An Open-World Extension To Knowledge Graph Completion Models

Final year project

Haseeb Shah, Abdul Qadir

In partial fulfillment
of the requirements for the degree
Bachelors of Engineering in Software Engineering

School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Islamabad, Pakistan

(2019)

DECLARATION

We hereby declare that this project report entitled **An Open-World Extension To Knowledge Graph Completion Models** submitted to the Department of Computing, is a record of an original work done by us under the guidance of Dr Faisal Shafait and Dr Seemab Latif, and that no part has been plagiarized without citations. Accordingly, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Software Engineering.

Team Members	**Signature**	**Dated**
Haseeb Shah	———————-	———————-
Abdul Qadir	———————-	———————-

Supervisors	**Signature**	**Dated**
Dr. Faisal Shafait	———————-	———————-
Dr. Seemab Latif	———————-	———————-

DEDICATION

This work is dedicated to our parents who have supported us throughout our education and have given us the freedom to forge our own career paths.

ACKNOWLEDGEMENTS

We would like to thank Dr. Adrian Ulges from Hoschule RheinMain and Dr. Faisal Shafait from NUST for their guidance and advice in all stages of this work. Their help has been instrumental in completing our work.

We are also profoundly thankful to Johannes Villmow, a PhD student at Hoschule RheinMain who has made several major contributions to this work. Most of this work would not have been possible in such a short amount of time without his help and expertise.

This work was partially funded by the German Federal Ministry of Education and Research (Program FHprofUnt, Project DeepCA / 13FH011PX6) and the German Academic Exchange Service (Project FIBEVID / 57402798).

PREFACE

This document is arranged as described in the following text. In Chapter 1, we introduce the problem, goals and the significance of our project. In Chapter 2, we perform the literature review. Here, we will go over some background like knowledge graphs, knowledge graph completion, word embeddings and then move on to related work such as DKRL and ConMask. In Chapter 3, we introduce the datasets that are used in the literature and also introduce our dataset, including some details on how it was constructed. In Chapter 4, we discuss several different evaluation metrics that are used in the literature. We discuss the Open-World Extension in Chapter 5 and the Open-World Extension using Entity Typing in Chapter 6. Finally, we conclude in Chapter 7.

Contents

List of Figures

List of Tables

ABSTRACT

In the Natural Language Processing domain, a thriving research topic is learning from large-scale knowledge graphs such as Wikidata, DBpedia, WordNet, or Freebase. These graphs contain large scales of triple-based facts such as (Obama, born_in, Hawaii). Several models have been proposed for representation learning of the graphs' entities and relations, which makes it possible to assess the plausibility of facts not yet present in the graph (Knowledge Graph Completion).

In this thesis, we explore methods that can be used to extend these knowledge graph completion models to determine the plausability of facts which contain entities that were not present in the knowledge graph, a task commonly referred to as Open-World Knowledge Graph Completion. More specifically, we present a novel extension which performs this task by utilizing the textual descriptiosn of the unknown entities. A transformation is learned to map the embeddings of an entity's name and description to the graph-based embedding space. We demonstrate competitive results on several datasets including FB20k, DBPedia50k and our new dataset FB15k-237-OWE. Our approach exploits the full knowledge graph structure even when textual descriptions are scarce, does not require a joint training on graph and text, and can be applied to any embedding-based Knowledge Graph Completion model.

Chapter 1

INTRODUCTION

Knowledge graphs are a vital source for disambiguation and discovery in various tasks such as question answering [6], information extraction [4] and search [21]. They are, however, known to suffer from data quality issues [12]. Most prominently, since formal knowledge is inherently sparse, relevant facts are often missing from the graph.

To overcome this problem, *knowledge graph completion* (KGC) or *link prediction* strives to enrich existing graphs with new facts. Formally, a knowledge graph $G \subset E \times R \times E$ consists of facts or triples $(head, rel, tail)$, where E and R denote finite sets of entities and relations respectively. Knowledge graph completion is targeted at assessing the probability of triples not present in the graph. To do so, a common approach involves representing the entities and relations in triples using real-valued vectors called embeddings. The probability of the triple is then inferred by geometric reasoning over the embeddings. Embeddings are usually generated by learning to discriminate real triples from randomly corrupted ones [11, 20, 25].

A key problem with most existing approaches is that the plausibility of links can be determined for known entities only. For many applications, however, it is of interest to infer knowledge about entities not present in the graph. Imagine answering the question "What is German actress *Julia Lindig* known for?", where *Julia Lindig* is not a known entity. Here, information can be inferred from the question, typically using word embeddings [9, 13, 14]. Similar to entity embeddings, these

techniques represent words with embedding vectors. These can be pre-trained on text corpora, thereby capturing word similarity and semantic relations, which may help to predict the plausibility of the triple $(Julia\_Lindig, starred\_in, Lola\_Rennt)$. This challenge is known as open-world (or zero-shot) KGC. To the best of our knowledge, few open-world KGC models have been proposed so far, all of which are full replacements for regular KGC models and require textual descriptions for all entities [27, 19].

1.1 PROBLEM DEFINITION

This project aims to propose better approaches for open-world knowledge graph completion. The goals of this project are:

Goal 1: Introduce a new dataset for open-world knowledge graph completion and evaluate current approaches on it.

Goal 2: Propose and implement a model-agnostic approach for open-world knowledge graph completion.

Goal 3: Incorporate entity typing approaches into the model.

1.2 SIGNIFICANCE

The main purpose of open-world knowledge graph completion task is to enhance the usability and effectiveness of existing knowledge graphs. Therefore, to highlight the importance of our work, we are going to discuss the significance of knowledge graphs:

1. Knowledge graphs store the semantic structure of information which is an essential component for building intelligent systems. One of the biggest examples of such a system is the Google Knowledge Graph, which is used to power the Google search engine [22]. This knowledge graph stores information about

570 million entities in 18 billion facts. Satori is another knowledge graph that is used by Microsoft to power the Bing search engine [16].

2. Knowledge graphs are used in some specialized domains. For example in the domain of life sciences, knowledge graphs such as Bio2RDF [1] and LinkedLife-Data [10] are used to support decision making and learning.

3. Knowledge graphs are the backbone of intelligent agents and chat-bots like Siri, Google Now and Cortana since they store information in a format that is easily consumed by the computers.

4. Storing information in a graph structure allows us to utilize decades of traditional graph research on real-world data.

Chapter 2

LITERATURE REVIEW

In this chapter, we will first discuss the background necessary to understand the contents of this thesis. Following that, we will discuss the published literature related to our work.

2.1 BACKGROUND

We will cover some essential topics in this section such as knowledge graphs, knowledge graph completion and word embeddings.

2.1.1 Knowledge Graphs

Knowledge Graphs represent data in form of entities and the relationships between them. Most knowledge graphs nowadays are published using W3C's Resource Description Format. This enables the knowledge graphs to be interlinked with each other. In RDF format of knowledge graph representation, the facts are represented in the form of $(head, relation, tail)$ triplets where $head$ and $tail$ are the entities.

An existance of a triplet in the knowledge graph indicates the presence of a true fact. However, there are two main interpretations for a triplet that does not exist:

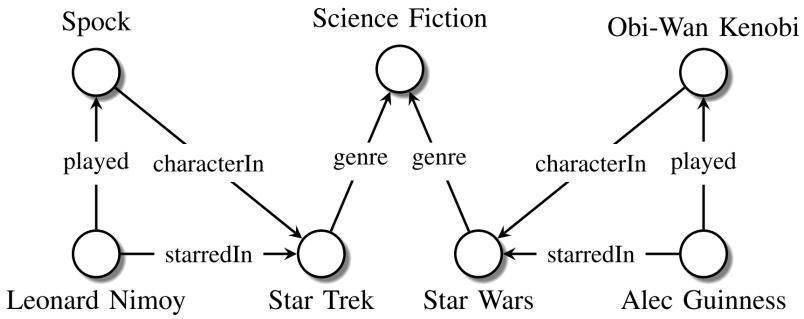


Figure 2.1: An example knowledge graph where the nodes are entities and the edges are the relations [11].

- **Closed-world assumption:** Under this assumption, a non-existent triplet represents a false or untrue fact.
- **Open-world assumption:** Under this assumption, a non-existent triplet represents a fact that is unknown. It may be true or false, but we cannot tell for sure since knowledge graphs are essentially incomplete. This assumption is mostly preferred for this reason.

2.1.2 Knowledge Graph Completion

Existing knowledge graphs are incomplete and continue growing at a rapid pace. To address this, Knowledge Graph Completion (KGC) aims to determine the plausibility of a given triplet. This allows us to find relations between entities in the knowledge graph without having to manually label every entity with each other. Interest in knowledge graph completion has increased in the past few years, with a focus on embedding-based methods. A concise survey of earlier works such as NTN [23] and TransE [3] is provided by [11]. TransE has been recently complemented by other models like DistMult [31], ComplEx [25], ProjE [20] and RDF2Vec [17].

A common approach is to estimate the probability of triples $(head, rel, tail)$ using a scoring function $\phi(u_{head}, u_{rel}, u_{tail})$, where u_x denotes the embedding of entity/relation x and is a real-valued or complex-valued vector. ϕ depends on the model and varies from simple translation [3] over bilinear forms [31] to complex-valued

forms [25]. Training happens by randomly perturbing triples in the graph and learning to discriminate real triples from perturbed ones, typically by using negative sampling.

2.1.3 Word Embeddings

Embedding-based representations of text have become a common approach in natural language processing [7]. They represent terms, sentences or documents in form of a real-valued vector. Word embeddings are known to capture term similarities and semantic relations [9, 13]. Word embeddings can also include sub-word information for out-of-training-vocabulary generalization [2, 14], and have been combined with anchor and link information obtained from Wikipedia to produce entity-specific embeddings [30].

2.1.4 Text-Enhanced Knowledge Graph Completion

While the knowledge graph completion models described above leverage only the triple structure of the graph, some approaches combine text information with the graph information. Some of these approaches regularize word embeddings based on semantic information, for example, by adding synonym and other relations from WordNet to the training set of contextual term pairs [5, 32] or by modelling relations like synonyms with translations [28]. These methods, however, are not targeted at KGC.

Closer to our work are actual KGC models that employ text information, such that for entities scarcely linked in the graph, extra information can be drawn from text. [23, 26] use averaged pre-trained word vectors (CBOW) as entity descriptions for geometric reasoning with affine mappings. [29] tests different aggregation functions for word embeddings (CBOW, LSTMs, attentive LSTMs) and proposes learning an entity-wise linear interpolation between graph-based and text-based embedding, which is combined with a translational KGC model. [24] enhances KGC

with a large set of relations extracted from dependency parses on a text corpus, and formulate a joint loss function for text-based and graph-based inputs. However, none of these works address the open-world setting that we target in this thesis.

2.2 RELATED WORK

In this section, we will discuss the details of the work that is related to our work. As will be highlighted in the following chapters, there is one major difference between our proposed approach and the related work: we train the graph-based and text-based embeddings separately.

2.2.1 DKRL

Description-Embodied Knowledge Representation Learning (DKRL) [27] proposes two types of representations of an entity: structure-based representation E_S that captures information from the structure of the knowledge graph and description-based representation E_D that captures the information from the textual description of an entity. They propose a scoring function that is defined as:

$$E = E_S + E_D \tag{2.1}$$

where E_S is the TransE [3] scoring function given as:

$$E_S = \left|\left|h + r - t\right|\right| \tag{2.2}$$

and E_D is the function that combines description-based representation with the structure-based one. It is defined as:

$$E_D = E_{DD} + E_{DS} + E_{SD} \tag{2.3}$$

here, $E_{SD} = \left|\left|h_s + r - t_d\right|\right|$ in which head has the structure-based representation and tail has the description-based representation. Similarly, $E_{DS} = \left|\left|h_d + r - t_s\right|\right|$ and $E_{DD} = \left|\left|h_d + r - t_d\right|\right|$.

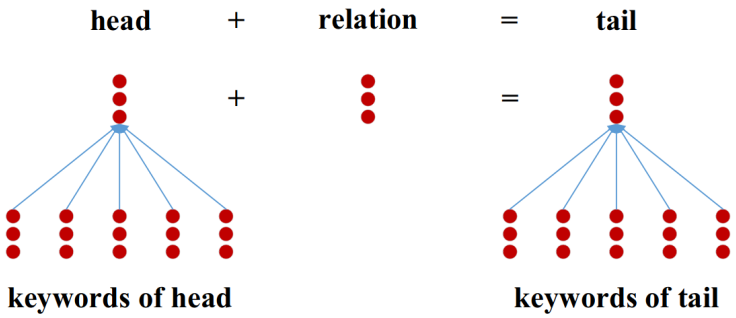


Figure 2.2: The CBOW encoder used by DKRL [27].

These description-based representations are obtained by their proposed encoders. One of the encoders they propose is a simple CBOW (Continuous Bag of Words) encoder. This encoder selects the top n keywords from the entity description and simply sums up their embeddings while ignoring the word orders to calculate E_D.

2.2.2 ConMask

ConMask [19] is a content masking approach for open-world KGC. A brief summary of its major components is given below:

1. Relationship-dependent content masking which assigns different weights to the embeddings of the words according to their relevance to the task. This is done by taking a cosine similarity of embedding of each word in the description of an entity with that of each word in the relationship and taking a maximum.

2. Target fusion which uses a CNN (Convolutional Neural Network) to extract a k-dimentional entity embedding by using the output of content masking.

3. Target entity resolution which calculates the similarity between the target entity candidates in the knowledge graph, extracted entity embeddings and other textual features to obtain a ranked list of target entities.

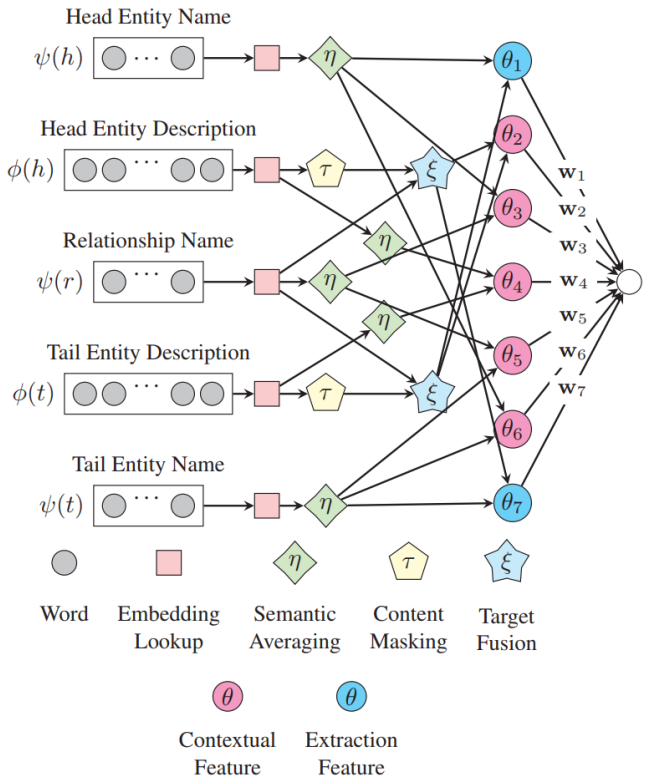


Figure 2.3: The proposed ConMask architecture for open-world knowledge graph completion [19].

The scoring function of ConMask is given as:

$$S\left(h,r,t,E^{\pm}\right) = \begin{cases} \frac{\exp(\text{ConMask}(h,r,t))}{\sum_{e\in E^{\pm}}\exp(\text{ConMask}(e,r,t))}, p_c > 0.5 \\ \frac{\exp(\text{ConMask}(h,r,t))}{\sum_{e\in E^{\pm}}\exp(\text{ConMask}(h,r,e))}, p_c \leq 0.5 \end{cases} \tag{2.4}$$

where p_c is the corruption probability drawn from a uniform distribution and $E^{\pm}$ is the set of all positive and negative entities.

Chapter 3

DATASETS

Closed-world knowledge graph completion models are usually evaluated on datasets such as WN18 and FB15K which where introduced by [3]. FB15K-237 is a subset of FB15K which avoids redundant relations. These datasets are subsets of WordNet and Freebase knowledge graphs and therefore, represent the information in form of triplets. However, these datasets cannot be used for open-world knowledge graph completion.

For open-world knowledge graph completion, FB20K [27] was introduced, which builds upon the FB15k dataset by adding test triples with unseen entities. Additionally, DBpedia50K and DBpedia500K [19] were also introduced. These datasets are capable of evaluating closed-world KGC in addition to open-world KGC. However, there are two main shortcomings of these datasets:

			Number of Triples		
Dataset	$\|E\|$	$\|R\|$	Train	Valid	Test
FB15k / FB20k	14,904	1,341	472,860	48,991	57,803
DBPedia50k	24,624	351	32,388	123	2,095
FB15k-237-OWE	12,324	235	242,489	12,806	-

Table 3.1: Dataset statistics for closed-world link prediction.

1. Both display a bias towards long entity descriptions. DBpedia50k has an av-

erage description length of 454 words while FB20k has length of 147 words.

2. For neither of the datasets, precautions have been taken to avoid redundant inverse relations, which allows models to exploit trivial patterns in the data [24].

To overcome these problems, we introduce a new dataset named FB15k-237-OWE. This dataset is based on the well-known FB15K-237 dataset, where redundant inverse relations have been removed. Also, we avoid a bias towards entities with longer textual descriptions by sampling test entities uniformly from FB15K-237, and using short WikiData descriptions (5 words on average).

		Head Pred.		Tail Pred.	
Dataset	$\lvert E^{open} \rvert$	Valid	Test	Valid	Test
FB20k	5,019	-	18,753	-	11,586
DBPedia50k	3,636	55	2,139	164	4,320
FB15k-237-OWE	2,081	1,539	13,857	9,424	22,393

Table 3.2: Dataset statistics for open-world link prediction. E^{open} is the set of novel entities not in the graph. Note that the corresponding closed-world training set is used for training the open-world models.

3.1 DEVELOPMENT OF FB15K-237-OWE

In this section, we will discuss the difference techniques we used to build our dataset.

3.1.1 Criteria

FB15K-237 is used as a seed to create FB15K-237-OWE. There are three conditions that have to be fulfilled:

1. Subject entites in the test/validation set must not occur as an entity in the training set.

2. Each object entity and each relation must be represented at least once in the training set.

3. Test/validation subject entities must not occur as test/validation object entities.

3.1.2 Approach 1: Dont discard any triplets

In this approach, we try to not discard any triplets in the original FB15K-237 while satisfying the conditions. Following steps are followed to create the dataset:

1. Combine the FB15k-237 train/test/validation splits into a total list.

2. Iterate through total list, add a triplet to the training split if the object or relation is not present atleast once in the training list.

3. Add all the triplets from the total list to training list whose subject entities exist in the training list as subject or object entities.

4. Add all the triplets from total list to training list whose subject entities exist as object entities in the total list.

5. Split the remaining total list randomly between test and validation splits.

Results: The test and validation splits obtained this way are very small (1.5k triplets each) when compared to training set (290k triplets).

3.1.3 Approach 2: Discard rare triplets

We observe that majority of object entities are represented rarely, as shown in the Fig 3.1. We can discard the triplets containing entities that occur less fre-

quently before applying Approach 1. This makes it easier to fulfill the condition 1 and increases the size of test and validation splits.

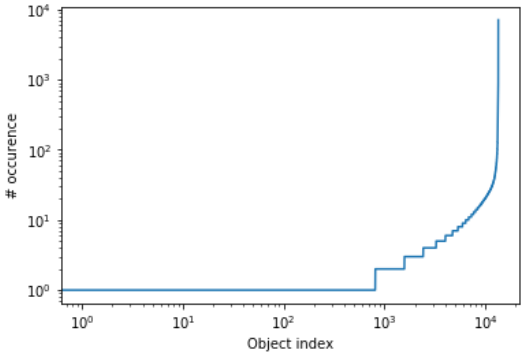


Figure 3.1: Showing the number of times each object entity occurs in the dataset. Horizontal axis shows the entity index while the vertical axis shows its frequency.

Results: The sizes of splits are much better as shown in Table 3.3

Max occurence	Training set	Test set
1	296892	6028
2	291895	9493
3	285054	13802
5	267961	23731

Table 3.3: Dataset statistics when entities with specific number of max occurences are removed from the dataset.

3.1.4 Approach 3: Random sampling

In this approach, we go through the following steps for a specific number of iterations k:

1. Put all triplets into the training set.
2. Pick a random subject entity s that is not already present as an entity in the test set.

3. Get all object entities o from the training set which are in relation with s.

4. Get the object entities o_i that are not in relation with s alone. Go to step 2 if this set is empty.

5. Move all triplets to test set which have s occurring as their subject entity and o_i as object entity.

6. Remove all triplets which have s occuring as their subject or object entity.

7. Go to step 2 until k iterations are complete.

Results: The sizes of splits are even better than the previous approach as shown in Table 3.4. Therefore, we use this approach to develop the final dataset.

k	Training set	Test set
1000	282761	16823
1500	273099	23951
2000	260585	28230
2500	259180	32984

Table 3.4: Dataset statistics when using the random sampling approach for k iterations.

Chapter 4

EVALUATION PROTOCOL

In this chapter, we will discuss various evaluation metrics that are used to evaluate the knowledge graph completion models. The objective of a KGC model is to correctly predict the most plausible object entity for a given set of subject entity and a relation. To evaluate such a model, the object entity in a triplet is replaced by all the object entities present in the knowledge graph and their scores are calculated by the scoring function of a given KGC model. Afterwards, these object entities are sorted in descending order by their scores to calculate their ranks. We will discuss some of the most commonly used metrics and protocols below:

Hits@k

The Hits@k metric gives the percentage of test triplets in which the rank of the target object entity is k or lower. In this thesis, we use 1, 3 and 10 as the values of k during evaluation.

Mean Rank

If Q is the set of test triplets, then mean rank (MR) is given by:

$$Hits@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} rank_i \tag{4.1}$$

Mean rank gives a single real number which denotes the average rank of the target object entity in all the queries. However, it is difficult to interpret the results using mean rank alone as it is sensitive to higher ranks.

Mean Reciprocal Rank

Mean reciprocal rank (MRR) is the average of reciprocal of ranks of object entities for a set of test triplets Q. It is given by:

$$Hits@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{4.2}$$

Mean reciprocal rank gives a real number between 0 and 1. However, we convert it to a percentage. This metric is not as sensitive to higher ranks as the mean rank is.

Filtered vs Raw Ranking

In some cases, multiple triplets with the same head and relation but different tails may occur in the dataset: $(h,r,t_1),...,(h,r,t_p)$. Following [3], when evaluating triple (h,r,t_i) we remove all entities $t_1,...,t_{i-1},t_{i+1},...,t_p$ from the result list. This is called the filtered ranking approach. In contrast, the raw ranking approach would give us true ranks of the queries. All results except *MRR(raw)* are reported using the filtered ranking approach.

Target Filtering

Target filtering is an approach introduced by [19]. When evaluating a test triple (h,r,t), tails t' are only included in the ranked result list if a triple of the form $(?,r,t')$ exists in the training data, otherwise it is skipped. We found this to improve quantitative results substantially, but it limits the predictive power of the model because it prevents the tail entities to be linked with relations if they were not linked in the training set before.

Chapter 5

OPEN-WORLD EXTENSION

In this chapter, we introduce the open-world extension for knowledge graph completion models. We will first go over the theory and workings of this model and then evaluate it on the common datasets.

Our approach starts with a regular link prediction model (in the following also referred to as the *graph-based* model) that is visualised in Fig. 5.1. The model scores triples (h,r,t):

$$score(h,r,t) = \phi(u_h, u_r, u_t) \tag{5.1}$$

where u_x denotes the embedding of entity/relation x. Typically, $u_x \in \mathbb{R}^d$, but other options are possible. For example, in ComplEx [25], u_x is complex-valued ($u_x \in \mathbb{C}^d$). ϕ is a scoring function that depends on the link prediction model and will be adressed in more detail in Section 5.0.1.

Closed-World Knowledge Graph Completion

Closed-world link prediction involves predicting facts about entities on which the link prediction model is trained on. For tail prediction, head and relation are given and the objective is to predict the tail entity with the highest score.

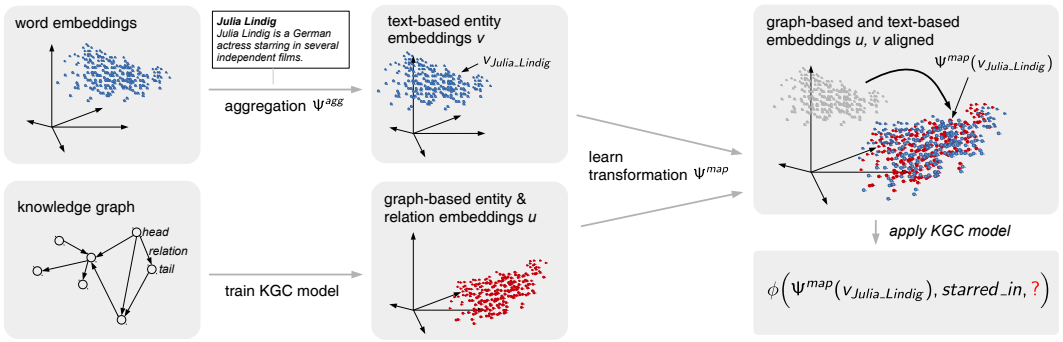


Figure 5.1: Our approach first trains a KGC model on the graph without using textual information (bottom left). For every entity we can obtain a text-based embedding v by aggregating the word embeddings for tokens in the name and description (top left). A transformation Ψ^{map} is learned on the training entities to map v to the space of graph-based embeddings (right). The learned mapping can then be applied to unknown entities, thus allowing the trained KGC model to be applied.

This is done by calculating the score of the (h,r) pair with every $t \in E$.

$$tail^* = \arg\max_{t \in E} \; score(h,r,t) \tag{5.2}$$

Similarly, for head prediction, the score of (r,t) is calculated with every $h \in E$:

$$head^* = \arg\max_{h \in E} \; score(h,r,t) \tag{5.3}$$

For the remaining part of the thesis, we will only discuss the task of tail prediction. The same concepts can be applied to the task of head prediction.

Open-world Extension

Open-world link prediction involves predicting facts about entities that the link prediction model was not trained on. Our contribution lies in extending the above graph-based model to perform open-world link prediction. We assume an unseen head entity $head \notin E$ to be represented by its name and textual description, which we concatenated into a word sequence $\mathscr{W} = (w_1, w_2, ..., w_n)$. Word embeddings (such as Word2Vec or Glove) pre-trained on a text corpus are then used to transform the sequence of tokens $\mathscr{W}$ into a sequence of embeddings $(v_{w_1}, v_{w_2}, ..., v_{w_n})$.

This sequence of embeddings is then aggregated with a function Ψ^{agg} to obtain a text-based embedding of the head entity $v_h \in \mathbb{R}^{d'}$:

$$v_h := \Psi^{agg}\left(v_{w_1}, v_{w_2}, ..., v_{w_n}\right) \tag{5.4}$$

Since text-based and graph-based embeddings are trained independently on different information sources, we cannot expect them to match. Therefore, a transformation Ψ^{map} is learned from text-based embedding space to graph-based embedding space such that $\Psi^{map}(v_h) \approx u_h$. Then we score triples with the unseen head entity by applying the graph-based model from Equation 5.1 with the mapped text-based head description:

$$score(h,r,t) = \phi\left(\Psi^{map}(v_h), u_r, u_t\right) \tag{5.5}$$

The single steps of our model are outlined in more detail in the following.

5.0.1 Link Prediction Models

Since our approach is independent of the specific link prediction model used, we test three commonly used models in this work:

1. TransE: $\phi(u_h, u_r, u_t) = -||u_h + u_r - u_t||_2$
2. DistMult: $\phi(u_h, u_r, u_t) = \langle u_h, u_r, u_t \rangle$
3. ComplEx: $\phi(u_h, u_r, u_t) = \mathrm{Re}(\langle u_h, u_r, \overline{u}_t \rangle)$

Note that the first two use real-valued embeddings, while ComplEx uses complex-valued embeddings (where $\overline{u} = \mathrm{Re}(u) - i \cdot \mathrm{Im}(u)$ denotes the complex conjugate of embedding u). All models are trained using their original loss functions and validated using closed-world validation data.

5.0.2 Word Embeddings and Aggregation

We use pre-trained word embeddings trained on large text corpora. Since the number of entities in the datasets used is limited and we found overfitting to be

an issue, we omit any refinement of the embeddings. We tested 200-dimensional Glove embeddings [13] and 300-dimensional Wikipedia2Vec embeddings [30].

Note that Wikipedia2Vec embeddings contain phrase embeddings, which we use as an embedding for entity names (like "Julia Lindig"). If no phrase embedding is available, we split the name into single tokens and use token-wise embeddings. If no embedding is available for a token, we use a vector of zeros as an "unknown" token.

To aggregate word embeddings to an entity embedding (function Ψ^{agg}, Equation 5.4), approaches in the literature range from simple averaging [13] over Long Short Term Memory Networks (LSTMs) [29] to relation-specific masking [19]. We use averaging as an aggregation function. Here, the word embedding vectors are averaged to obtain a single representative embedding. To prevent overfitting, we apply dropout during training, i.e. embeddings of some words are randomly replaced by the unknown token before averaging.

5.0.3 Transformation Functions

The key to open-world prediction is the mapping from text-based entity embeddings v_e to graph-based ones u_e. Several different transformation functions Ψ^{map} can be learned for this task. In this thesis, we discuss three options:

Linear: A simple linear function $\Psi^{map}(v) = A \cdot v$. For ComplEx, separate matrices are used for the real and imaginary part: $\Psi^{map}(v) = A \cdot v + i \cdot A' \cdot v$

Affine: Here, Ψ^{map} is an affine function $\Psi^{map}(v) = A \cdot v + b$. For ComplEx, separate matrices and vectors are trained just like above: $\Psi^{map}(v) = (A \cdot v + b) + i \cdot (A' \cdot v + b')$

MLP: Ψ^{map} is a four layer Multi-Layer Perceptron (MLP) with ReLU activation functions. The output layer is affine. We did not perform an extensive hyperparameter search here.

To train the transformations, first a link prediction model is trained on the full graph, obtaining entity embeddings $u_1, ..., u_n$. We then choose all entities $e_{i_1}, ..., e_{i_m}$ with textual metadata (names and/or descriptions), and extract text-based embedding $v_{i_1}, ..., v_{i_m}$ for them using aggregation (see above). Finally, Ψ^{map} is learned by minimizing the loss function

$$L(\Theta) = \sum_{k=1}^{m} \left|\left| \Psi^{map}_{\Theta}(v_{i_k}) - u_{i_k} \right|\right|_2 \tag{5.6}$$

using batched stochastic gradient descent, where Θ denotes the parameters of Ψ^{map} (e.g., the weight matrices and bias vectors A, b). For ComplEx, the above loss is summed for real and imaginary parts, and training happens on the sum. We apply no fine-tuning, neither on the graph nor on the text embeddings.

5.1 EXPERIMENTS

In this section, we study the impact of our model's parameters (Ψ^{agg}, Ψ^{map}, text embeddings) on prediction performance. We also provide mappings of selected open-world entities, and compare our results with the state-of-the-art.

5.1.1 Experimental Setup

We perform multiple runs using different KGC models, transformation types, training data, and embeddings used. For each run, both KGC model and transformation Ψ^{map} are trained on the training set: the KGC model without using any textual information and the transformation using entity names and descriptions. We manually optimize all hyperparameters on the validation set. Due to the lack of an open-world validation set on FB20k, we randomly sampled 10% of the test triples as a validation set.

For training TransE and DistMult, we use the OpenKE framework[1] which provides implementations of many common link prediction models. For closed-

[1] OpenKE framework: https://github.com/thunlp/OpenKE

world graph embedding, we use both OpenKE and our own implementation after validating the equivalence of both.

For training the transformation Ψ^{map}, we used the Adam optimizer with a learning rate of 10^{-3} and batch size of 128. For DBPedia50k we use a dropout of 0.5, while for FB20k and FB15k-237-OWE we use no dropout. The embedding used is the pretrained 300 dimensional Wikipedia2Vec embedding and the transformation used is affine unless stated otherwise.

5.1.2 Comparison with State of the Art

We first compare our model ComplEx-OWE with other open-world link prediction models in Table 5.1. For a fair comparison, all the results are evaluated using target filtering. For all models and all datasets, 200-dimensional Glove embeddings were used, except for the Complex-OWE300, which uses 300-dimensional Wikipedia2Vec embeddings. The effect of different embeddings will be studied further in Section 5.1.4.

The results for Target Filtering Baseline, DKRL and ConMask were obtained by the implementation provided by [19]. The Target Filtering Baseline is evaluated by assigning random scores to all targets that pass the target filtering criterion. DKRL uses a two-layer CNN over the entity descriptions. ConMask uses a CNN over the entity names and descriptions along with the relation-based attention weights.

It can be seen from Table 5.1 that our best model, Cmplx-OWE-MULTI, performs competitively when compared to ConMask. On DBPedia50k, our model performs best on all metrics except Hits@10. On FB20k, Cmplx-OWE-300 is outperformed by a small margin by ConMask but performs better on Hits@1. On FB15k-237-OWE our model outperforms all other models significantly. We believe that this is due to FB15k-237-OWE having very short descriptions. ConMask generally relies on extracting information from the description of entities with its attention

Model	DBPedia50k				FB15k-237-OWE				FB20k			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
Target Filt. Base.	4.5	9.7	23.0	11.0*	6.4	14.2	23.3	12.7	17.5	32.1	41.2	27.2
DKRL	-	-	40.0	23.0	-	-	-	-	-	-	-	-
ConMask	47.1	64.5	**81.0**	58.4*	21.5	39.9	45.8	29.9	42.3	**57.3**	**71.7**	**53.3**
Cmplx-OWE-200	49.0	62.3	73.6	57.7	29.1	41.0	52.7	37.3	44.2	55.9	68.2	52.3
Cmplx-OWE-300	51.9	65.2	76.0	60.3	31.6	43.9	56.0	40.1	**44.8**	57.1	69.1	53.1
Cmplx-OWE-MULTI	**55.7**	**69.1**	80.1	**64.3**	**33.2**	**46.0**	**58.8**	**42.0**	-	-	-	-

Table 5.1: Comparison with other open-world KGC models on tail prediction. Cmplx-OWE-MULTI learns a separate transformation function for each relation while the other OWE approaches learn only one function. Note that we used the same evaluation protocol with target filtering as in ConMask. The asterisk (*) denotes that the result differs from the one published, because the MRR is calculated differently.

mechanism, whereas our model relies more on extracting information from the textual corpus that the word embedding were trained on. This enables our model to provide good results without relying on having long descriptions.

5.1.3 Analysis of Different Link Prediction Models and Transformations

Model	MRR		HITS@		
	Filt.	Raw	1	3	10
TransE-OWE	28.7	22.9	21.9	31.7	41.0
DistMult-OWE	34.4	25.7	26.6	37.7	**49.2**
ComplEx-OWE	**35.2**	**26.1**	**27.8**	**38.6**	49.1

Table 5.2: Open-world tail prediction by applying the transformation to different closed-world link prediction models on the FB15k-237-OWE dataset without target filtering.

Our OWE extension for open-world link prediction can be used with any

common KGC model. Therefore, we evaluate three commonly used options, namely TransE, DistMult, and ComplEx. Results are displayed in Table 5.2: All three models are trained with embedding dimensionality $d = 300$ on the closed-world dataset. For text embeddings, Wikipedia2Vec embeddings of the same dimensionality were used. It can be seen that the performance on the open-world setting matches the expressiveness of the models: ComplEx-OWE with its ability to model asymmetric relations yields the best results, while the TransE performs comparatively poorly as it cannot model asymmetric relations.

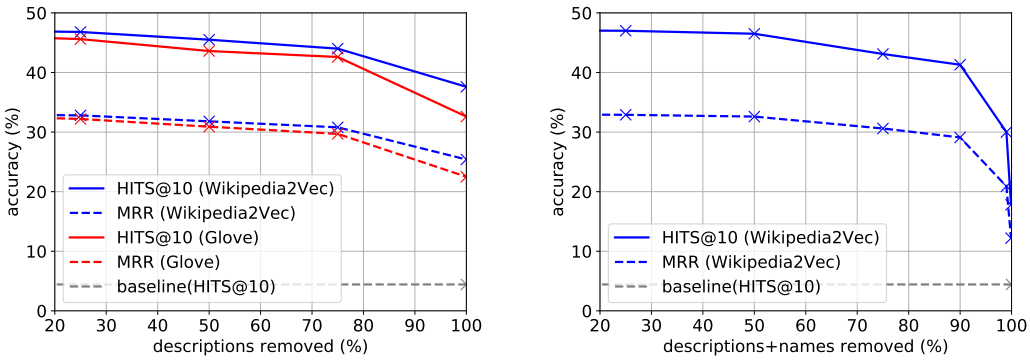
We also test different transformation functions Ψ^{map} as illustrated in Table 5.3. It can be observed that quite simple transformations achieve the strong results: The best performance is achieved by the affine transformation with 49.1% HITS@10 by a margin of 2–4 percent.

	MRR		HITS@		
Transformation	Filt.	Raw	1	3	10
Linear	33.2	25.2	26.1	36.4	46.5
Affine	**35.2**	**26.1**	**27.8**	**38.6**	**49.1**
MLP	32.2	24.7	25.0	35.3	45.6

Table 5.3: Comparison of different transformation functions with ComplEx-OWE-300 on the FB15k-237-OWE dataset without target filtering.

5.1.4 Text Embeddings and Robustness To Missing Entity Metadata

In some cases, the knowledge graph may lack textual metadata (both the name and description) for some or all of its entities. Other models like ConMask and DKRL are dependant on textual descriptions, e.g. ConMask uses attention mechanisms to select relation-specific target words from long texts. Therefore, ConMask and DKRL would require completely dropping triples without metadata and be unable to learn about the link structure of such entities as they use joint training. How-



(a) Scarce Entity Descriptions

(b) Scarce Entity Metadata

Figure 5.2: Performance on FB15k-237-OWE with ComplEx-OWE-300 without target filtering when dropping (a) the entity descriptions or (b) both descriptions and names. The x-axis shows the amount of textual data removed. Even for scarce textual data, learning the transformation Ψ^{map} is robust.

ever, in our approach, we have to drop such entities only during the phase where the transformation Ψ^{map} is learned, while the link prediction model can still be learned on the full graph.

To demonstrate the robustness of our approach to missing entity meta-data, we re-evaluate accuracy when randomly dropping metadata for training entities. Fig. 5.2 outlines the performance for two scenarios:

- **Dropping descriptions**: We remove only the textual descriptions for a varying percentage of randomly selected entities (between 20% to 100%). The names of these entities are not removed and therefore, we still train Ψ^{map} on them.
- **Dropping all meta-data**: We randomly select entities and remove both their descriptions and names, effectively removing these entities from the training set altogether when training Ψ^{map}.

We also included a baseline experiment to simulate an unsuccessful learning of Ψ^{map}. In this baseline, when evaluating a test triple, we replace its head by the embedding of another random head from the training data. Note that this baseline still gives some reasonable hits for triples where the *relation* is a strong indicator.

For example, if we have a triplet $(X, time\_zone, ?)$: Even if the head X is unknown, a model can achieve reasonable accuracy by simply "guessing" time zones as tails.

Overall, Fig. 5.2 suggests that transformation learning is able to generalize well even with very limited training data. In Fig. 5.2a only the descriptions of entities have been removed. For Wikipedia2Vec embeddings, this removal has virtually no effect on prediction accuracy. We believe that this is because Wikipedia2Vec embeddings are trained such that we can lookup strong entity embeddings by the name alone. Even when removing 100% of descriptions (i.e., only training on the entity names), accuracy is only 2-3% lower than training on the full graph. However, in case of Glove embeddings, the drop in performance is very significant, especially when the description is dropped for all the entities.

In Fig. 5.2b, we remove not only descriptions but also entity names. Even in this case, learning is robust. If half of the $12,324$ training entities are removed, the drop in accuracy is less than 1%. Only when removing 90% of training data (leaving 123 training entities), performance starts to deteriorate significantly. This highlights the ability of our model to learn from a limited amount of training data, when it is important to be able to train the KGC model itself on *all* the entities.

5.1.5 Selected Results

Finally, we inspect sample prediction results for ComplEx-OWE-300 in Table 5.4. Besides the final prediction, we also test whether our transformation from text-based to semantic space is successful: For each test triple, we represent the open-world head entity by its text-based embedding v_{head}, match it to a graph-based embedding $\Psi^{map}(v_{head})$, and estimate the nearest neighbor entities in this space. We use the Euclidean distance on the real part of the ComplEx embeddings, but found results to be similar for the imaginary part.

If the transformation works well, we expect these nearest neighbors to be semantically similar to the head entity. This is obviously the case: For *Bram Stoker*

(the author of Dracula), the nearest neighbors are other authors of fantasy literature. For *Parma*, the neighbors are cities (predominantly in Italy). For *Bachelor of Science*, the model predicts appropriate entities (namely, Universities) but – even though we apply filtering – the predictions are not rewarded. This is because the corresponding triples, like *(Bachelor of Science, /.../institution, Harward Law School)*, are missing in the knowledge graph.

Test Triple	Head Description	Top 4 Nearest Neighbors	Top 4 Predictions
(**Bram Stoker**, /.../profession, **Writer**)	Irish novelist and short story writer, best known today for his 1897 Gothic novel Dracula	1. Ursula K. Le Guin 2. Charles Stross 3. Larry Niven 4. Kurt Vonnegut	**1. Writer** 2. England 3. United Kingdom 4. Author
(**Parma**, /.../country, **Italy**)	Italian comune	1. Essen 2. Mantua 3. Bergamo 4. Siena	1. Portugal 2. Netherlands 3. Germany **4. Italy**
(**Bachelor of Science**, /.../institution, **Kingston University)**	Academic degree	1. Masters Degree 2. Doctoral Degree 3. Bachelor of Arts 4. Master of Arts	1. Harvard Law School 2. Wesleyan University 3. Panjab University 4. Baylor University **32. Kingston University**
(**Amtrak**, /.../headquarters/.../town, **Washington DC)**	Intercity rail operator in the United States	1. AT&T 2. Southwest Airlines 3. Starbucks 4. Delta Airlines	1. Dublin 2. New York City 3. United States Dollar 4. Charlotte **72. Washington DC**

Table 5.4: Selected results on FB15k-237-OWE. For each test triple (Column 1), the head's name and description (Column 2) is mapped to graph-based embedding space. The nearest training entities in that space (Column 3) indicate a good semantic match. The model predicts reasonable tails, in the first two cases successfully, in the others not.

Chapter 6

OPEN-WORLD EXTENSION USING ENTITY TYPING

Most of the existing knowledge graph completion models only concentrate on the structural information present in triples, and ignore the information present in the types of the entities. Entities in most of the knowledge graphs have more than one type, which could serve as supplementary information for link prediction.

Some of the recent methods have focused on the utilization of rich information present in the hierarchical types of entities. One of the methods [18] uses several different relationship specific type encoders to construct type projection matrices for each entity. As each entity can have multiple types and only a specific type is being used in a triple, this method learns different type projection matrices for each entity in different triples.

Another method [15] learns multiple vector representations for each entity and relation to capture structural as well as type information without utilizing any entity typing dataset. This method learns two vector representations for each entity to encode both structural information and type information. For each relation the model learns three vector representation to encode information about expected types for head and tail entities and the structural information of the relation in the triple. This methods only learns six or seven most generic entity types.

The methods discussed above only utilize the entity typing for closed-

world link-prediction models. In our work, we utilize the entity typing for open-world link prediction.

6.1 HIERARCHICAL ENTITY TYPES

In Freebase dataset each entity has multiple types which are arranged in a hierarchy from a generic entity type to a more specific one. Figure 6.1 shows the multiple hierarchical types of entities **William Shakespeare** and **Romeo and Juliet** in the triple. The entity types in the first layer i.e. music, award, book and TV are the generic entity types whereas the types in the second layer i.e artist, award nominee, author, written work and TV subject are more precise. The solid lines indicate the most significant role played by the head and tail entities in this triple. Although the entity **William Shakespeare** has multiple types but in this specific triple he is playing the role of author. Similarly the entity type of **Romeo and Juliet** in this triple is book or written work not a TV subject (movie).

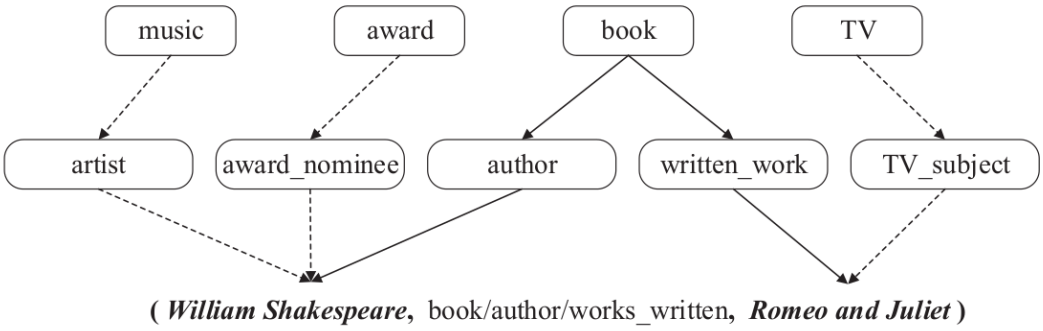


Figure 6.1: Example of hierarchical entity types in Freebase dataset taken from [18].

6.1.1 Dataset

We have prepared the hierarchical entity typing dataset for FB15k-237-OWE using the FB15k entity typing dataset provided by [18]. FB15k-237-OWE is the dataset for open-world link prediction and is the subset of FB15k dataset.

In the hierarchical entity typing dataset each entity has multiple types ar-

ranged in a hierarchy from generic entity type to the more specific one as shown in the figure 6.2. When we move from top to the bottom in an entity type hierarchy we get more precise information about the entity. The entity type 'media common' of **Jerry Maguire** doesn't give much information about the entity but the sub-type 'Netflix title' tells us that **Jerry Maguire** is a movie and is also available on the Netflix.

Having multiple types means that the entity plays different roles in different triples.

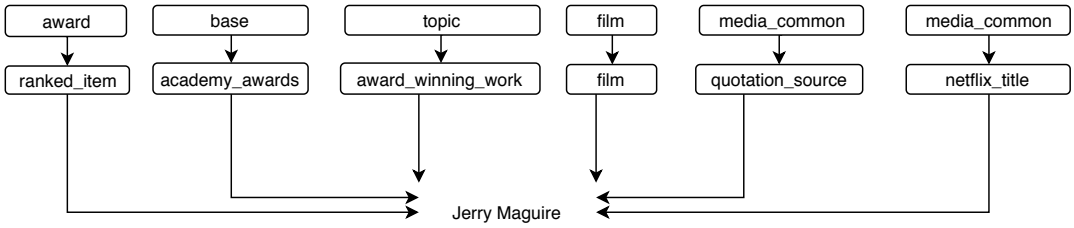


Figure 6.2: Example of hierarchical entity types. In this example the entity **Jerry Maguire** has multiple hierarchical types as shown by arrows. The first row represents the generic entity types whereas the entity types in the second row are more specific ones.

In a triple only those heads and tails can be combined with the relation whose types are compatible with the relation. For example: we cannot combine a tail entity of type 'food' with the relationship 'film directed by'. In hierarchical entity typing dataset, each relation has multiple expected types for head and tail entities as shown in the figure 6.3.

Figure 6.3: Example of the expected types of head and tail for a relation from hierarchical entity typing dataset [18]. In this example the relations are **film** and **film art direction by**. The expected head and tail types for these relations are **film/movie** and **film/film art director** respectively.

6.1.2 Average Type Encoder

In average type encoder we calculate the projection vector T_e of an entity e by taking summation of the embeddings of all types of the entity. We calculate the type embedding of each entity type using Wikipedia2vec [30]. Wikipedia2vec embeddings contain phrase embeddings which we use for entity types (like "Film Director"). If no phrase embedding is available we split the entity type into tokens and use token-wise embeddings. These token-wise embeddings are finally added to get the entity type embedding.

$$T_e = T_1 + T_2 + T_3 + ... + T_n \tag{6.1}$$

where n is the number of types entity e has and T_i is the type embedding of entity type i.
Finally the projection vector is multiplied with corresponding entity embedding.

$$v_e = v_e * T_e \tag{6.2}$$

where v_e is the text-based entity embedding vector.

6.1.3 Results

We are using Open-world Extension (OWE) model as baseline for experimentation. In the table 6.1 we show the comparison of OWE and OWE using average type encoder. It can be seen that there is small improvement in the results by using average type encoder. We got maximum improvement on Hits@10 which is around 1%.

6.2 TYPE FILTERING DURING EVALUATION

During evaluation, in order to predict the tail entity for a head entity and the relation, the link-prediction model takes all the tails in the knowledge graph and

Model	MRR		HITS@		
	Filt.	Raw	1	3	10
OWE	35.2	26.1	27.8	38.6	49.1
OWE-AverageTypeEncoder	**35.82**	**26.58**	**28.21**	**39.16**	**50.06**

Table 6.1: Comparison of open-world tail prediction using average type encoder and baseline model OWE.

calculates scores for each of them. The scores are then sorted in descending order to pick the correct tail entity. The model calculates the score for each tail entity without considering its type compatibility with the relation in the triple.

Based on type compatibility between a relation and the head and tail entities [8] has published an entity filtering dataset for FB15k and DBPedia50. For each relation the dataset contains all the head and tail entities whose types match the expected types of the head and tail entities. As the dataset FB1k-237-OWE is subset of FB15k so we have prepared an entity filtering dataset for FB15k-237-OWE using FB15K type filtering dataset. During evaluation we set the score of those entities to -infinity whose types don't match the expected types of the relation using type filtering datasets provided by [8].

6.2.1 Results

Model	DBPedia50k				FB15k-237-OWE			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
OWE	34.64	46.37	53.66	42.84	27.8	38.6	49.1	35.2
OWE-Entity-Filtering	**39.56**	**51.04**	**60.49**	**47.05**	**28.4**	**39.15**	**50.0**	**35.92**

Table 6.2: Comparison of open-world tail prediction using entity filtering with the OWE on DBPedia50k and FB15k-237-OWE.

Table 6.2 shows the comparison of OWE-EntityFiltering and OWE on FB15k-237-OWE and DBPedia50k datasets. It can be seen that the entity filtering has improved the results on both datasets. On FB15k-237-OWE dataset we got maximum improvement on H@10 which is 0.96%. However on DBPedia50k dataset the results have significantly improved. The maximum improvement is around 7% on Hits@10.

Chapter 7

CONCLUSION

In this work, we have presented a simple yet effective extension to embedding-based knowledge graph completion models (such as ComplEx, DistMult and TransE) to perform open-world prediction. Our approach – which we named OWE – maps text-based entity descriptions (learned from word embeddings) to the pre-trained graph embedding space. In experiments on several datasets (including the new FB15K-237-OWE dataset we introduced in this work), we showed that the learned transformations yield semantically meaningful results, that the approach performs competitive with respect to the state of the art, and that it is robust to scarce text descriptions.

An interesting direction of future work will be to combine our model with approaches like ConMask [19], which (1) exploit more complex aggregation functions and (2) use relation-specific attention/content masking to draw more precise embeddings from longer descriptions.

Bibliography

[1] BELLEAU, F., NOLIN, M.-A., TOURIGNY, N., RIGAULT, P., AND MORISSETTE, J. Bio2rdf: Towards a mashup to build bioinformatics knowledge system. *Journal of biomedical informatics 41* (04 2008), 706–16.

[2] BOJANOWSKI, P., GRAVE, E., JOULIN, A., AND MIKOLOV, T. Enriching Word Vectors with Subword Information. *TACL 5* (2017), 135–146.

[3] BORDES, A., USUNIER, N., GARCIA-DURAN, A., WESTON, J., AND YAKHNENKO, O. Translating Embeddings for Modeling Multi-relational Data. In *Adv. in Neural Information Processing Systems* (2013), pp. 2787–2795.

[4] DONG, X. L., GABRILOVICH, E., HEITZ, G., HORN, W., LAO, N., MURPHY, K., STROHMANN, T., SUN, S., AND ZHANG, W. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proc. KDD* (2014), pp. 601–610.

[5] FARUQUI, M., DODGE, J., JAUHAR, S. K., DYER, C., HOVY, E. H., AND SMITH, N. A. Retrofitting Word Vectors to Semantic Lexicons. In *NAACL HLT* (2015), pp. 1606–1615.

[6] FERRUCCI, D., BROWN, E., CHU-CARROLL, J., FAN, J., GONDEK, D., KALYANPUR, A. A., LALLY, A., MURDOCK, J. W., NYBERG, E., PRAGER, J., SCHLAEFER, N., AND WELTY, C. Building Watson: An Overview of the DeepQA Project. *AI Magazine 31*, 3 (2010), 59–79.

[7] GOLDBERG, Y. A Primer on Neural Network Models for Natural Language Processing. *J. Artif. Int. Res. 57*, 1 (Sept. 2016), 345–420.

[8] HAN, XU, S. C. X. L. Y. L. Z. L. M. S., AND LI, J. Openke: An open toolkit for knowledge embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (2018), pp. 139–144.

[9] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G., AND DEAN, J. Distributed representations of words and phrases and their compositionality. *CoRR abs/1310.4546* (2013).

[10] MOMTCHEV, V., PEYCHEV, D., PRIMOV, T., AND GEORGIEV, G. Expanding the pathway and interaction knowledge in linked life data. In *In Proc. of International Semantic Web Challenge* (2009).

[11] NICKEL, M., MURPHY, K., TRESP, V., AND GABRILOVICH, E. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE 104*, 1 (2016), 11–33.

[12] PAULHEIM, H. Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. *Semantic Web 8*, 3 (2017), 489–508.

[13] PENNINGTON, J., SOCHER, R., AND MANNING, C. D. Glove: Global Vectors for Word Representation. In *Proc. EMNLP* (2014).

[14] PETERS, M. E., NEUMANN, M., IYYER, M., GARDNER, M., CLARK, C., LEE, K., AND ZETTLEMOYER, L. Deep contextualized word representations. In *Proc. NAACL* (2018).

[15] PRACHI JAIN, PANKAJ KUMAR, M. A. C. Type-sensitive knowledge base inference without explicit type supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2018), pp. 75–80.

[16] QIAN, R. Understand your world with bing. https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/, 2013. [Online; accessed 24-April-2019].

[17] RISTOSKI, P., AND PAULHEIM, H. RDF2Vec: RDF Graph Embeddings for Data Mining. In *Proc. ISWC* (2016), pp. 498–514.

[18] RUOBING XIE, ZHIYUAN LIU, M. S. Representation Learning of Knowledge Graphs with Hierarchical Types. In *the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)* (2016), pp. 2965–2971.

[19] SHI, B., AND WENINGER, T. Open-World Knowledge Graph Completion. *CoRR abs/1711.03438* (2017).

[20] SHI, B., AND WENINGER, T. ProjE: Embedding Projection for Knowledge Graph Completion. In *Proc. AAAI* (2017), pp. 1236–1242.

[21] SINGHAL, A. Introducing the Knowledge Graph: Things, not Strings. https://googleblog.blogspot.com /2012/05/introducing-knowledge-graph-things-not.html, last retrieved: Aug 2018), 2012.

[22] SINGHAL, A. Introducing the knowledge graph: things, not strings. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/, 2012. [Online; accessed 24-April-2019].

[23] SOCHER, R., CHEN, D., MANNING, C. D., AND NG, A. Y. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* (USA, 2013), NIPS'13, Curran Associates Inc., pp. 926–934.

[24] TOUTANOVA, K., AND CHEN, D. Observed Versus Latent Features for Knowledge Base and Text Inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality* (July 2015).

[25] TROUILLON, T., WELBL, J., RIEDEL, S., GAUSSIER, É., AND BOUCHARD, G. Complex Embeddings for Simple Link Prediction. In *Int. Conference on Machine Learning* (2016), pp. 2071–2080.

[26] WANG, Z., AND LI, J. Text-enhanced Representation Learning for Knowledge Graph. In *Proc. International Joint Conference on Artificial Intelligence* (2016), pp. 1293–1299.

[27] XIE, R., LIU, Z., JIA, J., LUAN, H., AND SUN, M. Representation Learning of Knowledge Graphs with Entity Descriptions. In *Proc. AAAI* (2016), pp. 2659–2665.

[28] XU, C., BAI, Y., BIAN, J., GAO, B., WANG, G., LIU, X., AND LIU, T.-Y. RC-NET: A General Framework for Incorporating Knowledge into Word Representations. In *Proc. Int. Conf. on Information and Knowledge Management* (2014), pp. 1219–1228.

[29] XU, J., QIU, X., CHEN, K., AND HUANG, X. Knowledge Graph Representation with Jointly Structural and Textual Encoding. In *Proc. Int. Joint Conference on Artificial Intelligence* (2017), pp. 1318–1324.

[30] YAMADA, I., SHINDO, H., TAKEDA, H., AND TAKEFUJI, Y. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proc. SIGNLL Conference on Computational Natural Language Learning* (Berlin, Germany, August 2016), pp. 250–259.

[31] YANG, B., YIH, W., HE, X., GAO, J., AND DENG, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *CoRR abs/1412.6575* (2014).

[32] YU, M., AND DREDZE, M. Improving Lexical Embeddings with Semantic Knowledge. In *Proc. ACL* (2014), pp. 545–550.