EmbedS: Scalable, Ontology-aware Graph Embeddings

Gonzalo I. Diaz
Department of Computer Science
University of Oxford
gonzalo.diaz@cs.ox.ac.uk

Achille Fokoue
IBM T. J. Watson Research Center
achille@us.ibm.com

Mohammad Sadoghi
Exploratory Systems Lab
University of California, Davis
msadoghi@ucdavis.edu

ABSTRACT

While the growing corpus of knowledge is now being encoded in the form of *knowledge graphs* with rich semantics, the current graph embedding models do not incorporate ontology information into the modeling. We propose a scalable and ontology-aware graph embedding model, EmbedS, which is able to capture RDFS ontological assertions. EmbedS models entities, classes, and properties differently in an RDF graph, allowing for a geometrical interpretation of ontology assertions such as type inclusion, subclassing, and alike.

1 INTRODUCTION

A growing corpus of knowledge is being encoded in the form of *knowledge graphs*, i.e. in the form of (s, p, o) triples which represent simple subject-predicate-object sentences. These triple-based formats consist of binary relational data, where an (s, o) pair is effectively declared to be related by the p binary relation. Although more complex–or higher arity–information is difficult to express in this way, the simple syntax enjoyed by these triple-based data models has proved highly useful for a wide range of applications and has thus been widely adopted. Today, the linked open data cloud consists of hundreds of interlinked datasets, including a few very large knowledge graphs such as Freebase and DBpedia, which contain billions of triples and millions of entities.

Modern knowledge graphs are used in applications such as web search, where graph data provides structured data that complements hyperlink answers, artificial intelligence question answering systems such as Watson and voice-based assistants, and semantic web query engines that run powerful declarative query languages such as SPARQL. However, there are still important issues to be resolved. Even the largest knowledge graphs are extremely incomplete (i.e. many true facts have yet to be encoded as triples) and prone to errors (often triples encoding incorrect facts are included) [12]. The main tasks of link prediction (which consists of predicting new triples) and triplet classification (which seeks to assign a probability that a certain triple–be it new or existing–is true or not) look to address these shortcomings. In this context, there is renewed interest in machine learning over (binary) relational data. The techniques used vary [9] from rule-based learning [5] to tensor factorisation, neural network-based approaches, etc. In this work we focus on latent feature-based techniques, which are also known as graph embeddings.

Graph embeddings correspond to latent feature statistical relational learning models in that they assume the existence of a set of n latent features–or random variables–that account for the predictions we desire (triplet classification or otherwise). As such, these latent features provide machine learning-friendly representations of graph data, which can then be used as input to further machine learning tools such as neural networks or logistic regression for classification. The input to a graph embedding model, as with other statistical relational learning techniques, is a knowledge graph, and the output is a mapping which associates to each entity in the graph the set of n real-valued latent features. The name *graph embedding* refers to the geometrical interpretation given to the obtained latent features: the entities are interpreted as points in an n-dimensional real coordinate space, and thus are said to have been embedded into said space. On the other hand, the relations of the knowledge graph are varyingly represented as translational vectors [3], matrix transformations [11], etc.

Graph embedding models are usually trained via the minimisation of a global cost function, which can be expressed as the sum of a cost assigned to each triple in the knowledge graph. The minimisation itself is achieved via stochastic gradient descent or similar methods. One of the greatest advantages of graph embedding models, and especially translational models (loosely defined as those models which represent relations as one or more translational vectors, as opposed to transformation matrices), is that they are able to perform efficiently for very large graphs and with high accuracy. Current graph embedding models achieve high performance on basic learning tasks such as link prediction–ranking the correct entity among the top ten candidates 95% of the time [10].

Despite the positive scenario described above, one of the greatest limitations of current graph embedding models is that they consider a very simple model of the data: they consider a knowledge graph to be a set of triples, where each triple mentions two entities and one relation, with no special semantics for any particular triple. In contrast, standard knowledge graphs are often accompanied by rich ontological information which encodes a wealth of metadata, including type hierarchies and other constraints. Current graph models are entirely agnostic to such metadata, and thus ontological triples are usually manually removed before training (or, if they are included, simply interpreted as plain data triples). One of the consequences of this is that current graph embedding models do not directly incorporate constraints that humans would consider obvious (e.g. knowing that a human cannot be friends with a building). A huge potential exists in the rich ontologies that inform modern knowledge graphs, and the objective of this work is to explore ways in which such ontologies can become first class citizens of graph embedding models.

Specifically, we explore the problem of graph embedding on ontology-rich knowledge graphs, where the ontology is specified in a standard language such as RDFS (more expressive languages such as OWL2 have been developed, but will not be considered in this study). Drawing on the geometrical interpretation that graph embeddings give to their latent features (namely, that entities are embedded as points in a real coordinate space), we explore the case where RDFS classes are correspondingly modelled as sets of points in the same coordinate space, and relations are embedded as sets of pairs of points. This generalisation of the basic geometrical interpretation allows for a natural expression of ontological constraints in the global cost function. The result is a model we name EmbedS, which is able to model RDFS ontological constraints as first-class citizens. Along with providing the precise definitions for the EmbedS cost function, we provide experimental results that show EmbedS to be comparable to state-of-the-art graph embedding techniques when measured on traditional benchmark knowledge graphs, while performing well on a new ontology-rich dataset we have prepared for the purposes of studying the new enriched geometrical interpretation that EmbedS provides.

The preliminary results presented in this paper showcase the potential of extending current graph embedding research to include ontological information, and will hopefully encourage further development of the area. This paper is organised as follows. In Section 2 we introduce necessary mathematical notation and preliminary definitions. In Section 3 we introduce the EmbedS model, its global cost function, and the geometrical interpretation induced on embedded entities and relations. In Section 4 we explain the experimental setting and evaluation metrics, including a discussion of an ontology-rich benchmark dataset we have prepared for testing the EmbedS model. Finally, in Section 5 we present our conclusions and suggest future avenues of work.

2 PRELIMINARIES AND RELATED WORK

Graph embedding models are usually defined on a *knowledge graph*, which is essentially a set of triples, similar to the RDF data model, but lacking specialised features of the latter, such as the precise definition of IRIs, literals, and the semantics associated with certain keyword IRIs. In this section we will introduce necessary definitions, noting the similarities and discrepancies with the RDF data model. We will then introduce needed concepts such as graph embeddings, and finally we will discuss related work.

Let E and P be two mutually disjoint and countably infinite sets of *entities* and *relations*, respectively. A knowledge graph is then defined as a finite set of triples of the form $(s, p, o) \in \mathsf{E} \times \mathsf{P} \times \mathsf{E}$. As such, a knowledge graph can be interpreted as a directed graph with labelled edges, or as a set of binary relations. As this work will seek to allow graph embeddings to operate on more expressive graph data, it is important to note the differences with the full RDF data model. RDF also considers an infinitely countable set of *literals* L, disjoint from E and P, and note that RDF triples are drawn from the more general set $(\mathsf{E} \cup \mathsf{P}) \times (\mathsf{E} \cup \mathsf{P}) \cup (\mathsf{E} \cup \mathsf{P} \cup \mathsf{L})$. For example, note that RDF allows relations to be mentioned in the subject position of a triple, thus blurring the distinction that exists between relations as edge labels and nodes of the graph. This is generally disallowed in the traditional definition of knowledge graph.

The base RDF data model described provides very little semantics other than the basic interpretation of the triple as a fact. To enrich an RDF dataset, several ontology languages have been developed, of which RDFS is one of the simplest. RDFS defines the following core keyword IRIs: `rdf:type`, `rdfs:subClassOf`, `rdfs:sub PropertyOf`, `rdfs:domain`, and `rdfs:range` (abbreviated `type`, `sc`, `sp`, `dom`, `range`, resp.), which are assigned special semantics in order for richer knowledge—including basic type systems—to be encoded into RDF format[1].

In what follows, we will consider, in addition to the sets E and P, a set C of *classes*, also infinitely countable and disjoint from the previous two. Furthermore, we define five special relations $\mathtt{type}, \mathtt{sc}, \mathtt{sp}, \mathtt{dom}, \mathtt{range} \in \mathsf{P}$. We will only consider triples $t = (s, p, o)$ for which one of the following hold:

- $(s, p, o) \in \mathsf{E} \times \mathsf{P} \times \mathsf{E}$ (called data triples),
- $(s, p, o) \in \mathsf{E} \times \{\mathtt{type}\} \times \mathsf{C}$ (called type triples),
- $(s, p, o) \in \mathsf{C} \times \{\mathtt{sc}\} \times \mathsf{C}$ (called subclass triples),
- $(s, p, o) \in \mathsf{P} \times \{\mathtt{sp}\} \times \mathsf{P}$ (called subproperty triples),
- $(s, p, o) \in \mathsf{P} \times \{\mathtt{dom}\} \times \mathsf{C}$ (called domain triples),
- $(s, p, o) \in \mathsf{P} \times \{\mathtt{range}\} \times \mathsf{C}$ (called range triples).

An RDF data graph (or, simply, a graph) D is a finite set of triples such that for every triple $t \in D$, t is a data triple or a type triple. An RDFS ontology (or, simply, an ontology) S is a finite set of triples such that for every triple $t \in S$, t is not a data triple or a type triple.

Example 2.1. Consider the graph $D = \{(\mathtt{anne}, \mathtt{type}, \mathtt{Woman}), (\mathtt{jo\ hn}, \mathtt{type}, \mathtt{Man}), (\mathtt{john}, \mathtt{knows}, \mathtt{anne})\}$. This data about people and their relationships can be enriched with the ontology $S = \{(\mathtt{Woman}, \mathtt{sc}, \mathtt{Person}), (\mathtt{Man}, \mathtt{sc}, \mathtt{Person})\}$ The full dataset is then $I = D \cup S$. Notice that the semantics of RDFS allow us to conclude facts which are not explicitly included in the dataset, such as $(\mathtt{anne}, \mathtt{type}, \mathtt{Person})$ and $(\mathtt{john}, \mathtt{type}, \mathtt{Person})$. □

Crucially, the semantics of RDFS allow for inferencing new triples using a series of inference rules. For example, the following is an inference rule for RDFS: $(x, \mathtt{type}, c), (c, \mathtt{sc}, b) \to (x, \mathtt{type}, b)$, which is read as follows: given a dataset $I = D \cup S$ which contains two triples of the form $(\mathtt{a}, \mathtt{type}, \mathtt{C})$ and $(\mathtt{C}, \mathtt{sc}, \mathtt{B})$, for any entity $a \in E$ and classes B, C, the triple $(a, \mathtt{type}, B)$ may be *inferred* to hold [1]. In this context, inferring a triple to hold would cause a query engine to return the inferred triple as an answer to a query.

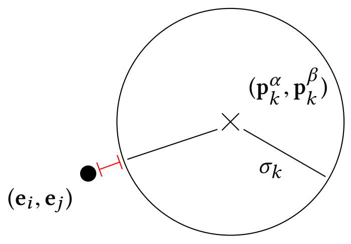
Graph embedding—and statistical relational learning—models use varying techniques in order to obtain machine-usable representations of knowledge graphs. In general, however, the main problem—that of knowledge base completion—gives a common direction to these techniques: constructing statistical models of the data that allow for link prediction (e.g. given an incomplete fact $(\mathtt{T\ arantino}, \mathtt{inspiredBy}, ?)$ return the entity that would complete the triple) and triple classification (assign a probability that a triple is true).

The most expressive models involve tensor or matrix factorisation techniques, although these are also the models with the highest complexity, measured in the number of parameters that must be trained. A well-known example of this is RESCAL [11], which explains triples using pairwise interactions of the latent features of entities. Thus, the *cost* associated to a triple x_{ijk} has the form:

$$\mathrm{cost}(x_{ijk}) = \mathbf{e}_i^\top \mathbf{W}_k \mathbf{e}_j.$$

Other highly expressive models use techniques such as matrix factorisation, neural tensor networks, and multilayer perceptrons [6, 8]. The latter, also known as word2vec, was strictly a word embedding model, but had as an interesting and unintended consequence a translational property among the latent representation of words: simple binary relations between words could be captured when interpreting the latent representation—*embedding*—of the relation as a *translational vector*. For example, after training on a textual corpus, researchers found that incomplete sentences such as $(\mathtt{Madrid}, \mathtt{capitalof}, \mathtt{Spain})$ had the property that the vector $\mathbf{e}_{\mathtt{Madrid}} + \mathbf{e}_{\mathtt{capitalof}}$ was nearest to $\mathbf{e}_{\mathtt{Spain}}$.

[1]The prefixes used are themselves abbreviations. Actually, `rdf:` expands to `http://www.w3.org/1999/02/22-rdf-syntax-ns#` and `rdfs:` expands to `http://www.w3.org/ 2000/01/rdf-schema#`.



$(\mathbf{p}_k^\alpha, \mathbf{p}_k^\beta)$
σ_k
$(\mathbf{e}_i, \mathbf{e}_j)$

Figure 1: Cost of a triple $(e_i, p_k, e_j) \in I$. The circle represents a $2n$-sphere of radius σ_k centered at $(\mathbf{p}_k^\alpha, \mathbf{p}_k^\beta)$. The pair (e_i, e_j) is embedded as the $2n$-dimensional point $(\mathbf{e}_i, \mathbf{e}_j)$. The (red) error line shows the cost.

The previous translational property spawned renewed interest in distance-based models that incorporated a *geometrical interpretation* for latent representations. The first of these was TransE [3], and was quickly followed by refinements such as TransH [13], and TransR [7]. While less expressive, translation-based models prove more scalable, as their model complexity is lower and a simpler cost structure allows for more efficient training.

The model proposed in this work draws from research in translation-based models. Our problem scenario focused on ontology-rich knowledge graphs, which contain specific semantics for keyword triples. This metadata has not been considered in previous work, to the best of our knowledge. The idea of considering ontological information in training statistical models, however, has been considered. In [4], type constraints are considered by removing type-constraint-violating triples, improving training speed and link predicition performance. They do not incorporate ontological information into the model as a first class citizen, however, and neither does the geometrical interpretation of the model adapt to reflect the presence of this metadata.

3 MODEL

Consider an RDF dataset $I = D \cup S$, and let $E_I \subseteq E$, $C_I \subseteq C$, and $P_I \subseteq P$ be the sets of entities, classes, and properties that appear in I, respectively. Define for each entity $e_i \in E_I$ an n-dimensional vector of parameters (i.e. variables) $\mathbf{e}_i = (e_{i1}, \ldots, e_{in})$; for each class $c_i \in C_I$ define an n-dimensional vector of parameters $\mathbf{c}_i = (c_{i1}, \ldots, c_{in})$ and a parameter ρ_i; and for each property $p_i \in P_I$ define two n-dimensional vectors of parameters $\mathbf{p}_i^\alpha = (p_{i1}^\alpha, \ldots, p_{in}^\alpha)$ and $\mathbf{p}_i^\beta = (p_{i1}^\beta, \ldots, p_{in}^\beta)$ and a parameter σ_i. We have thus defined a total of $|E_I| \cdot n + |C_I| \cdot (n+1) + |P_I| \cdot (2n+1)$ parameters for our model.

For what follows, we first define a distance function **dist** which assigns to every pair of n-dimensional vectors $\mathbf{x} = (x_1, \ldots, x_n)$, and $\mathbf{y} = (y_1, \ldots, y_n)$ a non-negative real value $\mathbf{dist}(\mathbf{x}, \mathbf{y})$, with the standard distance function properties[2]. In this paper, we choose to set $\mathbf{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sum_{i=1}^{i=n} (x_i - y_i)^2$, that is, the L_2 norm of $\mathbf{x} - \mathbf{y}$. We also define an activation function **act**, for which we choose the rectifier function, $\mathbf{act}(x) = \max(0, x)$.

Assuming that $|E_I| = N_E$, $|C_I| = N_C$, and $|P_I| = N_P$, the cost $\mathcal{L}$ will be a function of all the variables previously defined:

$$\mathcal{L} = \mathcal{L}(\, \mathbf{e}_1, \ldots, \mathbf{e}_{N_E}; \mathbf{c}_1, \ldots, \mathbf{c}_{N_C}, \rho_1, \ldots, \rho_{N_C}; \mathbf{p}_1^\alpha, \ldots, \mathbf{p}_{N_P}^\alpha, \mathbf{p}_1^\beta, \ldots, \mathbf{p}_{N_P}^\beta, \sigma_1, \ldots, \sigma_{N_P} \,).$$

We now define the precise form of the cost function $\mathcal{L}$. For the entire dataset, define $\mathcal{L}^I = \sum_{t \in I} \mathcal{L}_t$, where the cost of each

[2](a) $\mathbf{dist}(\mathbf{x}, \mathbf{y}) \geq 0$, (b) $\mathbf{dist}(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$, (c) $\mathbf{dist}(\mathbf{x}, \mathbf{y}) = \mathbf{dist}(\mathbf{y}, \mathbf{x})$, and (d) $\mathbf{dist}(\mathbf{x}, \mathbf{z}) \leq \mathbf{dist}(\mathbf{x}, \mathbf{y}) + \mathbf{dist}(\mathbf{y}, \mathbf{z})$.

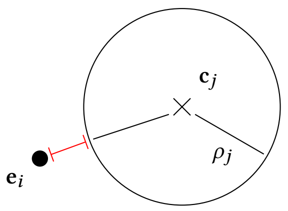


Figure 2: Cost (red error line) of $t = (e_i, \texttt{type}, c_j)$, where class c_j is embedded as an n-sphere at $\mathbf{c}_j$ with radius ρ_j.

triple $t = (e_i, p_k, e_j) \in I$ (note that $e_i, e_j \in E_I \cup C_I \cup P_I$ and $p_k \in P_I$), is defined as follows:

$$\mathcal{L}_t = \mathbf{act}\Big(\mathbf{dist}(\mathbf{e}_i, \mathbf{p}_k^\alpha) + \mathbf{dist}(\mathbf{e}_j, \mathbf{p}_k^\beta) - \sigma_k\Big).$$

The geometrical interpretation of this cost is provided in Section 3.1, and is visualized in Figure 1. Next, we define a cost term for each possible RDFS assertion. For each $t \in S$:

(1) If $t = (e_i, \texttt{type}, c_j) \in S$, where $e_i \in E_I$ and $c_j \in C_I$, define:
$$\mathcal{L}_t^S = \mathbf{act}\Big(\mathbf{dist}(\mathbf{e}_i, \mathbf{c}_j) - \rho_j\Big),$$
(2) If $t = (c_i, \texttt{sc}, c_j) \in S$, where $c_i, c_j \in C_I$, define:
$$\mathcal{L}_t^S = \mathbf{act}\Big(\mathbf{dist}(\mathbf{c}_i, \mathbf{c}_j) - (\rho_j - \rho_i)\Big),$$
(3) If $t = (p_i, \texttt{sp}, p_j) \in S$, where $p_i, p_j \in P_I$, define:
$$\mathcal{L}_t^S = \mathbf{act}\Big(\mathbf{dist}(\mathbf{p}_i^\alpha, \mathbf{p}_j^\alpha) + \mathbf{dist}(\mathbf{p}_i^\beta, \mathbf{p}_j^\beta) - (\sigma_j - \sigma_i)\Big),$$
(4) If $t = (p_i, \text{dom}, c_j) \in S$, where $p_i \in P_I$ and $c_j \in C_I$, define:
$$\mathcal{L}_t^S = \mathbf{act}\Big(\mathbf{dist}(\mathbf{p}_i^\alpha, \mathbf{c}_j) - \sigma_i\Big),$$
(5) If $t = (p_i, \texttt{range}, c_j) \in S$, where $p_i \in P_I$ and $c_j \in C_I$, define:
$$\mathcal{L}_t^S = \mathbf{act}\Big(\mathbf{dist}(\mathbf{p}_i^\beta, \mathbf{c}_j) - \sigma_i\Big).$$

Finally, we sum, for every triple $t \in S$, the cost of t depending on the RDFS relation it mentions. In that way, we define the cost term $\mathcal{L}^S = \sum_{t \in S} \mathcal{L}_t^S$, and thus define the final cost to be $\mathcal{L} = \mathcal{L}^I + \mathcal{L}^S$.

3.1 Geometrical interpretation

We now give a geometrical interpretation to the model definition. By embedding entities as single n-dimensional vectors we are modelling them as *points* in the n-dimensional euclidean space. Classes, on the other hand, are modelled as *regions* of the euclidean space, being embedded as a vector and a radius, representing n-spheres. This allows for the following geometrical interpretation: if an entity is embedded within the region defined by the embedding of a class, then it is interpreted to be of that type, and vice-versa (see Figure 2). Finally, properties, insofar as they represent binary relations, are modelled as $2n$-spheres which constitute a set of *pairs of points.* Thus, each relation $p_k \in P_I$ has an embedding which consists of two n-dimensional vectors $\mathbf{p}_k^\alpha$ and $\mathbf{p}_k^\beta$ and a radius ρ_k. The corresponding geometrical interpretation is analogous to the previous case: in $2n$-space, a pair (e_i, e_j) is interpreted to be related by a relation p_k if the $2n$-point $(\mathbf{e}_i, \mathbf{e}_j)$ is in the region defined by the $2n$-sphere centered at $(\mathbf{p}_k^\alpha, \mathbf{p}^\beta)_k)$ with radius σ_k (see Figure 1).

The main advantage conferred by this ontology-aware geometrical interpretation is that RDFS classes and ontological assertions are now first-class citizens of the model. By modelling classes as regions of the euclidean space, for example, certain properties are obtained for free, such as type containment transitivity: if after training the entity `aturing` is (correctly) embedded within the class `Researcher`, and said class is (correctly) embedded fully contained within the region corresponding to class `Person`, then the transitive fact that `aturing` is a `Person` will be provided for free. In this way, the embedding space will presumably encode ontological information geometrically.

4 EXPERIMENTAL EVALUATION

In this section we provide preliminary experimental results showing that EmbedS can perform at state-of-the-art levels on a standard benchmark dataset, while providing a new complementary triple classification method based on the geometrical interpretation which shows encouraging results. An exhaustive experimental evaluation will be left for future work. We use the following datasets for experimental evaluations:

wn18 Dataset extracted from WordNet. This dataset consists of 151,442 triples, 40,943 entities, and 18 relations.

dbpedia32k RDF dataset extracted from DBpedia, consisting of 340,827 triples, 32,657 entities, and 296 relations.

The first dataset has become a standard benchmark in the graph embedding field [3], and allows for an apples to apples comparison between our model and existing work. However, EmbedS has been designed for a fundamentally different problem setting: that of embedding in *ontology-rich* RDF data. In order to test the performance of our model, we have prepared a dataset, named 'dbpedia32k' which includes an RDFS ontology.

We now describe the construction of the dbpedia32k dataset. It initially draws from three distinct downloads, which are freely available: the 'DBpedia Ontology' file, which contains ontology triples, the 'Instance Types' file, which contains data triples of the form $(a, \texttt{type}, C)$ for some entity a and some class C, and the 'Mappingbased Objects' file, which contains general data triples, representing facts on entities present in Wikipedia articles. From the Mappingbased Objects file first define a relation to be useful if it appears in at least 1000 triples. We define an entity to be useful if it is mentioned at least 10 times in the file. A uniform sample of useful entities is built, keeping only triples which mention useful relations and these sampled entities only. Finally, we recalculate useful entities (in the filtered dataset) and remove triples which mention non-useful entities. We thus obtain the final set of Mappingbased Objects triples to be used. We complete the dataset by extracting, from Instance Types, triples $(a, \texttt{type}, C)$ where a is mentioned in the previous dataset, and similarly we extract relevant ontology triples from DBpedia Ontology. The resulting complete dataset is split uniformly into three subsets for training, validation, and testing, with proportions 0.8, 0.1, and 0.1, respectively.

Selection of hyperparameters of the model is achieved via random search. Although more sophisticated methods have been proposed, random search is competitive with these systems [2] and thus serves our purposes. For each dataset-model choice (e.g. dbpedia32k with EmbedS), a suitable bounding box for the hyperparameters of the model is chosen, and training is performed over 500 epochs for 1,000 different random hyperparameter values.

To measure the performance of the model, we use the standard *filtered hits@10* and *filtered mean reciprocal rank* metrics. The final model selected after training is that which maximizes the estimated mean reciprocal rank for the validation dataset.

As EmbedS allows for a geometrical interpretation of triples, we also evaluate a triple classification performance. For each triple in the dataset, and an equal amount of randomly generated false triples (i.e. triples not in the dataset), if $t = (e_i, p_k, e_j)$, the binary classification consists in asking whether the $2n$-dimensional point $(\mathbf{e}_i, \mathbf{e}_j)$ is contained in the sphere centered at $(\mathbf{p}_k^\alpha, \mathbf{p}_k^\beta)$ with radius σ_k or not. Precision and recall values are obtained for this test.

EmbedS was trained on the wn18 dataset, optimizing for best validation (filtered) hits@10 value, obtaining 94.9%, which is comparable to state-of-the-art models such as HolE [10]. HolE is clearly superior in the mean reciprocal rank metric, however, with a value of 0.938, compared to 0.560 for EmbedS. It must be noted, though, that these values correspond to *harmonic* mean ranks of 1.07 for HolE and 1.79 for EmbedS. If EmbedS is now instructed to optimise the geometrical interpretation, we achieve a precision of 84.2% and a recall of 83.9%, corresponding to an f-measure of 84.0%.

On the dbpedia_v2 dataset, we find that EmbedS achieves a performance on hits@10 of 22.7% and a mean reciprocal rank of 0.133 (corresponding to a harmonic mean rank of 7.52). TransE, on the other hand, performs at 11.6% hits@10 and 0.054 mean reciprocal rank (corresponding to a harmonic mean rank of 18.52).

5 CONCLUSIONS

In this paper we study the new problem of training graph embeddings on ontology-rich datasets. We propose a model which considers RDFS classes and other ontological information as first-class citizens, providing a geometrical interpretation for triples and for ontology assertions. Preliminary experimental results show that the model can perform at state-of-the-art levels on standard benchmark datasets, while on ontology-rich datasets it is also able to provide an alternative form of triple classification which takes advantage of the geometrical interpretation. An exhaustive experimental evaluation is required to be able to fully understand the limitations of this new model, although the encouraging results shown seem to indicate that incorporating ontological information into graph embedding models can potentially open a new avenue of research.

REFERENCES

[1] M. Arenas, C. Gutierrez, and J. Pérez. Foundations of RDF databases. In *Reasoning Web'09*.

[2] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *JMLR'12*.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *NIPS'13*.

[4] Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. Typed Tensor Decomposition of Knowledge Bases for Relation Extraction. In *EMNLP'14*.

[5] L. A. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In D. Schwabe, V. A. F. Almeida, H. Glaser, R. A. Baeza-Yates, and S. B. Moon, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 413–422. International World Wide Web Conferences Steering Committee / ACM, 2013.

[6] Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. Link Prediction in Multi-relational Graphs using Additive Models. In *SeRSy'12*.

[7] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI'15*.

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR'13*.

[9] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE'16*.

[10] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic Embeddings of Knowledge Graphs. In *AAAI'16*.

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. In *ICML'11*.

[12] M. Sadoghi, K. Srinivas, O. Hassanzadeh, Y. Chang, M. Canim, A. Fokoue, and Y. A. Feldman. Self-curating databases. In *EDBT'16*.

[13] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI'14*.