώσης είναι γνωστό ότι είναι πολύ ελλιπείς, πράγμα που σημαίνει ότι πολλές τριάδες θα έχουν λανθασμένη επισήμανση βάσει αυτής της υπόθεσης. Επιπλέον, ο αριθμός των παραδειγμάτων που επισημαίνονται αρνητικά είναι $|\mathcal{R}||\mathcal{V}|^2 - |\mathcal{T}^+|$ που γίνεται πολύ μεγάλο καθώς ο αριθμός των οντοτήτων $|\mathcal{V}|$ αυξάνεται καθώς οι περισσότεροι γράφοι είναι εξαιρετικά αραιοί. Αυτό οδηγεί σε έναν δυσανάλογα μεγάλο αριθμό αρνητικών παραδειγμάτων που μπορούν να βλάψουν την προγνωστική ισχύ του μοντέλου ενώ ταυτόχρονα περιορίζουν το μέγεθος του γράφου εισόδου.

Μια πιο συνηθισμένη προσέγγιση είναι η τοπικά κλειστή υπόθεση κόσμου (LCWA), η οποία είναι να υποθέσουμε ότι μια αρνητική τριάδα μπορεί να είναι μόνο το αποτέλεσμα της αντικατάστασης της κεφαλής ή της ουράς μιας θετικής τριάδας με οποιαδήποτε άλλη οντότητα, αποφεύγοντας έτσι παραδοχές σχετικά με την μη εμφάνιση τριάδων που δεν μοιράζονται το ίδιο υποκείμενο ή αντικείμενο με καμιά θετική τριάδα από την ίδια σχέση. Ενώ ο αριθμός των αρνητικών υποψηφίων τριάδων κάτω από το LCWA μειώνεται σε $\mathcal{O}(|\mathcal{T}^+||\mathcal{V}|)$, είναι ακόμα αρκετά μεγάλος για να θέσει τα ίδια ζητήματα με το CWA εάν όλοι οι υποψήφιοι χαρακτηρίζονται αρνητικοί. Δεδομένου ότι δεν υπάρχει τρόπος προσδιορισμού μιας προτίμησης μεταξύ δύο υποψηφίων αρνητικών τριάδων που διαφέρουν μόνο σε μία από τις οντότητες, οι περισσότερες μέθοδοι ενσωμάτωσης γράφου γνώσης δημιουργούν ένα διαφορετικό τυχαίο δείγμα αρνητικών τριάδων για κάθε εποχή κατά τη διάρκεια της εκπαίδευσης ενός μοντέλου σύμφωνα με κάποια ευρετική. Με αυτόν τον τρόπο όλες οι υποψήφιες αρνητικές τριάδες έχουν την ευκαιρία να εξεταστούν, ενώ η αναλογία θετικών προς αρνητικών τριάδων σε κάθε εποχή μπορεί να ρυθμιστεί ελεύθερα. Το σύνολο των αρνητικών τριάδων σημειώνεται με $\mathcal{T}^-$, το σύνολο όλων των τριάδων ετικετών είναι $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$

και

$$y_{hrt} = \begin{cases} 1 & \text{if } (h,r,t) \int \mathcal{T}^+ \\ -1 & \text{if } (h,r,t) \int \mathcal{T}^- \end{cases} \tag{8.1}$$

είναι η ετικέτα της τριάδα $(h,r,t) \in \mathcal{T}$.

Η πρόβλεψη συνδέσμου είναι το καθήκον της κατάταξης των μη επισημασμένων τριάδων με βάση την πιθανότητα να είναι θετικά. Για το σκοπό αυτό, οι μέθοδοι ενσωμάτωσης γράφου γνώσης που βασίζονται στην πρόβλεψη συνδέσμου "μαθαίνουν" μια συνάρτηση $f : \mathcal{V} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ που προσδιορίζει μια βαθμολογία σε κάθε πιθανή τριάδα . Το μοντέλο έχει εκπαιδευτεί για να αποδίδει υψηλές βαθμολογίες σε θετικές τριάδες και χαμηλές βαθμολογίες σε αρνητικές. Ορίζοντας το f ως τη σύνθεση δύο συναρτήσεων, έναν κωδικοποιητή και έναν αποκωδικοποιητή, οι μέθοδοι ενσωμάτωσης γράφου γνώσης είναι σε θέση να αντιπροσωπεύουν οντότητες και σχέσεις ως διανύσματα στον ενδιάμεσο τομέα που ονομάζεται χώρος ενσωμάτωσης.

Ο κωδικοποιητής ENC είναι το μέρος του μοντέλου που δημιουργεί τις αναπαραστάσεις. Μια τριάδα (h,r,t) κωδικοποιείται εφαρμόζοντας έναν κωδικοποιητή οντοτήτων $ENC_{\mathcal{V}} : \mathcal{V} \rightarrow \mathbb{R}^{d_e}$ που χαρτογραφεί τις οντότητες h και t στα d_e -διάστατα διανύσματα ενσωμάτωσης $\mathbf{e}_h$ και $\mathbf{e}_t$ και έναν κωδικοποιητή συσχέτισης $ENC_{\mathcal{R}} : \mathcal{R} \rightarrow \mathbb{R}^{d_r}$ που αντιστοιχίζει τη σχέση r με d_r -διάστατη αναπαράσταση $\mathbf{w}_r$. Η σχέση ενσωμάτωσης διάστασης d_r είναι μια συνάρτηση της ιδιότητας ενσωμάτωσης οντοτήτων d_e η οποία είναι μια υπερ-παράμετρος.

Ο αποκωδικοποιητής DEC : $\mathbb{R}^{d_e} \times \mathbb{R}^{d_r} \times \mathbb{R}^{d_e} \rightarrow \mathbb{R}$, επίσης ονομάζεται συνάρτηση βαθμολόγησης, χαρτογραφεί τις αναπαραστάσεις που δημιουργούνται από τον κωδικοποιητή σε μια βαθμολογία. Το σκορ είναι συνήθως ένα μέτρο ομοιότητας μεταξύ των διανυσμάτων ενσωμάτωσης του θέματος και των οντοτήτων αντικειμένου υπό μια μεταμόρφωση που εξαρτάται από την αναπαράσταση της σχέσης. Ενώ έχουν προταθεί ορισμένοι παραμετρικοί αποκωδικοποιητές, αυτή η μελέτη επικεντρώνεται σε αποκωδικοποιητές χωρίς παραμέτρους που είναι απλώς συναρτήσεις των ενδιάμεσων διανυσμάτων.

Το συνολικό μοντέλο είναι η σύνθεση του κωδικοποιητή και του αποκωδικοποιητή:

$$f(h,r,t) = DEC(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t)$$

Οι περισσότερες μέθοδοι ενσωμάτωσης γράφου γνώσης χρησιμοποιούν απευθείας κωδικοποιητές που παραμετροποιούνται από έναν πίνακα ενσωμάτωσης οντοτήτων $E \in \mathbb{R}^{|\mathcal{V}| \times d_e}$ και μια σχέση που ενσωματώνει τον πίνακα $W \in \mathbb{R}^{|\mathcal{R}| \times d_r}$ που εξάγουν απλώς μια διαφορετική σειρά E και W για κάθε οντότητα και σχέση αντίστοιχα. Σε αυτήν την περίπτωση, τα διανύσματα ενσωμάτωσης των οντοτήτων και οι σχέσεις είναι οι μόνες παράμετροι του προβλήματος βελτιστοποίησης και βελτιστοποιούνται άμεσα.

8.3.2 Loss Functions

Μπορούν να χρησιμοποιηθούν διάφορες συναρτήσεις κόστους για την εκπαίδευση ενός μοντέλου ενσωμάτωσης γράφου γνώσης. Μια συνάρτηση κόστους κατά σημείο

L εκχωρεί ένα κόστος σε κάθε τριάδα ξεχωριστά:

$$\min_{E,W} \sum_{(h,r,t)\in\mathcal{T}} L(y_{hrt}, f(h,r,t)) \tag{8.2}$$

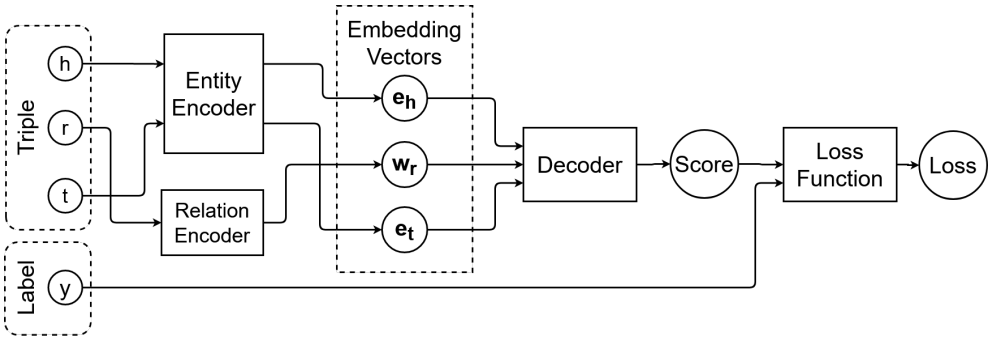

Figure 8.3: Model architecture for pointwise loss functions.

Μια κοινή επιλογή είναι η negative log likelihood του λογιστικού μοντέλου, γνωστή και ως logistic loss, που δίνεται από τη σύνθεση της λογιστικής συνάρτησης, η οποία είναι μια συνάρτηση σιγμοειδούς που αντιστοιχεί στο $\mathbb{R}$ στο διάστημα $(-1, 1)$, με το προϊόν $-y_{hrt}f(h,r,t)$:

$$L(y_{hrt}, f(h,r,t)) = \sum_{(h,r,t)\in\mathcal{T}} \log(1 + e^{-y_{hrt}\cdot f(h,r,t)}) \tag{8.3}$$

Μια συνάρτηση κόστους (loss function) κατά ζεύγη εκχωρεί ένα κόστος σε κάθε ζεύγος ενός θετικού και ενός αρνητικού παραδείγματος που διαφέρει μόνο σε μία από τις οντότητες θέματος και αντικειμένου ως συνάρτηση της διαφοράς των βαθμολογιών τους. Εάν το L είναι μια συνάρτηση κόστους (loss function) κατά ζεύγη, επιλύεται το ακόλουθο πρόβλημα:

$$\min_{E,W} \sum_{(h,r,t)\in\mathcal{T}^+} \left(\sum_{(h',r,t)\in\mathcal{T}^-} L\left(f(h,r,t) - f(h',r,t)\right) + \sum_{(h,r,t')\in\mathcal{T}^-} L\left((f(h,r,t) - f(h,r,t')\right) \right) \tag{8.4}$$

Η συνάρτηση κόστους (loss function) με βάση το περιθώριο (MRL) [4] είναι η πιο συχνά χρησιμοποιούμενη συνάρτηση κόστους (loss function) κατά ζεύγη:

$$L\left(f(h,r,t) - f(h',r',t')\right) = \max(0, \gamma + f(h,r,t) - f(h',r',t')) \tag{8.5}$$

Εκχωρεί ένα κόστος σε κάθε ζεύγος όπου το σκορ της θετικής τριάδας δεν είναι μεγαλύτερο από το σκορ της αρνητικής τριάδας κατά τουλάχιστον κάποιο περιθώριο γ. Οι κοινές τιμές για την υπερ-παράμετρο περιθωρίου είναι $0, 5$ και 1 .

Τέλος, οι συνολικές συναρτήσεις κόστους, αποδίδουν ένα κόστος σε κάθε θετικό παράδειγμα, λαμβάνοντας υπόψη όλα τα αρνητικά παραδείγματα που δημιουργούνται από τη αλλοίωση του. Για παράδειγμα, [34] πρότεινε μια συνάρτηση κόστους βάσει μιας προσέγγισης του αρνητικού αρχείου καταγραφής (NLL) καθορίζοντας:

$$p(t|h,r) = \frac{e^{f(h,r,t)}}{\sum_{(h,r,t')\in\mathcal{T}^-} e^{f(h,r,t')}} \tag{8.6}$$

$$p(h|t,r) = \frac{e^{f(h,r,t)}}{\sum_{(h',r,t)\in\mathcal{T}^-} e^{f(h',r,t)}} \tag{8.7}$$

και στη συνέχεια ελαχιστοποιώντας την ακόλουθη συνάρτηση:

$$\min_{E,W} \sum_{(h,r,t)\in\mathcal{T}} (-\log(p(t|h,r)) - \log(p(h|t,r))) \tag{8.8}$$

Το overfitting είναι μια κοινή κατάσταση στη μηχανική εκμάθηση όπου ένα μοντέλο εκπαιδεύεται για την αναπαραγωγή των ετικετών των παραδειγμάτων εκπαίδευσης με πολύ ακρίβεια, αλλά κάνει πολύ κακές προβλέψεις σε παραδείγματα δοκιμών που είναι αρκετά διακριτές από οποιοδήποτε παράδειγμα εκπαίδευσης. Στην Ενσωμάτωση Γράφου Γνώσης, διάφορες τεχνικές ομαλοποίησης έχουν προταθεί για τη μείωση της υπερβολικής προσαρμογής. Μια επιλογή είναι να κανονικοποιήσετε τις ενσωματώσεις οντοτήτων σε μοναδιαία διανύσματα ομαλοποιώντας τα μετά από κάθε ενημέρωση [4] Ένα άλλο είναι να προσθέσετε τις απόλυτες τιμές ($p = 1$) ή τα τετράγωνα ($p = 2$) των παραμέτρων που κλιμακώνονται από ορισμένες σταθερές λ_E και λ_W στη συνολική συνάρτηση κόστους $\mathcal{L}$, με αποτέλεσμα την ομαλοποίηση L_1 ή L_2 αντίστοιχα:

$$\min_{E,W} \mathcal{L}(E,W) + \lambda_e \sum_{v\in\mathcal{V}} \|\mathbf{e}_v\|_p^p + \lambda_w \sum_{r\in\mathcal{R}} \|\mathbf{w}_r\|_p^p \tag{8.9}$$

8.3.3 Αρνητική Δηγματοληψία

Για αποτελεσματική εκμάθηση κάτω από το LCWA απαιτείται ευρετική δειγματοληψία αρνητικών τριάδων . Οι κοινές προσεγγίσεις δημιουργούν αρνητικές τριάδες αλλοιώνοντας κάθε θετική τριάδα $(h,r,t) \in \mathcal{T}^+$ αντικαθιστώντας είτε h είτε t με μια τυχαία οντότητα. Μπορούν να δημιουργηθούν πολλαπλές αρνητικές τριάδες για κάθε θετικό. Κανονικά, όλες οι θετικές τριάδες αποκλείονται από υποψήφιες αρνητικές τριάδες . Στην πράξη, ωστόσο, ο έλεγχος εάν μια αλλοιωμένη τριάδα είναι θετική είναι μια σχετικά δαπανηρή διαδικασία και από το $|\mathcal{V}^2| \gg |T^+|$, η πιθανότητα δημιουργίας μιας θετικοής τριάδας μέσω της τυχαίας αλλοίωσης ενός άλλου είναι πολύ χαμηλή, επομένως αυτός ο έλεγχος συχνά αποφεύγεται.

Στην ομοιόμορφη αρνητική δειγματοληψία, υπάρχει η ίδια πιθανότητα αντικατάστασης h και t. Στο [37] παρατήρησαν ότι ορισμένες σχέσεις είναι μία προς πολλές, που

σημαίνει ότι περισσότερες οντότητες εμφανίζονται ως αντικείμενα παρά ως θέματα, ενώ άλλες είναι πολλές προς μία. Για το πρώτο, η αντικατάσταση της ουράς έχει μεγαλύτερη πιθανότητα να οδηγήσει σε θετική τριάδα και το αντίστροφο για το δεύτερο. Για την αντιμετώπιση αυτού του ζητήματος, προτάθηκε αρνητική δειγματοληψία Bernouli, η οποία αντικαθιστά την αρχική οντότητα με πιθανότητα $p_r = \frac{tph}{tph+hpt}$ και την ουρά με πιθανότητα $1 - p_r$ για τριάδα που ανήκει στη σχέση r, όπου tph είναι ο λόγος των μοναδικών οντοτήτων ουράς προς τις μοναδικές οντότητες κεφαλής και το hpt είναι η αναλογία μοναδικών οντοτήτων κεφαλής προς μοναδικές οντότητες ουράς σε σχέση με r

8.3.4 Decoders

Ο αποκωδικοποιητής, που αναφέρεται επίσης ως συνάρτηση βαθμονόμησης, είναι αναμφισβήτητα το πιο σημαντικό συστατικό, καθώς καθορίζει ποια στοιχεία των διανυσμάτων ενσωμάτωσης επιτρέπεται να αλληλεπιδρούν μεταξύ τους. Καθορίζει επίσης τη διάσταση ενσωμάτωσης σχέσης d_r σε σχέση με την επιλεγμένη ιδιότητα ενσωμάτωσης οντοτήτων d_e.

TransE

Το TransE [3] είναι μία από τις πιο απλές συνάρτησεις βαθμονόμησης. Ονομάζεται μεταφραστικό μοντέλο επειδή κωδικοποιεί τις σχέσεις ως μεταφράσεις που μετακινούν τις ενσωματώσεις των θεματικών οντοτήτων όσο το δυνατόν πιο κοντά στις αντίστοιχες οντότητες αντικειμένων. Για παράδειγμα, εάν ο γράφος εισαγωγής περιέχει χώρες και πόλεις ως οντότητες και μια σχέση που συνδέει χώρες με τις πρωτεύουσες τους, τα διανύσματα ενσωμάτωσης της Αθήνας, της Ελλάδας, του Παρισιού και της Γαλλίας αναμένεται να ικανοποιήσουν:

$$\mathbf{e}_{\text{athens}} - \mathbf{e}_{\text{greece}} \approx \mathbf{e}_{\text{paris}} - \mathbf{e}_{\text{france}} \approx \mathbf{w}_{\text{capital\_of}} \tag{8.10}$$

Η ιδιότητα ενσωμάτωσης σχέσης είναι ίδια με την ιδιότητα ενσωμάτωσης οντοτήτων και ο αποκωδικοποιητής δίνεται από:

$$DEC(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = -\|\mathbf{e}_h + \mathbf{w}_r - \mathbf{e}_t\|_p \tag{8.11}$$

όπου $\|\mathbf{a}\|_p$ είναι το p -norm του $\mathbf{a}$ και το p είναι είτε 1 είτε 2. Οι συγγραφείς πρότειναν τον περιορισμό $\|\mathbf{e}_v\| = 1, \forall v \in \mathcal{V}$ ως μορφή κανονικοποίησης. Το TransE είναι πολύ αποτελεσματικό να εκπαιδεύεται, αφού η κλίση μπορεί να υπολογιστεί σε $\mathcal{O}(d_e)$ time.

RESCAL

RESCAL [24] αντιπροσωπεύει κάθε σχέση ως γραμμικό χάρτη κάτω από τον οποίο οι εικόνες των διανυσμάτων ενσωμάτωσης των θεματικών οντοτήτων είναι παρόμοιες με τα διανύσματα ενσωμάτωσης των αντίστοιχων οντοτήτων αντικειμένου. Η

διάσταση ενσωμάτωσης σχέσης είναι το τετράγωνο της διάστασης ενσωμάτωσης οντότητας και ο αποκωδικοποιητής είναι διπλής μορφής:

$$DEC(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = \mathbf{e}_h^\top W_r \mathbf{e}_t \tag{8.12}$$

όπου $\text{vec}(W_r) = \mathbf{w}_r$. Συνήθως συνδυάζεται με κανονικοποίηση L_2.

Το RESCAL είναι ένα από τα πιο εκφραστικά μοντέλα δεδομένου ότι όλα τα στοιχεία των διανυσμάτων ενσωμάτωσης του θέματος και των οντοτήτων αντικειμένων επιτρέπεται να αλληλεπιδρούν μεταξύ τους, όπως φαίνεται στο Σχήμα 8.4, αλλά είναι επίσης σχετικά ακριβό να εκπαιδεύεται για τον ίδιο λόγο.

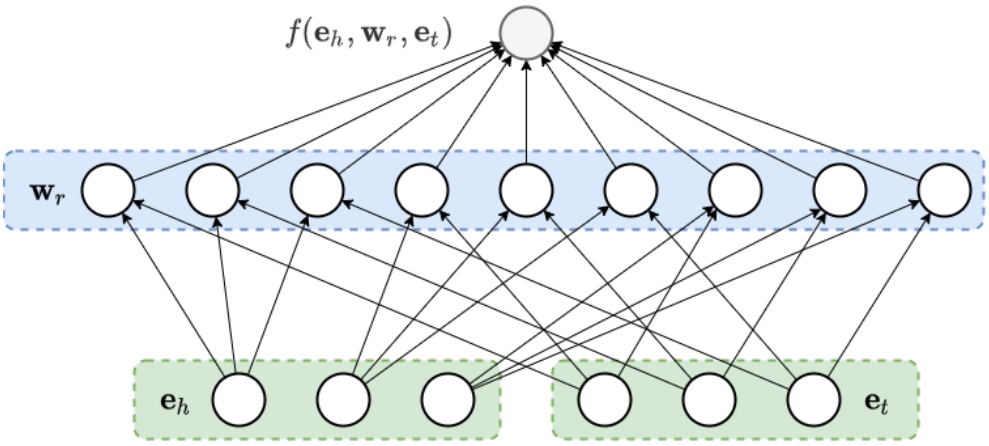

Figure 8.4: Interactions of the elements of the embedding vectors for RESCAL

DistMult

Το DistMult [40] χρησιμοποιεί μια περιορισμένη διπλή μορφή όπου η μήτρα περιορίζεται να είναι διαγώνια:

$$DEC(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = \mathbf{e}_h^\top W_r \mathbf{e}_t \tag{8.13}$$

όπου $W_r = \text{diag}(\mathbf{w}_r)$ είναι η διαγώνια μήτρα που σχηματίζεται από τα στοιχεία του διανύσματος $\mathbf{w}_r$. Το DistMult είναι πολύ αποτελεσματικό στον υπολογισμό και έχει δείξει καλή απόδοση στην πρόβλεψη συνδέσμου. Ένας περιορισμός είναι ότι η διάθεση των οντοτήτων θέματος και αντικειμένου δεν αλλάζει το σκορ, πράγμα που σημαίνει ότι το DistMult αγνοεί την κατεύθυνση των συνδέσμων.

ComplEx

Το ComplEx [35] επεκτείνει το DistMult σε σύνθετες αναπαραστάσεις:

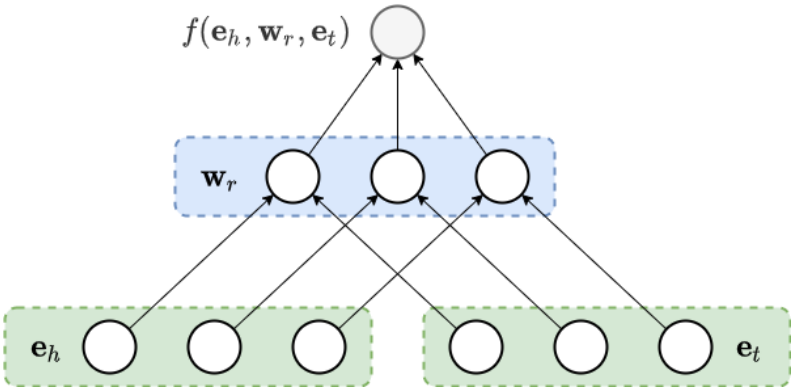

Figure 8.5: Interactions of the elements of the embedding vectors for DistMult

$$\begin{aligned} DEC(\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t) = \text{Re}(\mathbf{e}_h^\top W_r \mathbf{e}_t^*) = &\langle \text{Re}(\mathbf{e}_h), \text{Re}(\mathbf{w}_r), \text{Re}(\mathbf{e}_t) \rangle \\ &+ \langle \text{Im}(\mathbf{e}_h), \text{Re}(\mathbf{w}_r), \text{Im}(\mathbf{e}_t) \rangle \\ &+ \langle \text{Re}(\mathbf{e}_h), \text{Re}(\mathbf{w}_r), \text{Im}(\mathbf{e}_t) \rangle \\ &- \langle \text{Im}(\mathbf{e}_h), \text{Im}(\mathbf{w}_r), \text{Im}(\mathbf{e}_t) \rangle \end{aligned} \tag{8.14}$$

όπου $\mathbf{e}_h, \mathbf{w}_r, \mathbf{e}_t \in \mathbb{C}^{d_e}$, $W_r = \text{diag}(\mathbf{w}_r)$, x^* είναι το σύνθετο σύζευγμα $x \in \mathbb{C}$ και $\langle \mathbf{a}, \mathbf{b}, \mathbf{c} \rangle = \sum_i a_i b_i c_i$. Σε αντίθεση με το DistMult, το ComplEx δεν είναι συμμετρικό σε σχέση με τις οντότητες του αντικειμένου και του αντικειμένου που του επιτρέπει να μοντελοποιεί ασύμμετρες σχέσεις.

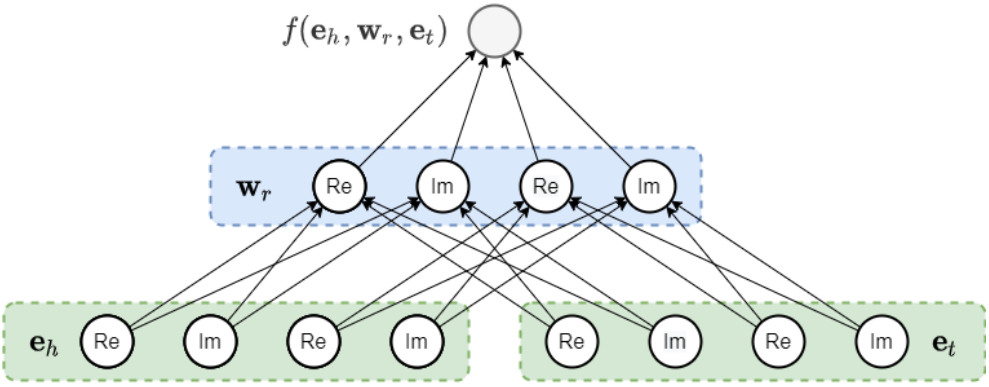

Figure 8.6: Interactions of the elements of the embedding vectors for ComplEx

8.3.5 Βελτιστοποίηση

Gradient Descent

Η πιο συνηθισμένη επιλογή της μεθόδου βελτιστοποίησης στην ενσωμάτωση γράφων γνώσης και στη μηχανική μάθηση γενικά είναι η μείωση κλίσης (GD) και οι παραλλαγές της. Το GD είναι μια επαναληπτική μέθοδος που λαμβάνει μια καλύτερη εκτίμηση των βέλτιστων παραμέτρων μετά από κάθε επανάληψη, που ονομάζεται επίσης μια εποχή, ξεκινώντας από μερικές αρχικές τιμές που συνήθως δειγματοληπτικά τυχαία από κάποια κατανομή. Σε κάθε εποχή, η παράμετρος vector θ ενημερώνεται κατά κάποιο ποσό ανάλογο με την κλίση της συνάρτησης κόστους (loss function) L:

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta L(\theta_i) \tag{8.15}$$

Η σταθερά αναλογικότητας η ονομάζεται ποσοστό εκμάθησης. Εάν το ποσοστό εκμάθησης είναι πολύ χαμηλό, θα χρειαστούν πάρα πολλές επαναλήψεις για τη σύγκλιση του GD, ενώ εάν είναι πολύ υψηλό, το GD θα υπερβεί και θα αρχίσει να ταλαντεύεται. Για αυτόν τον λόγο, είναι συνηθισμένο να ξεκινάτε με ένα σχετικά υψηλό ποσοστό εκμάθησης και να χρησιμοποιείτε ένα πρόγραμμα μαθησιακού ρυθμού, το οποίο είναι μια στρατηγική για τη μείωση του ρυθμού μάθησης με την πάροδο του χρόνου. Σύμφωνα με ένα πρόγραμμα εύλογου ρυθμού εκμάθησης, το GD συγκλίνει στο ολικό ελάχιστο για κυρτές συναρτήσεις κόστους (loss function) και σε ένα τοπικό ελάχιστο για μη κυρτές. Το GD ευνοείται όταν η συνάρτηση είναι ως επί το πλείστον ομαλή και κυρτή και το πρόβλημα είναι αρκετά μεγάλο για να είναι ανέφικτες οι μέθοδοι δεύτερης τάξης. Το GD έχει δύο κύρια μειονεκτήματα:

- Μπορεί εύκολα να παγιδευτεί στη λεκάνη έλξης ενός τοπικού ελάχιστου για εξαιρετικά μη κυρτές συναρτήσεις.
- Όταν η συνάρτηση κόστους δίνεται ως άθροισμα σε ένα σύνολο παραδειγμάτων, το GD έχει γραμμική πολυπλοκότητα χρόνου σε σχέση με τον αριθμό των παραδειγμάτων ανεξάρτητα από τυχόν πλεονασμό που υπάρχει. Για παράδειγμα, η αναπαραγωγή κάθε παραδείγματος εκπαίδευσης θα διπλασιάσει τον χρόνο που απαιτείται για τη σύγκλιση του GD στην ίδια λύση.

Το Stochastic Gradient Decsent (SGD) αντιμετωπίζει και τα δύο αυτά ζητήματα. Αντί να ακολουθεί την ακριβή κλίση της πλήρους συνάρτησης κόστους (loss function) , η SGD ακολουθεί μια στοχαστική προσέγγιση της πραγματοποιώντας κάθε ενημέρωση παραμέτρων με βάση τη κατεύθυνση μείωσης της συνάρτησης κόστους (loss function) ενός τυχαία επιλεγμένου παραδείγματος. Αυτό επιτρέπει στο SGD να ξεφύγει από τη λεκάνη έλξης ενός τοπικού ελάχιστου και να "πηδήξει" προς ένα δυνητικά καλύτερο. Επιπλέον, ο χρόνος που απαιτείται για κάθε ενημέρωση παραμέτρων είναι ανεξάρτητος από τον αριθμό των παραδειγμάτων, επομένως το SGD μπορεί να συγκλίνει ταχύτερα από το GD παρουσία πλεονασμού.

Ενώ η στοχαστική φύση του SGD μπορεί να οδηγήσει σε καλύτερες λύσεις για εξαιρετικά μη κυρτές συναρτήσεις, περιπλέκει επίσης τη σύγκλιση για συναρτήσεις που είναι κυρίως κυρτές. Μια ισορροπία μεταξύ GD και SGD μπορεί να επιτευχθεί με mini-batch στοχαστική κάθοδο κλίσης (MB-SGD) που χρησιμοποιεί περισσότερα

από ένα παραδείγματα εκπαίδευσης για να προσεγγίσει την κλίση για κάθε ενημέρωση παραμέτρων. Αυτό επιτρέπει στο MB-SGD να διατηρεί κάποια ποσότητα στοχαστικότητας χωρίς να θυσιάζει τα καλά χαρακτηριστικά σύγκλισης του GD σε ομαλές συναρτήσεις. Στην πράξη, το κύριο μειονέκτημα του SGD είναι ότι δεν μπορεί να εκμεταλλευτεί τις δυνατότητες παράλληλης εκτέλεσης του σύγχρονου υπολογστικού υλικού επειδή οι ενημερώσεις παραμέτρων πρέπει να συμβούν διαδοχικά, ενώ η αξιολόγηση της κατεύθυνσης μείωσης για ένα παράδειγμα εκπαίδευσης είναι απίθανο να δημιουργήσει κορεσμό στους περισσότερους σύγχρονους αγωγούς υλικού (hardware pipelines). Με το GD και το MB-SGD, αυτό το ζήτημα μπορεί να εξαλειφθεί αξιολογώντας παράλληλα την κλίση πολλαπλών παραδειγμάτων.

8.3.6 Αξιολόγηση

Η απόδοση ενός μοντέλου KGE στην εργασία πρόβλεψης συνδέσμου μπορεί να αξιολογηθεί χρησιμοποιώντας το πρωτόκολλο κατάταξης οντοτήτων. Το μοντέλο εκπαιδεύεται χρησιμοποιώντας μόνο ένα υποσύνολο των θετικών τριάδων ενώ τα υπόλοιπα παραμένουν στην άκρη ως σύνολο δοκιμών. Κατά τη διάρκεια της αξιολόγησης, για κάθε θετική τριάδα (s, r, o) στο σύνολο δοκιμών, οι βαθμολογίες όλων των τριάδων της φόρμας $(?, r, o)$ ταξινομούνται και η κατάταξη των δοκιμαστικών τριάδων $(s), r, o)$ καταγράφεται. Το ίδιο ισχύει για όλες τις τριάδες της μορφής $(s, r, ?)$. Οι καταγεγραμμένες τάξεις μπορούν να συγκεντρωθούν με διάφορους τρόπους. Οι μετρήσεις που χρησιμοποιούνται πιο συχνά είναι:

- **Mean Rank (MR)**: Ο μέσος όρος των καταγεγραμμένων τάξεων.
- **Mean Reciprocal Rank (MRR)**: Ο μέσος όρος των αντιστρόφων των καταγεγραμμένων τάξεων.
- **Hits@k**: Το ποσοστό των καταγεγραμμένων τάξεων που ήταν μικρότερο ή ίσο με k.

Με την εξαίρεση θετικών τριάδων που παρατηρήθηκαν κατά τη διάρκεια της εκμάθησης κατά τον υπολογισμό της κατάταξης κάθε τριάδας δοκιμής, λαμβάνεται η φιλτραρισμένη έκδοση αυτών των μετρήσεων.

8.3.7 R-GCN

Οι μέθοδοι ενσωμάτωσης γράφου που περιγράφονται μέχρι στιγμής εκπαιδεύονται σε έναν στόχο πρόβλεψης συνδέσμου. Παρόλο που οι αναπαραστάσεις που παράγουν μπορούν να χρησιμοποιηθούν ως χαρακτηριστικά σε διάφορες εφαρμογές μηχανικής μάθησης, δεν υπάρχει καμία εγγύηση ότι θα έχουν αρκετή διακριτική ισχύ αφού εκπαιδεύτηκαν για διαφορετικό στόχο. Εάν το μοντέλο f δεν εξαρτάται με κανέναν τρόπο από τον γράφο εισόδου, απαιτείται ένας στόχος που βασίζεται στην ανακατασκευή ακμών για να εισέλθει η εικόνα στον γράφο εισόδου. Τα νευρωνικά δίκτυα γράφου καταργούν αυτόν τον περιορισμό χρησιμοποιώντας έναν κωδικοποιητή που εξαρτάται από τον γράφο εισόδου που επιτρέπει στις ετικέτες να χρησιμοποιούνται για οποιονδήποτε εποπτευόμενο ή ημι-εποπτευόμενο στόχο.

Ένα δημοφιλές μοντέλο νευρωνικού δικτύου γράφων για πολυ-σχεσιακά γράφοι είναι το Relational Graph Convolutional Network (R-GCN) [32]. Με δεδομένη μια ανα-

παράσταση πηγής $\mathbf{h}_v^0$ για κάθε κόμβο $v \in \mathcal{V}$, το R-GCN μπορεί να παράγει μια σειρά από νέες αναπαραστάσεις $\mathbf{h}_v^l$ συνδυάζοντας τις αναπαραστάσεις όλων των κόμβων στη γειτονιά l -hop v. Εάν είναι διαθέσιμες αριθμητικές τιμές για τις οντότητες, μπορούν να χρησιμοποιηθούν ως αναπαραστάσεις πηγής, διαφορετικά οι αρχικοί συντάκτες προτείνουν τη χρήση ενός μοναδικού ενδιάμεσου διανύσματος για κάθε οντότητα. Για μεγάλα γράφοι, η κωδικοποίηση one-shot καθίσταται αναποτελεσματική και μπορεί να χρησιμοποιηθεί ένας άμεσος κωδικοποιητής για την παροχή των αναπαραστάσεων πηγής.

Ειδικότερα, εάν το $\mathcal{N}_v^r$ είναι το σύνολο γειτόνων του κόμβου $v \in \mathcal{V}$ σε σχέση $r \in \mathcal{R}$, η αναπαράσταση v in το επίπεδο l -th δίνεται από:

$$\mathbf{h}_v^{l+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_v^r}\left(\frac{1}{c_{v,r}}W_r^l\mathbf{h}_u^l + W_0^l\mathbf{h}_v^l\right)\right) \tag{8.16}$$

όπου το σ είναι μια συνάρτηση ενεργοποίησης με βάση το στοιχείο, $\mathrm{ReLU}(x) = max(0, x)$, $c_{v,r}$ είναι είτε μια σταθερά κανονικοποίησης όπως η $|\mathcal{N}_v^r|$ ή μια παράμετρος που θα προκύψει από την εκμάθηση, $W_r^l, W_0^l \in \mathbb{R}^{d_{l+1}\times d_l}$ είναι μήτρες βάρους και d_l είναι η διάσταση των αναπαραστάσεων επιπέδου l -th που είναι μια υπερπαράμετρος. Το Σχήμα 8.7 δείχνει

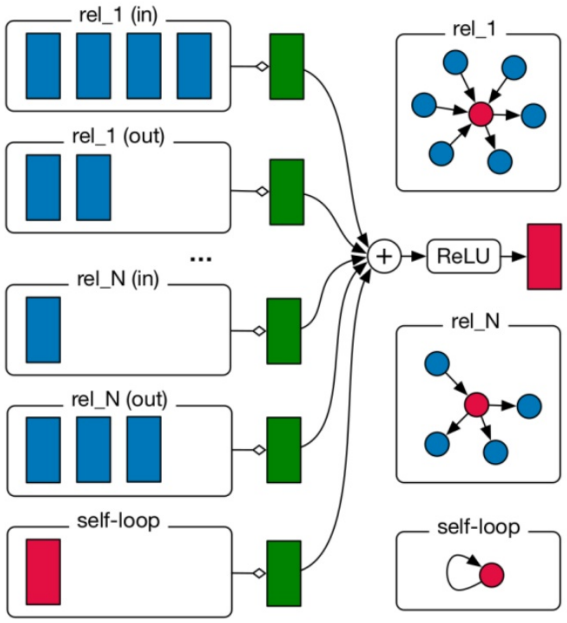

Figure 8.7: The computational graph of a layer of R-GCN. The representations of the previous layers are shown in blue. (source: [31])

Επιλέγοντας μια κατάλληλη διάσταση d_L για το τελικό επίπεδο L, το δίκτυο μπορεί να εκπαιδευτεί σε διάφορες εποπτευόμενες εφαρμογές. Η παλινδρόμηση (regression) μπορεί να πραγματοποιηθεί επιλέγοντας $d_L = 1$ και χρησιμοποιώντας μια συνάρ-

τηση κόστους (loss function) για τη μέτρηση του σφάλματος μεταξύ της αναπαράστασης εξόδου h_v^L και της ετικέτας y_v του κόμβου v. Για την ταξινόμηση οντοτήτων, εάν το C είναι το σύνολο των τάξεων, μπορεί να επιλεγεί $d_L = |C|$ και μπορεί να χρησιμοποιηθεί μια συνάρτηση ταξινόμησης, όπως η συνάρτηση εγκάρσιας εντροπίας (cross entropy loss). Η πρόβλεψη σύνδεσης είναι επίσης δυνατή συνδυάζοντας το R-GCN με έναν αποκωδικοποιητή, οπότε $d_L = d_e$ και η έξοδος του R-GCN είναι τα διανύσματα ενσωμάτωσης κόμβου.

8.3.8 Μεθοδολογία

Περισσότερες λεπτομέρειες για την μεθοδολογία που ακολούθησα στην υλοποίηση των αναφερομένων στο Τμήμα 8.3 αναφέρονται στο Section 4.

8.4 Πειραματική Αξιολόγηση

8.4.1 Link Prediction

Δεδομένα

Τα FB15K και WN18 είναι ευρέως χρησιμοποιούμενα σύνολα δεδομένων για συγκρίσεις μεθόδων ενσωμάτωσης γράφων γνώσης στην πρόβλεψη συνδέσμου.

FB15K
Το FB15K εξήχθη [3] από το Freebase, μια μεγάλη συνεργατική βάση γνώσεων που αποτελείται κυρίως από τα μέλη της κοινότητάς της. Είναι ένα δείγμα 592.213 τριάδων με 14.951 οντότητες και 1.345 σχέσεις. Εκείνη την εποχή το Freebase είχε περίπου 1,2 δισεκατομμύρια τριάδες και πάνω από 80 εκατομμύρια οντότητες. Από τότε, το Freebase έχει σταματήσει υπέρ του Wikidata, αλλά το FB15K εξακολουθεί να χρησιμοποιείται για τη σύγκριση μεθόδων ενσωμάτωσης. Στο [34] παρατήρησαν ότι για το 81 % του τεστ τριάδων (h, r, t) στο FB15K υπήρχε τουλάχιστον μια τριάδα της μορφής (h, r', t) ή (t, r', h). Λόγω αυτής της μη ρεαλιστικής συσχέτισης μεταξύ δεδομένων εκπαίδευσης και δοκιμών, εξαιρετικά απλές μέθοδοι μπόρεσαν να ξεπεράσουν τα υπερσύγχρονα μοντέλα. Το FB15K-237 δημιουργήθηκε με την κατάργηση σπάνιων, σχεδόν διπλότυπων και αντίστροφων σχέσεων, με αποτέλεσμα ένα σύνολο δεδομένων με 237 σχέσεις και χωρίς τριάδες τεστ να μοιράζονται και τις δύο οντότητες με μια τριάδα εκμάθησης.

WN18
Το WordNet (WN) είναι μια βάση γνώσης που περιέχει αισθήσεις λέξεων (δύο αισθήσεις της ίδιας λέξης αντιπροσωπεύονται ως διαφορετικές οντότητες) και λεξικές σχέσεις μεταξύ τους. Παραδείγματα σχέσεων είναι η υποωνυμία, η υπερνυμία και ο παράνομος. Το WN18 είναι ένα δείγμα WN που εξάγεται [13] αποκλείοντας σπάνιες σχέσεις και οντότητες. Το WN18 πάσχει από τα ίδια προβλήματα με το FB15K και το WN18RR δημιουργήθηκε για να τα αντιμετωπίσει χρησιμοποιώντας την ίδια διαδικασία με το FB15K-237.

Parameter	Distribution	Q	Candidate values
Embedding dimension	Uniform	10	$[50, 300]$
Negative triples per positive one	Uniform	2	$[2, 100]$
Loss function	Categorical	-	{ MRL, Logistic, NLL }
Regularization method	Categorical	-	{ Normalization, L_p-Regularization }
Learning Rate	LogUniform	-	$[0.01, 0.000001]$
Batch size	Categorical	-	{1000, 2000, 5000}
L_p-Regularization p	Categorical	-	{ 1, 2, 3 }
L_p-Regularization coefficient	LogUniform	-	$[0.1, 0.00001]$
TransE norm p	Categorical	-	{ 1, 2 }

Table 8.1: Hyper-parameter ranges for KGE models. Some parameters were quiantized with step Q.

Πειραματικές Ρυθμίσεις

Για καθένα από τα παραπάνω σύνολα δεδομένων (FB15K, FB15K-237, WN18 και WN18RR) αύξησα πρώτα τα δεδομένα με αντίστροφες τριάδες , προσθέτοντας μια νέα σχέση r' για κάθε σχέση r και μια νέα τριάδα (t, r', h) για κάθε θετική τριάδα (h, r, t) και μετά αξιολόγησα την απόδοση 3 μοντέλων: TransE, DistMult, ComplEx. Δυστυχώς, το RESCAL απαιτούσε πολύ μνήμη και χρόνο για το εύρος αυτής της μελέτης. Για κάθε μοντέλο έκανα βελτιστοποίηση υπερπαραμέτρων για να μεγιστοποιήσω τη μέση αμοιβαία κατάταξη (MRR) στο σύνολο επικύρωσης χρησιμοποιώντας μια εφαρμογή του BOHB [1] με συντελεστή μισού από 2, ελάχιστο προϋπολογισμό 100 εποχών και μέγιστο προϋπολογισμό 800 εποχές. Ο πίνακας 8.1 δείχνει τις υπερπαραμέτρους που βελτιστοποιήθηκαν και τα αντίστοιχα εύρη τους.

Για όλα τα πειράματα χρησιμοποίησα ομοιόμορφη αρνητική δειγματοληψία με ίση συχνότητα αλλοίωσης για τις οντότητες του υποκειμένου και του αντικειμένου χωρίς να ελέγξω εάν οι τριάδες που δημιουργήθηκαν είναι πραγματικά θετικές. Τα μοντέλα έχουν βελτιστοποιηθεί με το Adam [17]. Η καλύτερη διαμόρφωση για κάθε μοντέλο και σύνολο δεδομένων, εκπαιδεύτηκε στη συνέχεια στα συνδυασμένα υποσύνολα εκπαίδευσης και επικύρωσης για 800 εποχές και αξιολογήθηκε στο σύνολο δοκιμών για να παράγει τα τελικά αποτελέσματα.

Αποτελέσματα

Τα αποτελέσματα των BOHB για TransE, DistMult και ComplEx στα FB15K-237 και WN18RR παρουσιάζονται στον Πίνακα 8.2 και 8.3 αντίστοιχα. Περιέχουν την τελική μέση κατάταξη (MR), τη μέση αμοιβαία κατάταξη (MRR) και την αναλογία των επιτυχιών στα κορυφαία 10 (Hits @ 10) που πέτυχε κάθε μοντέλο στο υποσύ-

[1] `https://github.com/automl/HpBandSter`

	Scores			Best hyper-parameters			
Model	MR	MRR	Hits@10	Loss	d_e	# Neg.	Regularization
TransE (p=1)	1149.73	0.1865	0.3064	NLL	120	84	Norm.
DistMult	**398.29**	**0.2292**	**0.3915**	MRL	190	4	L_2 ($\lambda = 0.0075$)
ComplEx	782.66	0.1508	0.2802	MRL	280	28	Norm.

Table 8.2: Link prediction results for FB15K-237.

	Scores			Best hyper-parameters			
Model	MR	MRR	Hits@10	Loss	d_e	# Neg.	Regularization
TransE (p=2)	9243.86	0.2307	0.3698	MRL	150	6	Norm.
DistMult	**4320.64**	**0.4164**	**0.4836**	NLL	120	88	Norm.
ComplEx	10925.93	0.3557	0.3628	Logistic	120	70	L_2 ($\lambda = 0.001$)

Table 8.3: Link prediction results for WN18RR.

νολο δοκιμών καθώς και τις καλύτερες τιμές που βρέθηκαν για ορισμένες σημαντικές υπερπαράμετρους.

Παρά την αδυναμία του να μοντελοποιήσει ασύμμετρες σχέσεις, το DistMult πέτυχε την καλύτερη απόδοση και στα δύο σύνολα δεδομένων. Είναι σημαντικό να σημειωθεί ότι ο διαθέσιμος προϋπολογισμός για ρύθμιση υπερπαραμέτρων ήταν σημαντικά χαμηλός. Τα μοντέλα KGE μπορούν να διαρκέσουν αρκετές χιλιάδες εποχές για να συγκλίνουν στις καλύτερες λύσεις και υπάρχουν πολλές υπερπαράμετροι για βελτιστοποίηση. Επιπλέον, το BOHB πραγματοποιεί μια απλή τυχαία αναζήτηση για τις πρώτες λίγες επαναλήψεις έως ότου συλλεχθούν αρκετά αποτελέσματα. Είναι πιθανό ότι θα χρειαζόταν υψηλότερος μέγιστος προϋπολογισμός ώστε το στοιχείο Bayesian Optimization του BOHB να έχει σημαντικό αποτέλεσμα. Επομένως, οι σχετικά χαμηλές βαθμολογίες TransE και ComplEx ενδέχεται να οφείλονται σε περιορισμούς πόρων.

8.4.2 Ταξινόμηση οντοτήτων

Δεδομένα

Για την ταξινόμηση οντοτήτων επέλεξα το σύνολο δεδομένων που δημοσιεύτηκε στο [8], το οποίο περιέχει όλες τις αλληλεπιδράσεις χρηστών από τον ιστότοπο κοινωνικής δικτύωσης tagged.com που συνέβησαν εντός περιόδου δειγματοληψίας 10 ημερών. Υπάρχουν 7 τύποι αλληλεπιδράσεων, συμπεριλαμβανομένης της προβολής του προφίλ άλλου χρήστη, της αποστολής αιτημάτων φίλων και της αποστολής μηνυμάτων. Για κάθε ενέργεια χρήστη, καταγράφηκε ο αποστολέας, ο παραλήπτης και η χρονική σήμανση. Ορισμένα δεδομένα για κάθε χρήστη είναι επίσης διαθέσιμα, συμπεριλαμβανομένων 3 χαρακτηριστικών και μιας ετικέτας που υποδεικνύει εάν αυτός

ο χρήστης βρέθηκε ότι ήταν κακόβουλος (spammer) από τη διαχειριστική ομάδα του ιστότοπου εντός κάποιου μη καθορισμένου χρονικού διαστήματος μετά το τέλος της περιόδου δειγματοληψίας των 10 ημερών.

Ένας γράφος πολλαπλών σχέσεων μπορεί να εξαχθεί από αυτό το σύνολο δεδομένων συμπεριλαμβάνοντας τους χρήστες ως κόμβους, τύπους ενεργειών ως σχέσεις και την ύπαρξη τουλάχιστον μιας ενέργειας μεταξύ δύο χρηστών ως συνδέσμου στην αντίστοιχη σχέση. Ένα μειονέκτημα αυτής της μεθόδου είναι ότι αγνοεί τις χρονικές σημάνσεις καθώς και την πολλαπλότητα των ενεργειών. Ωστόσο, ο στόχος αυτής της μελέτης δεν είναι απαραίτητα να βρει την καλύτερη μέθοδο για την ανίχνευση spammer αλλά να μελετήσει την απόδοση διαφόρων μεθόδων για την ταξινόμηση οντοτήτων.

Ο γράφος πολλαπλών σχέσεων που εξάγεται από αυτό το σύνολο δεδομένων, περιέχει 5.607.454 οντότητες και περισσότερους από 300 εκατομμύρια συνδέσμους. Ο μεγάλος αριθμός οντοτήτων οδηγεί σε υψηλή απαίτηση μνήμης για κάθε διάσταση των ενσωματώσεων οντοτήτων. Δεδομένου ότι οι περισσότερες παράμετροι ενημερώνονται για κάθε mini-batch, έτσι πρέπει να είναι αποθηκευμένα στην GPU ανά πάσα στιγμή. Λόγω της περιορισμένης χωρητικότητας της μνήμης GPU, δεν μπορούσα να χωρέσω αρκετές διαστάσεις για να αντιπροσωπεύσω με ακρίβεια τις οντότητες. Επιπλέον, λόγω του μεγάλου αριθμού τριάδων , η βελτιστοποίηση θα μπορούσε να πάρει πολλές ώρες για να συγκλίνει.

Λόγω αυτών των περιορισμών και προκειμένου να αποφευχθεί η κατανεμημένη εκπαίδευση, αποφάσισα να εξαγάγω μικρότερα δείγματα από τον γράφο. Γενικά, είναι αδύνατο να διατηρηθούν όλες οι ιδιότητες ενός γράφου κατά την εξαγωγή ενός δείγματος. Με ομοιόμορφη τυχαία δειγματοληψία κόμβων, ο μέσος βαθμός κόμβου μειώνεται προς το 0 για αρκετά μικρά μεγέθη δείγματος. Η ομοιόμορφη δειγματοληψία ακμών είναι επίσης προβληματική, καθώς ο αριθμός των περιλαμβανόμενων οντοτήτων γίνεται πολύ μεγάλος ακόμη και για σχετικά μικρά μεγέθη δείγματος. Οι πιο προηγμένες μέθοδοι δειγματοληψίας γράφων [14] είναι σε θέση να διατηρήσουν καλύτερα την κατανομή του βαθμού κόμβου. Χρησιμοποιώντας τη μονάδα python Little Ball of Fur [2], δημιούργησα δείγματα διαφόρων μεγεθών με τις ακόλουθες μεθόδους δειγματοληψίας γράφων:

- **ForestFire Sampling (FFS)** [19] δημιουργεί μια ακολουθία διαχωριστικών συνόλων κόμβων ξεκινώντας με ένα μικρό υποσύνολο $V_0 \subset \mathcal{V}$ κόμβων και δημιουργώντας ένα νέο σύνολο V_i στο κάθε i επανάληψη που περιέχει έναν γεωμετρικά κατανεμημένο τυχαίο αριθμό των γειτόνων V_{i-1} που δεν έχουν ήδη επισκεφτεί.
- **Random Walk Sampling (RWS)** [12] ξεκινά με έναν μόνο κόμβο και ακολουθεί τυχαία έναν γείτονα κάθε φορά.
- **Frontier Sampling (FS)** [28], που ονομάζεται επίσης πολυδιάστατη δειγματοληψία τυχαίας πορείας, ξεκινά με ένα σύνολο κόμβων V_0. Σε κάθε επανάληψη i, ένας κόμβος $v \in V_{i-1}$ επιλέγεται τυχαία με πιθανότητα ανάλογη με τον βαθμό v και έναν γείτονα με ομοιόμορφο δείγμα u προστίθεται στο σύνολο, δηλαδή.

[2] `https://github.com/benedekrozemberczki/littleballoffur`

Sampler	Entities	Edges	Spammer ratio
Random Walk (RW)	10000	100938	0.1292
Random Walk (RW)	50000	1000594	0.1153
Forest Fire (FFS)	10000	113246	0.1374
Forest Fire (FFS)	50000	1184603	0.1346
Frontier Sampling (FS)	10000	87088	0.1237
Frontier Sampling (FS)	50000	1009920	0.1197

Table 8.4: Statistics of the sampled datasets for spammer classification.

$V_i = \{u\} \cup V_{i-1}$.

Τα λεπτομερή δεδομένα της δειγματοληψίας του γράφου δίνονται στον Πίνακα 8.4.

Πειραματικές Ρυθμίσεις

Έκανα ταξινόμηση δυαδικών οντοτήτων από άκρο σε άκρο χρησιμοποιώντας το μοντέλο R-GCN με την loss function εγκάρσιας εντροπίας. Χρησιμοποίησα μια εφαρμογή του R-GCN που περιλαμβάνεται στο DGL [3] και προσδιόρισα τις υπερ-παραμέτρους του μεγιστοποιώντας τη Μέση ακρίβεια στο σύνολο επικύρωσης. Όταν το R-GCN εκπαιδεύεται ως ταξινομητής από άκρο σε άκρο, η GDE επαναλαμβάνεται πάνω στις οντότητες (με ετικέτα) οι οποίες είναι πολύ λιγότερες από όλες τις θετικές και αρνητικές τριάδες που χρησιμοποιούνται για την εκπαίδευση μοντέλων KGE. Επίσης, το R-GCN χρησιμοποιεί το Tree-Structured Panzer Estimator (TPE), το οποίο είναι μια προσέγγιση Bayesian βελτιστοποίησης, όπως εφαρμόζεται στην ενότητα Python Hyperopt [4]. Οι υπερ-παράμετροι που επέλεξα να προσδιορίσω και τα αντίστοιχα εύρη τους εμφανίζονται στον πίνακα 8.5. Τα μοντέλα βελτιστοποιήθηκαν χρησιμοποιώντας το Adam με 10 % των παραδειγμάτων ως το τελικό υποσύνολο δοκιμών, το 10 % ως υποσύνολο επικύρωσης για το HPO, ένα άλλο 10 % ως υποσύνολο επικύρωσης για πρόωρη διακοπή και το υπόλοιπο 70 % για εκπαίδευση. Τα τελικά αποτελέσματα παρήχθησαν μετά την επανεκπαίδευση του μοντέλου χρησιμοποιώντας το 80 % των παραδειγμάτων, διατηρώντας παράλληλα το 10 % για πρόωρη διακοπή.

Αποτελέσματα

Αξιολόγησα την απόδοση των βέλτιστων διαμορφώσεων χρησιμοποιώντας τις ακόλουθες μετρικές:

- Ακρίβεια: Ο λόγος των σωστά προβλεπόμενων ετικετών.
- Balanced Accuracy: Ο αριθμητικός μέσος όρος της ευαισθησίας (πραγματικός θετικός ρυθμός) και της ειδικότητας (πραγματικός αρνητικός ρυθμός). Αυτή η

[3] `https://github.com/dmlc/dgl`

[4] `https://github.com/hyperopt/hyperopt`

Parameter	Type	Quantization	Range
Hidden dimension	Uniform	10	$[50, 300]$
Number of Bases	Uniform	1	$[1, \lvert\mathcal{R}\rvert]$
Dropout	Uniform	-	$[0, 0.5]$
Self loop	Categorical	-	$\{\text{True}, \text{False}\}$
Learning Rate	LogUniform	-	$[0.0001, 1]$

Table 8.5: Hyper-parameter ranges for RGCN.

μέτρηση ταιριάζει καλύτερα σε μη ισορροπημένα σύνολα δεδομένων από την κανονική ακρίβεια.

- Μέση ακρίβεια (AP): Η μέση τιμή της ακρίβειας ως συνάρτηση της ανάκλησης σε όλα τα πιθανά κατώφλια. Είναι ίσο με την περιοχή κάτω από την καμπύλη Precision-Recall.
- Περιοχή κάτω από την καμπύλη Receiver Operating Characteristic (ROC-AUC): Η καμπύλη ROC είναι μια γραφική παράσταση του True Positive Rate, η οποία είναι η αναλογία των θετικών παραδειγμάτων που ταξινομήθηκαν ως τέτοια και είναι επίσης γνωστή ως ευαισθησία ή ανάκληση, έναντι του False Negative Rate, το οποίο η αναλογία αρνητικών παραδειγμάτων που ταξινομήθηκαν ως θετικά.

Dataset		Scores			
Sampler	Entities	Accuracy	Balanced acc.	AP	ROC-AUC
RW	10000	0.855	0.5645	0.2399	0.6659
RW	50000	0.8842	0.5813	0.3733	0.7801
FFS	10000	0.859	0.6204	0.3355	0.6629
FFS	50000	0.8736	0.6212	**0.4774**	**0.8008**
FS	10000	0.859	0.5365	0.2597	0.6352
FS	50000	**0.8922**	**0.6719**	0.4769	0.7861

Table 8.6: Entity classification results for R-GCN

Οι καμπύλες ROC για τις καλύτερες διαμορφώσεις στα διάφορα σύνολα δεδομένων φαίνονται στο Σχήμα 5.1. Ενώ ήταν καλύτερο από το τυχαίο, ο ταξινομητής δεν είχε καλή απόδοση στα μικρότερα σύνολα δεδομένων. Οι καμπύλες Precision-Recall φαίνονται στο Σχήμα 5.2.

8.5 Συμπεράσματα

Οι γράφοι είναι πανταχού παρόντες και φέρουν πληροφορίες που μπορούν να χρησιμοποιηθούν για την επίλυση διαφόρων προβλημάτων. Οι παραδοσιακές μέθοδοι μηχανικής εκμάθησης δεν είναι άμεσα εφαρμόσιμες σε δεδομένα με δομή γράφου, αλλά οι μέθοδοι ενσωμάτωσης γράφου παρέχουν έναν τρόπο κωδικοποίησης των πληροφοριών που υπάρχουν στο γράφο σε χώρο χαμηλών διαστάσεων, επιτρέποντας έτσι διάφορες εφαρμογές. Στο πλαίσιο αυτής της διατριβής, μελέτησα και εφάρμοσα αρκετές μεθοδολογίες ενσωμάτωσης γράφου γνώσης και τις αξιολόγησα σε πολλά σύνολα δεδομένων και εφαρμογές σε μια προσπάθεια να εκτιμήσω την ικανότητά τους να εξαγάγουν χρήσιμες πληροφορίες από μεγάλους πολυ-σχεσιακους γράφους.

Το πρώτο σύνολο πειραμάτων αφορούσε την εφαρμογή πρόβλεψης συνδέσμου για την οποία εφάρμοσα μια μονάδα python βασισμένη κυρίως στο PyTorch. Συγκρίνω τα μοντέλα TransE, DistMult και ComplEx σε δύο σύνολα δεδομένων, FB15K-237 και WN18RR, αφού προσδιόρισα έως και 9 από τις υπερ-παραμέτρους τους χρησιμοποιώντας BOHB με περιορισμένο προϋπολογισμό όσον αφορά τις εποχές εκπαίδευσης. Προηγούμενες μελέτες έχουν δείξει ότι η απόδοση αυτών των μοντέλων θα πρέπει να είναι πολύ παρόμοια μετά τη βελτιστοποίηση των υπερπαραμέτρων τους. Τα αποτελέσματα ήταν ευνοϊκά για το DistMult και στα δύο σύνολα δεδομένων, υποδεικνύοντας ότι είναι λιγότερο ευαίσθητο στις επιλεγμένες τιμές σε σύγκριση με τα άλλα μοντέλα που απαιτούσαν υψηλότερο προϋπολογισμό για βελτιστοποίηση.

Το δεύτερο σύνολο πειραμάτων αφορούσε την ταξινόμηση οντοτήτων σε ένα κοινωνικό δίκτυο όπου οι χρήστες συνδέονται με 7 τύπους σχέσεων που δείχνουν διαφορετικές αλληλεπιδράσεις μεταξύ τους και όπου ορισμένοι από αυτούς έχουν χαρακτηριστεί ως spammers. Το σύνολο δεδομένων ήταν πολύ μεγάλο για επεξεργασία, για αυτό χρησιμοποίησα 3 διαφορετικούς αλγορίθμους δειγματοληψίας για να δημιουργήσω δείγματα δύο μεγεθών. Αξιολόγησα την απόδοση του R-GCN ως ταξινομητή από άκρο σε άκρο σε αυτά τα δείγματα μετά από βελτιστοποίηση 5 εκ των υπερπαραμέτρων του χρησιμοποιώντας TPE. Τα αποτελέσματα δείχνουν ότι το R-GCN ήταν σε θέση να αποδώσει σημαντικά καλύτερα από έναν τυχαίο ταξινομητή που δείχνει ότι ήταν σε θέση να εξαγάγει διακριτές πληροφορίες από τον γράφο.

Bibliography

[1] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. arXiv preprint arXiv:2006.13365, 2020.

[2] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. Pykeen 1.0: A python library for training and evaluating knowledge graph emebddings. arXiv preprint arXiv:2007.14175, 2020.

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

[4] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In AAAI. AAAI Press, 2011.

[5] Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. LibKGE - A knowledge graph embedding library for reproducible research. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 165--174, 2020.

[6] Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, Nicholas McCarthy, and Pedro Tabacof. AmpliGraph: a Library for Representation Learning on Knowledge Graphs, March 2019.

[7] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014, pages 601--610, 2014. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz.

[8] Shobeir Fakhraei, James Foulds, Madhusudana Shashanka, and Lise Getoor. Collective spammer detection in evolving multi-relational social networks. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, page 1769–1778, New York, NY, USA,

2015. Association for Computing Machinery.

[9] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale, 2018.

[10] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[11] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[12] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In 2010 Proceedings IEEE INFOCOM, pages 1--9, 2010.

[13] Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data, 2013.

[14] Pili Hu and Wing Cheong Lau. A survey and taxonomy of graph sampling, 2013.

[15] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In Arthur Gretton and Christian C. Robert, editors, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, volume 51 of Proceedings of Machine Learning Research, pages 240--248, Cadiz, Spain, 09--11 May 2016. PMLR.

[16] Georgios Katsimpras and Georgios Paliouras. Class-aware tensor factorization for multi-relational classification. Information Processing & Management, 57(2):102068, 2020.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR (Poster), 2015.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[19] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '05, page 177–187, New York, NY, USA, 2005. Association for Computing Machinery.

[20] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.

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

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[23] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. Proc. IEEE,

104(1):11--33, 2016.

[24] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In ICML, pages 809--816. Omnipress, 2011.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024--8035. Curran Associates, Inc., 2019.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825--2830, 2011.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[28] Bruno Ribeiro and Don Towsley. Estimating and sampling graphs with multidimensional random walks, 2010.

[29] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. pages 498--514, 10 2016.

[30] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. Rdf2vec: RDF graph embeddings and their applications. Semantic Web, 10(4):721--752, 2019.

[31] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks, 2017.

[32] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In ESWC, volume 10843 of Lecture Notes in Computer Science, pages 593--607. Springer, 2018.

[33] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[34] Kristina Toutanova and Danqi Chen. 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, July 2015.

[35] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria Florina Balcan and Kilian Q. Weinberger, editors, Proceedings of The 33rd International Conference on Machine Learning, volume 48 of Proceedings of Machine Learning Research, pages 2071--2080, New York, New York, USA, 20--22 Jun 2016. PMLR.

[36] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315, 2019.

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14, page 1112–1119. AAAI Press, 2014.

[38] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. pages 515--526, 04 2014.

[39] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24, Jan 2021.

[40] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In ICLR (Poster), 2015.