Embeddings for the $\mathcal{EL}^{++}$ Description Logic

Sutapa Mondal

IIIT-D-MTech-CS-DE-20-MT18056
July, 2020

Indraprastha Institute of Information Technology
New Delhi

Thesis Advisors
Dr. V. Raghava Mutharaju
Dr. Sumit Bhatia

Submitted in partial fulfillment of the requirements
for the Degree of M.Tech. in Computer Science & Engineering,
with Specialization in Data Engineering

CERTIFICATE

This is to certify that the thesis titled **" Embeddings for the $\mathcal{EL}^{++}$ Description Logic"** submitted by **Sutapa Mondal** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by her under our guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. V. Raghava Mutharaju
Indraprastha Institute of Information Technology, New Delhi

Dr. Sumit Bhatia
IBM Research, New Delhi

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude towards my advisors **Dr. V. Raghava Mutharaju** and **Dr. Sumit Bhatia**, for their continuous support throughout my studies and research work. Under their mentor-ship, I had enjoyed and learned to carry out research in a systemic manner. Whenever, I have approached them to discuss ideas or generic problems faced while working, I have always found an eager listener. Without their invaluable advice and assistance it would not have been possible for me to complete this thesis. I wholeheartedly acknowledge their full cooperation that I have received from the very beginning of this work. I would like to thank them for their constant encouragement at every stage of the thesis work. I'm grateful for this opportunity and look forward to continue my interactions with them in future.

I would also like to thank my friends and everyone at *"Knowledgeable Computing and Reasoning Lab(KRaCR)"* for their continuous support and guidance whenever needed. I'm happy to be associated with KRaCR lab which offers a positive environment of knowledge sharing and research.

Abstract

Knowledge graph (KG) embedding models have recently gained increased attention. However, most of the existing models for KG embeddings ignore the structure and characteristics of the underlying ontology. KGs are not always representative of the underlying configuration knowledge, they tend to capture the semantics at higher level. However, Ontologies are much generalized semantic data models which can capture more complex relationships between entities than KGs.
This research work proposes EmEL^{++} embeddings – an ontology-based embedding model for theories in Description Logic $\mathcal{EL}^{++}$. EmEL^{++} maps the classes and relations in an ontology to an n-dimensional vector space such that the relations between classes and relations in the ontology are preserved in the vector space. We evaluate the proposed embeddings on six different datasets and show that the proposed embeddings outperform the traditional knowledge graph embeddings on the subsumption reasoning task.

Keywords: Ontology, $\mathcal{EL}^{++}$, Description Logic ,Geometric Embeddings, Reasoning

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Methods for learning embedding functions that map the underlying entities (such as words, concepts, documents, nodes and edges in a graph) to a vector space has gained significant attention in the recent times. Different methods for learning embedding functions try to preserve the critical properties of, and relations between, the underlying entities in the n-dimensional vector space. For instance, the word embedding methods such as *word2vec* [23] try to map semantically similar words near each other in the vector space. Likewise, network embeddings such as *node2vec* [13] map nodes sharing similar structural properties close to each other in the vector space. These embeddings can be used for machine learning, similarity search, or similar tasks. In this work, we the study the limitations posed by knowledge graph embedding over ontology embedding. Further, we will look at ontology based embedding model with special focus on theories in Description Logic $\mathcal{EL}^{++}$. We discuss in detail the related background in Chapter 2 required to understand our work.

1.1 Motivation

A variety of embeddings for Knowledge Graphs [5, 20, 27, 34, 35, 36] have been proposed. These different methods differ in terms of the underlying properties of the knowledge bases preserved in the vector space and the techniques used to learn the mapping functions. However, most of the knowledge graph embedding models focus on capturing the structural properties of the graph and the interaction between the entities and *do not take into account the constraints and characteristics of the underlying ontology.* Consequently, the embeddings produced by such methods are not suited for reasoning tasks such as classification, satisfiability and consistency checking. This in-turn acts as a motivation to come up with techniques to generate embedding for an ontology.

1.2 Problem Statement

In this work, we aim to study and propose an approach to generate ontology embedding. We explore ontologies belonging to OWL 2 EL profile. Since, for more expressive ontologies like OWL 2 DL or very large size ontologies performing reasoning tasks is complex. Thus, to the best of our knowledge, we present the first attempt at performing reasoning tasks by embedding ontologies with OWL 2 EL profile in a vector space.

1.3 Thesis Contribution

The overall contribution of our work can be summarized as follows:

- We showcase the limitations with knowledge graph embedding models to perform reasoning tasks such as classification.
- We propose a framework for ontology embedding that offers complete coverage of the $\mathcal{EL}^{++}$ semantics (Chapter 4).
- We show the impact of relations on geometric orientation of classes when the embeddings are mapped to vector space.
- We show how the resulting embeddings can be used for performing the subsumption reasoning task. This is important as Baader et al. [2] have shown that all the standard reasoning tasks in $\mathcal{EL}^{++}$ ontologies can be reduced to subsumption task.

Further, we would also like to emphasize that the capability of performing reasoning in the vector space is critical as it has the potential to speed up the reasoning process significantly. As we describe in Chapter 5, the subsumption task in vector space involves computing distances between the source class and all the other classes in the ontology. In the worst case, this is an $O(n)$ operation where n is the number of classes. Thus, irrespective of the complexity of the underlying ontology, the subsumption task could be performed in $O(n)$ time. Further, with uses of techniques such as semantic hashing or binarized embeddings [25], the similarity based search operations can be performed in $O(1)$ time. Therefore, we believe that embedding based approaches, despite their lower accuracies than standard reasoners and no theoretical guarantees of performance, offer a promising direction of future research to develop more efficient reasoners, especially for more complex ontologies (such as OWL2 DL).

1.4 Thesis Outline

Chapter 2 gives an outline of the background needed to understand the work, followed by a literature review on existing embedding methods. First, we explain the different KG

embedding approaches and their limitations. Chapter 4 presents our proposed methodology to generate ontology embedding, wherein we outline the associated background knowledge, training and implementation approaches. Chapter 5 gives the detailed experimental setup carried out on different datasets followed by observations and analysis. The final chapter concludes our thesis and gives some leads for future work.

Chapter 2

Preliminaries

With the unprecedented explosion in the amount of information being generated and collected, an appropriate representation of the data becomes necessary. Herein, ontology and knowledge graphs (KGs) came into being. Both an ontology and knowledge graph represent knowledge bases, i.e., any collection of information.

2.1 Ontology

Ontologies [14] are semantic data models that define the types of things that exist in our domain and the properties that can be used to describe them.

An ontology is composed of three main components given as below:

- Classes: the distinct types of things that exist in our data.
- Relations: properties that connect two classes.
- Attributes: properties that describe an individual class.

For simplicity one can consider both relationships and attributes as properties.

Figure 2.1 represents an example of ontology for a given set of information on books, author and publishers. Since an ontology is a general data model, meaning that we don't want to include information about specific books in our ontology. Instead, we create a reusable framework we could use to describe any books in the future.

2.2 Knowledge Graphs

Knowledge graphs [15] is a multi-relational graph where nodes in KG represent the entities and the directed edges correspond to the relations between the entities. Together, these

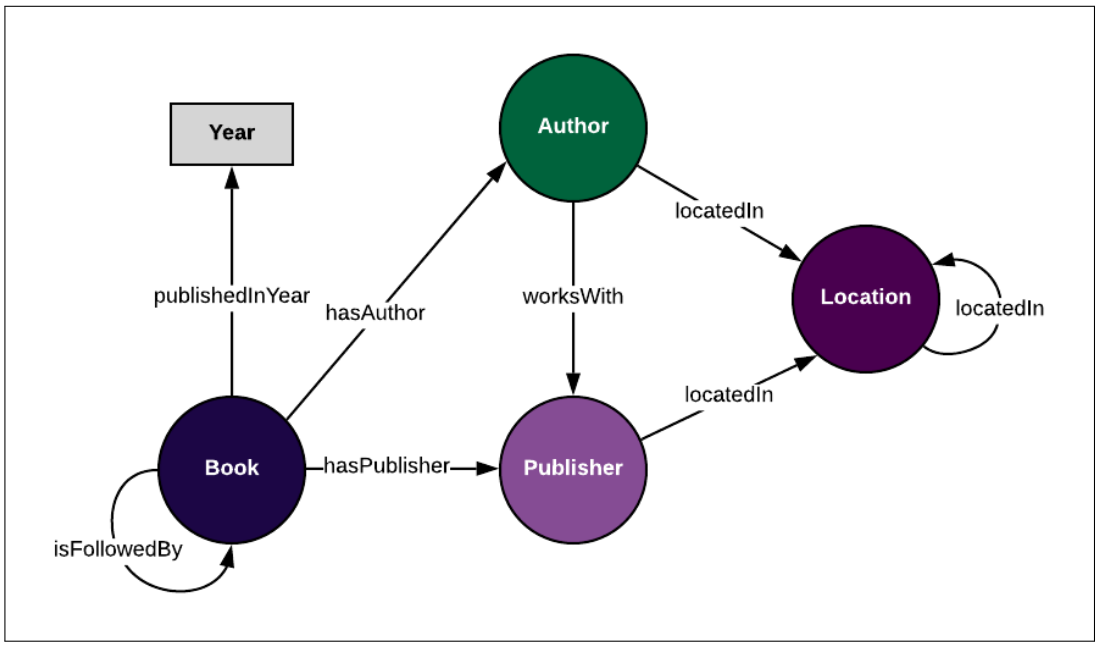

Figure 2.1: An example of Ontology

form facts in the KGs.These facts are represented in the form of triples as *(h,r,t)*, where h is the head entity, t is the tail entity and r is the relation associating the head with the tail entities. Using our ontology as a framework, we can add data to create a knowledge graph.

For example, with reference to ontology defined in Figure 2.1 we can add in real data about individual books, authors, publishers, and locations to create a knowledge graph represented by Figure 2.2. Thus, an ontology reflects the underlying configuration knowledge of a KG.

2.3 Description Logic

Description logics (DLs) are a family of formal knowledge representation languages, most of these are decidable fragments of First Order Logic(FOL)[4] [18]. Thus, DL is used for configuration knowledge representation such as ontologies.

2.3.1 DL Basics

Description logic is composed of three elements given as below:

- Concepts/Classes: Concepts are equivalent to unary predicates (e.g. Book,Author).
- Roles/Relations: Roles are equivalent to binary predicates. (e.g. hasAuthor, hasPublisher)

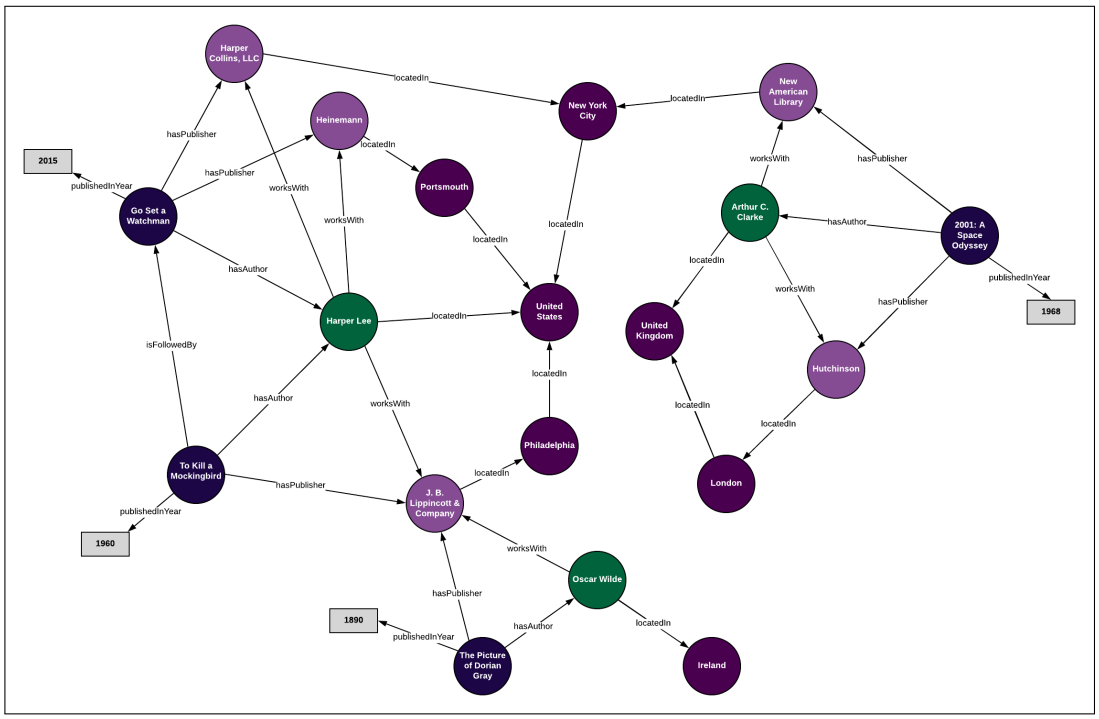

Figure 2.2: An example of Knowledge Graph

- Individuals: Individual names are equivalent to constants. (e.g. Harper Lee)

Thus, DL models concepts, roles and individuals, and their relationships present in an ontology. The fundamental modeling concept of a DL is the axiom, a logical statement relating roles and/or concepts. The logical statements in DL are formed by applying one or more of the following constructors on one or more of the atomic concepts: negation ($\neg$) of a concept, equality ($=$), intersection ($\sqcap$), union ($\sqcup$), and logical inclusion ($\sqsubseteq$) between two concepts. Also, there are two special concepts called $\top$ (every concept) and $\bot$ (empty concept). Finally, the operators universal restriction ($\forall R.C$) and existential restriction ($\exists R.C$). The axioms can be categorized as TBox and ABox. TBox is a set of terminology definitions (i.e. complex descriptions of concepts or roles) (e.g. $Human \sqsubseteq Mammal$) and ABox is a set of assertions about named individuals (e.g. Person(james)).

2.3.2 DL Languages

Each description logic describes a language, and each language differ in expressibility and reasoning complexity, defined by allowing or disallowing different constructs (e.g. conjunction, disjunction, negation, quantifiers, etc.) in their language. There are many varieties of description logics and there is an informal naming convention, roughly describing the operators allowed. The expressivity is encoded in the label for a logic starting with one of the following basic logics:

- $\mathcal{AL}$ Attributive language is the base language which allows atomic negation (negation of concept names that do not appear on the left-hand side of axioms), concept intersection, universal restrictions and limited existential quantification.
- $\mathcal{FL}$ Frame based description language which allows concept intersection, universal restrictions, Limited existential quantification and Role restriction.
- $\mathcal{EL}$ Existential language allows concept intersection and existential restrictions (of full existential quantification).

These languages can be further extended to support different constructs. For Example, $\mathcal{ALC}$, is simply $\mathcal{AL}$ with complement of any concept allowed, not just atomic concepts. Similary, $\mathcal{EL}^{++}$ is an extension of $\mathcal{EL}$.

Moreover, DLs are the underpinning for web ontology language(OWL). The current version of the OWL specification is OWL 2 as standardised in 2009. OWL is a family of knowledge representation languages for authoring ontologies. An OWL 2 profile (commonly called a fragment or a sublanguage in computational logic) is a trimmed down version of OWL 2. There are three profiles of OWL 2,OWL 2 EL is a fragment that has polynomial time reasoning complexity; OWL 2 QL is designed to enable easier access and query to data stored in databases; OWL 2 RL is a rule subset of OWL 2. In this work we focus on $\mathcal{EL}^{++}$ ontologies where $\mathcal{EL}^{++}$ is the underlying DL for the OWL 2 EL profile. .

2.3.3 DL Semantics

The formal meaning of DL axioms is given by their model-theoretic semantics. In particular, the semantics specifies what the logical consequences of an ontology. The formal semantics is therefore the main guideline that computes logical consequences of DL ontologies.

Ontologies usually cannot fully specify the situation that they describe. On the one hand, there is no formal relationship between the symbols we use and the objects that they represent: for example, an individual name james, is just a syntactic identifier with no intrinsic meaning. Indeed, the intended meaning of the identifiers in our ontologies has no influence on their formal semantics: what we know about them stems only from the ontological axioms. On the other hand, the axioms in an ontology may not provide complete information. Thus, the DL semantics generally considers all the possible situations where the axioms of an ontology would hold.

2.3.4 DL Reasoning

The capability of inferring additional knowledge increases the modelling power of DLs. Description logic reasoning involves deriving facts or patterns that are not expressed explicitly in an ontology. Description logics are created with the focus on tractable reasoning where some of the reasoning tasks can be classification/subsumption of concept,

satisfiability of a concept, consistency checking etc. Herein, we focus on subsumption of concepts as a reasoning task, i.e.,determine whether concept C subsumes concept D.

2.4 $\mathcal{EL}^{++}$ Ontology

The most notable application of DLs is in biomedical informatics where DL assists in the representation of biomedical knowledge. In this work we focus on ontologies belonging to OWL 2 EL because $\mathcal{EL}^{++}$ holds better algorithmic properties compared to others. Moreover, the expressive power of $\mathcal{EL}^{++}$ enables it for use in more applications. For example, SNOMED the Systematized Nomenclature of Medicine employs $\mathcal{EL}^{++}$ [33]. Further, large parts of GALEN medical knowledge can also be expressed[29].

The quest for tractable DLs that are expressive enough to be useful in practice led to $\mathcal{EL}^{++}$, an extension of $\mathcal{EL}$. While considering concepts descriptions for subsumption made tractability of DLs unattainable as investigated by Donini et. al. [8]. However, Baader et al. [2] proved that subsumption problem remains tractable even with addition of standard DL constructors such as concept descriptions for $\mathcal{EL}$ DL. Further, he proved that $\mathcal{EL}^{++}$ expressive power is enough to reduce all standard reasoning tasks to subsumption problem and vice-versa. Also, he showed that role inclusions ($r \sqsubseteq s$) generalize means of expressivity important in ontology applications: role hierarchies and transitive roles. In this work we particularly focus on subsumption reasoning task and also consider the impact of roles in our model discussed in detail in chapters 4 and 5. Table 2.1 summarizes the syntax and semantics supported by $\mathcal{EL}^{++}$ description logic.

Table 2.1: $\mathcal{EL}^{++}$ syntax and semantics

Name	**Syntax**	**Semantics**
Top concept	$\top$	Δ^{I}
Bottom concept	$\bot$	$\varnothing$
Nominal	$\{a\}$	$\{a^{\mathrm{I}}\}$
Negation	$\neg$C	$\neg C^{\mathrm{I}}$
Conjunction	$C \sqcap D$	$C^{\mathrm{I}} \cap D^{\mathrm{I}}$
Disjunction	$C \sqcup D$	$C^{\mathrm{I}} \cup D^{\mathrm{I}}$
Existential restriction	$\exists R.C$	$\{d_1 \mid \text{there exists } (d_1, d_2) \in R^{\mathrm{I}} \text{ with } d_2 \in C^{\mathrm{I}}\}$
Concept inclusion	$C \sqsubseteq D$	$C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$
Role inclusion	$R \sqsubseteq S$	$R^{\mathrm{I}} \subseteq S^{\mathrm{I}}$
Role chain	$R_1 \circ R_2 \sqsubseteq R$	${R_1}^{\mathrm{I}} \circ {R_2}^{\mathrm{I}} \sqsubseteq R^{\mathrm{I}}$
Concept assertion	$A(a)$	$a \in A^{\mathrm{I}}$
Role assertion	$r(a,b)$	$(a^{\mathrm{I}}, b^{\mathrm{I}}) \in R^{\mathrm{I}}$

Let $\mathbf{O} = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ be an $\mathcal{EL}^{++}$ ontology where $\mathbf{C}$ is the set of classes, $\mathbf{R}$ is the set of relations, $\mathbf{I}$ is the set of individuals.

Baader et al. [3] have shown that subsumption in $\mathcal{EL}^{++}$ can be reduced in linear time to subsumption with respect to normalized forms in $\mathcal{EL}^{++}$. Thus, accordingly an $\mathcal{EL}^{++}$ ontology can be reduced to a normalized form such that

- all the concept inclusions can take one of the following forms:
  1. $C_1 \sqsubseteq D$;
  2. $C_1 \sqcap C_2 \sqsubseteq D$;
  3. $\exists r.C_1 \sqsubseteq D$;
  4. $C_1 \sqsubseteq \exists r.C_2$;
- the bottom concept ($\bot$) can only appear on the right side of the concept inclusions, and can only appear in the first three forms mentioned above.
- all role inclusions are of the form $r \sqsubseteq s$ or $r_1 \circ r_2 \sqsubseteq s$

Further, the concept assertion and role assertion axioms in the ABox can be converted into TBox axioms as follows:

$$C(a) \longrightarrow \{a\} \sqsubseteq C$$
$$r(a, b) \longrightarrow \{a\} \sqsubseteq \exists r.\{b\}$$

Thus, with the above transformations, an $\mathcal{EL}^{++}$ ontology can be reduced to a normalized form and the task of embedding ontologies in a vector space requires us to learn mapping functions for both classes and relations in the ontology.

Chapter 3

Related Work

A wide range of methods for computing KG embeddings have been proposed. These methods aim to encode the entities and the relations in the knowledge graphs. Node2Vec proposed by Grover et. al. [13] initiated the idea of learning features for networks addressing the scalability challenge. It mainly focused upon preserving the neighborhood information of nodes in the graph/network. In this work features correspond to low dimensional vectors i.e. the embeddings and the problem derives its analogy from skip-gram architecture in Natural Language Processing (NLP) [23]. Although their results are decent with respect to the link prediction task but they make assumptions on conditional independence and symmetry of the feature space which may not hold true in real world scenarios. Some relations in a graph can be composition of other relations, in that case the embeddings generated with such assumptions may not be a good representation of nodes and relations in the graph. Further, bordes et. al. [6] looked into the problem of learning representation of elements in low dimensional vector space for a given knowledge base such that these embeddings can be used into statistical learning systems. They construct a low-dimension vector per entity and low-dimension matrix per relation. They define a notion of similarity between two entities and the training set consisting of triplets. Also, they use ranking to assess the quality of the embeddings learned.

This concept eventually got popularized with knowledge graphs wherein, a fact is represented as a triple of the form (h,r,t). A typical KG embedding technique generally consists of three steps (i) representing entities and relations, (ii) defining a scoring function, (iii) learning entity and relation representations. The first step specifies the form in which entities and relations are represented in a continuous vector space. Entities are usually represented as vectors. Then, in the second step, a scoring function is defined on each fact to measure its plausibility. Facts observed in the KG tend to have higher scores than those that have not been observed. Finally, to learn those entity and relation representations (i.e., embeddings), the third step solves an optimization problem that maximizes the total plausibility of observed facts.

Different KG embedding models were proposed. **Semantic matching models** for gener-

ating KG embeddings are based on similarity measures. These models exploit similarity based scoring functions and match latent semantics of entities and relations based on vector representations. Nikel et. al. [27] proposed RESCAL method which is categorized as semantic matching model. RESCAL uses multiple matrices to represent relations among entities. It introduces a huge number of parameters as it stores relations between each of the entities in matrices. Thus, scalability remains an issue with RESCAL. Further, Yang et. al. [36] proposed DistMult to overcome challenges of RESCAL. Although, DistMult is similar to RESCAL but they differ on the number of parameters. DistMult ensures low number of parameters for relations by restricting the matrices. Instead of using more complex matrices it utilizes diagonal matrices.

Later, **translational based models** for KG embeddings were introduced that use distance based scoring functions. Embedding models based on this technique gained most attention due to its simplicity to measure the correctness of a fact. It measures the plausibility of a fact as distance between the entities after translation carried out by relation. There are different variants of these translation based models, such as TransE [5] which is the most representative model. TransE represents the relations as translations such that given a fact (h,r,t), relation vector r minimizes the distance between h and t in vector space. This intuition originates from [24], which intends to learn the distributed word representations to capture linguistic regularities. For example, $Inception - ChristopherNolan \approx Avatar - JamesCameron$, such an analogy holds for multi-relational data because of a certain relation like *DirectorOf* in this case. Using these relations we can get $Inception + DirectorOf \approx ChristopherNolan$ and $Avatar + DirectorOf \approx JamesCameron$. Herein, the translation of *DirectorOf* relation over the subject entity i.e. the movie captures the idea behind TransE model. The scoring function is then defined as the negative distance between $h + r$ and t wherein, (h,r,t) denotes a fact. The score is expected to be large if the fact holds. Later to this, other variants such as TransH and TransR were introduced which tries to overcome drawbacks of TransE with respect to dealing with 1-to-N, N-to-N and N-to-1 relations[35][20]. TransH which interprets a relation as a translating operator on a hyperplane. Herein, each relation is associated to two vectors, the norm vector of the hyperplane and the translation vector on the hyperplane. It then uses the vectors obtained by projections of h and t onto the hyperplane i.e. $h_\perp$ and $t_\perp$ and there exists a vector on hyperplane that represents the relation between $h_\perp$ and $t_\perp$. These identified triple on the hyperplane then follows a training mechanism like TransE model. TransE and TransH models assume that the entity and relations are vectors in same semantic space, therefore similar entity will be close to each other in same entity space. However, each entity can have different aspects which may vary according to relations respectively. TransR [20] addresses this issue wherein, it models entities and relations in two distinct spaces namely, entity space and relation specific entity spaces. Thus, performs translation on corresponding relation space.

Further, Ristoski et. al. [30] came up with Rdf2vec which generates graph embeddings using language modelling approach. It requires to convert the graph into a sequence of entities to which CBOW and skip-gram models are applied. The model then estimates

the likelihood of the sequence. Garg et. al. [10] worked on a novel approach inspired by the theory of quantum logic to embed a Knowledge Base (KB) represented as an ontology. Their work is confined to simplest form of description logic i.e. ALC. Although they map quantum logic constraints to each of the logical structures in KB but do not talk about chaining constructs w.r.t. relations in KB. Moreover, the work is restricted to ontologies with ALC profiles whereas large ontologies for example, that of life sciences have been formulated in the Web Ontology Language (OWL) [12] and falls majorly under OWL 2 EL profile.

Existing works focusing on ontology embedding such as Onto2vec [32] was proposed. It uses word2vec as an underlying model which treats axioms in an ontology as sentences to generate vector representations. Most of this work focuses on encoding the entities and relations, but they lack in handling the complex relations in an ontology. Moreover, the existing works do not generalize the embeddings for all kinds of tasks. For example, most of them take up link prediction as the main task for evaluation of embeddings. Apart from this, approaches involving RDFs does not support quantifiers whereas an ontology includes cases of complex relations with quantifiers. Thus, proposed embedding models with KG or RDF alone cannot perform the reasoning tasks on ontologies accurately. However, Kulmanov et. al. [17] learns embeddings for ontology using $\mathcal{EL}^{++}$ description logics. Although, it tries to overcome the drawbacks of KG embeddings but it does not address all the $\mathcal{EL}$ constructs that are relevant to capture the relations present in an ontology. Further, the evaluation is focused only along link prediction task which may not be an appropriate way to measure the geometric notion of embeddings. EmEL^{++} builds upon existing models to improve embeddings for ontology focusing on subsumption reasoning task.

Chapter 4

Embedding $\mathcal{EL}^{++}$ Ontologies in a Vector Space

This chapter details out the approach involved in our ontology embedding framework EmEL^{++}. The below sections provide the intuition followed by methodology associated with our model.

4.1 Intuition

Typically, methods for mapping the entities of interest to a vector space learn the mapping function subject to certain constraints, encoded in the form of an objective function that is optimized during the training phase. These objective functions are designed such that certain specific properties of the underlying entities are also retained in the vector space. For example, the word2vec [23] model for word embeddings minimizes the distance between contextually similar words, RDF2Vec [30] adapts language modeling approach to capture local information from the graph sub-structures, and TransE [5] model for knowledge base embeddings models the relationship vectors as translation operation on the entities. Similarly, we are interested to learn mapping functions that can embed $\mathcal{EL}^{++}$ ontologies in a vector space while maintaining the semantics of the underlying ontologies. In order to do so, we build upon and extend the framework proposed by Kulmanov et al. [17] that interprets a class in the ontology as an n-ball (defined by its radius and center) in the vector space.

Let us consider two classes C and D such that $C \sqsubseteq D$. Let these two classes be represented by their respective n-balls b_c and b_d in the vector space such that $b_c : \{\vec{c}, r_c\}$ and $b_d : \{\vec{d}, r_d\}$; where $\vec{c}$ and $\vec{d}$ are the centers and r_c and r_d are the radii of the respective n-balls. Geometrically, if $C \sqsubseteq D$, the mapping function should aim to ensure that the b_c lies inside b_d (Figure4.1 (a)). Similarly, if C and D are disjoint, the respective n-balls should not overlap with each other in the vector space (Figure4.1 (b)). Further, similar to

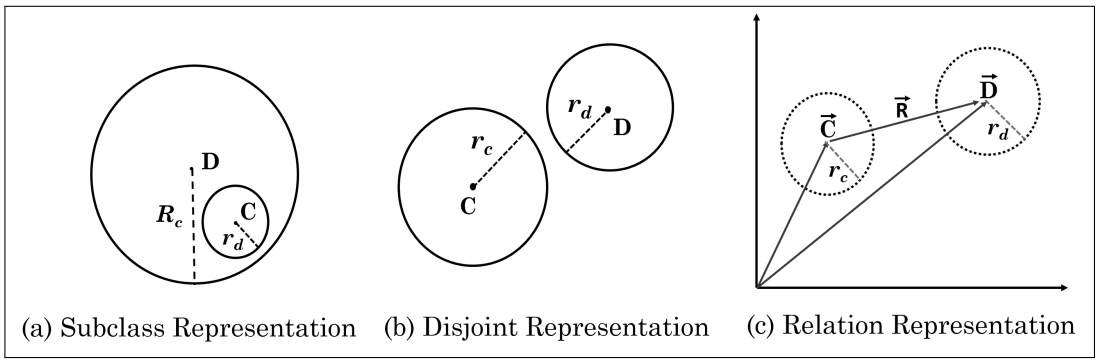

Figure 4.1: Geometric Representation of classes and relations

the TransE model [5], the relations in the ontology are interpreted as translations operating on the classes. More specifically, if $C \sqsubseteq \exists R.D$, the center of n-ball representing C can be moved to the center of the n-ball representing D translating via relation R (Figure4.1 (c)).

4.2 Loss Functions

With the intuitive framework described above, let us now describe the objective functions that should be optimized during the training phase to learn the mapping functions. Let $e_v : \mathbf{C} \cup \mathbf{R} \longmapsto \mathbb{R}^n$ be the mapping function that maps each class and relation to a unique vector in the n-dimensional embedding space. For $C_i \in \mathbf{C}$, the resulting vector corresponds to the centre of the n-ball representing the class. Further, let $e_r : C \longmapsto \mathbb{R}^+$ be the mapping function that maps each class into a non-negative real number, that represents the *radius* of the n-ball corresponding to class C. Thus, the pair (e_v, e_r) of functions represents the operations needed to *embed* an $\mathcal{EL}^{++}$ ontology into an n-dimensional space. We now describe the various loss functions to represent the different constructs in $\mathcal{EL}^{++}$. The total loss that needs to be minimized during the learning process is the sum of the individual loss functions.

Loss Functions for the Normal Forms:

As described before, the first normal form ($C \sqsubseteq D$) when embedded in a vector space can be interpreted geometrically as two n-balls, such that the n-ball corresponding to class C lies inside the n-ball corresponding to D. Hence, our mapping functions e_v and e_r should bring the centers of the two classes closer to each other, and give the sub-class a smaller radius than the super-class. The loss function presented in Equation 4.1 captures this intuition and penalizes the mappings that do not adhere to these constraints. Also note that in addition to the above constraints, we also add margin loss (γ) and a normalization

loss that brings the centres of n-balls of all the classes on the unity sphere.

$$\begin{aligned}\mathcal{L}_{C\sqsubseteq D}(c,d) = \max\left(0, \left(\underbrace{\|e_v(c)-e_v(d)\|}_{\substack{\text{penalize if the two}\\ \text{centers are far away}}} + \underbrace{e_r(c)-e_r(d)}_{\substack{\text{penalize if sub-class}\\ \text{has larger radius}}} - \gamma\right)\right) \\ + \quad \Big|\|e_v(c)\|-1\Big| + \Big|\|e_v(d)\|-1\Big|\end{aligned} \tag{4.1}$$

In the vector space, the second normal form, i.e., $C \sqcap D \sqsubseteq E$, implies that the n-ball for class E should completely engulf the area of intersection of n-balls for classes C and D. The first term in the loss function (Equation 4.2) imposes a penalty if the classes C and D are disjoint. The second and third terms together enforce that the center of the n-ball for class E lies in the area of intersection of n-balls for classes C and D (optimal position being the midpoint of the line joining their centers). Finally, the fourth term requires the radius of the n-ball of E to be greater than the smaller of the radii of n-balls of C and D.

$$\begin{aligned}\mathcal{L}_{C\sqcap D\sqsubseteq E}(c,d,e) &= \max\Big(0, \big(\|e_v(c)-e_v(d)\| - e_r(c) - e_r(d) - \gamma\big)\Big) \\ &+ \max\Big(0, \big(\|e_v(c)-e_v(e)\| - e_r(c) - \gamma\big)\Big) \\ &+ \max\Big(0, \big(||e_v(d)-e_v(e)|| - e_r(d) - \gamma\big)\Big) \\ &+ \max\Big(0, \big(\min\left(e_r(c), e_r(d)\right) - e_r(e) - \gamma\big)\Big) \\ &+ \Big|\|e_v(c)-1\|\Big| + \Big|\|e_v(d)-1\|\Big| + \Big|\|e_v(e)-1\|\Big|\end{aligned} \tag{4.2}$$

The first two normal forms are concerned with the mappings of classes and properties of their respective n-balls in the vector space. The next two normal forms involve relations and how they are associated with the classes. Recall that similar to TransE [5], we consider the relations in ontology as translations that operate on classes. Consider the normal form $C \sqsubseteq \exists R.D$. In the vector space, C and D are represented as two n-balls b_c and b_d, respectively. If $e_v(R)$ is the vector for R in vector space, then adding $e_v(R)$ to a point in b_c should move it to a point in b_d (i.e., R translates the points in b_c to points in b_d). The following loss functions capture these semantics as expressed by the third and fourth normal forms.

$$\begin{aligned}\mathcal{L}_{C\sqsubseteq\exists R.D}(c,d,r) &= \max\Big(0, \big(\|e_v(c)+e_v(r)-e_v(d)\| + e_r(c) - e_r(d) - \gamma\big)\Big) \\ &+ \quad \Big|\|e_v(c)\|-1\Big| + \Big|\|e_v(d)\|-1\Big|\end{aligned} \tag{4.3}$$

$$\begin{aligned}\mathcal{L}_{\exists R.C\sqsubseteq D}(c,d,r) &= \max\Big(0, \big(\|e_v(c)-e_v(r)-e_v(d)\| - e_r(c) - e_r(d) - \gamma\big)\Big) \\ &+ \quad \Big|\|e_v(c)\|-1\Big| + \Big|\|e_v(d)\|-1\Big|\end{aligned} \tag{4.4}$$

Handling Bottom Concept ($\bot$)**:**

Recall from the discussion in Section 2.4 that the bottom concept can appear only on the right hand side of the first three normal forms [2]. We now present the loss functions for each of the three special cases. The resulting first normal form $C \sqsubseteq \bot$ indicates that class C is unsatisfiable. Thus, in the vector space, we represent this constraint by reducing the radius of class C to zero. This is achieved by the following loss function.

$$\mathcal{L}_{C \sqsubseteq \bot}(c) = e_r(c) \tag{4.5}$$

Next, the second normal form with the bottom concept is $C \sqcap D \sqsubseteq \bot$ indicating that C and D are disjoint. In the vector space, this indicates that the n-balls of classes C and D are non-overlapping. This is captured by the following loss function.

$$\begin{aligned} \mathcal{L}_{C \sqcap D \sqsubseteq \bot}(c, d) = \max\Big(0, \big(e_r(c) + e_r(d) - \|e_v(c) - e_v(d)\| + \gamma\big)\Big) \\ + \quad \Big|\|e_v(c)\| - 1\Big| + \Big|\|e_v(d)\| - 1\Big| \end{aligned} \tag{4.6}$$

Finally, the third normal form $\exists R.C \sqsubseteq \bot$ indicates that in the vector space *translating* C by R results in an unsatisfiable class. We already require the radius of unsatisfiable classes to be zero (Equation 4.5) and since translation does not change the radius of the original class, we have the following loss function.

$$\mathcal{L}_{\exists R.C \sqsubseteq \bot}(c, r) = e_r(c) \tag{4.7}$$

Loss Functions for Role Inclusions and Role Chains:

The role vectors in our proposed framework serve the purpose of translating one class to another class. The constraints considered until now have imposed restrictions on the role vectors based on their relations with the n-balls of the concerned classes. We now present two loss functions to capture the constraints imposed by role inclusions and role chains in the ontology. The role inclusion of $R \sqsubseteq S$ implies that the vectors $e_v(R)$ and $e_v(S)$ in the vector space should be nearby because any translation produced by R should also be producible by S plus both the vectors should be in the same direction. This intuition is captured by the following loss function represented by Equation 4.8. Herein, the first term is indicative of the distance that ensures the vectors $e_v(R)$ and $e_v(S)$ lie in near vicinity of each other. The second term captures the directional aspect of roles in vector space such that they tend to be in same direction.

$$\begin{aligned} \mathcal{L}_{R \sqsubseteq S}(r, s) = \max\Big(0, \|e_v(s) - e_v(r)\| - \gamma\Big) \\ + \quad \left|1 - \frac{e_v(r).e_v(s)}{\|e_v(r)\|\|e_v(s)\|}\right| \\ + \quad \Big|\|e_v(r)\| - 1\Big| + \Big|\|e_v(s)\| - 1\Big| \end{aligned} \tag{4.8}$$

Next, we consider the hierarchy defined by the role chain $R_1 \circ R_2 \sqsubseteq S$. In the vector space, this implies that if class C can be translated to class E by successive application of R_1 and R_2, it can also be translated to E directly by the vector for role S while preserving the direction of role vectors. The following loss function captures this behavior represented by Equation 4.9.

$$\begin{aligned}\mathcal{L}_{R_1 \circ R_2 \sqsubseteq S}(r1, r2, s) = \max\Big(0, &\|e_v(s) - e_v(r_1) - e_v(r_2)\| - \gamma\Big) \\ + \quad &\left| 1 - \frac{(e_v(r_1) + e_v(r_2)).e_v(s)}{\|(e_v(r_1) + e_v(r_2))\|\|e_v(s)\|} \right| \\ + \quad &\Big|\|e_v(r_1)\| - 1\Big| + \Big|\|e_v(r_2)\| - 1\Big| + \Big|\|e_v(s)\| - 1\Big|\end{aligned} \tag{4.9}$$

Often, negative sampling is employed during the training phase to learn better embeddings as negative samples can be easily generated to enhance the training data available. In order to incorporate negative samples in the training phase, the following loss function is employed.

$$\begin{aligned}loss_{C \not\sqsubseteq \exists R.D}(c, d, r) = \max\Big(0, &e_r(c) + e_r(d) - \|e_v(c) + e_v(r) - e_v(d)\| + \gamma\Big) \\ + \quad &\Big|\|e_v(c)\| - 1\Big| + \Big|\|e_v(d)\| - 1\Big|\end{aligned} \tag{4.10}$$

Thus, the total loss for learning the embedding function is the sum of all the loss functions given by Equations 4.1- 4.10. Further, we also add the constraint that radius of the satisfiable classes are non-negative and penalize the total loss for learning negative radius for classes.

4.3 Training and Implementation

Given an $\mathcal{EL}^{++}$ ontology, we first normalize the ontology to generate the normal forms. These normal forms then constitute as a set of TBox statements wherein each axiom is treated as a positive sample. This normalization is performed using the OWL APIs and the APIs provided by the jCel reasoner which implements the normalization rules [22]. We then introduce negative samples using the third normal form (refer to Equation 4.3). We randomly generate corrupted axioms following $C \sqsubseteq \exists R.D$, by replacing C or D with C' or D' such that neither $C' \sqsubseteq \exists R.D$ nor $C \sqsubseteq \exists R.D'$ are asserted axioms in the ontology. Therefore, based on the facts the training process learns ontology embedding such that

the facts hold true.

Algorithm 1: Algorithm for generating Ontology Embeddings with $\mathcal{EL}^{++}$ profile

Input: Ontology $\mathbb{O} = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ in OWL format with axioms ax; Margin γ; Dimensions n; Epochs $epochs$; batchsize bs;

Output: Embeddings $(e_v(c), e_r(c)$ for classes and $e_v(r)$ for relations

/* Apply $\mathcal{EL}^{++}$ normalization rules */

$(ax_{nf1}, ax_{nf2}, ax_{nf3}, ax_{nf4}, ax_{dis}, ax_{ri}, ax_{rc}) = normalize(ax)$

/* Generate negative samples with third normal form */

$neg_{nf3} = negatives(ax_{nf3})$

$D \leftarrow ax_{nf1} \cup ax_{nf2} \cup ax_{nf3} \cup ax_{nf4} \cup ax_{dis} \cup ax_{ri} \cup ax_{rc}$

/* Initialize embeddings with given dimension */

$e_v(c) = uniform(0, 1)$ for each $c \in \mathbf{C}$

$e_r(c) = uniform(0, 1)$ for each $c \in \mathbf{C}$

$e_v(r) = uniform(0, 1)$ for each $r \in \mathbf{R}$

for ($i = 0;\ i < epochs;\ i = i + 1$) {

  /* sample mini batches of size bs */

  $(s_{nf1}, s_{nf2}, s_{nf3}, s_{nf4}, s_{dis}, s_{ri}, s_{rc}) = samples(D, bs)$

  /* Update embeddings */

  $\sum \nabla loss(s_{nf1}, s_{nf2}, s_{nf3}, s_{nf4}, s_{dis}, s_{ri}, s_{rc})$

The code for training of embeddings and optimization is implemented using Python and Tensorflow library, and Adam optimizer [16] is used for updating the embeddings. Algorithm 1 details out the training process to generate embeddings. The training is divided into two phases, in the first phase the ontology in OWL format is reduced to the $\mathcal{EL}^{++}$ normal forms. This normalization is performed using the OWL API and the APIs provided by the jCel reasoner which implements the $\mathcal{EL}^{++}$ normalization rules [22]. The ontology with the normal forms is parsed to identify the axioms representing the different normal forms defined in $\mathcal{EL}^{++}$ ontology denoted as ax_{nf1}, ax_{nf2} , ax_{nf3}, ax_{nf4} etc. in algorithm 1 where ax_{nf*} represents normal forms for concepts and ax_{dis} for disjoint cases. Moreover, ax_{ri} and ax_{rc} represents for role inclusions and role chains. In the second phase, the training of embeddings and optimization is performed. The classes and relations are encoded as integer values. Thus, the normal forms are also encoded into integer values accordingly with respect to the classes and relation. We define our linear model which takes the identified normal forms as input. The model uses two parallel embedding layers, one for the classes and another for the relations to map them to n-dimensional vector space. The dimensions defined in embedding layers for relation vectors is n and for classes is (n+1) where the last dimension represents the radius of the class. The weights in the embedding layer finally determines the generated embedding for classes and relations. These weights form the parameters of the neural network which are adjusted during training to minimize the loss on the task. These embedding layers are then followed by another layer that computes the total loss i.e. the sum of the losses incurred by each of the normal forms described above (Eq.4.1-Eq.4.10). In order to compute the total loss we define functions

for each of the losses defined for normal forms and the parameters to these functions are defined in their respective equations. Therefore, overall we start the learning process by initializing the embedding weights for classes and relations by random values. After which, we process the training samples in mini-batches for each of the losses defined for the normal forms described above (Eq.4.1-Eq.4.10) and update the embeddings depending upon the total loss i.e. the sum of all the loss functions. The update process is carried till saturation or for a fixed number of epochs. Finally, the resulting embedding weights determine the embedding vectors associated with each of the classes and relations.

Chapter 5

Experiments,Results and Observations

We evaluate the effectiveness and performance of EmEL^{++} on different ontologies. Further, we describe the datasets used, followed by experimental setup, evaluation metrics and analysis of results.

5.1 Datasets

We use following six different ontologies of varying size and different characteristics.

1. **SNOMED CT:** SNOMED Clinical Terms [9] ontology conceptualizes the medical terms used for clinical documentation and reporting. The primary purpose of SNOMED CT is to encode the meanings that are used in health information and to support the effective clinical recording of data. Further, it consists of 989k TBox statements with 307k classes and 60 relations.

2. **Anatomy:** It is a reference ontology for domain of anatomy [26]. It focuses on the representation of phenotypic diversity and linking phenotypes to genes. It consists of 278k TBox statements along with 106k classes and 218 relations.

3. **Foundational Model of Anatomy (FMA):** FMA [31] is a domain ontology that represents a coherent body of explicit declarative knowledge about human anatomy.It consists of 211k TBox statements along with 84k classes and 87 relations.

4. **Gene Ontology(GO):** Gene Ontology [7] unifies the representation of gene and gene product attributes across all species. It consists of 130k TBox statements that conceptualize the domain. These statements introduce 45k classes and 16 relations that belong to GO.

Table 5.1: Different ontologies used in this work and count of different types of axioms. NFi represents the i^{th} normal form as described in Chapter 4

Ontology	NF1	NF2	NF3	NF4	Disjoint	Role Inclusion	Role Chain
NCI	32909	0	13961	70	0	0	0
GALEN	28890	13595	28118	13597	0	958	58
GO	85480	12131	20324	12129	30	3	6
FMA	84444	0	126796	22	0	15	0
ANATOMY	122142	2121	152289	2143	184	89	31
SNOMED CT	446628	27779	482330	32449	0	11	1

5. **GALEN:** Galen [28] also represents clinical information. It consists of 84k TBox statements with 24353 classes and 1010 relations.

6. **National Cancer Institute (NCI):** NCI [11] ontology includes broad coverage of the cancer domain, including cancer related diseases, findings and associated abnormalities.It consists of 46k TBox statements with 27k classes and 71 relations.

Table 5.1 presents the six chosen ontologies in the increasing order of their size (number of axioms) and highlights the differences between them in terms of the coverage of different types of axioms. For instance, we note that SNOMED has 0 disjoint axioms and only 1 role chain axiom, despite being the largest amongst the six ontologies. On the other hand, GALEN, being one of the smaller ontologies, has the highest number of role inclusion (958) and role chain (58) axioms. Also, observe that ANATOMY is the only ontology considered that has the representation of all EL constructs considered in this work.

5.2 Baselines

The standard knowledge graph embedding (KGEs) models have been majorly used for KG completion or link prediction tasks. We consider some commonly used KGEs for comparison in order to understand their limitations for reasoning tasks. Also, since the underlying relational model in our proposed EmEL^{++} embedding model is similar to distance-based KGEs. Further, we draw close comparison of EmEL^{++} with an $\mathcal{EL}$ embedding model [17] which is the most recent state-of-the-art for ontology embedding with $\mathcal{EL}^{++}$ profile. Thus, the baselines chosen are described as below:

1. **TransE** [5], one of the most frequently used embedding model for knowledge graph, introduced the idea of translation based embeddings where the relations between entities is interpreted as a translation operation between the entities.

2. **TransH** [35], is an extension of TransE that better handles reflexive, one-to-many, many-to-one, and many-to-many relations. Unlike TransE, where relations are repre-

sented as vectors in the embedding space, TransH considers relations as hyperplanes in the embedding space. The translation operation is then performed over the projections of entities on the hyperplane.

3. **DistMult** [36], a matrix factorization based embedding model, has been found empirically to perform well at compositional reasoning tasks.

4. **EL Embeddings (ElEm)** [17] is one of the first embedding models for the $\mathcal{EL}^{++}$ description logic based model. Our proposed model is also an extension of ElEm embeddings and enhances ElEm by introducing additional constraints for a more comprehensive coverage of $\mathcal{EL}^{++}$ description logic.

We use the pykeen framework [1] for implementations of TransE, TransH, and DistMult embedding models. For ElEm embeddings, we used the source code provided by the authors [1]. Our implementation of EmEL^{++} is available on research group's GitHub page[2].

5.3 Experimental Protocol

For learning the embeddings by different models, we first normalize the ontologies as described in Chapter 4. Next, we remove 30% of the subclass relation pairs from the normalized ontology to be used for validation (20%) and testing (10%). The remaining ontology along with 70% sub-class relation pairs is used as the training set for learning the embedding functions. The training is carried out for 1000 epochs or till a saturation is reached. We perform hyper-parameter tuning using the 20% validation set and report the performance of fine-tuned models on the test set. The hyper-parameters to tune for all the models are the dimensions of the embedding vectors, and the margin parameter γ. We consider $n = \{50, 100, 200\}$ and $\gamma = \{-0.1, 0.0.1\}$ yielding nine different settings. The best performing hyper-parameters for each of the models are reported in Table 5.2.

5.4 Results and Observations

We chose subsumption as the main task to evaluate the effectiveness of the proposed EmEL^{++} embeddings. Baader et al. [2] have shown that all the other standard reasoning tasks (such as concept satisfiability, ABox consistency, and instance problem) can be reduced to the subsumption task in $\mathcal{EL}^{++}$ ontologies.

Note that once we have embedded the ontologies in a vector space, we have to reduce all the tasks we want to accomplish to operations that can be performed in an n-dimensional space.

[1] https://github.com/bio-ontology-research-group/el-embeddings
[2] https://github.com/kracr/EmELpp

Table 5.2: Best performing Hyper-parameters for each model. n indicates the dimension of embedding vectors and γ is the margin loss parameter.

	EmEL^{++}		ElEm		TransE		TransH		DistMult	
	n	γ	n	γ	n	γ	n	γ	n	γ
NCI	200	-0.1	50	0.1	100	-0.1	100	0.1	200	-0.1
GALEN	50	0.0	50	0.0	100	-0.1	100	-0.1	100	-0.1
GO	100	-0.1	100	-0.1	100	-0.1	100	-0.1	100	0.1
FMA	200	0.1	50	-0.1	100	0.1	100	-0.1	100	0.0
ANATOMY	200	-0.1	200	-0.1	100	-0.1	100	-0.1	100	-0.1
SNOMED CT	100	-0.1	100	-0.1	50	-0.1	50	-0.1	50	-0.1

Typically, distance-based metrics (such as Euclidean distance) are employed to perform various tasks in the embeddings space. For example, in the case of word embeddings, the task of finding similar words and related concepts is accomplished by finding the input word's nearest words (or concepts). Similarly, the missing links in knowledge bases are predicted by ranking the nodes in the graph based on their distance with the source node. We also reduce the task of subsumption in the embedding vector space as a distance-based operation. Given a test instance of the form $C \sqsubseteq D$, we take D as our source class and rank all the other classes in the ontology in increasing order of their distance from D in the vector space. We then compare the effectiveness of different embedding models based on the rank at which C is present in the ranked list. An embedding model that successfully captures the subclass relation between the two classes should be able to assign vector representations to the two classes that are very close to each other, hence, producing a lower rank for C.

Table 5.3 summarizes the performance of different embedding models for the subsumption task for the six datasets. We evaluate the performance using five metrics. Hits at ranks 10 and 100 report the fraction of test cases for which the expected class was found within top 10 and 100 ranks, respectively. A median rank of m means that for 50% of the test cases, the correct answer was found below rank m. 90^{th} percentile rank denotes the rank value below which the correct class was found for 90% of the test cases.

The first observation that we make from Table 5.3 is that ElEm and EmEL^{++} embeddings perform better than the three commonly used knowledge graph embeddings (TransE, TransH, and DistMult). This observation highlights the inadequacy of traditional knowledge graph embeddings that do not consider the ontological constructs and rely only on the structural properties of the underlying graph. Both ElEm and EmEL^{++} embeddings incorporate specific constraints and charactersitics of $\mathcal{EL}^{++}$ description logic, and hence, the embeddings produced by these models are better at retaining the properties of the underlying ontologies in the vector space.

Next, we note from Table 5.3 that there is no clear winner among ElEm and EmEL^{++}

Table 5.3: Embedding Models Ranking based Performance

Dataset	Metric	DistMult	TransE	TransH	ELEm	EmEL^{++}
NCI	**Hits@10**	0.00	0.00	0.00	0.10	**0.17**
	Hits@100	0.00	0.03	0.00	0.24	**0.34**
	AUC	0.50	0.68	0.50	0.89	**0.90**
	Median Rank	10818	4934	10533	672	**388**
	90th Percentile Rank	19437	16663	18882	10200	**9169**
GALEN	**Hits@10**	0.00	0.00	0.00	**0.05**	0.00
	Hits@100	0.00	0.03	0.00	**0.42**	0.24
	AUC	0.50	0.77	0.53	0.91	**0.92**
	Median Rank	10900	2829	10724	**169**	193
	90th Percentile Rank	19966	14201	19122	8605	**6804**
GO	**Hits@10**	0.00	0.01	0.00	0.01	0.01
	Hits@100	0.01	0.08	0.00	0.05	0.05
	AUC	0.53	0.72	0.47	**0.93**	0.92
	Median Rank	10084	3664	14260	**657**	874
	90th Percentile Rank	19268	16688	19137	**8710**	8930
FMA	**Hits@10**	0.00	0.00	0.00	0.01	**0.03**
	Hits@100	0.00	0.00	0.00	0.1	**0.30**
	AUC	0.50	0.57	0.51	0.85	**0.90**
	Median Rank	29350	23618	29123	2330	**256**
	90th Percentile Rank	51856	50232	52353	47705	**16605**
ANATOMY	**Hits@10**	0.00	0.00	0.00	**0.04**	0.02
	Hits@100	0.00	0.01	0.00	0.25	**0.29**
	AUC	0.50	0.64	0.49	0.85	**0.91**
	Median Rank	29214	16774	29640	481	**206**
	90th Percentile Rank	52403	47263	53205	40909	**15525**
SNOMED CT	**Hits@10**	0.00	0.00	0.00	**0.01**	0.00
	Hits@100	0.00	0.05	0.01	**0.14**	0.03
	AUC	0.51	0.57	0.54	**0.90**	0.85
	Median Rank	13228	9571	9574	1300	**1002**
	90th Percentile Rank	24634	19248	19605	14140	**12604**

embeddings. For NCI and FMA ontologies, EmEL^{++} outperforms ElEm across all the metrics. Also, note that ANATOMY which comprises of all the forms in sufficient number outperforms ElEm across almost all the metrics. For GALEN,GO and SNOMED ontologies, there is no clear winner, and each of the two methods performs better on some metrics and has a lower performance on other metrics. This observation is consistent with previous empirical studies comparing different link prediction methods that found that no single method outperforms across a variety of datasets [19, 21]. We speculate that this divergence in performance could be attributed to the different distributions of different types of axioms in the ontology (ref. Table 5.1). Over (or under) representation of certain types of axioms may lead to the optimization process giving more (or less) weight to the corresponding

loss functions during the training phase. Understanding the exact mechanism behind the performance characteristics of different ontologies is a crucial and challenging area of future research.

Table 5.4: Accuracies achieved by the ElEm and EmEL^{++} embeddings in terms of geomteric interpretation of the classes in different ontologies.

	Training		*Validation*		*Testing*	
	ElEm	EmEL^{++}	ElEm	EmEL^{++}	ElEm	EmEL^{++}
NCI	0.1878	**0.3345**	0.1873	**0.3218**	0.1059	**0.1348**
GALEN	0.2734	**0.6431**	0.2687	**0.6420**	0.2030	**0.5337**
GO	0.4527	**0.5925**	0.4484	**0.5956**	0.3476	**0.4438**
FMA	0.0204	**0.1213**	0.0212	**0.1209**	0.0041	**0.0136**
ANATOMY	0.0865	**0.4780**	0.0892	**0.4795**	0.0692	**0.2156**
SNOMED CT	0.2455	**0.5547**	0.2447	**0.5534**	0.1835	**0.3410**

Next, we compare the ElEm and EmEL^{++} embedding models in terms of their capability to retain the underlying characteristics of the ontology in the vector space. Recall that both the models map the classes in an ontology to n-balls in the vector space. Further, the mapping is such that the n-ball of a super-class subsumes the n-balls of its sub-classes. Thus, for a test instance $C \sqsubseteq D$, we check that the n-ball of class C lies inside the n-ball of class D in the vector space. Note that since we have the centers and radii of the corresponding n-balls, this can be checked easily. Table 5.4 presents the training, validation, and testing accuracy obtained for the two embedding models for this task. We report accuracy values, i.e., the fraction of instances where the subsumption relation between the classes was maintained in the vector space. Note that accuracy is a much stricter criterion for even if the sub-class n-ball is slightly outside the n-ball of the superclass, it will be considered a failure. We observe from Table 5.4 that EmEL^{++} outperforms the ElEm embeddings for all the datasets and across all settings. This indicates that EmEL^{++} embeddings are better at preserving the class relationships in the mapped vector space than ElEm embeddings. Also, the difference in accuracy values obtained with EmEL^{++} and ElEm embeddings indicate that EmEL^{++} is superior in terms of geometric representation.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We proposed EmEL^{++}, an ontology embedding model for $\mathcal{EL}^{++}$ ontologies. EmEL^{++} builds upon and extends the previously proposed ElEm embeddings by incorporating constructs focusing on role inclusions and role chains and offers a more complete coverage of $\mathcal{EL}^{++}$ constructs. Experiments with six different ontologies showed that EmEL^{++} outperforms traditional knowledge base embeddings on the subsumption reasoning task. Further, when compared with ElEm embeddings, it is able to better preserve the underlying semantics of the ontologies in the vector space along with maintaining the geometric properties associated with classes. We have also shown how to perform the subsumption reasoning task in a vector space, which is an $O(n)$ operation in the worst case. We believe this is an important capability and it offers exciting directions for future work. Developing models for embedding more complex ontologies, and non-tractable description logics in the vector space can allow us to build more efficient reasoners. Moreover, the work can be oriented towards checking inconsistencies in an ontology. Further, as part of future directions exploring different relational models as the basis of our loss functions might give new insights into impact on geometric properties of classes and relations in vector space. Moreover, evaluating the embeddings on the basis of inferences drawn from reasoners can help evaluate their quality better.

References

[1] Ali, M., Jabeen, H., Hoyt, C.T., Lehmann, J.: The KEEN Universe. In: International Semantic Web Conference. pp. 3–18. Springer (2019)

[2] Baader, F., Brandt, S., Lutz, C.: Pushing the EL Envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany (2005), see http://lat.inf.tu-dresden.de/research/reports.html.

[3] Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: IJCAI. vol. 5, pp. 364–369 (2005)

[4] Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D., et al.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)

[5] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)

[6] Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)

[7] Consortium, G.O.: The Gene Ontology (GO) database and informatics resource. Nucleic acids research 32(suppl_1), D258–D261 (2004)

[8] Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W.: The complexity of concept languages. Information and Computation 134(1), 1–58 (1997)

[9] Donnelly, K.: SNOMED-CT: The advanced terminology and coding system for ehealth. Studies in health technology and informatics 121, 279 (2006)

[10] Garg, D., Ikbal, S., Srivastava, S.K., Vishwakarma, H., Karanam, H., Subramaniam, L.V.: Quantum Embedding of Knowledge for Reasoning. In: Advances in Neural Information Processing Systems. pp. 5595–5605 (2019)

[11] Golbeck, J., Fragoso, G., Hartel, F., Hendler, J., Oberthaler, J., Parsia, B.: The National cancer Institute's thesaurus and ontology. Journal of Web Semantics First Look 1_1_4 (2003)

[12] Graua, B.C., Horrocksa, I., Motika, B., Parsiab, B., Patel-Schneiderc, P., Sattlerb, U.: Web semantics: Science, Services and Agents on the world wide web. Web Semantics: Science, Services and Agents on the World Wide Web 6, 309–322 (2008)

[13] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)

[14] Guarino, N., Oberle, D., Staab, S.: What is an ontology? In: Handbook on ontologies, pp. 1–17. Springer (2009)

[15] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J.E.L., Kirrane, S., Neumaier, S., Polleres, A., et al.: Knowledge graphs. arXiv preprint arXiv:2003.02320 (2020)

[16] Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014) (2014)

[17] Kulmanov, M., Liu-Wei, W., Yan, Y., Hoehndorf, R.: EL embeddings: geometric construction of models for the description logic el++. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. pp. 6103–6109. AAAI Press (2019)

[18] Levesque, H.J.: Knowledge representation and reasoning. Annual review of computer science 1(1), 255–287 (1986)

[19] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology 58(7), 1019–1031 (2007)

[20] Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI conference on artificial intelligence (2015)

[21] Lü, L., Zhou, T.: Link prediction in complex networks: A survey. Physica A: statistical mechanics and its applications 390(6), 1150–1170 (2011)

[22] Mendez, J.: jcel: A Modular Rule-based Reasoner. In: ORE (2012)

[23] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[24] Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies. pp. 746–751 (2013)

[25] Misra, V., Bhatia, S.: Bernoulli embeddings for graphs. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[26] Mungall, C.J., Torniai, C., Gkoutos, G.V., Lewis, S.E., Haendel, M.A.: Uberon, an integrative multi-species anatomy ontology. Genome biology 13(1), R5 (2012)

[27] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Icml. vol. 11, pp. 809–816 (2011)

[28] Rector, A., Rogers, J., Pole, P.: The GALEN high level ontology (1996)

[29] Rector, A., Horrocks, I.: Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In: Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97), Stanford, CA. pp. 321–325 (1997)

[30] Ristoski, P., Paulheim, H.: Rdf2vec: Rdf graph embeddings for data mining. In: International Semantic Web Conference. pp. 498–514. Springer (2016)

[31] Rosse, C., Mejino, J.L.: The foundational model of anatomy ontology. In: Anatomy Ontologies for Bioinformatics, pp. 59–117. Springer (2008)

[32] Smaili, F.Z., Gao, X., Hoehndorf, R.: Onto2vec: Joint vector-based representation of biological entities and their ontology-based annotations. Bioinformatics 34(13), i52–i60 (2018)

[33] Spackman, K.A.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with snomed rt. heart 13, 14 (2000)

[34] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML) (2016)

[35] Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Twenty-Eighth AAAI conference on artificial intelligence (2014)

[36] Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)