ResqLP: Relation Sequences for Link Prediction in Knowledge Graphs

Emil S. Bækdahl
ebakda16@student.aau.dk
Department of Computer Science
Aalborg University

18th June 2021

Summary

This project paper focuses on the link prediction (LP) problem for knowledge graphs (KGs). A KG is a collection of structured data that captures facts, or knowledge, about a domain by linking entities with relations. These facts collectively make up a directed graph. The LP problem is concerned with inferring new facts in a KG given its current state. This problem has many practical use cases. For instance, in bioinformatics, LP can be used to predict new possible drug targets and to reason about the interactions of chemical substances. LP is also found useful in question answering and recommender systems.

In recent years, as deep learning continues to impact the machine learning (ML) world, research in LP models has gained momentum. Even though many of the proposed models exhibit increasingly better performance, most of them are hard to interpret. In other words, they can predict new facts but cannot explain why and how they do it. Since explainability is becoming more and more important in ML, we choose to develop an approach to LP where the ability to explain predictions is integrated into the model itself. We call this model ResqLP.

In the paper, we start by introducing the LP problem and explain a handful of existing approaches to solving it. Here, we find models based on recurrent neural networks (RNNs) interesting since they have two properties that complement each other well: they are efficient to train and they come with the ability to provide explanations for their prediction in the form of first-order logic rules. However, they suffer from scalability issues since they learn from sequences of relations found by enumerating paths in the KG which is a hard, though well-known, problem.

We then propose ResqLP that builds upon these RNN-based models but furthermore includes an enhanced feature extraction phase that aims to cope with the mentioned scalability issue. In this phase, the model only considers a subset of the entire KG when finding relation sequences. Furthermore, we propose to derive a notion of semantics of the entities based on observable features in the graph. The relation sequences and entity semantics are encoded using an RNN that outputs a probability describing how likely it is that the two entities should be linked.

While describing the architecture of ResqLP, we analyse real KGs to guide our design. After that, we evaluate the model on these KGs and find that its performance varies between them. ResqLP performs best or next-best in some cases and worst in others. This is due to the fact that the KGs are structured differently. We find that ResqLP is best suited for KGs that are sparsely connected and have many different types of relations while it exhibits worse performance on KGs with few relations. On one of the KGs, we examine the logic rules that ResqLP learns and find a varying degree of quality and usefulness. Finally, we end the paper by proposing pointers to future work on this type up LP model.

Abstract

Knowledge graphs capture domain knowledge in the form of facts about entities and have many use cases in information systems. One of these is link prediction which is the task of inferring new facts in a knowledge graph. Even though research in the field gains momentum, few models focus on being able to explain their predictions. In this paper, we introduce ResqLP, a link prediction model that learns probabilistic first-order logic rules from both entity semantics and sequences of relations. The model includes a feature extraction phase that, by the use of enclosing subgraphs, aims to cope with the inherently hard path enumeration problem. We also present a general way to capture entity semantics based on relations found in entity neighbourhoods.

We find that using enclosing subgraphs speeds up feature extraction even though large knowledge graphs still cause problems in some cases. Furthermore, even though the quality varies, ResqLP is able to learn non-trival logic rules. The model performs the best on relatively sparsely connected knowledge graphs with many different relations.

1 Introduction

In an increasingly complex society, information systems play a more and more important role and are set out to solve equally complicated tasks. Traditionally, many areas, such as medicine and finance, have relied on domain experts with highly specialised knowledge. Over recent years, we have seen this domain knowledge making its way into the information systems themselves [14, 15]. Due to this evolution, research in knowledge representation and appropriate machine learning (ML) agents gains momentum. In such applications, knowledge graphs (KGs) are widely used as an intuitive representation of knowledge. A KG is a structured collection of data that captures knowledge in the form of facts expressed as a relations between two entities. For instance, in a KG over fiction literature, the fact that Margaret Mitchell wrote *Gone with the Wind* can be captured by linking the entities M. MITCHELL and GONE WITH THE WIND using the relation WROTE. Collectively, such facts form a directed graph as shown in Figure 1. KGs like this are used to provide real-world knowledge to ML agents in different scenarios such as recommender systems [36] and question answering [40].

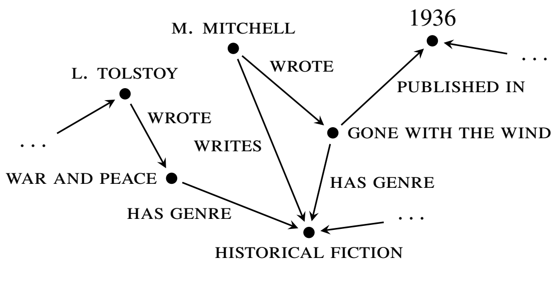


Figure 1: A segment of an example KG over fiction literature.

KGs are typically very large [35], automatically constructed [30], or continuously changing [19]. Thus, manually verifying the completeness and validity of their content is infeasible. Therefore, there is an increasing interest in automating this task. One central problem here is to infer new knowledge in an existing KG. This task is commonly referred to as link prediction (LP) as it aims to predict whether two entities in a KG should be connected with a given relation. A practical example of this is found in the medical domain where LP can be used to predict new targets or possible side-effects of drugs [11]. In this case, the entities in the KG are chemical substances while the relations between them describe how they interact. The LP task is then to predict how two chemical substances, that are not already linked in the KG, interact.

Since the LP problem has practical applications in many fields, it has been approached in just as many different ways. Some techniques are based on classic graph theory measures such as common neighbours and PageRank [12]. However, more recent LP literature focuses on developing ML models that either learn vector representations of the KG [2, 5, 9, 13, 23, 29, 31, 33, 39] or infer probabilistic first-order logic rules [7, 8, 22, 26]. The former group, called latent feature models (LFMs), have the advantage of being efficient to train and perform relatively well on common benchmark KGs. However, since they learn latent representations that do not translate directly to observable features or patterns in the KG, such models are hard to interpret. Models in the latter group, called logic rule models (LRMs), come with a higher degree of interpretability. They learn first-order logic rules based on common patterns in the KG. These rules can provide valuable explanations as support to the predictions of the model.

In this paper, we propose an LRM based on a recurrent neural network (RNN) that learns from entity semantics and sequences of relations found in the KG. We call this model ResqLP; an abbreviation of 'relation sequence link prediction'. The RNN itself is inspired by [8, 22]. However, ResqLP includes a feature extraction phase that aims at coping with large search spaces, inspired by [31, 39]. In this phase, we first extract a subset of the KG that encloses the two entities for which we are to predict a link. Then, we enumerate the sequences of relations that connect the two entities in that subgraph. We find that this approach significantly speeds up the enumeration process for some KGs. However, in other cases, we see that the entities are so densely connected that the enclosing subgraph does not provide a noteworthy speedup. In those cases, we sample an even smaller subgraph to make the task feasible.

By comparing ResqLP to other LP models, we find that, even though ResqLP performs the best and next-best in some cases, its performance depends on the structure of the KG in question. We find that ResqLP is best suited for KGs that have many different relations and where the entities are relatively sparsely connected. In such KGs, the sequences of relations between two entities are fewer but more varied.

The rest of the paper is structured in the following way. In Section 2, we introduce preliminary concepts behind KGs and LP and define them formally, while Section 3 describes some existing approaches to the problem and outlines their advantages and disadvantages. We describe the overall architecture and idea behind ResqLP in Section 4 and go further in depth with the feature extraction phase and RNN in Sections 5 and 6, respectively. In those sections, we consult real KGs to identify key properties that guide our decisions in the design of the model. Finally, we evaluate ResqLP and compare its performance to other models in Section 7. We end with concluding remarks and pointers to future work in Section 8.

2 Preliminaries

In the following two sections, we formally define KGs and introduce relevant notation. Given this definition, we formulate the LP as a binary classification problem.

2.1 Knowledge Graphs

A knowledge graph (KG) is a collection of structured data that describes facts, i.e. knowledge, about how different entities are related. An entity can be a concrete or abstract object such as a person, an event, a film, or a genre. A pair of entities is connected by zero or more relations that describe associations between them. For instance, in Figure 1, the relation HAS GENRE connects the entities WAR AND PEACE and HISTORICAL FICTION. Together, this relation and these entities describe a unit of knowledge; in this case that *War and Peace* is historical fiction. This unit can intuitively be represented as a so-called triple, (WAR AND PEACE, HAS GENRE, HISTORICAL FICTION). The left entity in a triple is called the head entity, while the right is called the tail entity. Formally, we define a KG in the following way.

Definition 1 (Knowledge graph)**.** A knowledge graph (KG) over the set of entities $\mathcal{E}$ and set of relations $\mathcal{R}$ is a set of triples $\mathcal{K} = \{(h, r, t) \mid h, t \in \mathcal{E} \land r \in \mathcal{R}\}$.

Remark. When referring to a triple in a context where one or more of its elements are irrelevant, we use dots in place of said constituents. For instance, $(h, r, \cdot)$ refers to any triple that has h and r as head entity and relation, respectively, while the tail entity can be any $e \in \mathcal{E}$.

A KG has a natural graphical representation in the form of a directed labelled multigraph. Each entity $e \in \mathcal{E}$ represents a node labelled with e while each triple $(h, r, t) \in \mathcal{K}$ represents a directed edge from h to t labelled with r. This representation allows us to intuitively reason about paths in a KG as we would in any graph. A path is an alternating sequence of entities and relations, such as $\langle e_1, r_1, e_2, r_2, e_3, \ldots, e_{n-1}, r_{n-1}, e_n\rangle$. The existence of this path is implied by $(e_1, r_1, e_2), (e_2, r_2, e_3), \ldots, (e_{n-1}, r_{n-1}, e_n) \in \mathcal{K}$. We make use of these paths later when describing ResqLP.

2.2 Link Prediction

Now that we have defined what a KG is, we can formulate the link prediction (LP) problem in this context. As described earlier, the LP problem is concerned with predicting whether a candidate triple (h, r, t) belongs in a given KG. In existing research, this is generally approached in one of two ways. The first considers LP as an information retrieval task. In this case, an ML model learns a score function $f(h, r, t)$ that assigns a real valued score to any candidate triple (h, r, t). The model learns to give relatively high values to triples $(h, r, t) \in \mathcal{K}$ and relatively low values to triples $(h', r', t') \notin \mathcal{K}$. When presented with a list of candidate triples, the model ranks them by score in descending order. The triples most likely to be belong in the KG appear at the top of the list while the least probable triples appear at the bottom.

The second perspective, sometime referred to as triple classification [28, 37], sees LP as a binary classification problem. Here, a model is trained to determine whether a candidate triple belongs or does not belong in the KG. This is typically done in a probabilistic manner where the model outputs the probability of h and t being connected by r, denoted $P(r \mid h, t)$. In this paper, we propose a model based on this view. Thus, we define the problem in the following way.

Definition 2 (Link prediction)**.** Let $\mathcal{K}^*$ be a KG of all true knowledge in a certain domain and let $\mathcal{K}$ be a KG of observed knowledge such that $\mathcal{K} \subset \mathcal{K}^*$. Given a triple $(h, r, t) \notin \mathcal{K}$, the link prediction (LP) problem is to suggest the probability of $(h, r, t) \in \mathcal{K}^*$, denoted $P(r \mid h, t)$.

Remark. We use the terms 'true' and 'false' about triples that belong and do not belong in a KG, respectively. Important, here is the distinction between $\mathcal{K}$ and $\mathcal{K}^*$. That fact that a triple is true does not necessarily mean that it can be found in $\mathcal{K}$ but rather that it is in $\mathcal{K}^*$. Thus, true triples that are not in $\mathcal{K}$ are so-called missing knowledge.

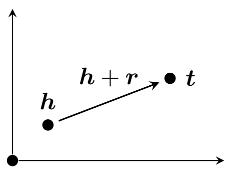
With a definition of KGs and the LP problem in place, we now look at how the task is approached in existing literature.

3 Related Work

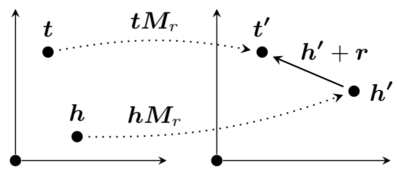
We divide approaches to solving the LP problem into two major groups: latent feature models (LFMs) and logic rule models (LRMs). The former is concerned with learning latent vector representations of entities and relations, while the latter aims at deriving probabilistic first-order logic rules based on how entities are connected through one or more triples. The following sections highlight some concrete models in each group and outline their advantages and disadvantages. Following that, we describe how ResqLP fits in.

3.1 Latent Feature Models

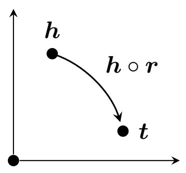
A latent feature model (LFM) learns latent feature embeddings of entities and relations. We subdivide this category further into three groups: models based on distance in the embedding space, models based on matrix factorisation, and models based on neural networks. With one exception,



(a) TransE [5]. $\boldsymbol{r}$ acts as translation from $\boldsymbol{h}$ to $\boldsymbol{t}$.



(b) TransR [13]. $\boldsymbol{h}$ and $\boldsymbol{t}$ are transformed into a relation-specific space using $\boldsymbol{M}_r$ before $\boldsymbol{r}$ acts as translations between them.



(c) RotatE [29]. $\boldsymbol{r}$ acts as a rotational transformation from $\boldsymbol{h}$ to $\boldsymbol{t}$ in the complex plane.

Figure 2: Graphical representation of the constraint in different distance-based LFMs.

all of the LFMs mentioned here are based on the same overall principle. They define a score function $f(h,r,t)$ in a way that assigns relatively high values to true triples and relatively low values to false triples. Using gradient descent, or a similar optimisation algorithm, the entity and relation embeddings are tuned such that the score function satisfies this requirement as good as possible.

Remark. In the following section we refer to latent embeddings of entities and relations by using bold letters. For instance, the entity e may be associated with the embedding $\boldsymbol{e}\in\mathbb{R}^d$ where d is the dimensionality of the embedding space.

Distance-Based Models A popular approach to learning latent representations of KGs is to define a constraint on the embeddings based on distances in their space. One of the first LP models based on this principle is TransE [5]. The model associates each entity $e\in\mathcal{E}$ and relation $r\in\mathcal{R}$ with the vectors $\boldsymbol{e}\in\mathbb{R}^d$ and $\boldsymbol{r}\in\mathbb{R}^d$, respectively. TransE then defines the following constraint. For each triple $(h,r,t)\in\mathcal{K}$, the entity and relation embeddings should approximate $\boldsymbol{h}+\boldsymbol{r}=\boldsymbol{t}$. The intuition here is that $\boldsymbol{r}$ acts as a translation from $\boldsymbol{h}$ to $\boldsymbol{t}$ in the embedding space. This is illustrated in Figure 2a where the embedding space is $\mathbb{R}^2$. This constraint is used in the score function $f(h,r,t)=\|\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}\|$.

One downside of TransE is it inability to properly learn one-to-many and many-to-one relations [37]. Consider, for instance triples on the form $(h,$ HAS GENRE, HISTORICAL FICTION$)$. One can easily imagine a KG with multiple $h\in\mathcal{E}$ for which this triple is true. However, since TransE aims for $\boldsymbol{h}+\boldsymbol{r}=\boldsymbol{t}$, the model will learn similar embeddings for all those entities. This is seldom ideal since they likely represent quite different things. This issue is addressed by several models that extend the basic idea of TransE. These extensions typically go in one of two directions: they improve the constraint or increase the number of parameters. An example of a model that works on the constraint is TransR [13]. In this model, entities and relations live in different spaces, $\mathbb{R}^d$ and $\mathbb{R}^{d'}$, and each relation $r\in\mathcal{R}$ is furthermore associated with a transformation matrix $\boldsymbol{M}_r\in\mathbb{R}^{d'\times d}$. This matrix is used to transform the entity embeddings $\boldsymbol{h}$ and $\boldsymbol{t}$ into a relation-specific space before making them subject to the constraint from TransE. This is illustrated in Figure 2b. The score function in TransR is thus defined as $f(h,r,t)=\|\boldsymbol{hM}_r+\boldsymbol{r}-\boldsymbol{tM}_r\|$. By following this approach, an entity can have different representations in each relation-specific space. This allows TransR to overcome the issue with one-to-many and many-to-one relations that TransE has.

Another model that builds on top of TransE is RotatE [29]. In RotatE, entities and relations are embedded in the same complex space $\mathbb{C}^d$ under the constraint $\boldsymbol{h}\circ\boldsymbol{r}=\boldsymbol{t}$ where $\circ$ is the element-wise product. The model uses the fact that such product of two complex vectors can be interpreted as a rotation in the complex plane. As such, the translational interpretation that relations have in TransE is altered to a rotational interpretation in RotatE as shown in Figure 2c. The score function of the model is $f(h,r,t)=\|\boldsymbol{h}\circ\boldsymbol{r}-\boldsymbol{t}\|$. This approach comes with a benefit. RotatE is able to capture some specific types of relations that TransE and TransR cannot. For instance, a relation r is said to be symmetric if both (h,r,t) and (t,r,h) are true triples. If TransE or TransR are to learn from such triples, they will end up with $\boldsymbol{h}=\boldsymbol{t}$ and $\boldsymbol{r}=\boldsymbol{0}$ which is probably not meaningful in many cases. RotatE, on the other hand, can capture symmetric relations by having $\boldsymbol{r}$ represent a 180° rotation from $\boldsymbol{h}$ to $\boldsymbol{t}$ and $\boldsymbol{t}$ to $\boldsymbol{h}$.

Matrix Factorisation Models Another group of LFMs uses matrix factorisation to decompose a three-dimensional matrix representation of a KG into smaller matrices or vectors. This representation is denoted with $\mathcal{X}\in\{0,1\}^{|\mathcal{E}|\times|\mathcal{E}|\times|\mathcal{R}|}$ where each entry has the value

$$\mathcal{X}_{h,t,r}=\begin{cases}1 & \text{if } (h,r,t)\in\mathcal{K}\\ 0 & \text{otherwise.}\end{cases}\tag{1}$$

A prime example of this type of model is RESCAL [23] where all entities are embedded as a single matrix $\boldsymbol{E}\in\mathbb{R}^{|\mathcal{E}|\times d}$ while each relation $r\in\mathcal{R}$ is embedded as its own matrix $\boldsymbol{R}_r\in\mathbb{R}^{d\times d}$. The objective of RESCAL is to learn the embeddings such that $\boldsymbol{ER}_r\boldsymbol{E}^\mathsf{T}\approx\mathcal{X}_r$ for all $r\in\mathcal{R}$. Here, $\mathcal{X}_r\in\{0,1\}^{|\mathcal{E}|\times|\mathcal{E}|}$ is the slice of $\mathcal{X}$ that represents the triples containing r. As opposed to the other LFMs described here, RESCAL does not learn its embeddings by optimising a score function with gradient descent. Instead, the model uses the ASALSAN matrix factorisation algorithm [1].

One of the main issues with RESCAL is that it has quadratic space complexity since each relation embedding requires d^2 values. This issue is addressed by the ComplEx model [33] which learns entity and relation embeddings in the complex space $\mathbb{C}^d$. The model restricts each relation matrix $\boldsymbol{R}_r \in \mathbb{C}^{d \times d}$ to be a diagonal matrix. In practice, this means that the model only needs to store a vector $\boldsymbol{r} \in \mathbb{C}^d$ for each relation. As such, ComplEx has the same linear space complexity as, for instance, TransE.

TuckER [2] is another example of an extension to RESCAL. The model is based on Tucker decomposition [34] which factorises a three-dimensional matrix into a smaller three-dimensional core matrix and three two-dimensional matrices. In TuckER, the core matrix is denoted $\mathcal{X}' \in \mathbb{R}^{d \times d' \times d}$, entities are embedded as $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{E}| \times d}$, and relations as $\boldsymbol{R} \in \mathbb{R}^{|\mathcal{R}| \times d'}$. The model learns these matrices such that $\mathcal{X}' \times_1 \boldsymbol{E} \times_2 \boldsymbol{R} \times_3 \boldsymbol{E} \approx \mathcal{X}$ where $\times_i$ is the tensor product along the ith mode. By using the core matrix, TuckER introduces an element of parameter sharing to the model. This turns out to beneficial since TuckER is the best performing LFM among the models described until now. The following models extend the idea of sharing parameters by using neural networks.

Neural Network Models LFMs based on neural networks aim at increasing model expressiveness without increasing the size of the embedding spaces. One example of this is the ConvE model [9] that uses a convolutional neural network (CNN) to accomplish this. Given a triple (h, r, t), ConvE first combines the embeddings $\boldsymbol{h}$ and $\boldsymbol{r}$ into a single two-dimensional matrix on which one or more convolutional filters are applied. The result is projected back into a vector in the the embedding space. The dot product of this transformed vector and $\boldsymbol{t}$ is then used as the output of the score function.

Even though ConvE shares parameters between the embeddings by using a convolutional layer, it, and other LFMs, still only learn from KG triples. Thus, possibly valuable relationships between entities that span more than one triple in the graph are ignored. This issue is addressed by the SEAL model [39] which is based on a graph neural network (GNN). In this type of neural network, the embedding of an entity is based on an aggregation of the embeddings of its neighbours. SEAL introduces the concept of an enclosing subgraph which is a subset of a graph containing overlapping neighbourhoods of two entities. The model learns which enclosing subgraphs are typical for entities that are connected with a given relation. SEAL is further extended in [21, 31] where elements like attention is added to the GNN. These models do show improved performance when compare to other LFMs but are also slower to train since the neural network is more intricate. Furthermore, the models are not interpretable and cannot provide explanations for their predictions since the latent features do not translate to observable patterns in the KG.

3.2 Logic Rule Models

The logic rule models (LRMs) described in this section derive probabilistic first-order logic rules from the KG. Besides being able to predict missing knowledge, the rules can be used to explain the predictions. Much of the notation and terminology regarding LRMs is borrowed from the logic programming field. This means that a triple $(h, r, t) \in \mathcal{K}$ is expressed as a logical fact $r(h, t)$. By chaining these facts, we can express logic rules that reflect patterns in the KG. For instance, we can imagine a larger version of the KG in Figure 1 where the following rule holds,

$$\textsc{wrote}(x, y) \land \textsc{has genre}(y, z) \implies \textsc{writes}(x, z).$$

This states that if an author x has written a book y of genre z, then the author x writes books in the genre z. This rule is, of course, universally applicable but it is also trivial. When we want to infer new knowledge in a KG, we may not only be interested in such trivialities. Instead, we want to discover more intricate relationships that might be less general. The LRMs described below capture this dynamic by associating each rule with a probability. In this way, the model can reason about rules that holds in some cases.

Inductive Logic Programming By looking at a KG as a logic database of facts, the LP problem can be formulated as an inductive logic programming task. That is, given at set of facts, we want to induce a logic program in which all the facts hold. However, as mentioned before, logic rules that always hold are either rare or trivial. To overcome this, [27] defines a probabilistic version of the logic programming language Prolog called ProbLog. In ProbLog, each possible program is associated with a probability of succeeding, i.e. that the program entails all the facts in the KG. The authors present an algorithm that approximates this probability for any ProbLog program and then uses the most probable one to infer missing knowledge. However, ProbLog does not scale well and so deciding on the most probable program quickly becomes an infeasible task as the KG grows.

TensorLog TensorLog [7] is a differentiable logic where first-order logical operators on variables are expressed as operations on matrices. In [38], the authors introduce an LP model called Neural Logic Programming (Neural LP) based on TensorLog. Here, each entity $e \in \mathcal{E}$ is associated with a unique one-hot vector $\boldsymbol{e} \in \{0, 1\}^{|\mathcal{E}|}$, and each relation $r \in \mathcal{R}$ with a matrix $\boldsymbol{R}_r \in \{0, 1\}^{|\mathcal{E}| \times |\mathcal{E}|}$. These relation matrices are defined in a way similar to Equation (1), namely that the entry (h, t) in $\boldsymbol{R}_r$ is 1 if $(h, r, t) \in \mathcal{K}$ and 0 otherwise. In Neural LP, these vectors and matrices are used to represent logic clauses. For instance, we know that the clause $r_1(h, e_1) \land r_2(e_1, t)$ holds for some $h, t \in \mathcal{E}$ if the product $\boldsymbol{h}^\intercal \boldsymbol{R}_{r_1} \boldsymbol{R}_{r_2} \boldsymbol{t}$ is non-zero. Further, if we know that $(h, r_3, t) \in \mathcal{K}$, then Neural LP assigns a high confidence α to the rule $r_1(h, e_1) \land r_2(e_2, t) \implies r_3(h, t)$. This is done by maximising $\boldsymbol{h}^\intercal(\alpha(\boldsymbol{R}_{r_1} \boldsymbol{R}_{r_2}))\boldsymbol{t}$ for all $h, t \in \mathcal{E}$ where $(h, r_3, t) \in \mathcal{K}$.

Similar to RESCAL, Neural LP has quadratic space complexity. Furthermore, its number of parameters depends on the size of $\mathcal{E}$ which makes it even less efficient than RESCAL. This is the main downside of Neural LP.

Recurrent Neural Networks Another group of LRMs is based on RNNs. This is, for instance, the case in [22] where

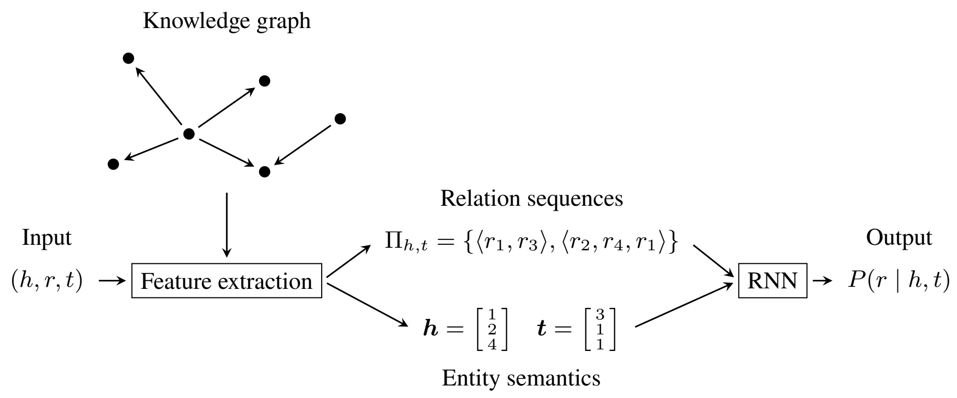

Knowledge graph
Input
(h, r, t)
Feature extraction
Relation sequences
$\Pi_{h,t} = \{\langle r_1, r_3\rangle, \langle r_2, r_4, r_1\rangle\}$
$\boldsymbol{h} = \begin{bmatrix}1\\2\\4\end{bmatrix}$ $\boldsymbol{t} = \begin{bmatrix}3\\1\\1\end{bmatrix}$
Entity semantics
RNN
Output
$P(r \mid h, t)$

Figure 3: Overview of the architecture of ResqLP.

the model learns relation embeddings based on paths between entity pairs. This allows the model to measure the similarity between a single relation and a sequence of relations. By observing which sequences of relations connect a pair of entities, the model can predict that they should connected by the most similar relation. This is analogous to a probabilistic first-order logic rule in that a sequence of relations implies a certain relation. Another advantage of this model is that since it only learns from relation sequences it is independent the entities found in the KG. The model is extended by [8] who, among other improvements, include entity embeddings as part of the hidden RNN states. Even though this improves the performance of the model, its ability to cope with unseen entities is lost.

These RNN models suffer from scalability issues since the number of paths between two entities increase exponentially with the size of the KG. However, we do find the nature of the model, i.e. the combination of the effectiveness of RNNs and interpretability of first-order logic rules, compelling. Therefore, we introduce an extension to this type of model called ResqLP. This model aims at coping with the scalability issues by taking inspiration from the enclosing subgraph concept introduced in [39]. Furthermore, ResqLP includes entity semantics from observable features in the KG. This allows the model to learn from both entity semantics and relation sequences, as in [8], while remaining independent of the specific entities, as in [22].

4 Overview of ResqLP

This section outlines the architecture of ResqLP which consists of a feature extraction phase and an RNN. Both build upon existing research on GNN and RNN-based LP models [8, 22, 31, 39].

Figure 3 illustrates the overall structure of ResqLP. As with any LP model, the input is a candidate triple (h, r, t) for which we are to output $P(r \mid h, t)$. Based on the entities of the candidate triple, we extract two types of features. The first is the set of sequences of relations that appear along the paths between h and t in the KG. The second is semantic feature vectors of h and t. Both of these features are directly observable in the KG. They are combined and encoded using an RNN that outputs our desired probability $P(r \mid h, t)$.

In broad terms, the RNN compares the similarity between r and sequences of relations that connect h and t. If we know that a relation sequence $\langle r_1, r_2, \ldots, r_n\rangle$ is similar to a relation r, we can formulate it as a first-order logic rule,

$$r_1(h, e_1) \wedge r_2(e_1, e_2) \wedge \cdots \wedge r_n(e_{n-1}, t) \implies r(h, t),$$

where $r_i(e, e')$ is satisfied if $(e, r_i, e') \in \mathcal{K}$. This logic rule is then associated with the probability $P(r \mid h, t)$. Such rules can be used to provide interpretable explanations of the predictions of ResqLP.

In Sections 5 and 6, we go in depth with the feature extraction phase and the RNN, respectively. To support our decisions in those sections, we look at properties of real KGs. So before we elaborate on details of ResqLP, we introduce said KGs.

4.1 Use Case Knowledge Graphs

Throughout the paper, we analyse different aspects of real KGs with the goal of identifying key features that may guide the design of ResqLP and tell us when it may, or may not, be applicable. The KGs that we have selected are FB15K237 [32], WN18RR [9], YAGO3-10 [9], and OpenBioLink [6]. They differ in terms of size and domain as well as how often they are used as benchmark datasets in the LP literature. FB15K237 and WN18RR are common benchmark KGs that allow use to compare the performance of ResqLP to other models. On the other hand, YAGO3-10 and OpenBioLink let us examine how ResqLP handles larger KGs that may pose more realistic prediction scenarios.

Table 1 shows the number of entities, relations, and triples in each KG.

Table 1: Basic statistics of the four use case KGs.

Knowledge graph	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{K}\rvert$
WN18RR [9]	41 105	11	93 003
FB15K237 [32]	14 541	237	310 116
YAGO3-10 [9]	123 182	37	1 089 040
OpenBioLink [6]	184 635	28	4 563 405

FB15K237 Freebase [3] is a KG containing general-domain data aggregated from multiple sources. Even though Freebase is a discontinued product, it continues to be widely used as a benchmark dataset for LP. Specifically, two subsets of the KG are used for this. FB15K [5] contains the 15 000 most popular entities in Freebase. This dataset is, however, often deemed 'unrealistic' since it contains many trivial prediction tasks. Most notably, many correct predictions can be made by simply reversing triples in the training data. In a revision of FB15K, commonly called FB15K237 [32], this issue is addressed. Relations that often appear in reversible triples are removed, reducing the total number of relations from over 1000 to 237.

WN18RR WordNet [18] is a lexical KG over the English language. It describes hierarchical linguistic structures and semantic relationships with relations like HYPONYM, HYPERNYM, and SYNONYM. WN18 [4] is a subset of WordNet constructed by selecting 18 relations and filtering out entities that do not appear in more than 15 triples. Even though frequently used in the LP literature, WN18 suffers from the same issues as FB15K. This is addressed with the refined subset WN18RR [9] that contains only 11 relations. Compared to Freebase, WordNet is a domain-specific KG.

YAGO3-10 YAGO (yet another great ontology) is a general-domain KG based on data extracted from Wikipedia. The KG has been refined through four versions and the third edition [16] has been used as a benchmark dataset for LP in previous research. Specifically, the subset YAGO3-10 [9] contains all the entities of YAGO3 that are related by ten or more relations. The general-domain nature of YAGO makes its content comparable to Freebase. However, YAGO10-3 is significantly larger than FB15K237 with over a million entities. On the contrary, YAGO10-3 has only 37 relations compared to 237 in FB15K237.

OpenBioLink LP has practical uses in bioinformatics for tasks like drug re-purposing [10] and predicting interactions between chemical substances [20]. Thus, the OpenBioLink [6] KG aims to be an LP benchmark dataset that mimics a real-world prediction scenario. On the other hand, FB15K237, WN18RR, and YAGO3-10 are, to a large extent, merely benchmark datasets. OpenBioLink is an aggregation of structured bioinformatics data from multiple public sources. Furthermore, OpenBioLink is significantly larger than the other KGs with more than four million triples. However, the ratio between the size and the number of relations is even smaller than in YAGO3-10.

5 Feature Extraction

The feature extraction phase of ResqLP is concerned with two types of features: relation sequences and entity semantics. The objective of the former is to capture how entities are related across multiple triples, while the latter captures what the individual entities actually represent. Both types of features are directly observable in the KG. In the following sections, we elaborate on how they are extracted.

5.1 Relation Sequences

In a KG, two entities h and t are likely connected through one or more paths. However, since paths in a graph, such as $\langle e_1, r_2, e_2, r_2, e_3, \ldots, e_{n-1}, r_{n-1}, e_1\rangle$, are unique by definition, they will not make useful features for the model. Instead, we consider only the sequences of relations in these paths.

Definition 3 (Relation sequence)**.** Let $\langle e_1, r_1, e_2, r_2, e_3, \ldots, e_{n-1}, r_{n-1}, e_n\rangle$ be path in a KG. Its corresponding relation sequence is $\langle r_1, r_2, \ldots, r_{n-1}\rangle$.

We use $\Pi_{h,t}$ to denote the set of relation sequences that connect h and t. An element in this set, i.e. a relation sequence, is denoted with π.

As mentioned in Section 4, the assumption for using relation sequences as features is that certain relation sequences can be predictive of certain relations. As an example, consider again the KG over fiction literature in Figure 1. One can easily imagine that the relation sequence $\langle$WROTE, HAS GENRE$\rangle$ between M. MITCHELL and HISTORICAL FICTION frequently occurs between other similar entity pairs. Since the two entities are already connected with the relation WRITES, ResqLP will learn that the sequence $\langle$WROTE, HAS GENRE$\rangle$ is predictive of that relation. Then, when the model sees the candidate triple (L. TOLSTOY, WRITES, HISTORICAL FICTION) at inference time, it finds the same relation sequence $\langle$WROTE, HAS GENRE$\rangle$ between the entities and predicts that the triple is true.

At first, the task of finding the relations sequences seems straight forward. We just traverse all paths between h and t and ignore the concrete entities along the way. It should be noted that we only consider simple paths, i.e. paths with no cycles, to avoid possibly infinite relation sequences. The common approach to finding a simple path between two nodes in a graph is to use a depth-first search which has the time complexity $\mathcal{O}(|\mathcal{E}| + |\mathcal{K}|)$. We are, however, interested in finding *all* paths. In the worst case, where the KG forms as complete graph, the are $\mathcal{O}((|\mathcal{E}||\mathcal{R}|)!)$ possible paths to enumerate. This can make the task of finding the relation sequences infeasible in practice even though the worst case scenario is far from reality in the four use case KGs. In the following section, we propose an approach to cope with these large search spaces.

5.1.1 Enclosing Subgraphs

One way to reduce the search space when enumerating relations sequences is to simply search a smaller graph. To do this, we build upon the concept of enclosing subgraphs introduced in [39]. An enclosing subgraph is a subset of a KG that encloses two specific entities. The definition in [39] is based on undirected graphs and, thus, not directly applicable to KGs. Therefore, we first propose a different definition that is suitable for our case.

The enclosing subgraph with respect to h and t is based on the neighbourhood of those entities. We say that the direct neighbourhood of an entity e is the set of triples in which e occurs. By recursively finding these direct neighbourhoods, we can find neighbourhoods of any depth.

Definition 4 (Entity neighbourhood)**.** Let $\mathcal{K}$ be a KG and $e \in \mathcal{E}$. The depth-k neighbourhood of e is a KG over the set of entities $\mathcal{E}_e^k$ and the set of relations $\mathcal{R}_e^k$ consisting of the triples $\mathcal{K}_e^k = \bigcup_{e' \in \mathcal{E}_e^{k-1}} \mathcal{K}_{e'}^1$. For $k = 1$, we have $\mathcal{K}_e^1 = \{(\cdot, \cdot, e), (e, \cdot, \cdot) \in \mathcal{K}\}$.

Given this definition, we can define the enclosing subgraph which combines two entity neighbourhoods of a certain depth.

Definition 5 (Enclosing subgraph)**.** Let $\mathcal{K}$ be a KG and $h, t \in \mathcal{E}$. The depth-k enclosing subgraph of $\mathcal{K}$ with respect to h and t is a KG containing the intersection of their depth-k neighbourhoods, $\mathcal{K}_{h,t}^k = \mathcal{K}_h^k \cap \mathcal{K}_t^k$.

Figure 4 shows an example of a depth-2 enclosing subgraph with respect to the entities h and t. Entities and relations outside the subgraph are coloured grey.

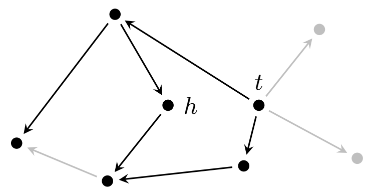


Figure 4: Depth-2 enclosing subgraph with respect to h and t. Entities and relations outside the subgraph are coloured grey.

When we want to enumerate the relation sequences $\Pi_{h,t}$, we search only $\mathcal{K}_{h,t}^k$, for some k, instead of $\mathcal{K}$ in its entirety. The assumption here is that entities and relations that are close to h and t are more relevant than those far away.

5.1.2 Effects of Using Enclosing Subgraphs

We now look at how enclosing subgraphs actually affect the relation sequence enumeration process. First, we examine the size of enclosing subgraphs in the different use case KGs. Figure 5 shows the proportion of all triples that an average depth-3 enclosing subgraph covers in each of the KGs. The data points behind the figure are made by uniformly sampling 100 entity pairs in each KG and extracting their enclosing subgraph. Clearly, the effect is most noticeable in WN18RR and YAGO3-10 where the average subgraph covers 0.36 % and 14.53 % of the entire KG, respectively. This means that relation sequence enumeration is significantly faster when using enclosing subgraphs. For WN18RR and YAGO3-10, we see a speedup factor in the order of 100 and 1000, respectively, when using enclosing subgraphs. Not only is this significant in itself but the absolute time it takes to enumerate the sequences is also reasonable for practical use. In WN18RR, the execution time is in order of 0.0001 s and in YAGO3-10 it is in the order of 0.01 s.

In FB15K237 and OpenBioLink, the situation is different. The average depth-k enclosing subgraph grows much quicker

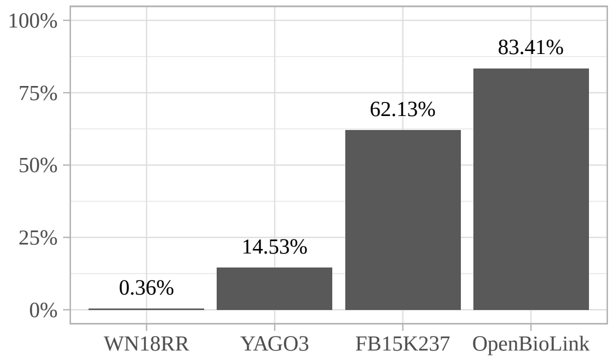


Figure 5: Proportion of the entire KG covered by an average depth-3 enclosing subgraph.

and, as Figure 5 shows, even $k = 3$ results in subgraphs that span more than half of the entire KG. By looking at Table 1, we can conclude that enclosing subgraphs in FB15K237 and OpenBioLink have sizes in the order of hundred thousands and millions, respectively. This is still too large to make relation sequence enumeration tractable. Actually, since the subgraph extraction procedure in itself takes time, it may even slow the process down.

5.1.3 Neighbourhood Sampling

To cope with the large enclosing subgraphs that we see in FB15K237 and OpenBioLink, we introduce an element of randomness to the subgraph extraction process. We use the term 'neighbourhood sampling' for this.

As mentioned in Definition 4, a depth-k neighbourhood is just a union of depth-1 neighbourhoods. When doing neighbourhood sampling, we uniformly sample a fraction s of the triples in each depth-1 neighbourhood. Seen from a graph perspective, this is equivalent of only following s of the outgoing edges from each node. Figure 6 shows how neighbourhood sampling with $s = 0.1$ affects the enclosing subgraph size in FB15K237 and OpenBioLink. This approach makes relation sequence enumeration faster in both KGs where we see a speedup factor in the order of 10. For FB15K237, this results in an execution time around 0.01 s while the it is around 1 s for OpenBioLink. There is, however, a significant caveat with this approach: Since we sample the triples uniformly, we cannot guarantee the usefulness of the relation sequences found in the subgraphs. We may very likely discard triples that are part of important paths between entities. This issue is discussed further in Section 8.

5.2 Entity Semantics

Until now, focus has been on relation sequences and their potential ability to be predictive of relations. However, we know from [8] that integrating entities in the RNN can be beneficial for the performance of the model. In [8], each entity $e \in \mathcal{E}$ is associated with a corresponding vector embedding $\boldsymbol{e} \in \mathbb{R}^d$. These embeddings are parameters in the RNN and thus learnt during training. This is quite similar to

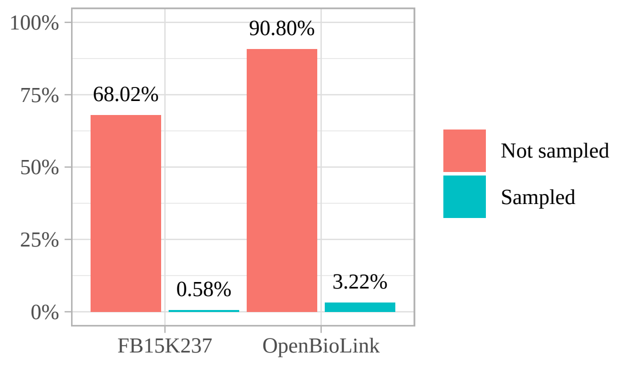


Figure 6: Proportion of entire KG covered by an average depth-3 enclosing subgraph with and without neighbourhood sampling.

the concept of LFMs like TransE and RotatE. The downside of this approach is that the model loses its independence of the entities in the KG. If we see an entity $e \notin \mathcal{E}$ on inference time, there is no corresponding embedding $\boldsymbol{e}$ to use and the model cannot make predictions. Since we consider this independence important, we want to preserve it. So instead of learning a latent vector representation of entities, we use features that are directly observable in the KG.

We present two different approaches to capturing entity semantics: a general, called relation frequency semantics, and a KG-specific. In both cases, the objective is to assign each entity $e \in \mathcal{E}$ a suitable feature vector $\boldsymbol{e}$.

5.2.1 Relation Frequency Semantics

The purpose of this first type of entity semantics is for it to be applicable in any KG. We define the relation frequency semantics of an entity $e \in \mathcal{E}$ based on the relations found in its direct neighbourhood $\mathcal{K}^1_e$. Specifically, we associate each entity $e \in \mathcal{E}$ with a vector $\boldsymbol{e} \in \mathbb{N}^{|\mathcal{R}|}$ where the ith entry corresponds to the frequency of ith relation in the neighbourhood of e,

$$\boldsymbol{e}_i = |\{(\cdot, r_i, \cdot) \in \mathcal{K}^1_e\}| \tag{2}$$

As an example, consider Figure 7 that shows a segment of a KG over $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$. If we want to find $\boldsymbol{e}$, we simply count the frequency of each relation in the direct neighbourhood of e. For instance, r_2 appears twice which means that $\boldsymbol{e}_2 = 2$, and the full feature vector looks like the following,

$$\boldsymbol{e} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 0 \end{bmatrix}.$$

The assumption behind using relation frequently semantics is that part of the semantics of an entity is defined by the relations it occurs in. For instance, continuing with our fiction literature KG, books probably participate more frequently in $(\cdot,$ HAS GENRE$, \cdot)$ triples than authors do. Similarly, genres will probably not appear in $(\cdot,$ WROTE$, \cdot)$ triples; instead, only authors and books will be present there. By

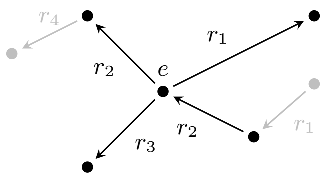


Figure 7: KG segment for determining the semantic vector of the entity e. Entities and relations outside $\mathcal{K}^1_e$ are coloured grey.

this logic, we assume that the relation frequency semantics is a good fit for KGs that describe many different types of entities, such as FB15K237 and YAGO3-10. If we look at some concrete entities, we can see that this holds in some cases by comparing the feature vectors using cosine similarity. For instance, the entities THOMAS JEFFERSON and HONG KONG FILM AWARD FOR BEST FILM in FB15K237 share only one non-zero entry in their feature vectors. This means that their cosine similarity is relatively small, 0.015. However, if we compare HONG KONG FILM AWARD FOR BEST FILM with another film award, GENIE AWARD FOR BEST ACHIEVEMENT IN EDITING we get a cosine similarity of 0.85.

An example of where relation frequency semantics may not be appropriate is in WN18RR. In this KG, there are relatively few relations, $|\mathcal{R}| = 11$, which means that the feature vectors vary less. In turn, this means that entities that indeed are very different have high cosine similarities. For instance, the entities TRIGGER (as a noun) and HEAT (as a verb) have a cosine similarity of 0.99.

5.2.2 Domain-Specific Semantics

An alternative to relation frequency semantics is to analyse what properties characterise the entities in the individual KGs. However, as opposed to [8], we want the features to be observable in the KG. To do this, we use a manual approach where we identify individual features that may capture the semantics of the entities. For instance, a book entity in the KG in Figure 1 may be described by its genre and publication year which can be extracted from triples on the form $(\cdot,$ HAS GENRE$, \cdot)$ and $(\cdot,$ PUBLISHED IN$, \cdot)$. By selecting a number of these features, we can construct feature vectors for each entity that can be used in ResqLP. Like relation frequency semantics, this approach preserves the model's ability to make predictions about unseen entities since the features are observable in the KG. In the following, we propose a definition of domain-specific entity semantics for the YAGO3-10.

As mentioned earlier, the YAGO3-10 KG describes a more general domain than, for instance, WN18RR and OpenBioLink. This means that some entities are more likely to participate in certain relations than others. We can use this information to identify some appropriate features. For instance, even though the entities in YAGO3-10 are not directly associated with a type, we can infer the type of many entities based on a number of different relations. If an entity participates as tail in triples on the form $(\cdot,$ HAS OFFICIAL LANGUAGE$, \cdot)$ we can assume that it is of type

$$\begin{tikzcd}
\boldsymbol{y_1} \arrow[r] & {[\boldsymbol{y_1};\boldsymbol{r_2}]} \arrow[r, "\sigma(\boldsymbol{A}\cdot)"] & \boldsymbol{y_2} \arrow[r] & {[\boldsymbol{y_2};\boldsymbol{r_3}]} \arrow[r, "\sigma(\boldsymbol{A}\cdot)"] & \boldsymbol{y_3} \arrow[r] & \cdots \arrow[r] & {[\boldsymbol{y_{n-1}};\boldsymbol{r_n}]} \arrow[r, "\sigma(\boldsymbol{A}\cdot)"] & \boldsymbol{y_n} \arrow[r] & {[\boldsymbol{h};\boldsymbol{y_n};\boldsymbol{t}]} \arrow[r, "\sigma(\boldsymbol{B}\cdot)"] & \boldsymbol{\pi}_{h,t} \\
\boldsymbol{r_1} \arrow[u] & \boldsymbol{r_2} \arrow[u] & & \boldsymbol{r_3} \arrow[u] & & & \boldsymbol{r_n} \arrow[u] & & \boldsymbol{h},\boldsymbol{t} \arrow[u] &
\end{tikzcd}$$

Figure 8: Overview of how the RNN in ResqLP processes relation embeddings $\boldsymbol{r_i} \in \mathbb{R}^d$ and entity features $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^{d'}$ when encoding the relation sequence $\pi = \langle r_1, r_2, r_3, \ldots, r_n \rangle$ between the entities h and t.

LANGUAGE. Similarly, we can assign PERSON as type to head entities in $(\cdot, \text{HAS GENDER}, \cdot)$ triples. The latter example also leads to another potential feature: the gender of a person. By manually examining the entities and relations in the KG, we propose a set of features that can be used as entity semantics in YAGO3-10. These features are TYPE, GENDER, OCCUPATION, LANGUAGE, and NATIONALITY. A description of how we derive the features for the individual entities can be found in Appendix A. Naturally, these features are all categorical and so to represent them in a numerical way, we use a one-hot encoding. This gives us entity semantic vectors $\boldsymbol{e} \in \{0,1\}^{165}$. In Section 7 we evaluate how ResqLP performs using domain-specific compare to relation frequency semantics.

6 Encoding Relation Sequences

Until now, we have explained how to extract the set of relation sequences $\Pi_{h,t}$ and entity semantics $\boldsymbol{h}$ and $\boldsymbol{t}$ from a KG given a candidate triple (h, r, t). In the following sections, we define the actual RNN that processes these features, how we train it, and how it can report explanations for its predictions. The RNN is an extension to the ideas presented in [8, 22]. Specifically, we present a new way to combine relation sequences with entity semantics.

6.1 Recurrent Neural Network

Since relation sequences, naturally, are discrete, we must first represent $\Pi_{h,t}$ in a continuous manner that fits the nature of neural networks. We do this by associating each relation $r \in \mathcal{R}$ with a vector embedding $\boldsymbol{r} \in \mathbb{R}^d$ where d is the dimensionality of the embedding space. This allows us to represent a relation sequence $\pi = \langle r_1, r_2, \ldots, r_n \rangle$ as a sequence of relation embeddings $\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_n}$. Given this sequence of length n, the RNN has n hidden states where each state $\boldsymbol{y_t} \in \mathbb{R}^d$ is computed by combining the previous state $\boldsymbol{y_{t-1}}$ and the relation $\boldsymbol{r_t}$. Formally, this recursive case is formulated as

$$\boldsymbol{y_t} = \sigma(\boldsymbol{A}[\boldsymbol{y_{t-1}}; \boldsymbol{r_t}]). \tag{3}$$

Here, $[\cdot;\cdot] \in \mathbb{R}^{2d}$ denotes the stacking of two vectors in $\mathbb{R}^d$, $\boldsymbol{A} \in \mathbb{R}^{2d \times d}$ is a transformation matrix that keeps the embeddings in $\mathbb{R}^d$ space, and $\sigma(\cdot)$ is the sigmoid function. The base case of Equation (3) is

$$\boldsymbol{y_1} = \boldsymbol{r_1}. \tag{4}$$

The final hidden state of the RNN is considered as the embedding of π, i.e. $\boldsymbol{\pi} = \boldsymbol{y_n}$. Figure 8 illustrates this process of combining hidden states and relation embeddings according to Equations (3) and (4).

The next step is to combine $\boldsymbol{\pi}$ with the semantic vectors of the two entities, h and t. Similar to Equation (3), these three vectors are first stacked and then transformed back into $\mathbb{R}^d$ space,

$$\boldsymbol{\pi}_{h,t} = \sigma(\boldsymbol{B}[\boldsymbol{h}; \boldsymbol{\pi}; \boldsymbol{t}]), \tag{5}$$

where $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^{d'}$, $\boldsymbol{\pi} \in \mathbb{R}^d$, and $\boldsymbol{B} \in \mathbb{R}^{2d'+d \times d}$. This final step in the RNN is show to the right in Figure 8. Worth noting here is that d' is the number of dimensions in the entity feature vectors. When using relation frequency semantics, we have $d' = |\mathcal{R}|$ while different domain-specific semantic give rise to different d'. For instance, for the YAGO3-10 semantics introduced in Section 5.2.2, we have $d' = 165$.

We now compute the similarity between the relation r from the candidate triple and $\boldsymbol{\pi}_{h,t}$. We do this by first passing the relation embedding $\boldsymbol{r}$ through the same stacking transformation as $\boldsymbol{\pi}$ in Equation (5). Then the dot product of $\boldsymbol{r}_{h,t}$ and $\boldsymbol{\pi}_{h,t}$ is used as the similarity score,

$$\operatorname{score}(r, \pi, h, t) = \boldsymbol{r}_{h,t} \cdot \boldsymbol{\pi}_{h,t}. \tag{6}$$

Next, we use this similarity to suggest the probability $P(r \mid h, t)$. First, we notice that in most cases, $\Pi_{h,t}$ will contain more than one relation sequence. Thus, we compute Equation (6) for each $\pi \in \Pi_{h,t}$ and aggregate the similarities to a single value that represents said probability. Generally, we define this probability as

$$P(r \mid h, t) = g(\{\operatorname{score}(r, \pi, h, t) \mid \pi \in \Pi_{h,t}\}) \tag{7}$$

where $g(\cdot)$ is an aggregation function. In [22], the authors define $g(\cdot)$ to simply output the highest similarity score and passing it through the sigmoid function $\sigma(\cdot)$,

$$g_{\max}(S) = \sigma\left(\max_{s \in S} s\right). \tag{8}$$

However, as noted in [8], this approach discards other scores that may potentially be useful. Therefore, they propose three different aggregation functions: one that averages all the similarities, one that averages the top-k similarities, and one based on LogSumExp function. The latter is defined as

$$g_{\mathrm{LSE}}(S) = \sigma\left(\log \sum_{s \in S} e^s\right), \tag{9}$$

and the authors find that it results in the best performance of their model. Therefore we use Equation (9) as aggregation function in ResqLP.

As mentioned earlier, the advantage of using observable features in ResqLP is that it makes interpretability easier. The similarity between a relation and a relation sequence, which we find in Equation (6), can be used to report explanations at inference time. In addition to aggregating the similarities and outputting the probability Equation (7), we can output the list of relation sequences found between h and t ranked by their similarity.

The parameters that will be adjusted while training ResqLP are the relation embeddings $\{\boldsymbol{r} \in \mathbb{R}^d \mid r \in \mathcal{R}\}$ and the two transformation matrices $\boldsymbol{A} \in \mathbb{R}^{2d \times d}$ and $\boldsymbol{B} \in \mathbb{R}^{d+2d' \times d}$. Here, d is the dimensionality of the relation embeddings and d' is the dimensionality of entity semantic vectors. Since d and d' are constants, the number of parameters in ResqLP only depends on the number of relations in the input KG. This makes the model parameter efficient when compared to common LFMs such as TransE where the number parameters grows with the number of entities.

6.2 Training With Negative Sampling

As mentioned earlier, we consider LP to be a binary classification problem: Either a candidate triple (h, r, t) is true, i.e. it belongs in the KG, or it is false. However, since the purpose of a KG is to provide true knowledge, it only contains actual true triples. Only using these triples for training will not contribute to the prediction capability of the model since it will not be exposed to examples of false triples. To cope with this, we utilise negative sampling [17] to generate false triples based on existing data. For each true triple $(h, r, t) \in \mathcal{K}$, we replace either h or t with another entity $e \in \mathcal{E}$ to generate a false triple, such as $(h, r, e) \notin \mathcal{K}$. Using this approach, we can generate a set of false triples $\mathcal{K}^-$ which, together with $\mathcal{K}$, forms the training data for ResqLP.

In the context of KGs, doing negative sampling in the way described above comes with a risk of generating false negatives. Just because a triple is not present in the KG does not mean that it is false. This issue is addressed in [37] where the authors propose that the choice of replacing either the head or tail entity in (h, r, t) should not be made uniformly at random. Instead, it should depend on the cardinality of r. They introduce the following heuristic. Consider the triple $(h, \text{PUBLISHED IN}, t)$. The relation PUBLISHED IN is many-to-one since multiple books can be published in the same year, but a specific book is only published in a specific year[1]. Due to this cardinality, it is more likely that $(h, \text{PUBLISHED IN}, e)$ is a actually false compared to $(e, \text{PUBLISHED IN}, t)$ for some $e \in \mathcal{E}$. Therefore, in this case, it is preferable to replace t.

The cardinality of a relation is determined by two statistics of the KG tails-per-head, $\mathrm{tph}(\cdot)$, and heads-per-tail, $\mathrm{hpt}(\cdot)$. Given a relation r, the tails-per-head statistic is the average number of tail entities that a head entity is related to by r; the inverse goes for heads-per-tail. We formulate these measures in the following way,

$$\mathrm{tph}(r) = \frac{\sum_{h \in \mathcal{E}} |\{t \mid (h, r, t) \in \mathcal{K}\}|}{|\{h \mid (h, r, \cdot) \in \mathcal{K}\}|} \tag{10}$$

$$\mathrm{hpt}(r) = \frac{\sum_{t \in \mathcal{E}} |\{h \mid (h, r, t) \in \mathcal{K}\}|}{|\{t \mid (\cdot, r, t) \in \mathcal{K}\}|}. \tag{11}$$

$\mathrm{tph}(\cdot)$ and $\mathrm{hpt}(\cdot)$ are used to determine the probability of replacing either the head or tail entity in (h, r, t) when generating a false triple. Specifically, we have

$$P(\text{replace } t \mid r) = \frac{\mathrm{hpt}(r)}{\mathrm{hpt}(r) + \mathrm{tph}(r)} \tag{12}$$

$$P(\text{replace } h \mid r) = \frac{\mathrm{tph}(r)}{\mathrm{hpt}(r) + \mathrm{tph}(r)}. \tag{13}$$

As mentioned above, this is just a heuristic. However, [37] finds that it improves the performance across different models and KGs. Therefore, we choose to employ this negative sampling approach when training ResqLP.

Now that we have a set of true and false training triples, we define the learning objective of the model as

$$\arg\max_{\theta} \sum_{(h,r,t) \in \mathcal{K}} P(r \mid h, t) + \sum_{(h',r',t') \in \mathcal{K}^-} 1 - P(r' \mid h', t'), \tag{14}$$

where θ denotes the parameters of the RNN, i.e. relation embeddings $\boldsymbol{r_i}$ and the transformation matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

7 Experiments

In this section, we conduct a series of experiments with ResqLP to evaluate its performance on the four use case KGs. These experiment are, first of all, concerned with comparing the performance of ResqLP to other models. However, we also look at the relation sequences that ResqLP learns as well as how different definitions of entity semantics affect its performance. In the following sections, we first describe the setup for these experiments as well as the metrics used to compare the models. Then, we present the results of the experiments and highlight our findings.

7.1 Experimental Setup

We conduct a number of different types of experiments with ResqLP. First, we do hyperparameter optimisation using a grid search over different parameter values. Second, the performance of the models with the best performing parameter configurations are compare to other LP models. Third, we evaluate the impact of using different entity semantics by training two different ResqLP models on the YAGO3-10 KG. Before describing each experiment in details, we present the evaluation metrics used and comment briefly on implementation.

Implementation and Evaluation Since much existing research views LP as an information retrieval task, performance metrics from this field is commonly used. The two

[1]This is, of course, a simplification since a book can be published in different editions in different years.

Table 2: Mean reciprocal rank (MRR) and hits-at-n (H@N) performance of ResqLP compared to other models on the four use case KGs.

Model	FB15K237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RotatE [29]	0.338	0.241	0.375	$\underline{0.533}$	$\underline{0.476}$	$\underline{0.428}$	**0.492**	**0.571**
TuckER [2]	$\underline{0.358}$	**0.266**	**0.394**	**0.544**	0.470	**0.443**	$\underline{0.482}$	$\underline{0.526}$
ConvE [9]	0.325	0.237	0.356	0.501	0.430	0.400	0.440	0.520
ResqLP	**0.523**	$\underline{0.262}$	$\underline{0.379}$	0.404	**0.605**	0.322	0.349	0.384
	YAGO3-10*				OpenBioLink†			
RESCAL [23]	—	—	—	—	—	**0.407**	—	**0.615**
ComplEx [33]	0.360	0.260	$\underline{0.400}$	$\underline{0.550}$	—	0.166	—	$\underline{0.525}$
ConvE [9]	$\underline{0.440}$	**0.350**	**0.490**	**0.620**	—	—	—	—
ResqLP	**0.570**	$\underline{0.335}$	0.336	0.345	**0.629**	$\underline{0.336}$	**0.357**	0.410

* Results from [9]. † Results from the OpenBioLink leaderboard: `https://github.com/OpenBioLink/OpenBioLink`

main metrics are mean reciprocal rank (MRR) and hits-at-n (H@N). Both are based on a list of test triples $\mathcal{S}$ ranked by an individual score in descending order. In the case of ResqLP, this score is simply $P(r \mid h, t)$ for a given triple $(h, r, t) \in \mathcal{S}$. We use $\operatorname{rank}(h, r, t)$ to denote its rank among all candidate triples. Then, MRR is defined as

$$\text{MRR} = \frac{1}{|\mathcal{S}|} \sum_{(h,r,t)\in\mathcal{S}} \frac{1}{\operatorname{rank}(h, r, t)}. \tag{15}$$

H@N is a type of precision measure on the ranked list of candidate triples. For any n, the metric is defined as

$$\text{H@N} = \frac{|\text{true triples in first } n \text{ elements of } \mathcal{S}|}{n}. \tag{16}$$

We have implemented ResqLP in Python using the deep learning library PyTorch [24] and the accompanying research-oriented framework PyTorch Lightning [25]. The implementation is available on GitHub[2].

We run each experiment in distributed mode across four Nvidia Tesla V100 GPUs each with access to four Intel Xeon Platimum CPU threads at 2.7 GHz. The KGs come pre-split into a training, validation, and test set from their sources[3]. We let ResqLP train for at most 100 epochs on the training data and evaluate it on the validation data every three epochs. We stop the training process early if the MRR score on the validation data does not improve after three validation runs. Then, we test the model on the test data.

Experimental Procedure The first experiment we run is hyperparameter optimisation. For each use case KG, we train ResqLP with different values for batch size, $b \in \{64, 128, 256\}$, size of the relation embedding space, $d \in \{10, 100, 1000\}$, and learning rate $\eta \in \{0.01, 0.001, 0.0001\}$. The parameters that control enclosing subgraph depth k and sampling size s are not part of the parameter search. Instead, we manually set them to values that we find reasonable based on the analysis of the KGs done in Section 5.1. We report the parameter configurations that lead to the best performance on each KG. These performance are then compared to other LP models. We have, however, not been able to find papers that report results on all four KGs presented in this paper. Therefore, for some KGs, we compare with different models.

The second experiment we conduct compares how different definitions of entity semantics affect the performance of ResqLP on the YAGO3-10 KG. For this, we train ResqLP using the domain-specific semantics introduced in Section 5.2.2 and compare its performance to the model trained on the same KG in the previous experiment.

7.2 Results

Table 3 shows the best parameter configurations for each KG. There are a few things worth noting here. First, WN18RR is the only KG where ResqLP prefers a higher embedding dimension $d = 1000$. This may be because WN18RR has so few relations that each relation needs more parameters to distinguish itself from the others. Second, with the exception of FB15K237, ResqLP uses larger batch sizes as the KG increases.

Table 3: The best performing parameter configurations for ResqLP on each KG.

Knowledge graph	b	k	d	η	s
FB15K237	256	3	100	0.0001	0.3
WN18RR	64	5	1000	0.0001	—
YAGO3-10	128	3	100	0.001	—
OpenBioLink	256	3	100	0.0001	0.1

b: batch size k: enclosing subgraph depth
d embedding space dimensionality
η: learning rate s: subgraph sampling size

The performance of ResqLP as well as other models is shown in Table 2 where the best and next-best scores in

[2]`https://github.com/emilbaekdahl/masters-code`.

[3]WN18RR, FB15K237, and YAGO3-10 is available here: `https://github.com/TimDettmers/ConvE`. OpenBioLink is available here: `https://github.com/OpenBioLink/OpenBioLink`.

each column are highlighted with bold and underlined font, respectively. For FB15K237 and WN18RR we compare with RotatE [29], TuckER [2], and ConvE [9]. The results for these models come from the original papers. Results on the YAGO3-10 and OpenBioLink KGs come from [9] and the OpenBioLink leaderboard[4], respectively.

From Table 2 we see that ResqLP consistently gets the best MRR across all KGs. However, for all H@N metrics, the model is at most next-best. Across all the KGs, ResqLP performs worst on WN18RR. One explanation for this is that the relation frequency semantics of the entities are too similar and that the relation sequences, in many cases, do not convey much meaning. We find the best MRR, H@1, and H@10 scores on the OpenBioLink KG, while the highest H@3 is found on FB15K237. Since these KGs give rise to the largest enclosing subgraphs, as explained in Section 5.1, we can also expect to find the most relation sequences there. Under our assumption that ResqLP actually learns for relation sequences, this is more or less as expected. However, the result on OpenBioLink is most surprising since it has relatively few relations. As such, we would expect that ResqLP would not find many meaningful relation sequences.

Table 4 shows how ResqLP performs on YAGO3-10 using relation frequency semantics and domain-specific semantics, as defined in Section 5.2.2. We see that the domain-specific semantics lead to increased results. Even though the MRR does not differ much between the two types of semantics, the larger increase in H@N indicates that ResqLP is more precise in its predictions using domain-specific semantics. This show that, even though relation frequency semantics may be a good place to start, it is worth investigating how to define good domain-specific entity semantics on KGs.

Table 4: Results of using relation frequency and domain-specific semantics on the YAGO3-10 KG.

Semantics	MRR	H@1	H@3	H@10
Relation frequency	<u>0.570</u>	<u>0.335</u>	<u>0.336</u>	<u>0.345</u>
Domain-specific	**0.573**	**0.348**	**0.351**	**0.470**

7.3 Predictive Relation Sequences

In this section, we examine the rules that ResqLP actually learns by looking at examples on the YAGO3-10 KG. We begin by looking at the most common rule that ResqLP finds,

$$\textsc{is affiliated to}(h,t) \implies \textsc{plays for}(h,t). \tag{17}$$

This rule is concerned with athletes and the sports teams they play for. The rule is more or less trivial since it is very likely that an athlete who is affiliated with a sports team also plays for it.

If we look at the most common rules that span more than one relation, we see similar trivialities. For instance, the following rules capture how airports are connected,

$$\begin{aligned}&\textsc{is connected to}(h,e_1) \wedge \textsc{is connected to}(e_1,t)\\ &\implies \textsc{is connected to}(h,t).\end{aligned} \tag{18}$$

Since only airport entities appear in triples on the form $(\cdot, \textsc{is connected to}, \cdot)$ and only very few pairs of airports are not connected, ResqLP assigns the probability 0.9 to Equation (18). This rule is, of course, useful in the sense that it predicts true facts. But the facts that it predicts can be inferred just as well using a manually defined rule without considering entity semantics.

An example of a rule that ResqLP learns to assign a high probability but at the same time never holds in the test data is the following,

$$\begin{aligned}&\textsc{was born in}(h,e_1) \wedge \textsc{is located in}(e_1,e_2)\\ &\wedge \textsc{deals with}(e_2,t) \implies \textsc{is citizen of}(h,t).\end{aligned} \tag{19}$$

where the relation DEALS WITH describes that two countries trade with each other. ResqLP assigns the probability 0.83 to this rule since it happens to hold in many cases in the training data. But, of course, we cannot derive the citizenship of a person based which countries their birth-country deals. This shows that ResqLP only learns rules based on the frequency of certain patterns in the KG. It does not learn what the relations actually mean.

The last example is a more intricate rule learned by ResqLP which captures how countries that import and export certain commodities trade with each other.

$$\begin{aligned}&\textsc{deals with}(h,e_1) \wedge \textsc{has neighbour}(e_1,e_2)\\ &\wedge \textsc{imports}(e_2,t) \implies \textsc{exports}(h,t).\end{aligned} \tag{20}$$

Here the HAS NEIGHBOUR relation refers to neighbouring countries. Equation (20) states that if some country h deals with a country whose neighbour imports a specific product t, then the country h is an exporter of the product t. The rule turns out to hold in all cases where it occurs and as such, ResqLP assigns a probability of 0.99. An example of where the rule holds is $h = \textsc{bulgaria}$ and $t = \textsc{clothing}$.

8 Conclusion

We introduce ResqLP, a link prediction (LP) model for knowledge graphs (KGs) that learns to predict how likely it is that a certain triple belongs in a KG. ResqLP is based on a recurrent neural network (RNN) and learns from sequences of relation found between entities as well as entity semantics. The task of finding the relation sequences is inherently hard since it depends on path enumeration. ResqLP includes a feature extraction component that aims to cope with this problem by only considering an enclosing subgraph of the KG, i.e. a subset that encloses the entities for which we are to predict a link. This decreases the time it takes to find the relation sequences significantly on some KGs. In other cases, we reduce the size of the enclosing subgraph more by uniformly sampling triples from it. Furthermore, we define a notion of entity semantics, called relation frequency

[4]The leaderboard can be found on the OpenBioLink GitHub repository: `https://github.com/OpenBioLink/OpenBioLink`.

semantics, that can be derived for any entity in any KG. ResqLP also takes these semantics into account in its RNN. We find that the model performs best on KGs with many different types of relations where the entities are sparsely connected. Furthermore, relation frequency semantics seems to be a good starting point for including entity semantics, but experiments show that a manually defined feature vectors results in better performance.

8.1 Future Work

As mentioned, we find that using enclosing subgraphs when enumerating relation sequences provides a significant speedup. This shows to be effective for some KGs while in other cases, the subgraphs are still to large. Our proposed solution to this is to uniformly sample a subset of these subgraphs. Even though this actually reduces the search space to a manageable size, we risk discarding valuable relation sequences. Furthermore, we cannot even guarantee that there are any relation sequences at all. Therefore, it is worth looking how to improve this sampling approach. This could, for instance, be done by associating each entity with an importance, such as eigenvector centrality, that can be used to define a probability distribution over entities to be employed in sampling. Given a sampling technique that is good at sampling important subgraphs, it can applied on all KGs, not only those where the search space is too large, to further improve path enumeration performance. This leads us to another area of improvement. Currently, we enumerate all the relation sequences in a given enclosing subgraph. However, as mentioned in Section 7.3, some of the rules that ResqLP are not interesting. So, if we instead only select the most important sequences, we may both improve execution time, since we traverse fewer paths, and performance, since the sequences we find are more relevant. This might be done in a way similar to the proposed enhanced subgraph sampling, i.e. by defining a relevant probability distribution over relations.

Another aspect that may be improved is the definition of entity semantics. As we conclude in Section 7.2, using domain-specific entity semantics on YAGO3-10 improves the performance of ResqLP. However, we have manually identified the features to include which, naturally, is not scalable to KGs where the number of relations and types of entities is high. Therefore, it is worth looking at how we automatically can derive semantics for the entities. In some cases, semantics may also come from external sources. For instance, in the WN18RR KG, word embeddings, which solve exactly the task of capturing word semantics, may be used. However, by doing so, the semantics are no longer observable in the KG. This is a property that we have referred to multiple times as being preferable since is allows ResqLP to handle previously unseen entities that may appear as the KG changes. One can argue, however, that in the case of WN18RR, this is not an issue since the set of entities in the KG, i.e. the words in the English language, very likely will not change.

Acronyms

CNN	convolutional neural network
GNN	graph neural network
H@N	hits-at-n
KG	knowledge graph
LFM	latent feature model
LP	link prediction
LRM	logic rule model
ML	machine learning
MRR	mean reciprocal rank
Neural LP	Neural Logic Programming
RNN	recurrent neural network
YAGO	yet another great ontology

References

[1] Brett W. Bader, Richard A. Harshman and Tamara G. Kolda. 'Temporal Analysis of Semantic Graphs Using ASALSAN'. In: *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*. IEEE Computer Society, 2007, pp. 33–42. DOI: 10.1109/ICDM.2007.54.

[2] Ivana Balazevic, Carl Allen and Timothy M. Hospedales. 'TuckER: Tensor Factorization for Knowledge Graph Completion'. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*. Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 5184–5193. DOI: 10.18653/v1/D19-1522.

[3] Kurt D. Bollacker et al. 'Freebase: a collaboratively created graph database for structuring human knowledge'. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*. Ed. by Jason Tsong-Li Wang. ACM, 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746.

[4] Antoine Bordes et al. 'A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation'. In: *Mach. Learn.* 94.2 (2014), pp. 233–259. DOI: 10.1007/s10994-013-5363-6.

[5] Antoine Bordes et al. 'Translating Embeddings for Modeling Multi-relational Data'. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges et al. 2013, pp. 2787–2795. URL: https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[6] Anna Breit et al. 'OpenBioLink: a benchmarking framework for large-scale biomedical link prediction'. In: *Bioinformatics* 36.13 (Apr. 2020). Ed. by Zhiyong Lu, pp. 4097–4098. DOI: `10.1093/bioinformatics/btaa274`.

[7] William W. Cohen, Fan Yang and Kathryn Mazaitis. 'TensorLog: A Probabilistic Database Implemented Using Deep-Learning Infrastructure'. In: *J. Artif. Intell. Res.* 67 (2020), pp. 285–325. DOI: `10.1613/jair.1.11944`.

[8] Rajarshi Das et al. 'Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks'. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. Ed. by Mirella Lapata, Phil Blunsom and Alexander Koller. Association for Computational Linguistics, 2017, pp. 132–141. DOI: `10.18653/v1/e17-1013`.

[9] Tim Dettmers et al. 'Convolutional 2D Knowledge Graph Embeddings'. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 1811–1818. URL: `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366`.

[10] Shobeir Fakhraei, Louiqa Raschid and Lise Getoor. 'Drug-target interaction prediction for drug repurposing with probabilistic similarity logic'. In: *Proceedings of the 12th International Workshop on Data Mining in Bioinformatics - BioKDD '13*. ACM Press, 2013. DOI: `10.1145/2500863.2500870`.

[11] Shobeir Fakhraei et al. 'Network-Based Drug-Target Interaction Prediction with Probabilistic Soft Logic'. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11.5 (Sept. 2014), pp. 775–787. DOI: `10.1109/tcbb.2014.2325031`.

[12] David Liben-Nowell and Jon M. Kleinberg. 'The link-prediction problem for social networks'. In: *J. Assoc. Inf. Sci. Technol.* 58.7 (2007), pp. 1019–1031. DOI: `10.1002/asi.20591`.

[13] Yankai Lin et al. 'Learning Entity and Relation Embeddings for Knowledge Graph Completion'. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by Blai Bonet and Sven Koenig. AAAI Press, 2015, pp. 2181–2187. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571`.

[14] Yang Liu et al. 'Anticipating Stock Market of the Renowned Companies: A Knowledge Graph Approach'. In: *Complexity* 2019 (Aug. 2019), pp. 1–15. DOI: `10.1155/2019/9202457`.

[15] Sarah J. MacEachern and Nils D. Forkert. 'Machine learning for precision medicine'. In: *Genome* 64.4 (Apr. 2021), pp. 416–425. DOI: `10.1139/gen-2020-0131`.

[16] Farzaneh Mahdisoltani, Joanna Biega and Fabian M. Suchanek. 'YAGO3: A Knowledge Base from Multilingual Wikipedias'. In: *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org, 2015. URL: `http://cidrdb.org/cidr2015/Papers/CIDR15_Paper1.pdf`.

[17] Tomás Mikolov et al. 'Distributed Representations of Words and Phrases and their Compositionality'. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges et al. 2013, pp. 3111–3119. URL: `https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html`.

[18] George A. Miller. 'WordNet: A Lexical Database for English'. In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: `10.1145/219717.219748`.

[19] T. Mitchell et al. 'Never-ending learning'. In: *Communications of the ACM* 61.5 (Apr. 2018), pp. 103–115. DOI: `10.1145/3191513`.

[20] Sameh K Mohamed, Vít Nováček and Aayah Nounu. 'Discovering Protein Drug Targets Using Knowledge Graph Embeddings'. In: *Bioinformatics* (Aug. 2019). Ed. by Lenore Cowen. DOI: `10.1093/bioinformatics/btz600`.

[21] Deepak Nathani et al. 'Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs'. In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2019, pp. 4710–4723. DOI: `10.18653/v1/p19-1466`.

[22] Arvind Neelakantan, Benjamin Roth and Andrew McCallum. 'Compositional Vector Space Models for Knowledge Base Completion'. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 2015, pp. 156–166. DOI: `10.3115/v1/p15-1016`.

[23] Maximilian Nickel, Volker Tresp and Hans-Peter Kriegel. 'A Three-Way Model for Collective Learning on Multi-Relational Data'. In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. Ed. by Lise Getoor and Tobias Scheffer. Omnipress, 2011, pp. 809–816. URL: `https://icml.cc/2011/papers/438_icmlpaper.pdf`.

[24] Adam Paszke et al. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 8024–8035. URL: `https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`.

[25] PyTorch Lightning. *PyTorch Lightning*. 13th June 2021. URL: `https://pytorchlightning.ai`.

[26] Meng Qu and Jian Tang. 'Probabilistic Logic Neural Networks for Reasoning'. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach et al. 2019, pp. 7710–7720. URL: https://proceedings.neurips.cc/paper/2019/hash/13e5ebb0fa112fe1b31a1067962d74a7-Abstract.html.

[27] Luc De Raedt, Angelika Kimmig and Hannu Toivonen. 'ProbLog: A Probabilistic Prolog and Its Application in Link Discovery'. In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. Ed. by Manuela M. Veloso. 2007, pp. 2462–2467. URL: http://ijcai.org/Proceedings/07/Papers/396.pdf.

[28] Richard Socher et al. 'Reasoning With Neural Tensor Networks for Knowledge Base Completion'. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by Christopher J. C. Burges et al. 2013, pp. 926–934. URL: https://proceedings.neurips.cc/paper/2013/hash/b337e84de8752b27eda3a12363109e80-Abstract.html.

[29] Zhiqing Sun et al. 'RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space'. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HkgEQnRqYQ.

[30] Thomas Pellissier Tanon, Gerhard Weikum and Fabian M. Suchanek. 'YAGO 4: A Reason-able Knowledge Base'. In: *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*. Ed. by Andreas Harth et al. Vol. 12123. Lecture Notes in Computer Science. Springer, 2020, pp. 583–596. DOI: 10.1007/978-3-030-49461-2_34. URL: https://doi.org/10.1007/978-3-030-49461-2_34.

[31] Komal Teru, Etienne Denis and Will Hamilton. 'Inductive Relation Prediction by Subgraph Reasoning'. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 9448–9457. URL: http://proceedings.mlr.press/v119/teru20a.html.

[32] Kristina Toutanova and Danqi Chen. 'Observed versus latent features for knowledge base and text inference'. In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, 2015. DOI: 10.18653/v1/w15-4007.

[33] Théo Trouillon et al. 'Complex Embeddings for Simple Link Prediction'. In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 2071–2080. URL: http://proceedings.mlr.press/v48/trouillon16.html.

[34] Ledyard R Tucker. 'Some mathematical notes on three-mode factor analysis'. In: *Psychometrika* 31.3 (Sept. 1966), pp. 279–311. DOI: 10.1007/bf02289464.

[35] Denny Vrandecic and Markus Krötzsch. 'Wikidata: a free collaborative knowledgebase'. In: *Commun. ACM* 57.10 (2014), pp. 78–85. DOI: 10.1145/2629489.

[36] Hongwei Wang et al. 'RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems'. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. Ed. by Alfredo Cuzzocrea et al. ACM, 2018, pp. 417–426. DOI: 10.1145/3269206.3271739.

[37] Zhen Wang et al. 'Knowledge Graph Embedding by Translating on Hyperplanes'. In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*. Ed. by Carla E. Brodley and Peter Stone. AAAI Press, 2014, pp. 1112–1119. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[38] Fan Yang, Zhilin Yang and William W. Cohen. 'Differentiable Learning of Logical Rules for Knowledge Base Reasoning'. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon et al. 2017, pp. 2319–2328. URL: https://proceedings.neurips.cc/paper/2017/hash/0e55666a4ad822e0e34299df3591d979-Abstract.html.

[39] Muhan Zhang and Yixin Chen. 'Link Prediction Based on Graph Neural Networks'. In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio et al. 2018, pp. 5171–5181. URL: https://proceedings.neurips.cc/paper/2018/hash/53f0d7c537d99b3824f0f99d62ea2428-Abstract.html.

[40] Yuyu Zhang et al. 'Variational Reasoning for Question Answering With Knowledge Graph'. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 6069–6076. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16983.

A Domain-Specific Entity Semantics for YAGO3-10

Here we describe how we infer the domain-specific entity semantics for the YAGO3-10 KG used in Section 5.2.2. By manually looking at the relations in the KG, we identify the following five features.

Type YAGO3-10 describes different types of entities. Many of these can be determined by looking at which relations they participate in. For instance, in triples on the form $(h, \text{ACTED IN}, t)$, the head entity is a person and the tail entity is a film. We have identified the following types: people (PERSON), geographical regions (GEO), events (EVENT), sports teams (TEAM), films (FILM), educational institutions (EDUCATION), gender (GENDER), awards (AWARD), languages (LANGUAGE), and musical instruments (INSTRUMENT). Table 5 shows the types we assign to the head and tail entities of different relations.

Table 5: Head and tail entities are assign a specific TYPE feature based on the relation they participate in.

Relation	Head type	Tail type
IS LOCATED IN	GEO	GEO
PLAYS FOR	PERSON	SPORTS TEAM
DIED IN	PERSON	GEO
ACTED IN	PERSON	FILM
GRADUATED FROM	PERSON	EDUCATION
WAS BORN IN	PERSON	GEO
HAS GENDER	PERSON	GENDER
HAPPENED IN	EVENT	GEO
HAS MUSICAL ROLE	PERSON	INSTRUMENT
HAS WON PRIZE	PERSON	AWARD
HAS OFFICIAL LANGUAGE	GEO	LAGUAGE
IS POLITICIAN OF	PERSON	GEO
DIRECTED	PERSON	FILM
CREATED	PERSON	FILM
IS CITIZEN OF	PERSON	GEO
DIRECTED	PERSON	FILM
CREATED	PERSON	FILM
IS CITIZEN OF	PERSON	GEO

Gender Triples on the form $(h, \text{HAS GENDER}, t)$ allow us to assign the gender t to the entity h.

Occupation A number of relations say something about a person's occupation. For instance, a triples such as $(h, \text{PLAYS FOR}, t)$ describes what sports team a person plays for. We can use this triple to say that h must be an athlete. Table 6 shows the relations that we infer occupations from.

Table 6: Certain relations let us infer an OCCUPATION feature.

Relation	Head entity occupation
ACTED IN	ACTOR
PLAYS FOR	ATHLETE
HAS MUSICAL ROLE	MUSICIAN
IS POLITICIAN OF	POLITICIAN
DIRECTED	PRODUCER
CREATED	PRODUCER

Language There are many geographical regions in YAGO3-10 and triples on the form $(h,$ HAS OFFICIAL LANGUAGE, $t)$ allow us to assign t as the LANGAUGE feature of h.

Nationality Similar to the language feature, we can infer the nationality of a person h by looking at the tail entity in $(h, \text{IS CITIZEN OF}, t)$.

In cases where entity has multiple values for a given feature, we select the most frequent one. For instance, if we are to assign an occupation to an entity that occurs in one $(h, \text{ACTED IN}, t)$ triple and two $(h,$ HAS MUSICAL ROLE, $t)$ triples, we choose MUSICIAN.