Ensemble-based Fact Classification with Knowledge Graph Embeddings

Unmesh Joshi and Jacopo Urbani

Department of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
u.n.joshi@vu.nl,jacopo@cs.vu.nl

Abstract. Numerous prior works have shown how we can use Knowledge Graph Embeddings (KGEs) for ranking unseen facts that are likely to be true. Much less attention has been given on how to use KGEs for fact classification, i.e., mark unseen facts either as true or false. In this paper, we tackle this problem with a new technique that exploits ensemble learning and weak supervision, following the principle that multiple weak classifiers can make a strong one. Our method is implemented in a new system called DuEL. DuEL post-processes the ranked lists produced by the embedding models with multiple classifiers, which include supervised models like LSTMs, MLPs, and CNNs and unsupervised ones that consider subgraphs and reachability in the graph. The output of these classifiers is aggregated using a weakly supervised method that does not need ground truths, which would be expensive to obtain. Our experiments show that DuEL produces a more accurate classification than other existing methods, with improvements up to 72% in terms of F_1 score. This suggests that weakly supervised ensemble learning is a promising technique to perform fact classification with KGEs.

1 Introduction

Knowledge Graphs (KGs) [26] have emerged as the de-facto standard to share large amounts of factual knowledge on the Web. A fundamental problem that concerns KGs is *link prediction*, i.e., the problem of predicting potential missing links in a KG.

Recently, numerous works [40, 25] have shown that Knowledge Graph Embeddings (KGEs) models can be used to identify the top k completions for link patterns (e.g., $\langle$`London`, `capitalOf`, `?`$\rangle$). This operation is useful to identify a smaller set of promising links, but more work is needed for selecting the correct ones. Consider, for instance, the case of a human KG curator who is searching for missing links. An embedding model can help they to identify the k most promising links, but in practice only a small fraction of such a subset is indeed correct. To recognize those, an additional evaluation is needed, which might be time consuming if it were conducted manually. This problem would be solved, or at least reduced, if we have a procedure that directly classifies potential links with a binary true/false label. Such a procedure could be used to implement a fully automated KG completion pipeline, or at least would lift the burden of interpreting ranked lists of potential completions off the user.

Surprisingly, performing fact classification with KGEs is a problem that is not yet well studied. So far, the research on KGEs has primarily focused on the model construction, using ranking as the main evaluation metric and leaving the task of making fact classification as future work [35, 42]. This makes existing KGEs models (e.g.,

TransE [5], ComplEx [38], RotatE [36], RDF2Vec [29]) not suitable in their current form. So far, the only proposal for performing fact classification with KGEs is to simply label all the top k completions as correct and all the others as incorrect [35]. In practice, however, this approach does not work well because not all correct completions appear in the top ranked positions; thus a small k would affect recall while a large one would affect precision. Another approach would be to include additional background knowledge such as ontologies to filter out incorrect links. For instance, if we know that a property is functional, then at most one of the k completions should be marked as correct. Unfortunately, such additional knowledge is not always available; thus we consider the setting where we do not have it.

In general, we can identify two main key challenges for performing fact classification using KGEs. *First*, KGs can be very incomplete and this affects negatively the accuracy of predictions. *Second*, it is hard to produce training samples, negative in particular, because KGs are built under Open World Assumption (OWA), thus potential links can be either missing or incorrect. One could address this problem by manually annotating the top k completions, but this is a time consuming operation which may require human experts.

The two challenges above increase significantly the difficulty of designing a single procedure, e.g., a supervised classifier, that relies solely on the embeddings to produce the classification. Fortunately, there is a well-known alternative approach in Machine Learning called *ensemble learning* that is designed to address precisely the cases when we do not have a classifier that is accurate enough. The idea behind ensemble learning is conceptually simple: instead of focusing on a single classifier, we can use multiple ones, following the principle that multiple weak classifiers can make a strong one.

Ensemble learning is a technique that has been successfully applied in multiple domains (e.g., see overview at [47]), but it has never been applied for fact classification with KGEs. In this paper, we cover this gap with a new ensemble learning method called DuEL (Dual Embedding-based Link prediction), that is specifically designed for fact classification with KGEs. With ensemble learning, the challenge is to identify a suitable set of classifiers and aggregation technique to exploit their predictive power as much as possible. Next to this, in our context we also need to face the problem that we lack ground truths to train any supervised classifier and aggregation model.

We address the aforementioned problems as follows. For the selection of a suitable set of classifiers, we considered state-of-the-art neural architectures, which can be seen as a natural choice for this type of problems. In particular, we selected three different models: an LSTM network [17], a convolutional neural network (CNN) [13], and a multi-layer perceptron (MLP) [4]. We selected these models because they interpret the input in different ways (e.g., with an LSTM it is a one-by-one sequence while with a CNN multiple facts are fed at the same time), hence each of them can capture signals that the others might miss. To function properly, however, all three models require ground truths for training, which we do not have. To fix this problem, we created (possibly wrong) training data assuming Closed World Assumption (CWA), which states that everything that is not in the KG is false by definition. A consequence of this assumption is that the training data might contain many false negatives. Hence, the classifier is trained with a bias towards rejecting potential good completions, which favors preci-

sion but harms recall. To mitigate this problem, we include two additional unsupervised classifiers which leverage subgraph embeddings [18] and shared paths in the KG, respectively. These two classifiers tend to have a higher recall. Therefore, they are a good complement to the first three classifiers.

For aggregating the classifiers' outputs, using a supervised classifier is problematic because we do not have ground truths. An alternative could be to rely on unsupervised techniques like majority voting. However, such approaches would not consider possible differences of the classifiers' accuracies, or latent correlations between them. To exploit those, we can leverage recent weakly supervised techniques that combine the outputs of classifiers without ground truths [28, 12]. In the literature, these models have been shown to be very effective for making predictions with noisy data (e.g., see the Snorkel project [27]). We show here that they are also valuable for performing link prediction, which is a problem for which they have not been applied yet.

While our solution is conceptually simple, our experiments using multiple embedding models confirmed that DuEL was able to perform a fact classification that is much more accurate than currently possible, with the major benefit that our solution can be trained only with the content of the (incomplete) KG and without high-quality manual annotations. For instance, DuEL outperformed existing methods producing predictions with an F_1 of 0.60 and 0.51 on FB15k237 and DBpedia50 respectively, which are two well-known benchmarks, with improvements that range between 72% and 24% against the second best approach.

2 Link Prediction with KGEs

A KG can be seen as a directed labeled (multi)graph $\mathcal{K} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ where vertices in $\mathcal{V}$ represent entities and every edge in $\mathcal{E}$ denotes a semantic relation labeled with type $r \in \mathcal{R}$. Given $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$, we write $\langle h, r, t\rangle$ to indicate the edge from h to t which is labeled with r, e.g., $\langle \texttt{London}, \texttt{capitalOf}, \texttt{UK}\rangle$. Throughout, we often refer to edges as *links*. We also introduce the expression *link pattern*, denoted $\langle h, r, ?\rangle$ ($\langle ?, r, t\rangle$), to refer to the set of all links from h (to t) with label r. Finally, we say that e is a *valid completion* for $\langle h, r, ?\rangle$ ($\langle ?, r, t\rangle$) *in* $\mathcal{K}$ if $\langle h, r, e\rangle \in \mathcal{K}$ ($\langle e, r, t\rangle \in \mathcal{K}$).

We assume that $\mathcal{K}$ is incomplete in the sense that some links are missing. This assumption implies the existence of another KG $K' = (\mathcal{V}, \mathcal{E}', \mathcal{R})$ where $\mathcal{E}' \supset \mathcal{E}$ is the set of all true links with a label in $\mathcal{R}$ between the entities in $\mathcal{V}$. Our goal is to predict all and only the links in $\mathcal{E}' \setminus \mathcal{E}$.

An *embedding* is a vector in $\mathbb{R}^d$ where $d > 0$ and an *embedding model* (or *model* for short) is a set of embeddings. Several techniques have been proposed to construct embedding models that are suitable for link prediction (e.g., [36, 7, 24, 38, 44, 41, 5, 32]). In this paper, we consider three techniques: ComplEx [38], RotatE [36], and TransE [5], which we selected as examples of factorization models (ComplEx) and translational models (RotatE, TransE). ComplEx and RotatE are among the techniques that returned the best performance according to [30] while TransE is included as it is one of the oldest and most frequently used techniques.

All three techniques assign a vector of d numbers to every entity in $\mathcal{V}$ and every relation in $\mathcal{R}$, effectively creating models with $(|\mathcal{V}| + |\mathcal{R}|) \times d$ parameters. In the case

of TransE, the numbers are real, while with RotatE and ComplEx the numbers are complex. These techniques first define a suitable scoring function for a candidate link $\langle h, r, t\rangle$. Then, the models are trained with different loss functions that combine the scoring functions of true and false links. With TransE, the scoring function is:

$$f_{tr}(\langle h, r, t\rangle) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}|| \tag{1}$$

where $||\cdot||$ is the L1 norm, $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ are the vectors associated to h, r, and t respectively (we follow convention of denoting the embeddings in boldface). With ComplEx, it is:

$$f_{co}(\langle h, r, t\rangle) = Re(\langle \mathbf{r}, \mathbf{h}, \bar{\mathbf{t}}\rangle) \tag{2}$$

where $\langle\cdot\rangle$ applied to vectors is the generalized dot product, $Re(\cdot)$ is real component, and $\bar{\cdot}$ is the conjugate for complex vectors. With RotatE, it is:

$$f_{ro} = ||\mathbf{h} \circ \mathbf{r} - \mathbf{t}|| \tag{3}$$

where $\circ$ denotes the element-wise product.

In the literature, empirical evaluations have shown that embedding models return higher scores for true links than for false ones [40, 25, 30]. This observation suggests a straightforward way to do link prediction, that is, to rank every entity t_i according to $f(\langle h, r, t_i\rangle)$, and consider the k entities with the highest ranks as potential valid completions. However, accepting indiscriminately all k is likely to yield a low precision because in practice many of the top k entities are not valid completions. One solution would be to reduce the k to retain only the most likely completions, but this would lower the recall since many correct completions will be missed. To improve the both precision and recall, we are called to critically look at the ranked list of entities and translate the numerical scores into binary decisions.

3 Our proposal

Let $\mathcal{K}$ be the input KG and $\mathcal{K}'$ be the (unknown) KG with all the true links. DuEL is designed to predict all the links in $\mathcal{K}'$ with a given label r that either start from or end to a given entity e. This equals to finding all valid completions for a pattern that is either of the form $\langle ?, r, e\rangle$ or $\langle e, r, ?\rangle$ in $\mathcal{K}'$. Thus, from now on we will assume that the input is a link pattern p and an embedding model M of $\mathcal{K}$, while the output is the set of valid completions for p in $\mathcal{K}'$.

As an example, Figure 1 gives a graphical overview of the functioning of DuEL with the pattern $p = \langle ?, \texttt{locatedIn}, \texttt{UK}\rangle$. The first step consists of computing the top k ranked entities for p with M, which are not valid completions in $\mathcal{K}$ (if they are, then the links are already known). Let us call E the set of such entities. DuEL considers only the entities in E, which are the most likely completions, and ignore all others.

Next, DuEL makes a binary decision for every entity $e \in E$, thus establishing the truth value of links (e.g., if the decision for $e = \texttt{London}$ is positive, then the link $\langle \texttt{London}, \texttt{locatedIn}, \texttt{UK}\rangle$ is correct). Each binary decision is a two-step process. First, multiple classifiers independently label every candidate entity. Then, the labels are aggregated to formulate a final correct/incorrect prediction.

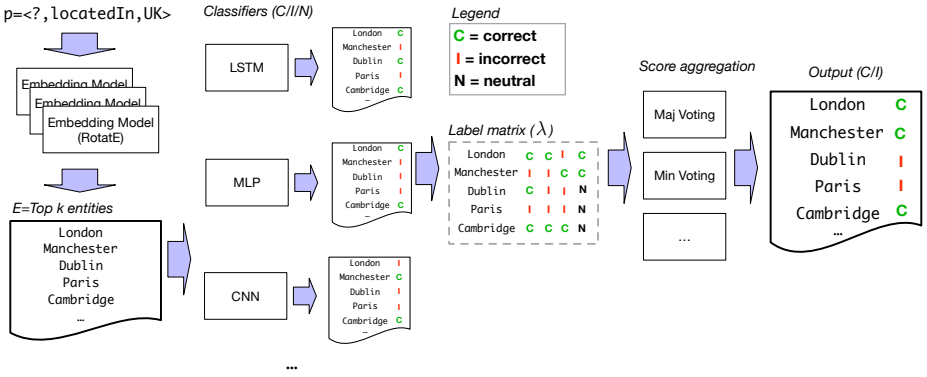

Fig. 1: Schematic overview of DuEL

3.1 Classifiers

In principle, DuEL can be configured to use an arbitrary number of classifiers. In general, we would like to have some classifiers that complement each other. For instance, some can learn to make the predictions based on the latent representations of the entities, and others that more generally consider the structure of $\mathcal{K}$. The classifiers do not need to be conceptually simple or fast to execute. For our purposes, they can be arbitrarily sophisticated as long as they do not require human input. With this desiderata in mind, we introduce five types of classifiers, **C1**,...,**C5**.

The classifiers **C1**,...,**C3** are supervised models. Thus, they require training data. Unfortunately, obtaining ground truth annotations involves a human intervention, which can be expensive. To give an idea, deciding whether a link was true often took more than a minute during the creation of our gold standard. To avoid this problem, we decided to use the content of $\mathcal{K}$ to label the training samples, effectively operating under CWA.

Since this data contains an approximation of the true labels, we train multiple classifiers hoping that mistakes will be corrected during a collective evaluation. Each classifier approaches the problem from a different perspective: **C1** is a MLP, one of the most conventional choices for classification; **C2** is an LSTM, thus it views the classification as a sequence labeling problem; **C3** is a CNN, thus it relies on the convolutional operator to perform a collective prediction of the top-k at once.

The classifiers **C4** and **C5** are added because they do not rely on supervised models: **C4** relies on ranked lists of subgraph embeddings while **C5** considers shared paths between the entities. Since they do not base their decisions using training data created under CWA, we expect them to have a higher recall than the first three which should have instead a higher precision.

For a given set E as input, every classifier returns the set $F \coloneqq \{\langle e, l\rangle \mid e \in E\}$ where $l \in \{\texttt{CORRECT}, \texttt{INCORRECT}\}$. We create two classifiers of each type, one for patterns of the form $\langle ?, r, t\rangle$ and another for $\langle h, r, ?\rangle$. Below, we describe each classifier in more detail.

C1 (MLP) MLPs are among the most popular neural architectures used for classification. Our MLP network is structured with two dense layers (each with n units) interleaved by two dropout layers (each with rate r) and sigmoid as final activation function. As input, the network receives the vectors $\mathbf{t}_1, \ldots, \mathbf{t}_k$ that represents the list of k entities. Each vector $\mathbf{t}_i$ is obtained by concatenating three vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, and the ranking score of the i^{th} ranked entity. Vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are created using the embedding model M. In particular, $\mathbf{a}$ and $\mathbf{b}$ equal to the embeddings of the entity and relation in the link pattern, respectively, while $\mathbf{c}$ is the embedding mapped to the i^{th} entity in the ranked list.

For instance, suppose that we want to construct the vector $\mathbf{t_i}$ that corresponds to the entity e_i, $p = \langle ?, r, t \rangle$, and M was created with TransE. In this case, $\mathbf{a} \coloneqq \mathbf{t}$, $\mathbf{b} \coloneqq \mathbf{r}$, and $\mathbf{c} \coloneqq \mathbf{e_i}$ and the score is $f_{tr}(\langle e_i, r, t \rangle)$.

C2 (LSTM) Using an LSTM for classifying the top-k answers entities is not a usual approach. This is because LSTM is a sequence model and in theory the truth value of each answer does not depend on the ones that are before or after it. However, we observed that there are some regularities in the number and positions of true links in the ranked list. For instance, if $p = \langle \texttt{Ferrary}, \texttt{isA}, ? \rangle$, then it is likely that the valid completions are few and concentrated in the top positions (since typically the number of classes is limited). However, if $p = \langle ?, \texttt{isA}, \texttt{Student} \rangle$, then the valid completions are (probably) many more. These observations hint that the sequence of completions is a useful asset for making binary predictions.

As input, the LSTM with n hidden units receives a sequence of vectors $\mathbf{t}_1, \ldots, \mathbf{t}_k$ that represents the list of k entities. Each $\mathbf{t}_i$ is constructed in the same manner as done for **C1**. Since our task is classification, we add, on top of the LSTM, one extra layer with n units, followed by a dropout layer with rate r and a final dense layer with a sigmoid activation function to produce a binary classification.

C3 (CNN) CNN networks are a very popular type of deep neural networks used primarily for image processing and other types of problems, like sentence classification [21], sentiment analys [9], or text ranking [33].

We construct a network with a single 2D convolutional layer, parametrized by a kernel of size $s_1 \times s_1$. We chose a 2D layer instead of a 1D layer because with the former we can model the interactions between the concatenated embeddings. As input, we provide a 2D matrix obtained by concatenating the $\mathbf{t}_1, \ldots, \mathbf{t}_k$ vectors. The convolutional layer returns an output with k channels, which is passed to a max pooling layer, parametrized by another kernel of size $s_2 \times s_2$ for further down sampling. This layer returns a 1D vector with k elements, which is post-processed by a sigmoid activation function such that it returns k binary predictions.

C4 (Subgraph Embeddings) This classifier uses subgraph embeddings, which were recently introduced by [18]. Subgraphs embeddings are created by aggregating the embeddings of the entities contained in them, and are meant to quickly provide an approximate ranking of the top k entities. In this context, subgraphs are defined as set of

entities that share the same neighbour with edges with the same label. Subgraphs can be of two types, depending on the direction of the edges to the common neighbour. Let us recall that $\mathcal{K} = (\mathcal{V}, \mathcal{E})$. We denote with $S_{\langle ?,r,t\rangle} \coloneqq \{h \mid \langle h,r,t\rangle \in \mathcal{E}\}$ the subgraph with all entities with outgoing edges to t which are labeled with r. Analogously, $S_{\langle h,r,?\rangle} \coloneqq \{t \mid \langle h,r,t\rangle \in \mathcal{E}\}$ denotes the subgraph with incoming edges from h. Subgraph embeddings are constructed by averaging the embeddings of their entities. With TransE, this equals to

$$\mathbf{S}_l \coloneqq \frac{\sum_{e \in S_l} \mathbf{e}}{|S_l|} \tag{4}$$

With ComplEx and RotatE, we take the average of the real ($Re(\cdot)$) and imaginary ($Im(\cdot)$) parts, respectively.

$$\mathbf{S}_l \coloneqq \langle \frac{\sum_{e \in S_l} Re(\mathbf{e})}{|S_l|}, \frac{\sum_{e \in S_l} Im(\mathbf{e})}{|S_l|} \rangle \tag{5}$$

The average embeddings computed by Equations 4 and 5 allow us to apply the scoring function using the subgraph embeddings rather than the embeddings of potential completions. Once the classifier receives as input p and E, it first ranks all the subgraphs with the scoring function of M and p. This operation produces a list $\mathcal{S} = \langle S_1, S_2, \ldots\rangle$ of subgraphs. Then, it retains in $\mathcal{S}$ only the top j subgraphs, where j is a threshold value that is dynamically computed using statistics from $\mathcal{K}$ [18].

Finally, the classifier labels every entity $e \in E$ as follows. If e appears as a member of any subgraph in $\mathcal{S}$, then it is labeled as `CORRECT`. Otherwise, e is labeled as `INCORRECT`. Notice that this classifier, unlike the previous three, does not require a training phase.

C5 (Shared Paths) The previous four classifiers rely on KG embeddings to make their predictions. In contrast, this classifier does not use embeddings, but considers instead shared paths between potential valid completions in $\mathcal{K}'$ (i.e., the entities in E) and any valid completion in $\mathcal{K}$.

First, let us assume that the input link pattern p is of the form $\langle ?, r, t\rangle$ (the other case is analogous). Then, let $\mathcal{P}_{a,b}$ be the set of all paths between a and b in $\mathcal{K}$ of maximum length 2 (note that the direction of the edges is not taken into account). Furthermore, let $\mathcal{P}_p = \{q \mid q \in \mathcal{P}_{a,t}, \langle a,r,t\rangle \in \mathcal{K}\}$ the set of all paths between t and any valid completion of p in $\mathcal{K}$.This classifier will label every entity $e \in E$ as `CORRECT` if there is a path $p \in \mathcal{P}_{e,t}$ and another path $q \in \mathcal{P}_p$ which differ only on the first entity in the paths. In other words, entity e is marked as `CORRECT` if it is connected to t with the same path as one of the valid completions in $\mathcal{K}$. Otherwise, e is labeled as `INCORRECT`.

3.2 Aggregation

The output of classifiers can be aggregated in several ways. Two techniques which are often used are *Min Voting* and *Majority Voting*. With the first technique, we label an entity as `CORRECT` if at least one classifier has labeled it as `CORRECT`. With the second one, we pick the label chosen by the majority of the classifiers (ties are broken

arbitrarily). A disadvantage of these techniques is that they do not consider latent correlations between the classifiers. To include those, we can use several approaches that were originally introduced for building machine learning models without using ground truth annotations [28, 12].

The problem of aggregating without ground truths can be modeled as follows. The input consists of a set data points $\mathcal{X}$. The goal consists of labelling each data point $X \in \mathcal{X}$ with a vector $\mathbf{Y} = [Y_1, \ldots, Y_t]^T$ of t categorical task labels. We assume that we do not have any ground truth that we can use for training. To recover from the lack of such data, we consider a set of *sources* that provide approximate labels for (a subset of) the t tasks. If they cannot provide a label, then the sources can *abstain*.

The sources might be potentially correlated and have an unknown accuracy. We can estimate those considering the observed agreement and disagreement rates of the emitted labels. To this end, we can construct a matrix $\boldsymbol{\lambda}$ of noisy labels produced by the sources, and then compute a *label model* $P_\mu(\mathbf{Y}|\boldsymbol{\lambda})$ where μ is the vector of parameters that encodes the correlations and accuracies. Then, we can use the label model to output a single probabilistic label vector $\tilde{\mathbf{Y}}$ from the noisy labels of an unseen data point X. The problem translates into computing the parameters μ. One way consists of estimating μ from the inverse covariance matrix among the sources [28]. Another approach consists of breaking down the original problem into a set of smaller problems that considers subsets of three sources. The advantage is that the subproblems have closed-form solutions, thus the parameters can be computed without iterative solutions [12].

In our context, we have a single task, the categorical task label represents the binary prediction `CORRECT` and `INCORRECT`, the data points are the potential completions for p (i.e., E), and there are five sources, **C1**, . . . , **C5**. Finally, since one of the strengths of such methods is to consider that a source might abstain, we slightly change the labeling of our classifiers as follows. Instead of predicting either `CORRECT` or `INCORRECT`, we introduce one extra label `NEUTRAL`. Then, we modify the output of **C1**, **C2**, and **C3** introducing two threshold values τ_1 and τ_2. If the output of the sigmoid is lower (higher) than τ_1 (τ_2) then the label is `INCORRECT` (`CORRECT`). Otherwise, if it is between τ_1 and τ_2, then it is `NEUTRAL`.

A problem that arise with our threshold-based approach is that we must find good values for τ_1 and τ_2. In practice, we observed that using grid search using a small held-out validation dataset yields satisfactory performance. For **C4** and **C5**, we replace `INCORRECT` with `NEUTRAL`, thus simulating a prediction under OWA where missing links are not automatically considered as incorrect.

To aggregate the classifiers' output, we first compute the label model by applying the classifiers on every potential completion and use their output to create the label matrix $\boldsymbol{\lambda}$. After computing μ, either exploiting the covariance ([28]) or the decomposition in triplets ([12]), we compute $\boldsymbol{\lambda}$ for an unlabeled completion e, pass it to the label model and use the value of the returned $\tilde{\mathbf{Y}}$ as the final (binary) label of e.

It is important to note that none of the components in the pipeline of DuEL needs large volumes of manually annotated data, with a consequent benefit in terms of scalability. The classifiers **C4**, **C5** do not require training, **C1**, **C2**, and **C3** are trained using the true links of $\mathcal{K}$, and the training of the label model only considering the provided $\boldsymbol{\lambda}$.

	FB15k237	DBpedia50
# Entities in $\mathcal{K}$	14k	50k
# Edges in $\mathcal{K}$	298k	32k
Parameters embedding models		
ComplEx: d (dimension) 256, lr (learning rate) 0.1, regularizer 0.95		
RotatE: d 400, lr 0.5, γ (margin) 6, sampling temperature 2.0		
TransE: d 200, lr 0.1, γ 1, batch size 1000		
*Statistics on the training datasets for **C1**, **C2**, and **C3***		
# Training data points for patterns $\langle ?, r, t\rangle$	56k	11k
# Training data points for patterns $\langle h, r, ?\rangle$	93k	29k

(a) KGs, model parameters, and training data

Parameter name	*Min*	*Max*
n (# hidden units)	10	500
d (drop out)	0.0	0.8
s_1 and s_2 (kernel sizes)	2	6
τ_1	0.01	0.8
τ_2	0.01	0.8

(b) Ranges for grid search to determine value hyperparameters

Fig. 2: Statistics, model parameters, and ranges used for grid search

4 Evaluation

KGs. As inputs, we considered datasets commonly used in the related literature. We selected FB15k237, a subset of Freebase used as benchmark in many works (e.g., [18, 36, 38]), and DBpedia50, a subset of DBpedia used in [30]. We used the parameters for the embedding models reported as optimal by [30]. Details are in Figure 2a. The embedding models were trained with Adagrad [10] for 1000 epochs.

Below, we set $k = 10$ ($|E|$) as default since this is a common threshold used for evaluating ranked lists of entities (*hit@10*). The experiments were performed on a machine with 64GB RAM and two 8-core CPU 2.4GHz. The code and other experimental data is available online[1].

Training C1, C2, and C3. These classifiers are trained under CWA using training data that was automatically generated. Every pattern and list of k completions is a data point used to train the networks. Since there is a variable number of link patterns of different types, the number of data points depends on the KG and type of pattern. Figure 2a reports the size of the training data sets. Training occurred by minimizing the binary cross entropy with Adam [19] for 10 epochs.

We performed grid search to find the optimal values for n (hidden units), r (dropout rate), s_1 and s_2 (kernel sizes), τ_1, and τ_2 optimizing for the best F_1 on the validation dataset. The ranges considered for the search are reported in Figure 2b. We observed that the values $n = 100$, $r = 0.2$, $s_1 = 3$, $s_2 = 2$, $\tau_1 = 0.2$ and $\tau_2 = 0.6$ work well with the models and KGs.

Gold standard. To test the performance of DuEL, we cannot rely on the content of the input KG since it is, by definition, incomplete. In particular, we cannot use a held-out dataset as it is typically done for evaluating the ranking capabilities of embedding models because such a dataset would contain only (some) links which we know are true (and not the ones which we know are false). To perform a more complete evaluation, we created a gold standard with data annotated by humans. We randomly selected

[1] https://github.com/karmaresearch/duel

	FB15k237	DBpedia50
Statistics on the gold standard		
# Annotated links	∼3900	∼3600
Ratio `CORRECT` links for $\langle ?, r, t \rangle$ in top k entities	14%	12%
Ratio `CORRECT` links for $\langle h, r, ? \rangle$ in top k entities	11%	3%

(a) Ratio of correct links in the gold standards with FB15k237 and DBpedia50

FB15k237	DBpedia50
`tv actor`	`producer of film`
`influenced by`	`birth place`
`graduate student`	`director of film`
`award category`	`soccer team`
`award nomination`	`starring in film`
`netflix genre`	`creative writer`
`film performance`	`country`
`film language`	`is part of`
`person profession`	`film language`
`music artist`	`music genre`

(b) Ten popular relations (ranked from the most popular to the least one) on FB15k237 and DBpedia50

Fig. 3: Details about gold standard

Fig. 4: Screenshot annotation interface

250 and 150 previously unseen link patterns of both types (50/50) for FB15k237 and DBpedia50 respectively, retaining 50 patterns of each type to construct a small validation dataset. Then, for every link pattern and embedding model, we manually annotated the top $k = 10$ entities that correspond to links that are not in $\mathcal{K}$, consulting external sources to verify the correctness of the links. The annotations were performed by two human annotators who independently annotated the links using a special web interface. Figure 4 reports a screenshot of (part of) the interface. The interface shows, for a given query, which are the top ranked answers provided by the three embedding models. Additional links to Google and Wikipedia are provided to help the human annotator to decide whether a particular answer is correct.

In total, the annotators labeled about 3900 links for FB15k237 and 3600 links for DBpedia50. Figure 3a reports the rate of `CORRECT` links in both datasets while Figure 3b reports the 10 most popular relations annotated in each dataset. Since the task of the annotators is to verify whether the fact is true, the degree of subjectivity is low. This is confirmed by a high Cohen's score: With FB15k237 is it 0.8137 while with DBpedia50 it is 0.869, which indicate a nearly perfect agreement between the annotators. Notice that the size of the gold standard is much smaller than the size of training data used to train the classifiers since the former requires a manual annotation while

	Method	ComplEx			RotatE			TransE		
		P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1
FB15k237	*RankClass*	**0.727**	0.229	0.348	**0.747**	0.232	0.353	**0.682**	0.221	0.334
	DeepPath	0.193	0.210	0.198	0.203	0.207	0.205	0.247	0.210	0.202
	AnyBurl	0.217	0.257	0.208	0.199	0.253	0.223	0.211	0.258	0.221
	C1	0.619	0.436	0.505	0.592	0.435	0.501	0.529	0.296	0.371
	C2	0.517	0.589	0.539	0.531	0.624	0.560	0.544	0.388	0.451
	C3	0.704	0.267	0.384	0.674	0.284	0.392	0.564	0.271	0.347
	C4	0.356	**0.883**	0.508	0.357	0.563	0.416	0.337	**0.876**	0.487
	C5	0.394	0.651	0.490	0.406	0.655	0.499	0.382	0.651	0.480
	MinV	0.317	*1.000*	0.481	0.324	*1.000*	0.490	0.308	*1.000*	0.471
	MajV	0.560	0.588	0.567	0.559	0.513	0.523	0.548	0.492	0.514
	Super.	0.565	0.537	0.550	0.595	0.486	0.533	0.522	0.507	0.513
	DuEL (M)	0.527	0.710	**0.599**	0.467	**0.824**	**0.569**	0.491	0.641	**0.537**
	DuEL (S)	0.528	0.728	0.594	0.472	0.740	0.540	0.504	0.586	0.530
DBpedia50	*RankClass*	0.516	0.353	0.403	0.740	0.273	0.381	0.183	0.096	0.120
	C1	0.620	0.105	0.175	**0.920**	0.070	0.129	0.375	0.027	0.051
	C2	**0.650**	0.287	0.350	0.500	0.005	0.010	0.000	0.000	0.000
	C3	0.354	0.415	0.363	0.551	0.326	0.314	0.240	0.205	0.215
	C4	0.162	0.112	0.132	0.253	0.094	0.137	0.135	0.109	0.121
	C5	0.187	0.288	0.222	0.259	0.305	0.280	0.151	0.209	0.175
	MinV	0.164	*1.000*	0.277	0.323	*1.000*	0.469	0.165	*1.000*	0.275
	MajV	0.520	0.149	0.210	0.917	0.052	0.095	0.344	0.050	0.087
	Super.	0.583	0.086	0.147	0.917	0.068	0.125	**0.400**	0.018	0.035
	DuEL (M)	0.555	0.457	**0.501**	0.557	**0.499**	**0.505**	0.191	0.751	0.274
	DuEL (S)	0.399	**0.625**	0.433	0.665	0.353	0.461	0.186	**0.961**	**0.301**

Fig. 5: Performance on gold standard (*P*, *R* and F_1 denote Precision, Recall, F_1 scores, respectively). The best results are marked in boldface

the latter can be automatically computed from the input KG. Also, notice that the ratio of correct links is fairly low, especially in DBpedia50. With such a low ratio, a supervised classifier trained with such data can achieve a high accuracy by simply returning always the label `INCORRECT`. Our method does not suffer from this problem since the aggregation does not need ground truths.

4.1 Performance of link prediction

Baselines. We consider three external baselines and several alternative approaches to the default pipeline, yielding a comparison against 10 other methods. The first external baseline, which we name *RankClassify*, is presented by [35] and it is, as far as we know, the only method that uses embeddings to perform binary predictions. The technique consists of marking as correct all the answers in the top k positions, where k is fine tuned upfront on a validation dataset. The second baseline is the state-of-the-art

method proposed by [43]. This method, which we label *DeepPath*, does not use embeddings. Instead, it uses reinforcement learning to learn reasoning paths on the KG. We configured it to do *fact prediction* on FB15k237 and mark the top k ranked facts as correct, similarly as before. The third baseline is the reinforcement learning version of AnyBURL [22, 23], which learns rules bottom-up for link prediction. AnyBURL is executed using the default parameters mentioned in the online documentation and trained for 1000 seconds. On DBpedia50, we report only the performance with RankClassify as it was the external baseline with the highest F_1.

To compare against more methods, we also apply the five classifiers in isolation. Finally, we compare our weak supervision approach (with the covariance [28]) denoted DuEL (M) and with the triplets [12], denoted DuEL (S)) against two unsupervised and one supervised alternatives. The unsupervised ones are the *minority* and *majority voting*, which are popular choices. The supervised one is a random forest which uses the scores of the classifiers as input features and the validation dataset as training labels.

Results. Figure 5 reports the precision, recall, and F_1 of the `CORRECT` predicted links with our gold standard. We make *six* main observations.

Observation 1. We observe that DuEL returns the highest F_1 in all cases. The improvement is 0.09 (DBpedia50,ComplEx) and 0.01 (FB15k237,RotatE) points better than the second-best non-DuEL result in the best and worst cases, respectively. If we compare against existing approaches in the literature, then the gain increases to 0.25 (FB15k237,ComplEx) and 0.1 (DBpedia50,ComplEx). We find remarkable that both DuEL (M) and denoted DuEL (S) achieve superior or comparable performance than the fully supervised model despite they do not make use of ground truth annotations.

Observation 2. The best performance is obtained with ComplEx, but the differences with the other models are not large, except for DBpedia50 where TransE does not perform as well as the others. A lower performance of TransE should be expected since it is a model that is often outperformed by the other two (e.g., see [30]). In general, we conclude that our approach generalizes well. Therefore, it can be used with different embedding models.

Observation 3. If we compare **C1**,. . . ,**C5**, then we notice that **C1**, **C2**, and **C3** mostly return a higher precision than the other two, as we expected. In contrast, **C4** and **C5** tend to return a higher recall. In some cases, some classifiers used in isolation returned remarkable performance. For instance, **C2** with RotatE and FB15k237 is close to return the best result. In other cases, the classifiers perform very poorly. For instance, **C2** with TransE and DBpedia50 never returned any positive answers.

Observation 4. There is not a classifier that is clearly outperforming the others, and they all contribute to improve the performance. To obtain further evidence, we performed an ablation study where we executed DuEL (M) excluding, each time, the labels of one classifier. Figure 6 reports the obtained F_1 during such a study with some representative KGs, models, and types of patterns. In this case, if the performance loss that we get when we remove one classifier is large, then it means that it provided a significant contribution. We observe that with ComplEx, FB15k237, and $\langle ?, r, t\rangle$ patterns, **C3**, **C4**, and **C5** gave the most significant contribution. In contrast, with $\langle h, r, ?\rangle$ patterns, **C1** and **C2** are more important. These differences highlight the benefit of weak supervision that takes all classifiers into account.

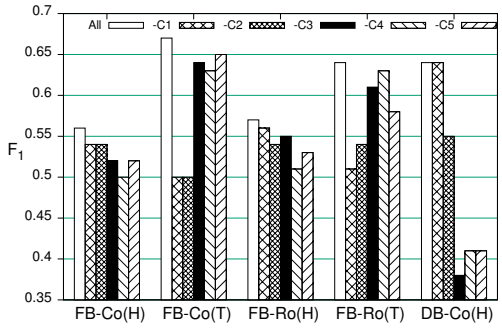

Fig. 6: Ablation study where the F_1 is computed with all classifiers but one. H refers to performance with patterns of the form $\langle ?, r, t\rangle$ while T with $\langle h, r, ?\rangle$. Other abbreviations: FB=FB15k237, DB=DBpedia50, Co=ComplEx, Ro=RotatE

Observation 5. It is interesting to compare the results of DuEL (M) against the ones of DuEL (S). Both methods return similar scores, but DuEL (M) has slightly better performance. Thus, we select it as default choice. However, DuEL (S) has the advantage that it is much faster. Thus, it is a good alternative for a large-scale deployment or for context where a timely prediction is needed.

Observation 6. Fact classification appears to be a hard problem. The absolute performance of external baselines is low, with F_1 scores that range from 0.1 to 0.4. This is partly due to the fact that these methods were mainly designed for ranking and not for classifying. The best F_1 values that we obtained with DuEL are between 0.5 and 0.6. This is due to the fact that many links are hard to label if we have only the KG as input. We observed higher F_1 values for $\langle h, r, ?\rangle$ patterns than for $\langle ?, r, t\rangle$, which is expected since there are typically fewer tails than heads. Despite the absolute values of the F_1 are not very high, the relative improvement brought by DuEL is significant. With DBpedia50 and TransE, DuEL returns an F_1 score that is 2.5 times better than existing techniques but this likely to be due to the low quality of the embedding model. With better models like ComplEx and RotatE, the relative improvement is still significant as it ranges between 72% and 24%. We believe that the performance can be further improved by including more sources, but this is a topic that deserves a dedicated study.

Hyperparameter tuning. We show the effect of some parameters on the overall performance. We report the results only with ComplEx and FB15k237 since they are representative of the other cases. Figure 7a shows the effect of the number of units in the networks of **C1** and **C2** (the tuning that we performed with grid search included more values than the shown ones). We observe that the impact of this parameter is limited since the performance does not change significantly. Figure 7b reports the F_1 if we set different τ_1 and τ_2. In this case, we observe that the performance changes significantly, and this makes τ_1 and τ_2 two important parameters. Figure 7c shows the impact of changing the kernel sizes in **C3**. We notice that also here the impact is noticeable.

Finally, Figure 7d shows how the performance varies if we change the top k considered completions with FB15k237 and ComplEx. As expected, we observe that the performance decreases with higher k since the problem becomes harder.

	n	P	R	F_1
C1	100	0.53	0.39	0.45
C1	200	**0.54**	**0.41**	**0.46**
C1	500	0.51	0.40	0.46
C2	10	0.50	0.48	0.53
C2	100	0.54	**0.58**	**0.56**
C2	200	**0.55**	0.52	0.53

(a)

s_1	s_2	P	R	F_1
2	1	**0.60**	0.26	0.36
3	2	0.49	**0.37**	**0.41**
6	3	0.57	0.27	0.37

(c)

C1		**C2**		**C3**		F_1
τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	
0.01	0.1	0.01	0.1	0.01	0.1	0.49
0.01	0.6	0.01	0.6	0.01	0.6	0.51
0.2	0.6	0.2	0.6	0.2	0.6	**0.57**
0.2	0.8	0.2	0.8	0.2	0.8	0.55

(b)

Top k	*P*	*R*	F_1	*Top k*	*P*	*R*	F_1
$k=1$	**0.62**	**0.96**	**0.76**	$k=5$	0.55	0.79	0.65
$k=3$	0.61	0.87	0.73	$k=10$	0.53	0.71	0.60

(d)

Fig. 7: Figures (a-c): Performance while changing multiple hyperparameters, with ComplEx, and FB15k237. The best results are marked in boldface. Figure (d): Performance with different k

5 Related work

Ranking vs. classifying. The problem that the prevailing evaluation paradigm of KGEs is based on ranking rather than classifying has been empirically studied in [42] and in [35]. Both works focus on the analysis of current methods rather than proposing new ones like ours. Previous work addressed the problem of a ranking-based evaluation by creating negative samples and measuring the accuracy [34], or with new metrics [42]. In contrast, we use a manually annotated dataset.

Link prediction on KGs. Links can be predicted also using rules, which can be either mined from KGs [23, 14] or learned with differentiable models [39]. These approaches propose themselves as alternatives to KGEs for ranking promising set of links. Therefore, they can be used as additional classifiers within our pipeline.

Another technique for finding new links is logic-based reasoning. In particular, rule-based reasoning based on Datalog [1] can compute new facts (links) in KGs with billions of edges [6]. Also, a recent work uses BERT [8] for link prediction [46], leveraging the labels of the entities, thus the language model. Our work differs from them because ours does not depend on external knowledge, like rules or language models.

(Knowledge) Graph embeddings. In our work, we considered three representative embedding models but there are many more that could be considered. A class of embedding models that has yield good results is the one that employs Graph Neural Networks (GNNs) [32]. In principle, our technique could also be used considering the embeddings produced by a GNN, but it is interesting, as future work, to study whether it is possible to exploit the graph-like structure of the GNN to produce more sophisticated classifiers. Another way to exploiting KGEs for fact classification could rely on a probabilistic interpretation of their scoring function. For instance, [11] has shown that it is possible to give to DistMult a probabilistic interpretation, which in turn can be used

for producing a binary classification. In this case, we can employ techniques used for calibrating the probabilities, like the ones presented in [37] and [31] to improve the accuracy of classification.

If we broad our horizon and consider also different types of graphs, then it is useful to mention a recent overview of (unlabeled) graph embeddings is given at [15]. An important task for these techniques consists of classifying the nodes (instead of links), e.g., [16, 20]. To achieve the best results, some techniques use semi supervision [45, 20]. We conclude by mentioning that there are several fully supervised techniques for binary link prediction designed for social networks [2].

6 Conclusion

We addressed the problem of performing (binary) fact classification with KGEs. Existing KGEs methods were designed and evaluated for link prediction via ranking and not via classification. There is an emerging consensus that this is an important limitation and that future methods should be evaluated also on classification next to ranking [35, 42, 3]. To make the problem worse, using KGEs for classification is not trivial also because embeddings may be too noisy if they are not sufficiently trained and we lack large volumes of ground truths to train effectively supervised classifiers on top of them.

Our proposal is the first of its kind. Instead of proposing yet another (embedding) model, the main novelty of our contribution is to show how we can leverage and combine the power of existing methods following the well-established paradigm of ensemble learning. By aggregating the output of multiple classifiers together, we are able to correct mistakes that may be due to noisy embeddings. Moreover, the aggregation takes place in a weakly-supervised manner without using ground truths.

Our experiments confirm the value of our approach. Although the absolute F_1 values show that we have not yet reached human-like levels, the improvement brought by our method is significant. For instance, with our approach the F_1 improved of 72% against the second best with FB15k237 and ComplEx (0.599 vs 0.348) and of 33% against the second best with DBpedia50 and RotatE (0.505 vs 0.381). These improvements indicate that ensemble learning methods are promising techniques to implement fact classification with KGEs.

In practice, we believe that DuEL can be used in several pipelines for knowledge extractions. For instance, it can be used to further assist human curators to further populate KGs or to cover the last mile to implement a fully automated end-to-end embedding-based system for KG completion. In this latter case, more work is needed in order to further improve the accuracy. One possible extension is to include additional classifiers, e.g., based on ontological constraints. Alternatively, it is worthwhile to study whether we can further reduce a potential bias introduced by training some of our classifiers under CWA. Another topic for future work could aim at combining the rankings produced by different embedding models. Moreover, it is interesting to study how we can include the knowledge that can be extracted from textual corpora, or to investigate whether we can build classifiers for some specific relations.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases, vol. 8. Addison-Wesley Reading (1995)
2. Al Hasan, M., Zaki, M.J.: A survey of link prediction in social networks. In: Social network data analytics, pp. 243–275. Springer (2011)
3. van Bakel, R., Aleksiev, T., Daza, D., Alivanistos, D., Cochez, M.: Approximate Knowledge Graph Query Answering: From Ranking to Binary Classification. In: GKR. pp. 107–124 (2021)
4. Bishop, C.: Pattern Recognition and Machine Learning. Springer-Verlag (2006)
5. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. In: NIPS. pp. 2787–2795 (2013)
6. Carral, D., Dragoste, I., González, L., Jacobs, C., Krötzsch, M., Urbani, J.: VLog: A Rule Engine for Knowledge Graphs. In: ISWC. pp. 19–35 (2019)
7. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D Knowledge Graph Embeddings. In: AAAI. pp. 1811–1818 (2018)
8. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: NAACL. pp. 4171–4186 (2019)
9. Dos Santos, C., Gatti, M.: Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In: COLING. pp. 69–78 (2014)
10. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research **12**, 2121–2159 (2011)
11. Friedman, T., Van den Broeck, G.: Symbolic Querying of Vector Spaces: Probabilistic Databases Meets Relational Embeddings. In: UAI. pp. 1268–1277 (2020)
12. Fu, D.Y., Chen, M.F., Sala, F., Hooper, S.M., Fatahalian, K., Ré, C.: Fast and Three-rious: Speeding Up Weak Supervision with Triplet Methods. In: ICML. pp. 3280–3291 (2020)
13. Fukushima, K., Miyake, S.: Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets, pp. 267–285. Springer (1982)
14. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. The VLDB Journal **24**(6), 707–730 (2015)
15. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78–94 (2018)
16. Grover, A., Leskovec, J.: node2vec: Scalable Feature Learning for Networks. In: KDD. pp. 855–864 (2016)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
18. Joshi, U., Urbani, J.: Searching for Embeddings in a Haystack: Link Prediction on Knowledge Graphs with Subgraph Pruning. In: WWW. pp. 2817–2823 (2020)
19. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 (2017), `http://arxiv.org/abs/1412.6980`
20. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 (2017), `http://arxiv.org/abs/1609.02907`
21. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent Convolutional Neural Networks for Text Classification. In: AAAI. pp. 2267–2273 (2015)
22. Meilicke, C., Chekol, M.W., Fink, M., Stuckenschmidt, H.: Reinforced anytime bottom up rule learning for knowledge graph completion. arXiv:2004.04412 (2020), `https://arxiv.org/abs/2004.04412`
23. Meilicke, C., Chekol, M.W., Ruffinelli, D., Stuckenschmidt, H.: Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In: IJCAI. pp. 3137–3143 (2019)

24. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.: A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In: NAACL. pp. 327–333 (2018)
25. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A Review of Relational Machine Learning for Knowledge Graphs. Proceedings of the IEEE **104**(1), 11–33 (2016)
26. Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale Knowledge Graphs: Lessons and Challenges. Communications of the ACM **62**(8), 36–43 (2019)
27. Ratner, A., Bach, S.H., Ehrenberg, H., Fries, J., Wu, S., Ré, C.: Snorkel: rapid training data creation with weak supervision. The VLDB Journal **29**(2), 709–730 (2020)
28. Ratner, A., Hancock, B., Dunnmon, J., Sala, F., Pandey, S., Ré, C.: Training Complex Models with Multi-Task Weak Supervision. In: AAAI. pp. 4763–4771 (2019)
29. Ristoski, P., Paulheim, H.: RDF2Vec: RDF graph embeddings for data mining. In: ISWC. pp. 498–514 (2016)
30. Ruffinelli, D., Broscheit, S., Gemulla, R.: You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings. In: ICLR (2020)
31. Safavi, T., Koutra, D., Meij, E.: Evaluating the Calibration of Knowledge Graph Embeddings for Trustworthy Link Prediction. In: EMNLP. pp. 8308–8321 (2020)
32. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling Relational Data with Graph Convolutional Networks. In: Proceedings of ESWC. pp. 593–607 (2018)
33. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: SIGIR. pp. 373–382 (2015)
34. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS. pp. 926–934 (2013)
35. Speranskaya, M., Schmitt, M., Roth, B.: Ranking vs. Classifying: Measuring Knowledge Base Completion Quality. In: AKBC (2020)
36. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In: ICLR (2019)
37. Tabacof, P., Costabello, L.: Probability Calibration for Knowledge Graph Embedding Models. In: ICLR (2019)
38. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex Embeddings for Simple Link Prediction. In: ICML. pp. 2071–2080 (2016)
39. Wang, P.W., Stepanova, D., Domokos, C., Kolter, J.Z.: Differentiable Learning of Numerical Rules in Knowledge Graphs. In: ICLR (2019)
40. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
41. Wang, Q., Wang, B., Guo, L.: Knowledge Base Completion Using Embeddings and Rules. In: IJCAI. p. 1859–1865 (2015)
42. Wang, Y., Ruffinelli, D., Gemulla, R., Broscheit, S., Meilicke, C.: On Evaluating Embedding Models for Knowledge Base Completion. In: The 4th Workshop on Representation Learning for NLP. pp. 104–112 (2019)
43. Xiong, W., Hoang, T., Wang, W.Y.: DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In: EMNLP. pp. 564–573 (2017)
44. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In: ICLR (2015)
45. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting Semi-supervised Learning with Graph Embeddings. In: ICML. pp. 40–48 (2016)
46. Yao, L., Mao, C., Luo, Y.: KG-BERT: BERT for Knowledge Graph Completion. arXiv:1909.03193 (2019), http://arxiv.org/abs/1909.03193
47. Zhou, Z.H.: Ensemble learning. In: Machine learning, pp. 181–210. Springer (2021)