Scalable Learning of Entity and Predicate Embeddings for Knowledge Graph Completion

Pasquale Minervini, Nicola Fanizzi, Claudia d'Amato, Floriana Esposito
Department of Computer Science - Università degli Studi di Bari Aldo Moro, Italy
{firstname.lastname}@uniba.it

***Abstract*—*Knowledge Graphs* (KGs) are a widely used formalism for representing knowledge in the Web of Data. We focus on the problem of *link prediction*, i.e. predicting missing links in large knowledge graphs, so to discover new facts about the world. Representation learning models that embed entities and relation types in continuous vector spaces achieve state-of-the-art results on this problem, while showing the potential to scale to very large KGs. A limiting factor is that the process of learning the optimal embeddings can be very computationally expensive, and may even require days for large KGs. In this work, we propose a principled method for reducing the training time by an order of magnitude, while learning more accurate link prediction models. Furthermore, we employ the proposed method for training a set of novel, scalable models, with high predictive accuracy. Our extensive evaluations show significant improvements over state-of-the-art link prediction methods on several datasets.**

I. Introduction

Knowledge Graphs (KGs) are graph-structured Knowledge Bases (KBs), where factual knowledge about the world is represented in the form of relationships between entities. They are widely used for representing relational knowledge in a variety of domains, such as citation networks and protein interaction networks. An example of their widespread adoption is the *Linked Open Data* (LOD) Cloud, a set of interlinked KGs such as Freebase [1] and WordNet [2]. As of April 2014, the LOD Cloud was composed by 1,091 interlinked KBs, globally describing over 8×10^6 entities and 188×10^6 relationships holding between them [1].

Despite their large size, KGs are still largely incomplete. For example consider Freebase [2], a core element in the Google Knowledge Vault project [3]: 71% of the persons described in Freebase have no known place of birth, 75% of them have no known nationality, and the coverage for less frequent predicates (relation types) can be even lower [3].

In this work we focus on the problem of automatically *completing missing links* in large KGs, so to discover new facts about the world. In the literature, this problem is referred to as *link prediction* or *knowledge graph completion*, and has received a considerable attention over the last years [4].

Recently, *representation learning models* [5] such as the *Translating Embeddings* model (TransE) [6] have been used for achieving new state-of-the-art link prediction results on large and Web-scale KGs. Such models learn a unique distributed representation for each entity and predicate in the KG: each entity is represented by a low-dimensional *embedding vector*, and each predicate is represented as an *operation* in the embedding vector space. These models are closely related to *distributional semantic* natural language processing models such as word2vec [7], which represent each word in a corpus of documents as a low-dimensional embedding vector. We refer to these models as *embedding models*, and to the learned distributed representations as *embeddings*. The embeddings of all entities and predicates in the KG are learned jointly: the learning process consists in minimizing a global loss function considering the whole KG, by back-propagating the loss to the embeddings. As a consequence, the learned entity and predicate embeddings retain global, structural information about the whole KG, and can be used to serve several kinds of applications: in *link prediction*, the confidence of each candidate edge can be measured as a function of the embeddings of its source entity, its target entity, and its predicate.

A major limitation of the models discussed so far is that the learning procedure, which consists in learning the distributed representations of all entities and predicates in the KG, can be very time-consuming: for instance, it may even require days of computations for large KGs [8]. As a solution to this problem, in this work we propose a novel, principled method for significantly reducing the learning time in KG embedding models. Furthermore, we employ the proposed method for training a variety of novel, more accurate models, achieving new state-of-the-art results on several link prediction tasks.

II. Basics

RDF Graphs

The most widely used formalism for representing knowledge graphs is the W3C *Resource Description Framework* (RDF) [3], a recommended standard for representing knowledge on the Web. An RDF KB, also referred to as *RDF graph*, is a set of *RDF triples* in the form $\langle s, p, o \rangle$, where s, p and o respectively denote the *subject*, the *predicate* and the *object* of the triple: s and o are *entities*, and p is a relation type. Each triple $\langle s, p, o \rangle$ describes a statement, which is interpreted as "*A relationship p holds between entities s and o*".

Example 2.1 (Shakespeare): The statement "*William Shakespeare is an author who wrote Othello and the tragedy Hamlet*" can be expressed by the following RDF triples:

```
⟨Shakespeare,   profession,   Author⟩
⟨Shakespeare,   author,       Hamlet⟩
⟨Shakespeare,   author,       Othello⟩
⟨Hamlet,        genre,        Tragedy⟩
```

[1]State of the LOD Cloud 2014: http://lod-cloud.net/
[2]Publicly available at https://developers.google.com/freebase/data
[3]http://www.w3.org/TR/rdf11-concepts/

TABLE I: Scoring functions used by several link prediction models, and the corresponding number of parameters: $n_e = |\mathcal{E}_G|$ and $n_r = |\mathcal{R}_G|$ respectively denote the number of entities and predicates in G, and $k, d \in \mathbb{N}$ are user-defined hyper-parameters.

Model	**Scoring function** $f_p(\mathbf{e}_s, \mathbf{e}_o)$	**Number of Parameters**
Unstructured [9]	$-\|\mathbf{e}_s - \mathbf{e}_o\|_2^2, \quad \mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$	$\mathcal{O}(n_e k)$
Translating Embeddings (TransE) [6]	$-\|(\mathbf{e}_s + \mathbf{e}_p) - \mathbf{e}_o\|_{\{1,2\}}, \quad \mathbf{e}_p \in \mathbb{R}^k$	$\mathcal{O}(n_e k + n_r k)$
Structured Embeddings (SE) [10]	$-\|\mathbf{W}_{p,1}\mathbf{e}_s - \mathbf{W}_{p,2}\mathbf{e}_o\|_1, \quad \mathbf{W}_{p,i} \in \mathbb{R}^{k \times k}$	$\mathcal{O}\left(n_e k + n_r k^2\right)$
Semantic Matching Energy (SME) [9] Linear	$-(\mathbf{W}_1\mathbf{e}_s + \mathbf{W}_2\mathbf{e}_p + \mathbf{b}_1)^T(\mathbf{W}_3\mathbf{e}_o + \mathbf{W}_4\mathbf{e}_p + \mathbf{b}_2)$ $\mathbf{e}_p \in \mathbb{R}^k, \mathbf{W}_i \in \mathbb{R}^{d \times k}, \mathbf{b}_j \in \mathbb{R}^d$	$\mathcal{O}(n_e k + n_r k + dk)$
Semantic Matching Energy (SME) [9] Bilinear	$-\left[(\mathbf{W}_1 \times_3 \mathbf{e}_p)\mathbf{e}_s\right]^T\left[(\mathbf{W}_2 \times_3 \mathbf{e}_p)\mathbf{e}_o\right]$ $\mathbf{e}_p \in \mathbb{R}^k, \mathbf{W}_i \in \mathbb{R}^{d \times k \times k}, \mathbf{b}_j \in \mathbb{R}^d$	$\mathcal{O}\left(n_e k + n_r k + dk^2\right)$
Neural Tensor Network (NTN) [11]	$-\mathbf{u}_p^T f\left(\mathbf{e}_s^T\mathbf{W}_p\mathbf{e}_o + \mathbf{W}_{p,1}\mathbf{e}_s + \mathbf{W}_{p,2}\mathbf{e}_o + \mathbf{b}_p\right)$ $\mathbf{W}_p \in \mathbb{R}^{k \times k \times d}, \mathbf{W}_{p,i} \in \mathbb{R}^{d \times k}, \mathbf{u}_p, \mathbf{b}_p \in \mathbb{R}^d$	$\mathcal{O}\left(n_e k + n_r d k^2\right)$

An RDF graph can be viewed as a *labeled directed multigraph*, where each entity is a vertex, and each RDF triple is represented by a directed edge whose label is a predicate, and emanating from its subject vertex to its object vertex. In RDF KBs, the *Open-World Assumption* holds: a missing triple does not mean that the corresponding statement is false, but rather that its truth value is unknown (it cannot be observed).

In the following, given an RDF graph G, we denote as $\mathcal{E}_G$ the set of all entities occurring as subjects or objects in G, and as $\mathcal{R}_G$ the set of all predicates occurring in G. Formally:

$$\begin{aligned}\mathcal{E}_G &= \{s \mid \exists\langle s, p, o\rangle \in G\} \cup \{o \mid \exists\langle s, p, o\rangle \in G\},\\ \mathcal{R}_G &= \{p \mid \exists\langle s, p, o\rangle \in G\}.\end{aligned}$$

For instance, in the case of the RDF graph shown in Ex. 2.1, we have that $\mathcal{E}_G = \{\texttt{Author}, \texttt{Shakespeare}, \texttt{Hamlet}, \texttt{Othello}, \texttt{Tragedy}\}$ and $\mathcal{R}_G = \{\texttt{profession}, \texttt{author}, \texttt{genre}\}$. Furthermore, we denote as $\mathcal{S}_G = \mathcal{E}_G \times \mathcal{R}_G \times \mathcal{E}_G$ as the space of *possible triples* of G, i.e. the set of all triples that can be created by using the entities and predicates in G (note that $G \subseteq \mathcal{S}_G$). We refer to all triples in G as *observed triples*, and to all triples in $\mathcal{S}_G \setminus G$ as *unobserved triples*.

Knowledge Graph Embedding Models

In the literature, several models and methods have been proposed for embedding KGs in continuous, low-dimensional vector spaces, by learning an unique *distributed representation* (or *embedding*) for each entity and predicate in the KG. For instance, in [12], authors cast the problem of learning the optimal entity and predicate embeddings as a three-way adjacency tensor factorization problem. In [10], [6], [11], [9], authors describe the interactions between entity and predicate embeddings by means of an energy-based model: in these works, the entity and predicate embeddings are learned jointly, by minimizing a loss functional measuring the discrepancy between the model and the KG by using Stochastic Gradient Descent. We refer to the corresponding articles for more details on each of these models.

Regardless of the learning procedure, the aforementioned models share a fundamental characteristic: given a KG G, they represent each entity $x \in \mathcal{E}_G$ by means of a continuous *embedding vector* $\mathbf{e}_x \in \mathbb{R}^k$, where $k \in \mathbb{N}$ is a user-defined hyper-parameter. Similarly, each predicate $p \in \mathcal{R}_G$ is associated to a *scoring function* $f_p : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$, also referred to as *energy function* [9]. For each pair of entities $s, o \in \mathcal{E}_G$, the score $f_p(\mathbf{e}_s, \mathbf{e}_o)$ measures the *confidence* that the statement encoded by the triple $\langle s, p, o\rangle$ holds true. For such a reason, such models can be considered as multi-relational *energy-based models* [9].

In a *link prediction* setting, KG embedding models are used as follows. First, the optimal embeddings for all entities and predicates in the KG are learned, by minimizing a loss function and back-propagating the loss to the embeddings. Then, the learned model is used for ranking unobserved triples in descending order: triples with higher scores have an higher probability of representing true statements, and are considered for a completion of the KG.

Consider the RDF graph shown in Ex. 2.1: we aim at learning a model that assigns a higher score (corresponding to a higher probability value) to the triple $\langle\texttt{Othello}, \texttt{genre}, \texttt{Tragedy}\rangle$, which is unobserved but represents the true statement "*Othello is a Tragedy*", and a lower score (corresponding to a lower probability value) to other unobserved triples, such as $\langle\texttt{Hamlet}, \texttt{genre}, \texttt{Author}\rangle$.

III. Knowledge Graph Embedding Models

Several embedding models have been proposed in the literature for addressing the problem of link prediction in KGs [10], [13], [6], [11], [9], [14]. As mentioned in Sect. II, they share a fundamental characteristic: they can be used for learning a *distributed representation* (or *embedding*) for each entity and predicate in the KG. We refer to such models as *embedding models*, and denote the distributed representation of an entity or predicate z by adding a subscript to the corresponding vector or matrix representation, as in $\mathbf{e}_z \in \mathbb{R}^k$.

Formally, let G be an RDF graph. For each entity $x \in \mathcal{E}_G$, embedding models learn a continuous vector representation $\mathbf{e}_x \in \mathbb{R}^k$, with $k \in \mathbb{N}$, called the *embedding vector* of x. Similarly, for each predicate $p \in \mathcal{R}_G$, they learn a *scoring function* $f_p : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$, defined on pairs of embeddings in the embedding vector space, and defined by a set of *embedding parameters*. The score $f_p(\mathbf{e}_s, \mathbf{e}_o)$ of a triple $\langle s, p, o\rangle$ is defined as a function of the distributed representations (embeddings) of its subject s, its predicate p and its object o, and indicates the probability that the corresponding statements holds true.

In Tab. I, we report the scoring functions adopted by several models proposed in the literature. For each model, we report the number of parameters needed for storing the distributed representations of all entities and predicates in the KG: $n_e = |\mathcal{E}_G|$ denotes the number of entities in the KG, $n_r = |\mathcal{R}_G|$ denotes the number of predicates, and $k, d \in \mathbb{N}$ are user-defined hyper-parameters. In general, if the number of parameters grows *super-linearly* with the number of entities and predicates in the KG, it becomes increasingly harder for the model to scale to large, highly-relational KGs [13].

The Translating Embeddings Model. Among the models outlined in Tab. I, the recently proposed *Translating Embeddings* model (TransE) [6] is particularly interesting:

(i) It achieves better results than other state-of-the-art link prediction methods on several datasets.
(ii) The number of parameters in TransE scales *linearly* in the number of entities n_e and predicates n_r in the KG.

The TransE model is very simple: each entity $x \in \mathcal{E}_G$ is represented by its embedding vector $\mathbf{e}_x \in \mathbb{R}^k$, and each predicate $p \in \mathcal{R}_G$ is represented by a (vector) *translation operation* $\mathbf{e}_p \in \mathbb{R}^k$. The score of a triple $\langle s, p, o \rangle$ is given by the similarity (negative L_1 or L_2 distance) between the translated subject embedding $(\mathbf{e}_s + \mathbf{e}_p)$ and the object embedding $\mathbf{e}_o$:

$$f_p(\mathbf{e}_s, \mathbf{e}_o) = -\|(\mathbf{e}_s + \mathbf{e}_p) - \mathbf{e}_o\|_{\{1,2\}}.$$

In the TransE model, the optimal embedding and translation vectors are learned jointly, as discussed in detail in Sect. IV. The number of parameters needed by the TransE model for storing all the embedding and translation vectors is $(n_e k + n_r k)$, a quantity that grows *linearly* with n_e and n_r. For such a reason, TransE can potentially scale to very large and highly-relational KGs [6].

A Novel Set of KG Embedding Models. In the following, we propose a set of variants of the TransE model. Let $\text{sim}(\mathbf{x}, \mathbf{y})$ be a *similarity function*, from the following set: $\text{sim}(\mathbf{x}, \mathbf{y}) \in \{-\|\mathbf{x} - \mathbf{y}\|_1, -\|\mathbf{x} - \mathbf{y}\|_2, \mathbf{x}^T\mathbf{y}\}$, i.e. chosen from the negative L_1 and L_2 distance, and the inner product. We propose the following embedding models, where each is defined by the corresponding scoring function f_p:

- TransE : $f_p(\mathbf{e}_s, \mathbf{e}_o) = \text{sim}(\mathbf{e}_s + \mathbf{e}_p, \mathbf{e}_o)$,
- TransE$^+$: $f_p(\mathbf{e}_s, \mathbf{e}_o) = \text{sim}(\mathbf{e}_s + \mathbf{e}_{p,1}, \mathbf{e}_o + \mathbf{e}_{p,2})$,
- ScalE : $f_p(\mathbf{e}_s, \mathbf{e}_o) = \text{sim}(\mathbf{e}_s \circ \mathbf{e}_p, \mathbf{e}_o)$,
- ScalE$^+$: $f_p(\mathbf{e}_s, \mathbf{e}_o) = \text{sim}(\mathbf{e}_s \circ \mathbf{e}_{p,1}, \mathbf{e}_o \circ \mathbf{e}_{p,2})$,

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^k$ are the embedding vectors of the entities appearing as the subject s and the object o of the triple; $\mathbf{e}_p, \mathbf{e}_{p,1}, \mathbf{e}_{p,2} \in \mathbb{R}^k$ are the embedding parameters of the predicate p of the triple, denoting either a *translation* or a *scaling* vector; and $\circ$ denotes the Hadamard (element-wise) product, corresponding to the vector *scaling* operation.

The scoring function in TransE is the same used in [6], but also allows using the inner product as a similarity measure between the (translated) subject and object embedding vectors, if it shows to improve the performance on the validation set. The TransE$^+$ model generalizes TransE by also translating the object embedding vector $\mathbf{e}_o$. The ScalE and ScalE$^+$ models are similar to the previous two models, but replace the vector *translation* with a *scaling* operation. The rationale behind ScalE and ScalE$^+$ is the following: scaling the embedding vector of an entity can be seen as *weighting* the (latent) features of such an entity in the embedding vector space.

All proposed models share the same advantages as the TransE model: (i) The required number of parameters is $\mathcal{O}(n_e k + n_r k)$, which grows *linearly* with n_e and n_r, and (ii) The gradients of the scoring functions w.r.t. the embedding of entities and predicates can be computed very efficiently.

Algorithm 1 Learning the model parameters via SGD [6]

```
Require: Learning rate $\eta$, Batch size $n$, Iterations $\tau$
Ensure: Optimal entity and predicate embeddings $\hat{\theta}$
1: Initialize entity and predicate embeddings $\theta_0$
2: for $t \in \langle 1, \ldots, \tau \rangle$ do
3:     $\mathbf{e}_x \leftarrow \mathbf{e}_x / \|\mathbf{e}_x\|, \forall x \in \mathcal{E}_G$    {Normalize embeddings}
4:     $T \leftarrow \text{SampleBatch}(G, n)$
5:     $g_t \leftarrow \nabla \sum_{(\langle s,p,o \rangle, \langle \tilde{s},p,\tilde{o} \rangle) \in T} \left[\gamma - f_p(\mathbf{e}_s, \mathbf{e}_o) + f_p(\mathbf{e}_{\tilde{s}}, \mathbf{e}_{\tilde{o}})\right]_+$
6:     $\Delta_t \leftarrow -\eta g_t$    {Compute the update}
7:     $\theta_t \leftarrow \theta_{t-1} + \Delta_t$    {Update the embeddings}
8: end for
9: return $\theta_\tau$
```

IV. Improving the Efficiency of the Representation Learning Process

In [10], [6], [9], authors propose a method for jointly learning the distributed representations of all entities and predicates in a KG G. The method relies on a *stochastic optimization process*, that iteratively updates the distributed representations by increasing the score of triples in G (observed triples) while lowering the score of triples in $\mathcal{S}_G \setminus G$ (unobserved triples). During the learning process, unobserved triples are randomly generated by means of a *corruption process*, which replaces either the subject or the object of each observed triple with another entity in G. Formally, given an observed triple $y \in G$, let $\mathcal{C}_G(y)$ denote the set of all corrupted triples obtained by replacing either its subject or object with another entity:

$$\mathcal{C}_G(\langle s, p, o \rangle) = \{\langle \tilde{s}, p, o \rangle \mid \tilde{s} \in \mathcal{E}_G\} \cup \{\langle s, p, \tilde{o} \rangle \mid \tilde{o} \in \mathcal{E}_G\}.$$

The distributed representations of all entities and predicates in the KG can be learned by minimizing a *margin-based ranking loss*. Formally, let $\theta \in \Theta$ denote a configuration for all entity and predicate embeddings (i.e. the *model parameters*), where Θ denotes the space of parameters. The optimal model parameters $\hat{\theta} \in \Theta$ can be learned by solving the following constrained optimization problem:

$$\begin{aligned} \underset{\theta \in \Theta}{\text{minimize}} \quad & \sum_{\langle s,p,o \rangle \in G} \sum_{\substack{\langle \tilde{s},p,\tilde{o} \rangle \in \\ \mathcal{C}_G(\langle s,p,o \rangle)}} \left[\gamma - f_p(\mathbf{e}_s, \mathbf{e}_o) + f_p(\mathbf{e}_{\tilde{s}}, \mathbf{e}_{\tilde{o}})\right]_+ \\ \text{subject to} \quad & \forall x \in \mathcal{E}_G : \ \|\mathbf{e}_x\| = 1, \end{aligned} \tag{1}$$

where $[x]_+ = \max\{0, x\}$, and $\gamma \geq 0$ is a hyper-parameter referred to as *margin*. The loss functional in Eq. 1 enforces the score of observed triples to be higher than the score of unobserved triples. The constraints in the optimization problem prevent the training process to trivially solve the problem by increasing the entity embedding norms.

Stochastic Gradient Descent. In related works on KG embedding [10], [6], [9], the constrained loss minimization problem in Eq. 1 is solved using *Stochastic Gradient Descent* (SGD) in mini-batch mode, as summarized in Alg. 1. On each iteration, the algorithm samples a batch of triples from the knowledge graph G. Batches are obtained by first randomly permuting all triples in G, partitioning them into n_b batches of the same size, and then iterating over such batches. A single pass over all triples in G is called an *epoch*. Then,

TABLE II: Statistics for the datasets used in the **Link Prediction** and **Triple Classification** tasks.

Dataset	Entities	Predicates	Training Triples	Valid. Triples	Test Triples
FREEBASE (FB15K) [6]	14,951	1,345	483,142	50,000	59,071
WORDNET (WN18) [6]	40,943	18	141,442	5,000	5,000
FREEBASE (FB13) [11]	75,043	13	316,232	11,816	47,466
WORDNET (WN11) [11]	38,588	11	112,581	5,218	21,088

for each triple y in the batch, the algorithm generates a *corrupted* triple $\tilde{y}$ uniformly sampled from $\mathcal{C}_G(y)$: this leads to a set T of observed/corrupted pairs of triples $\langle y, \tilde{y} \rangle$. The observed/corrupted triple pairs are used for computing the gradient of the objective (loss) function in Eq. 1 w.r.t. the current model parameters θ. Finally, θ is updated in the steepest descent direction of the objective function. This procedure is repeated until convergence.

In this work, we argue that *the choice of SGD is strongly sub-optimal*. In fact, the main drawback of SGD is that it requires a careful tuning of the global learning rate η, which is then used for updating all model parameters, regardless of their peculiarities. However, if an entity $x \in \mathcal{E}_G$ occurs in a limited number of triples in G, the corresponding embedding vector $\mathbf{e}_x \in \mathbb{R}^k$ will be updated less often, and it will require a much longer time to be learned. For such a reason, SGD may be very time-consuming, and the training process may require days of computation for large KGs [8]. Intuitively, a possible solution to this problem consists in associating *smaller learning rates* to parameters updated more often, such as the embedding vectors of entities appearing more frequently, and *larger learning rates* to parameters updated less often.

Adaptive Learning Rates. In order to reduce the time required for learning all entity and predicate embeddings, in this work we propose leveraging *Adaptive Per-Parameter Learning Rates*. While SGD uses a global, fixed learning rate η for updating all parameters, we propose relying on methods for estimating the *optimal* learning rate for each parameter, while still being tractable for learning very large models.

We consider two highly-scalable criteria for selecting the optimal learning rates, namely the *Momentum method* [15] and *AdaGrad* [16]: they specify alternate ways of computing the parameters update Δ_t, defined in Alg. 1 on line 6.

Momentum Method. The idea behind this method is to accelerate the progress along dimensions where the sign of the gradient does not change, while slowing the progress along dimensions where the sign of the gradient continues to change. The update rule in this method is defined as follows:

$$\Delta_t \leftarrow \rho \Delta_{t-1} - \eta_m g_t,$$

where $\eta_m \in \mathbb{R}$ is a user-defined hyper-parameter.

AdaGrad. This method is based on the idea that the learning rate of each parameter should grow with the inverse of gradient magnitudes. The update rule in AdaGrad is:

$$\Delta_t \leftarrow - \frac{\eta_a}{\sqrt{\sum_{j=1}^{t} g_j^2}} g_t,$$

where $\eta_a \in \mathbb{R}$ is a user-defined hyper-parameter. AdaGrad adds nearly no complexity, it has very strong convergence guarantees [16], and it has shown remarkable results on large scale learning tasks in distributed environments [17].

V. EMPIRICAL EVALUATIONS

This section is organized as follows. In Sect. V-A we describe experimental settings, datasets and evaluation metrics. In Sect. V-B, we show that *adaptive learning rates* sensibly improve both the efficiency of the learning process, and the predictive accuracy of embedding models. In Sect. V-C, we empirically evaluate the novel embedding models proposed in Sect. III, by training them using adaptive learning rates.

A. Experimental Settings

In the experiments, we followed the same evaluation protocols adopted in [6] and [11]. Specifically, in *Link Prediction* experiments, we used the two datasets released in [6], namely WORDNET (WN18) and FREEBASE (FB15K). In *Triple Classification* experiments, we used the two datasets released in [11]: WORDNET (WN11) and FREEBASE (FB13). Each dataset is composed by a *training*, a *validation* and a *testing* set of triples: their size is summarized in Tab. II.

Link Prediction. In these experiments, we used the metrics proposed in [6] for evaluating the *rank* of each test triple. In particular, for each test triple $\langle s, p, o \rangle$, its object o is replaced with every entity $\tilde{o} \in \mathcal{E}_G$ in the KG G in turn, generating a set of *corrupted* triples in the form $\langle s, p, \tilde{o} \rangle$. The scores of corrupted triples are first computed by the model, then sorted in descending order, and used to compute the rank of the correct triple. This procedure is repeated by corrupting the subject. Aggregated over all the test triples, this procedure leads to the following two metrics: the *averaged rank*, denoted by MEAN RANK, and the *proportion of ranks not larger than* 10, denoted by HITS@10. This is referred to as the RAW setting. In the FILTERED setting, corrupted triples that exist in either the training, validation or test set are removed before computing the rank of each triple. In both settings, a lower MEAN RANK is better, while a higher HITS@10 is better.

Triple Classification. In these experiments, we used the metrics proposed in [11] for evaluating the classification accuracy of the model on test triples, each labeled either *positive* or *negative*. Each test triple $\langle s, p, o \rangle$ is classified as *positive* iff its score is higher than a predicate-specific threshold δ_p, i.e. $f_p(\mathbf{e}_s, \mathbf{e}_o) \geq \delta_p$. Otherwise, it is classified as *negative*. Predicate-specific thresholds δ_p are determined by maximizing the classification accuracy on the validation set.

B. Evaluation of Adaptive Learning Rates

Learning Time. For comparing Momentum and AdaGrad with SGD on the task of solving the optimization problem in Eq. 1, we empirically evaluated such methods on the task of learning the parameters in TransE on WN18 and FB15K, using the optimal hyper-parameter settings reported in [6]: $k = 20$, $\gamma = 2$, $d = L_1$ for WN18, and $k = 50$, $\gamma = 1$, $d = L_1$ for FB15K. Following the evaluation protocol in [18],

Fig. 1: Average loss (the lower, the better) across 10 TransE parameters learning tasks on the WordNet (WN18) and Freebase (FB15k) datasets, using the optimal TransE settings reported in [6]. For each optimization method, we report the hyper-parameter values that achieve the lowest average loss after 100 epochs, and the corresponding average loss values.

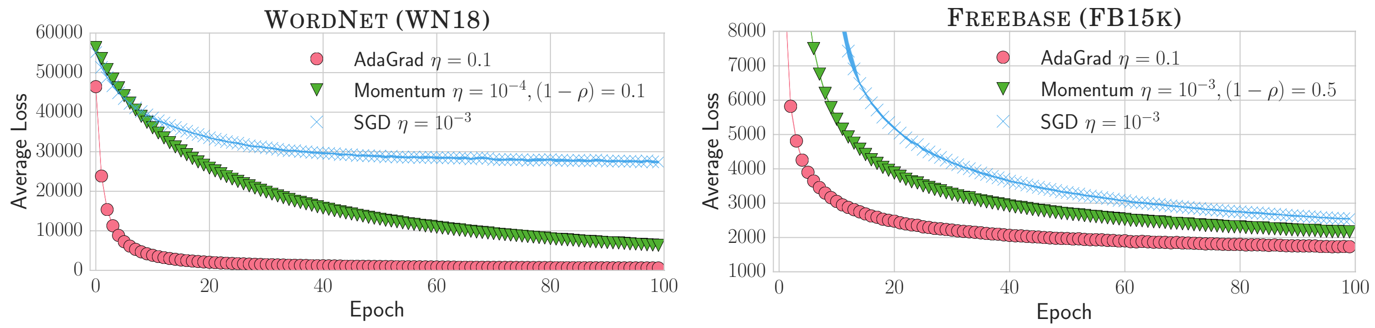

TABLE III: **Link Prediction** and **Triple Classification** Results: Test performance of several link prediction methods on the WordNet and Freebase datasets. Results show the Mean Rank (the lower, the better) and Hits@10 (the higher, the better) for both the Raw and the Filtered settings [6], and the Classification Accuracy (the higher, the better) [11].

Dataset	WordNet (WN18)				Freebase (FB15k)			
Metric	Mean Rank		Hits@10 (%)		Mean Rank		Hits@10 (%)	
	Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.
Unstructured [9]	315	304	35.3	38.2	1,074	979	4.5	6.3
RESCAL [12]	1,180	1,163	37.2	52.8	828	683	28.4	44.1
SE [10]	1,011	985	68.5	80.5	273	162	28.8	39.8
SME Linear [9]	545	533	65.1	74.1	274	154	30.7	40.8
SME Bilinear [9]	526	509	54.7	61.3	284	158	31.3	41.3
LFM [13]	469	456	71.4	81.6	283	164	26.0	33.1
TransE [6]	263	251	75.4	89.2	243	125	34.9	47.1
TransE^A	**169**	**158**	**80.5**	**93.5**	**189**	**73**	**44.0**	**60.1**

Dataset	WordNet (WN11)	Freebase (FB13)
Metric	Acc. (%)	Acc. (%)
SME bil. [9]	70.0	63.7
TransE [6]	75.85	70.9
TransE^A	**85.84**	**74.91**

we compared the optimization methods by using a large grid of hyper-parameters. Let $\mathcal{G}_\eta = \{10^{-6}, 10^{-5}, \ldots, 10^{1}\}$ and $\mathcal{G}_\rho = \{1 - 10^{-4}, 1 - 10^{-3}, \ldots, 1 - 10^{-1}, 0.5\}$. The grids of hyper-parameters considered for each of the optimization methods were the following:

- **SGD** and **AdaGrad:** rate $\eta, \eta_a \in \mathcal{G}_\eta$.
- **Momentum:** rate $\eta_m \in \mathcal{G}_\eta$, decay rate $\rho \in \mathcal{G}_\rho$.

For each possible combination of optimization method and hyper-parameter values, we performed an evaluation consisting in 10 learning tasks, each using a different random initialization of the model parameters.

Fig. 1 shows the behavior of the objective (loss) function for each of the optimization methods, using the best hyper-parameter settings selected after 100 training epochs. We can immediately observe that, for both WordNet (WN18) and Freebase (FB15k), AdaGrad (with rate $\eta_a = 0.1$) yields sensibly lower values of the loss function than SGD and Momentum, even after very few iterations (< 10 epochs). The duration of each epoch was similar in all methods: each epoch took approx. **1.6 seconds** in WordNet (WN18), and approx. **4.6 seconds** in Freebase (FB15k) on a single i7 CPU.

Quality of Learned Models. We also measured the *quality* of models learned by AdaGrad, in terms of the Mean Rank and Hits@10 metrics, in comparison with SGD. For this purpose, we trained TransE using AdaGrad with $\eta_a = 0.1$ for 100 epochs (denoted by TransE^A) and compared it with results obtained with TransE from the literature, on *Link Prediction* and *Triple Classification* tasks on the WordNet and Freebase datasets. Hyper-parameters were selected according to the performance on the validation set, using the same grids of hyper-parameters used for TransE in [6] for the *Link Prediction* tasks, and in [14] for the *Triple Classification* tasks. The results obtained by TransE^A, in comparison with state-of-the-art results reported in [6] and [14], are shown in Tab. III. Despite the sensibly lower number of training iterations (we trained the model using AdaGrad for only 100 epochs, while in [6] and [14] TransE was trained using SGD for 1,000 epochs), TransE^A yields more accurate link prediction models (i.e. with lower Mean Rank and higher Hits@10 and Accuracy values) than every other link prediction model in the comparison.

C. Evaluation of the Proposed Novel Embedding Models

In this section, we evaluate the embedding models inspired by TransE and proposed in Sect. III: ScalE, TransE$^+$ and ScalE$^+$. Model hyper-parameters were selected according to the performance on the validation set: we selected the embedding vector dimension k in $\{20, 50, 100, 200, 300\}$, the *similarity function* $\text{sim}(\mathbf{x}, \mathbf{y})$ in $\{-\|\mathbf{x}-\mathbf{y}\|_1, -\|\mathbf{x}-\mathbf{y}\|_2, \mathbf{x}^T\mathbf{y}\}$, and the margin γ in $\{1, 2, 5, 10\}$. All models were trained using AdaGrad, with $\eta_a = 0.1$, for only 100 epochs.

Results are summarized in Tab. IV. We can see that, despite the very different geometric interpretation, the proposed embedding models achieve sensibly higher results in terms of Hits@10 in comparison with every other link prediction models outlined in Sect. III. An explanation is that TransE [6] and the proposed models TransE$^+$, ScalE and ScalE$^+$ have

TABLE IV: **Link Prediction** Results: Test performance of the embedding models proposed in Sect. III on the WordNet and Freebase datasets. Results show the Mean Rank (the lower, the better) and Hits@10 (the higher, the better) [6].

Dataset		WordNet (WN18)				Freebase (FB15k)			
Metric		Mean Rank		Hits@10 (%)		Mean Rank		Hits@10 (%)	
		Raw	Filt.	Raw	Filt.	Raw	Filt.	Raw	Filt.
TransE from [6]	SGD	263	251	75.4	89.2	243	125	34.9	47.1
TransE	AdaGrad	161	150	80.5	93.5	183	63	47.9	68.2
TransE$^+$ from Sect. III	AdaGrad	**159**	**148**	79.6	92.6	196	78	44.9	62.4
ScalE from Sect. III	AdaGrad	187	174	82.7	94.5	194	62	**49.8**	**73.0**
ScalE$^+$ from Sect. III	AdaGrad	298	287	**83.7**	**95.5**	**185**	**59**	50.0	71.5

a *limited statistical capacity* in comparison with other models. For such a reason, they are *less prone to underfitting* than other more expressive link prediction models, such as RESCAL [12], SME [9] and NTN [11].

In each experiment, the proposed models ScalE and ScalE$^+$ always improve over TransE in terms of Hits@10. We can clearly see that, by leveraging: (i) *Adaptive* learning rates, and (ii) The proposed embedding models ScalE and ScalE$^+$, we were able to achieve a record **95.5%** Hits@10 on WordNet, and a **73.0%** Hits@10 on Freebase. These results are sensibly higher than state-of-the-art results reported in [6]. It is also remarkable that, during learning, the proposed method required a much lower learning time (100 epochs, approx. **30 minutes** on Freebase, on a single CPU) in comparison with [6] (1,000 epochs, and careful rate tuning).

A significantly lower training time – from days, as reported by [8], to minutes – can sensibly improve the applicability of this family of link prediction models in the Web of Data.

VI. Conclusions

We focused on the problem of link prediction in large Knowledge Graphs [4]: recently, KG embedding models such as TransE [6] achieved new state-of-the-art results on this problem. In this paper, we proposed a method for sensibly reducing the learning time in embedding models, and proposed a set of new models with interesting scalability properties. We extensively evaluated the proposed methods in several experiments on real world large datasets: our results show a significant improvement in terms of predictive accuracy over state-of-the-art link prediction methods, while significantly reducing the required training time by an order of magnitude. This contribution improves both the effectiveness and applicability of embedding models on large and Web-scale KGs. Source code and datasets for reproducing the experiments in this paper are available on-line, with an open source license [4].

References

[1] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008*, J. T. Wang, Ed. ACM, 2008, pp. 1247–1250.

[2] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[3] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, S. A. Macskassy *et al.*, Eds. ACM, 2014, pp. 601–610.

[4] A. Bordes and E. Gabrilovich, "Constructing and mining web-scale knowledge graphs: KDD 2014 tutorial," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, 2014, p. 1967.

[5] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26*, C. Burges *et al.*, Eds. Curran Associates, Inc., 2013, pp. 2787–2795.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," C. J. C. Burges *et al.*, Eds., 2013, pp. 3111–3119.

[8] K. Chang, W. Yih, B. Yang, and C. Meek, "Typed tensor decomposition of knowledge bases for relation extraction," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, A. Moschitti *et al.*, Eds. ACL, 2014, pp. 1568–1579.

[9] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[10] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011*, W. Burgard *et al.*, Eds. AAAI Press, 2011.

[11] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems 26*, C. Burges *et al.*, Eds. Curran Associates, Inc., 2013, pp. 926–934.

[12] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, L. Getoor *et al.*, Eds. Omnipress, 2011, pp. 809–816.

[13] R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski, "A latent factor model for highly multi-relational data," in *Advances in Neural Information Processing Systems 25*, F. Pereira *et al.*, Eds. Curran Associates, Inc., 2012, pp. 3167–3175.

[14] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, C. E. Brodley *et al.*, Eds. AAAI Press, 2014, pp. 1112–1119.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Wilson, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[16] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[17] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira *et al.*, Eds. Curran Associates, Inc., 2012, pp. 1223–1231.

[18] T. Schaul, I. Antonoglou, and D. Silver, "Unit tests for stochastic optimization," in *International Conference on Learning Representations*, 2014.

[4] Source code and datasets: https://github.com/pminervini/DeepKGC