Semantic Web 0 (0) 1
IOS Press

Generalized neural embeddings for link prediction in knowledge graphs: training, evaluation, explanation

Asan Agibetov [a,*], Matthias Samwald [a]

[a] *Section for Artificial Intelligence and Decision Support, Medical University of Vienna, Vienna, Austria*
E-mail: asan.agibetov@meduniwien.ac.at

Abstract. Link prediction is the task of finding missing or unknown links among inter-connected entities in knowledge graphs. It can be accomplished with a classifier that outputs the probability of a link between two entities. However, the way in which entities and networks are represented strongly determines how well classification works. Recently, several works have successfully used neural networks to create entity embeddings which can be fed into binary classifiers. Moreover, it was proposed in literature that creating specialized embeddings separately for each relation type in the knowledge graph yields better classifier performance, as opposed to training a single embedding for the entire knowledge base. The drawback of these specialized approach is that they scale poorly as the number of relation types increases. In this work we formalize a unified methodology for training and evaluating embeddings for knowledge graphs, which we use to empirically investigate if, and when, the generalized neural embeddings – trained once on the entire knowledge graph – attain performance similar to specialized embeddings. This new way of training the neural embeddings and evaluating their quality is important for scalable link prediction with limited data. We perform an extensive statistical validation to empirically support our claims, and derive relation-centric connectivity measures for knowledge graphs to explain our findings. Our evaluation pipeline is made open source, and we aim to draw more attention of the community towards an important issue of transparency and reproducibility of the neural embeddings evaluations.

Keywords: knowledge graphs, neural embeddings, link prediction

1. Introduction

Link prediction is the task of finding missing or unknown links among inter-connected entities. This assumes that entities and links can be represented as a graph, where entities are nodes and links are edges (if relationships are symmetric) or arcs (if relationships are asymmetric). Link prediction was first characterized in the social network analysis community [1], but soon became popular in other domains such as large-scale knowledge bases [2], where it is used to add missing data and discover new facts. When dealing with link prediction in knowledge bases, the semantic information contained within is usually encoded as a knowledge graph (KG). For the purpose of this manuscript, we treat a knowledge graph as a graph where the links may have different types. We refer the reader to [3] for a more detailed presentation of knowledge graphs. In addition, we conform to the *closed-world* assumption. This means that all the existing (asserted) links are considered *positive*, and all the links which are unknown are considered *negative*. For what follows, consider a knowledge graph KG in Figure 1, consisting of three entities e_1, e_2, e_3 interconnected with links of two types r_1 and r_2, where bold arcs indicate the known links (positive), and the dotted arcs unknown (negative).

This separation into positive and negative links (examples) naturally allows us to treat the link prediction problem as a supervised classification problem with binary predictors. However, while this separa-

*Corresponding author. E-mail: asan.agibetov@meduniwien.ac.at.

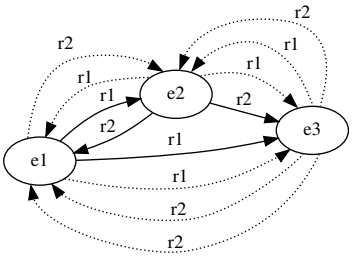

Fig. 1. A sample KG with three entities and two relation types. Positive links are drawn in bold, negative in dotted edge styles.

tion makes it possible to use a wide array of well-studied machine learning algorithms for link prediction, the main challenge is how to find the best representations for the links. This is the core subject of the recent research trend in learning suitable representations for knowledge graphs, largely dominated by so-called *neural embeddings* (initially introduced for language modeling [4]). Neural embeddings are numeric representations of nodes, and/or relations of the knowledge graph, in some continuous and dense vector space. These embeddings are learned through neural networks by optimizing a specific objective function. Usually, the objective function models the constraints that the neighboring nodes are embedded close to each other, and the nodes that are not directly connected, or separated via long paths in the graph, are embedded to stay far apart. A link in a knowledge graph is then represented as a combination of node and/or relation type embeddings. See [2, 5] for an overview of these approaches.

Our work is an extension of the framework to both learn and evaluate neural embeddings for the knowledge graphs, as proposed in [6]. Throughout this manuscript we refer to this approach as the *specialized embeddings* approach. This approach learns and evaluates specialized embeddings for each relation type r_i of entities of KG as follows: a) we generate the retained graph where we delete some of the triples involving r_i, then b) we compute the embeddings of the entities on this resulting retained graph, finally, c) we assess the quality of these specialized embeddings on relation r_i by training and testing binary predictors on positive and negative triples involving r_i. These three steps are detailed in Figure 2 in the *specialized embeddings* box. The arrows labeled with "a" in Figure 2 symbolize the generation of retained graphs for relations r_1 and r_2, those marked with "b" computation of the entity embeddings, and "c" represents the training and testing binary classifiers for each relation type. The drawback of this approach is that we need to compute entity embeddings for each relation type separately, this will soon become a scalability issue when the number of relation types in the knowledge graph becomes big.

1.1. Related work

There are two major ways of measuring the quality of (neural) embeddings of entities in a knowledge graph for link prediction tasks, inspired by two different fields: information retrieval [2, 7–10] and graph-based data mining [6, 11–15]. Information retrieval inspired approaches seem to favor node-centric evaluations, which measure how well the embeddings are able to reconstruct the *immediate* neighbourhood for each node in the graph; these evaluations are based on the mean rank measurement and its variants (mean average precision, top k results, mean reciprocal rank). Graph-based data mining approaches are based on the standard evaluation measurements of classifiers' performance based on false positive rate to true positive rate curves (e.g., ROC AUC, F-measure). In the literature, the former measures are also referred to as *node-equality*, and the latter as *link-equality* [16].

Some research examined issues that come up during the data splitting in train and test phases. The issue of imbalanced classes when the link prediction in graphs is treated as a classification problem is well presented in [13]. In the bioinformatics community the problem of imbalanced classes for the binary classification problem can be circumvented by considering negative links that have a biological meaning, thus omitting many potential negative links that are highly improbable biologically [6]. Other work demonstrated that if no care is taken while splitting datasets, one might end up producing biased train and test examples, such that implicit information from the test set may leak into the train set [17, 18]. Kadlec et al. [8] have mentioned that fair optimization of hyperparameters for competing approaches should be considered, as some of the reported KG completion results are significantly lower than what they potentially could be. Evaluation of machine learning tasks for the semantic web domain that use neural embeddings from the knowledge graphs is explored in [19, 20]. In the life sciences domain, the time-sliced graphs as generators for train and test examples have been proposed as a more realistic evaluation benchmark [16], as opposed to the randomly generated slices of graphs.

In [16, 18] authors use node-centric connectivity measures (e.g., incoming/outcoming node degree, or

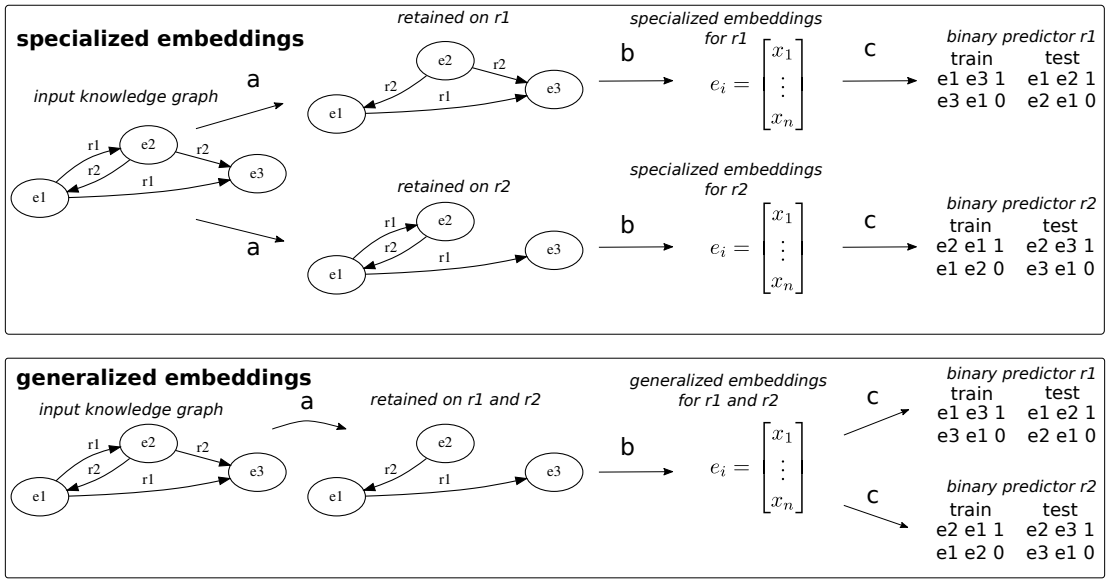

Fig. 2. Overview of the pipeline for training specialized and generalized neural embeddings for link prediction.

PageRank) to explain the performance of different neural embedding techniques.

1.2. Contribution of this work

We investigate empirically and analyze statistically the potential of training *generalized* neural embeddings for all relation types once, as opposed to training *specialized* embeddings for a specific relation r_i (Figure 2, *generalized embeddings* box). Specifically, we generate only one retained graph, where we delete a fraction of triples for each relation type r_i (arrow marked with "a" on the bottom of Figure 2). This retained graph is then used as a corpus for the computation of entity embeddings ("b"), which are then assessed with binary predictors for each relation type r_i as in the specialized case (arrows marked with "c" on the bottom of Figure 2). Evidently, this approach is more scalable and economic, since we only compute and keep one set of entity embeddings per knowledge graph. To explain the correlations with the classifiers' performance, we introduce *relation-centric* connectivity measures for the knowledge graphs to quantify these conditions. Additionally, in our analysis we take into consideration the ratios of missing examples during train and test phases. Our statistical validation tests the robustness of the neural embeddings, by evaluating their performances with different amounts of available information (i.e., percentage of available positive links), to simulate realistic scenarios where we have only limited data. We believe that the *relation-centric* connectivity measures, the analysis of missed examples, and statistical validation with limited data are an important addition to the state-of-the-art evaluation toolbox for knowledge graphs, which has not been proposed before. Our evaluation pipeline is formalized and made [1]open source, and with this we aim to draw more attention of the community towards an important issue of transparency and reproducibility of the results.

The rest of this manuscript is organized as follows: in the Methods section (Section 2) we present the datasets we used to evaluate our approach (Section 2.2), and we introduce formally our methodology (Section 2.3). In Section 3 we report our results and analysis, which we complement with a discussion on limitations of our approach and possible directions (Section 4). Finally, we conclude our manuscript in Section 5.

2. Methods

2.1. Notation and terminology

Throughout this manuscript we use a *triple*-oriented representation of knowledge graphs. As such, a knowl-

[1]https://github.com/plumdeq/neuro-kglink

edge graph KG is simply a set of triples $(e_i, r_i, e_j) \in KG$, where $e_i, e_j \in \text{ENTITIES}(KG)$ are some entities, and $r_i \in \text{RELATIONS}(KG)$ are its relation types, or simply relations. We assume that entities and relation types are disjoint sets, i.e., $\text{ENTITIES}(KG) \neq \text{RELATIONS}(KG)$. The character _ denotes a *free* variable in the triple, any value is assumed in its place. For instance, when we write $\forall(e_i, \_, e_j) \in KG$ we refer to a set of triples where e_i, e_j are fixed, and the relation type $\_ \in \text{RELATIONS}(KG)$ is free (i.e., all the links of different relation types that connect fixed entities e_i, e_j).

Let Pos_{KG} denote all the existing triples in the KG, i.e., triples $(e_i, r_i, e_j) \in KG$, and let Neg_{KG} denote the non-existing triples $(\bar{e}_i, \bar{r}_i, \bar{e}_j) \notin KG$, such that $\bar{e}_i, \bar{e}_j \in \text{ENTITIES}(KG)$ and $\bar{r}_i \in \text{RELATIONS}(KG)$. Similarly, Pos_{r_i} and Neg_{r_i} denote the existing and non-existing triples involving a relation r_i, respectively. Obviously, in every triple of Pos_{r_i} or Neg_{r_i} the relation type is fixed to r_i. For each relation type r_i, $\text{DOMAIN}(r_i) = \{e_i | \forall(e_i, r_i, \_) \in KG\}$, and $\text{RANGE}(r_i)\{e_j | \forall(\_, r_i, e_j) \in KG\}$ indicate the entities that belong to the domain and range of a relation r_i (i.e., left and right entities). To describe the process of sampling some triples, we use the notation $\alpha X, \alpha \in [0, 1]$, where X is any set of triples. For instance, $(\alpha = 0.8)Pos_{r_i}$ is a sampled set of triples involving r_i, and consisting of 80% of triples from Pos_{r_i}.

For each relation type $r_i \in \text{RELATIONS}(KG)$, $G_{r_i} = (V_{r_i}, E_{r_i})$ refers to a graph obtained by extracting all the triples involving the relation type r_i from the knowledge graph (i.e., $V_{r_i} = \text{DOMAIN}(r_i) \bigcup \text{RANGE}(r_i)$, and $E = \forall(\_, r_i, \_) \in KG$).

2.2. Datasets and their characterization

We run our experiments on four different knowledge graphs: WN11 [17] (subset of original WordNet dataset [21] brought down to 11 types of relations, and without inverse relation assertions), FB15k-237 [17] (a subset of Freebase knowledge graph [22] where inverse relations have been removed), UMLS (subset of the Unified Medical Language System [23] semantic network) and BIO-KG (comprehensive biological knowledge graph [6]). WN11, FB15k-237 and UMLS have been downloaded (December 2017) from the ConvE [18] [2]GitHub repository, and BIO-KG has been downloaded (September 2017) from the official link indicated in the [3]supplementary material for [6]. Details on the derivation of subsets for Wordnet and Freebase knowledge graphs can be found in [17, 18].

2.2.1. Properties of the knowledge graphs

Each of the four knowledge graphs has distinct properties. To better assess these differences, in Table 1 we provide different statistics of these four datasets. We describe a) *global properties* with the total number of types of relations and entities; b) *relation-based properties* with minimum, maximum and mean number of triples for all relations in a knowledge graph (Equations 1, 2, 3, respectively); and c) *multi-relatedness* – a term that we use to quantify how many pairs of entities are connected with more than one relation type (see Figure 3).

$$min_{|Pos_{r_i}|} = min(|Pos_{r_1}|, \ldots, |Pos_{r_n}|), \tag{1}$$

$$max_{|Pos_{r_i}|} = max(|Pos_{r_1}|, \ldots, |Pos_{r_n}|), \tag{2}$$

$$mean_{|Pos_{r_i}|} = 1/|\text{RELATIONS}(KG)| \sum_{r_i} |Pos_{r_i}|. \tag{3}$$

Formally, the *multi-relatedness* $\phi(KG)$ (Equation 4) is computed as the ratio of the size of the set of all triples, where a fixed pair of entities is connected with more than one link of different relation types, to the total number of triples in the knowledge graph.

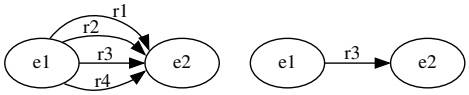

Fig. 3. One the left we have an example of a pair of entities (e_1, e_2) connected with four relation types, i.e., four positive links for the same entities. The same pair of entities is connected with only one relation type on the right. The *multi-relatedness* is bigger on the left.

$$\phi(KG) = \frac{|\{(e_i, r_1, e_j), \ldots, (e_i, r_n, e_j) | \forall(e_i, \_, e_j), n \geqslant 2\}|}{|Pos_{KG}|}. \tag{4}$$

2.2.2. Relation-based connectivity distributions

We define μ_{r_i} (Equation 5) as the measure of connectedness of G_{r_i}, a graph consisting of all the triples involving relation r_i of a knowledge graph (Figure 4)

$$\mu_{r_i} = \frac{|Pos_{r_i}|}{|\text{DOMAIN}(r_i)| \times |\text{RANGE}(r_i)|}. \tag{5}$$

[2]https://github.com/TimDettmers/ConvE

[3]http://aber-owl.net/aber-owl/bio2vec/bio-knowledge-graph.n3

dataset	Multi-relatedness			Relation-based properties			Global properties	
	$\phi(KG) \times \lvert Pos_{KG}\rvert$	$\lvert Pos_{KG}\rvert$	$\phi(KG)$ (%)	$max_{\lvert Pos_{r_i}\rvert}$	$min_{\lvert Pos_{r_i}\rvert}$	$mean_{\lvert Pos_{r_i}\rvert}$	$\lvert\text{RELATIONS}(KG)\rvert$	$\lvert\text{ENTITIES}(KG)\rvert$
WN11	124	93003	0.133 %	37221	86	8459	11	40943
FB15k-237	23700	310116	7.642 %	16391	45	1308	237	14541
UMLS	1343	6527	20.576 %	1021	1	142	46	137
BIO-KG	0	1619239	0.000 %	554366	6159	179915	9	346225

Table 1
Global and relation-specific properties of the four knowledge graphs used.

To this end, the maximum number of triples for relation r_i is attained when all entities in the domain of the relation are connected to all the entities in the range (i.e., $\mu_{r_i} = 1$, see Figure 4, right). Another interpretation of μ is the potential for the generation of negative examples, that is, low values of μ_{r_i} suggest that there might be many negative examples, i.e., the graph G_{r_i} is highly incomplete. We hypothesize that the performance of a binary link predictor of type r_i should be positively correlated with μ_{r_i}, i.e., the more training examples of type r_i there are (the more connected G_{r_i} is) the better is the performance of the binary predictor for r_i.

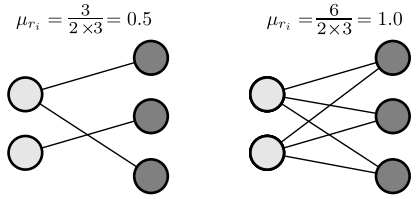

Fig. 4. μ_{r_i} - measure of connectedness of graph G_{r_i} for the relation r_i. Lighter nodes indicate *domain* and darker nodes indicate *range* of r_i.

Distribution of μ_{r_i} for all relations r_i in each of the four graphs is shown in Figure 5. We also provide the distributions of normalized number of triples z_{r_i} (Equation 6) for each relation r_i for all knowledge graphs. We normalize the number of triples for each relation type, where 0 (minumum) is the relation with the least number of triples, and 1 (maximum) is the relation with the biggest number of triples in the knowledge graph, with

$$z_{r_i} = \frac{|Pos_{r_i}| - min_{|Pos_{r_i}|}}{max_{|Pos_{r_i}|} - min_{|Pos_{r_i}|}}. \quad (6)$$

All distributions in Figure 5 are estimated and normalized with kernel density interpolation from the actual histograms.

2.2.3. Connectivity characterization of datasets

Globally, we have one small (in terms of number of entities) knowledge graph (UMLS), two medium-sized graphs (WN11, FB15K-237) and one very large biological graph (BIO-KG). UMLS is however much denser (in terms of multi-relatedness $\phi(KG)$) and has a more uniform μ_{r_i} distribution over all relation types as compared to all the others. We hypothesize that this denseness is what makes relation-based link prediction an easier task for binary classifiers. WN11 and BIO-KG have similar μ_{r_i} and z_{r_i} distributions, for both the graphs G_{r_i} are sparsely connected. Differently to WN11, BIO-KG has much more training examples for the classifier (i.e., $mean_{|Pos_{r_i}|}(\text{BIO-KG}) \gg mean_{|Pos_{r_i}|}(\text{WN11})$). Overall, this variability in the properties allows us to better evaluate our approach and identify the specific use-cases when such an approach for link prediction will be most beneficial.

2.3. Link prediction with neural embeddings

Herein, we formalize the pipeline for link prediction with specialized and generalized neural embeddings, which we briefly presented in Figure 2. The pipeline consists of three main steps: a) generation of retained graphs and train/test examples, b) training neural embeddings on the retained graphs, c) assessment of the quality of the neural embeddings with binary predictors. In the rest of this section we give a thorough description of these steps.

2.3.1. Generation of retained graphs (step a)

The preparation of datasets for neural embedding training is inspired by the methodology presented in [6]. In the following we present our generalized approach to this problem, as we think it is crucial for the transparent and reproducible evaluation pipeline, and has not been detailed enough in [6]. In addition to *specialized* retained graphs, where only triples of a specified relation r_i were removed (as used in [6]), we also consider *generalized* retained graphs for all relations $\forall r_i \in KG$.

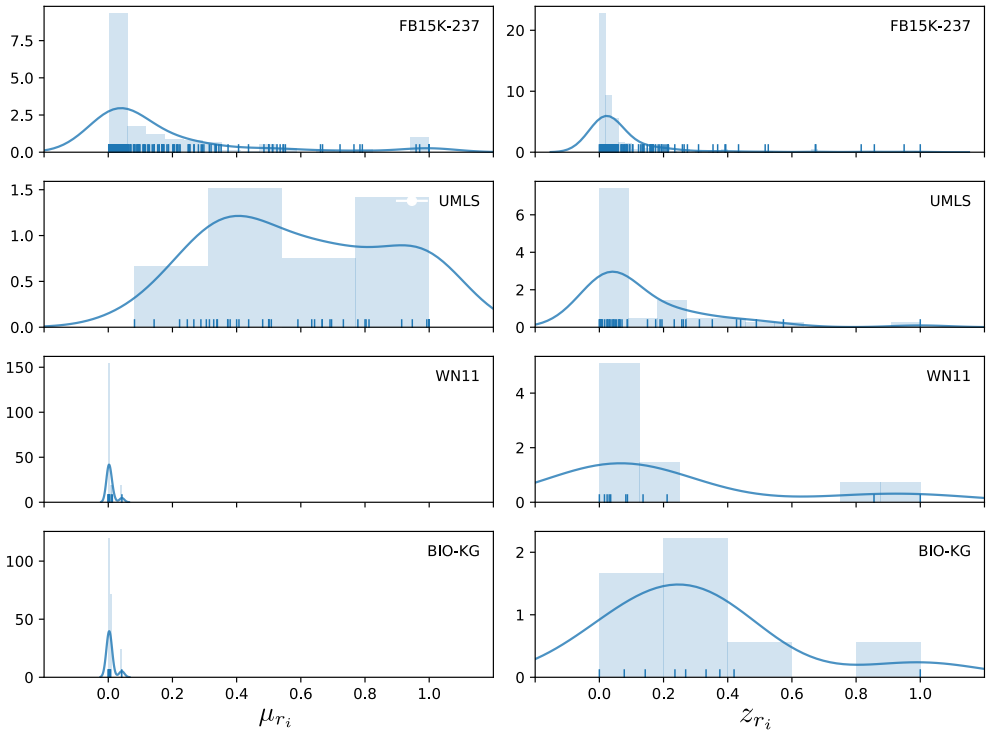

Fig. 5. Distributions of connectivity measure (μ_{r_i}) and the normalized number of available triples (z_{r_i}) per relation for four graphs. Left-skewed distribution of μ_{r_i} reveals sparse connectivity of the knowledge graph (i.e., the graph is highly incomplete). Uniform distribution of z_{r_i} represents a well-balanced prediction task, i.e., each classifier will have an equal amount of train/test examples.

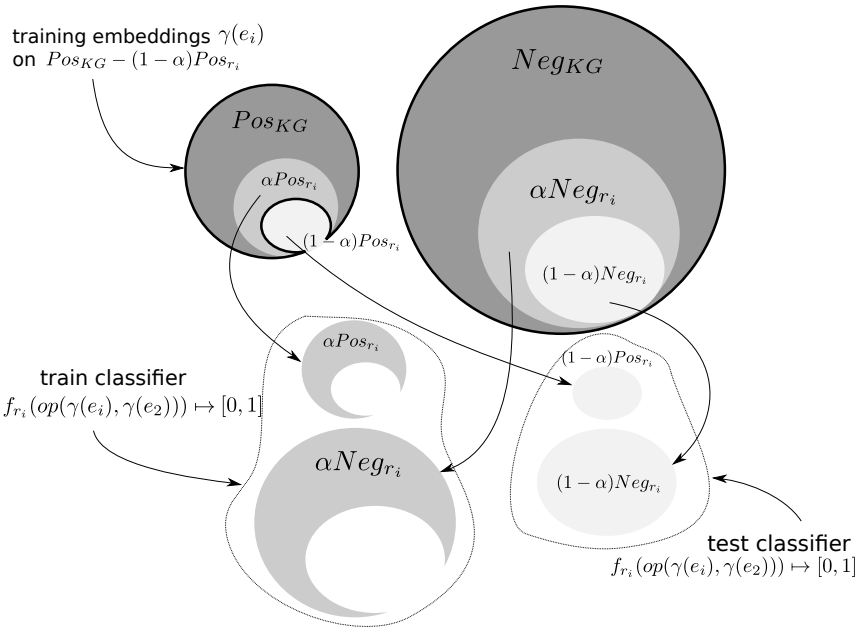

Fig. 6. Schematic representation of the pipeline for the evaluation of the embeddings. Neg_{KG} and its derivations (e.g., αNeg_{r_i}) appear bigger visually to indicate that the elements are sampled from a much bigger set of all possible negative links.

By treating the problem of evaluation of the quality of the embeddings in a set-theoretic approach, we can define the following *datasets*:

1. $Pos_{KG} - (1-\alpha)Pos_{r_i}$ a *specialized* retained graph on r_i (in Figure 6 this set is demarcated with bold contour in the upper left corner),
2. $Pos_{KG} - \bigcup_{r_i}(1-\alpha)Pos_{r_i}$ a *generalized* retained graph on all relations r_i ,
3. $\forall r_i, \alpha Pos_{r_i} \bigcup \alpha Neg_{r_i}$ – train examples for the binary classifier for r_i,
4. $\forall r_i, (1-\alpha)Pos_{r_i} \bigcup (1-\alpha)Neg_{r_i}$ – test examples for the binary classifier for r_i.

We also note the following properties, which must hold and serve as validation criteria for the generation of train and test data. In particular,

1. $Pos_{KG} - (1-\alpha)Pos_{r_i} \subseteq Pos_{KG}, Pos_{KG} - \bigcup_{r_i}(1-\alpha)Pos_{r_i} \subseteq Pos_{KG}$ retained graphs must be subgraphs of full graph,
2. $Pos_{KG} - \bigcup_{r_i} \alpha Pos_{r_i} = \bigcup_{r_i}(1-\alpha)Pos_{r_i}$ the difference between the full and the retained generalized graph are the positive links, which we use in the test set, i.e., the embeddings will be used to predict these positive links,
3. $\forall r_i, \alpha Pos_{r_i} \bigcap (1-\alpha)Pos_{r_i} = \varnothing, \alpha Neg_{r_i} \bigcap (1-\alpha)Neg_{r_i} = \varnothing$ there should be no link shared between the train and test sets,
4. $\forall r_i \forall x_i, \bar{x_i} \in Neg_{r_i}, \bar{x_i} \notin KG$ all generated negative links do not exist in the original full graph.

The last two properties reflect what is usually done in the literature during the generation of negative links, and for this work we also conform to these two properties. However, we elaborate more on the issue, where the negative examples set generation is disjoint from the positive examples set in Section 4.2.2.

2.3.2. Neural embedding model (step b)

In this work we employ a shallow unsupervised neural embedding model [15], which aims at learning *entity embeddings* in a dense d-dimnesional vector space. The model is simple and fast, and it embeds the entities that appear in the positive triples close to each other, and places the entities that appear in negative triples farther appart. As in many neural embedding approaches, the weight matrix of the hidden layer of the neural network serves the role of the look-up matrix (the matrix of embeddings - latent vectors). The neural network is trained by minimizing, for each positive triple $x_i = (e_i, \_, e_j)$ in the specialized ($x_i \in Pos_{KG} - (1-\alpha)Pos_{r_i}$), or generalized graphs ($x_i \in Pos_{KG} - \bigcup_{r_i}(1-\alpha)Pos_{r_i}$), the following loss function

$$\sum_{(e_i,\_,\bar{e_j}) \in Neg_{KG}} \psi(e_i, e_j), \psi(e_i, \bar{e_1}), \ldots, \psi(e_i, \bar{e_k}).$$

Where, for each positive triple x_i, we embed entities e_i, e_j close to each other, such that e_i stays as far as possible from the k *negative* entities $\bar{e_1}, \ldots, \bar{e_k}$. The similarity function ψ is task-dependent and should operate on d-dimensional vector representations of the entities (e.g., standard Euclidean dot product).

2.3.3. Missing examples at train and test phases

Since we are learning embeddings for the entities from the retained graphs (some links are excluded), the algorithm may *miss* to learn an embedding for an entity, which will result in missing examples at train or test phases. To see it clearer, suppose that during the generation of the retained graph all triples $(e_k, \_, \_) \in KG$ (only e_k fixed) involving entity e_k are not assigned to the retained graph (i.,e., $(e_k, \_, \_) \notin Pos_{KG} - (1-\alpha)Pos_{r_i}$), then the algorithm will not learn an embedding $\gamma(e_k)$, which will lead us to a situation where all the triples $(e_k, \_, \_) \in KG$ (examples at train/test) will be missing during training or testing of the binary classifier (depending whether these are assigned to to the train or test sets). Intuitively, the amount of possible missing examples is inversely proportionate to the α parameter, i.e., the more information we include during the embedding learning phase, the fewer embeddings will be missed. We also hypothesize that the sparsely connected relations (i.e., sparsely connected graphs G_{r_i}) and the relations with very few examples (i.e., small $|Pos_{r_i}|$) will yield higher amounts of missed examples.

2.3.4. Link prediction evaluation with binary classifiers (step c)

To quantify confidence in the trained embeddings, we perform the repeated random sub-sampling validation for each classifier f_{r_i}. That is, for each relation r_i we generate k times: retained graph $Pos_{KG} - (1-\alpha)Pos_{r_i}$ corpus for unsupervised learning of entity embeddings $\gamma_{r_i}(e_i)$) and train $\alpha Pos_{r_i} \bigcup \alpha Neg_{r_i}$ and test $(1-\alpha)Pos_{r_i} \bigcup (1-\alpha)Neg_{r_i}$ splits of positive and negative examples. Link prediction is then treated as a binary classification task with a classifier $f_{r_i} : op(\gamma(e_i), \gamma(e_j))) \mapsto [0,1]$, where op is a binary operator that combines entity embeddings $\gamma(e_i), \gamma(e_j)$ into one single representation of the link (e_i, r_i, e_j) (e.g., element-wise sum or multiplication, or vector concatenation). The performance of the classifier is measured

with the standard performance measurements (e.g., F-measure, ROC AUC).

2.4. Experiments

Algorithm 1 Evaluation of specialized and generalized knowledge graph embeddings

Precondition: KG, α, J

for each sub-sample validation run
for $j \in 1, \ldots, J$ **do**
  generate retained graph on ***all*** r_i*,*
  and compute ***generalized*** *embeddings*
    $X \leftarrow Pos_{KG} - \bigcup_{r_i}(1-\alpha)Pos_{r_i}$
    $\gamma(e_i) \leftarrow \text{Embeddings}(X)$

  for $r_i \in \text{RELATIONS}(KG)$ **do**
    generate retained graph on r_i*,*
    and compute ***specialized*** *embeddings*
      $X_{r_i} \leftarrow Pos_{KG} - (1-\alpha)Pos_{r_i}$
      $\gamma_{r_i}(e_i) \leftarrow \text{Embeddings}(X_{r_i})$

    generation of train/test examples for r_i
      $train_{r_i} \leftarrow \alpha Pos_{r_i} \bigcup \alpha Neg_{r_i}$
      $test_{r_i} \leftarrow (1-\alpha)Pos_{r_i} \bigcup (1-\alpha)Neg_{r_i}$
    evaluate quality of specialized embeddings
      $F1^j_{r_i} \leftarrow f_{r_i}(train_{r_i}, test_{r_i}, \gamma_{r_i}(e_i))$
  end for
    evaluate quality of generalized embeddings
  $F1^j \leftarrow f_{r_i}(train_{r_i}, test_{r_i}, \gamma(e_i))$
end for

average specialized embeddings evaluations
for $r_i \in \text{RELATIONS}(KG)$ **do**
  $\widetilde{F1}_{r_i} \leftarrow \sum_j F1^j_{r_i}$
end for

average generalized embeddings evaluations
$\widetilde{F1} \leftarrow \sum_j F1^j$

The goal of our experiments is to empirically investigate if, and when, the generalized neural embeddings attain similar performance as the specialized embeddings, for the four considered datasets. To do so, we first generate the retained graphs, and the train and test datasets. The retained graphs are generated for each relation type r_i in the case of specialized embeddings, and only once for the generalized embeddings. We always keep the 1:1 ratio for the positive and negative examples. When we sample the negatives for a relation r_i, we only consider the triples $(\bar{e_i}, r_i, \bar{e_j})$ where the entities come from the domain ($\bar{e_i} \in \text{DOMAIN}(r_i)$) and the range ($\bar{e_i} \in \text{RANGE}(r_i)$) of r_i. The embeddings are computed from the retained graphs, and then evaluated on the train and test datasets. Note that we only provide the results for the generalized embeddings for FB15k-237, since the computation of specialized embeddings for 237 relations of FB15k-237 would take months (on our machine) to finish (the computation of specialized embeddings grows linearly with the number of relations, and exponentially in the number of repeated sub-sample validation runs). The evaluation of the embeddings for one relation type r_i is performed with the logistic regression classifier $f_{r_i}(concat(\gamma(e_i), \gamma(e_j)))$ (we use vector concatenation for link representation in this work, see Section 4.2.1 for a detailed discussion on the choice of operators). To test the robustness of the embeddings we perform the evaluations with limited information, i.e., the size of the retained graphs controlled by $\alpha \in \{20, 50, 80\}$, and we analyze the amount of missed embeddings in all experiments. All of our results are presented as averages of 10 repeated random sub-sampling validations. We thus report mean F-measure scores and their standard deviations. This pipeline is formalized in Algorithm 1.

The neural embeddings are trained with the StarSpace toolkit [9] in train mode 1 (see StarSpace specification) with the fixed hyperparameters: the embedding size is set to $d = 50$, and number of epochs is set to 10. All other hyperparameters for StarSpace are set to default, in particular, the maximum number of negative entities per one positive triple is 10 ($k = 10$). Classification results are obtained with the scikit Python library [24], grouping of classification results and their statistical analysis are performed with Pandas [25]. All of our experiments were performed on a modern desktop PC with a quad core Intel i7 CPU (clocked at 4GHz) and 32 Gb of RAM.

2.4.1. Hyperparameter search

While we report our statistical evaluation only for a fixed set of hyperparameters for the neural embedding model, we did experiment with different setting of the hyperparameters. In particular, we experimented with the different dimension sizes ($d = 5, 10, 20, 50, 100, 512$), and with the different number of epochs ($e = 5, 10, 20, 50$) for a smaller number of validation runs, and on medium-sized knowledge graphs only. In general, only slight improvements on the classification performance were observed. Convergence of distributions for specialized and generalized embeddings being our primary target, we decided to perform full-scale statistical evaluation with the chosen fixed set of hyperparameters ($d = 50, e = 10$).

3. Results

3.1. Generalized vs. specialized embeddings: comparing distributions

In Figure 7 we present the distributions of averaged F1 scores, which measure the performance of the embeddings, and the ratios ($[0, 1]$) of missed examples at training and testing of the binary classifiers f_{r_i}. As such, the overall performance of the specialized or generalized embeddings on one knowledge graph is characterized by these three distributions over all relations in the given knowledge graph. The performance of embeddings is compared with varying amount of information present at the time of training of the neural embeddings (parameter α). All distributions in Figure 7 are estimated and normalized with kernel density interpolation from the actual histograms.

In the three knowledge graphs: BIO-KG, UMLS and WN11, the distributions converge as we increase the amount of available information (e.g., $\alpha \to 1$), which supports the main hypothesis of this manuscript, that the generalized embeddings may yield the similar (if not the same) performance as the specialized embeddings. When we consider BIO-KG, the F1 and missing examples distributions for the specialized and generalized neural embeddings converge almost to identical distributions, even when the overall amount of information is low (e.g., only 20 % of available triples). This may be explained by a relatively big size of available positive examples per relation type $|Pos_{r_i}|$ (hundreds of thousands of available triples per relation). Though, distributions of z_{r_i} and μ_{r_i} (Figure 5) are very similar for BIO-KG and WN11, the differences between the specialized and generalized embeddings for WN11 are much more characterized, than in the case of BIO-KG. In particular, the neural embeddings for WN11 are very sensitive to α, the less information there is the more the generalized and specialized distributions diverge (different shapes of distributions for F1 and the ratio of missed examples). The amount of missed examples is very high for both specialized and generalized cases, for smaller values of $\alpha < 0.8$, and the distributions converge when $\alpha = 0.8$. The most regular behavior is demonstrated by the neural embeddings trained on UMLS corpora, where the missing examples rates are all almost zero, even when $\alpha = 0.2$. The shapes of the F1 distributions are very similar for all the values of α, the intra-discrepancies are very low. These observations allow us to hypothesize that the similar trends might exist for the FB15k-237 knowledge graph, since UMLS and FB15k-237 have similar uniform distributions of μ_{r_i} and z_{r_i}.

To summarize, as we increase the amount of available information during the computation of neural embeddings ($\alpha \to 1$) the intra-discrepancies between the specialized and the generalized embeddings become negligible. And this is good news, since training generalized embeddings is $|\text{RELATIONS}(KG)|$-times faster than training the specialized embeddings for each relation r_i, with the strong evidence that if we have enough information we can achieve the same performance.

3.2. Generalized vs. specialized embeddings: comparing means and stds of distributions

We recall that each distribution's sampled point is obtained by averaging the results of k repeated experiments for one relation r_i. To directly compare the distributions, we compare their means and standard deviations, and, as such, we are comparing the average performance of $|\text{RELATIONS}(KG)|$ binary classifiers for specialized and generalized neural embeddings, with the varying parameter α. Figure 8 depicts the average performance of all binary classifiers and its standard deviation for the four knowledge graphs. As expected, the performance of the specialized embeddings is better than the performance of the generalized embeddings, however the differences are very slim. BIO-KG and UMLS demonstrate that, as we increase α, the average F1 score increases in both cases, however, so does the standard deviation as well. WN11, on the other hand, demonstrates a counter-intuitive trend where the best performance of specialized embeddings occurs when less information is available. For the specialized embeddings, the F1 score decreases slightly when we include more information during the computation of neural embeddings. This may be explained by an increased amount of missing examples, both during training and testing of the binary classifier. Due to a very sparse connectivity of graphs G_{r_i} of WN11, when we only consider 20% of available triples – we exclude 80 % of available links –, many entities are likely to become disconnected. This means that no embeddings are learned for them, and, as a result, the binary classifier is both trained and tested on fewer examples.

3.3. Correlation of average performance with the connectivity

Our experiments show that almost in all settings there is a positive correlation between the performance of a

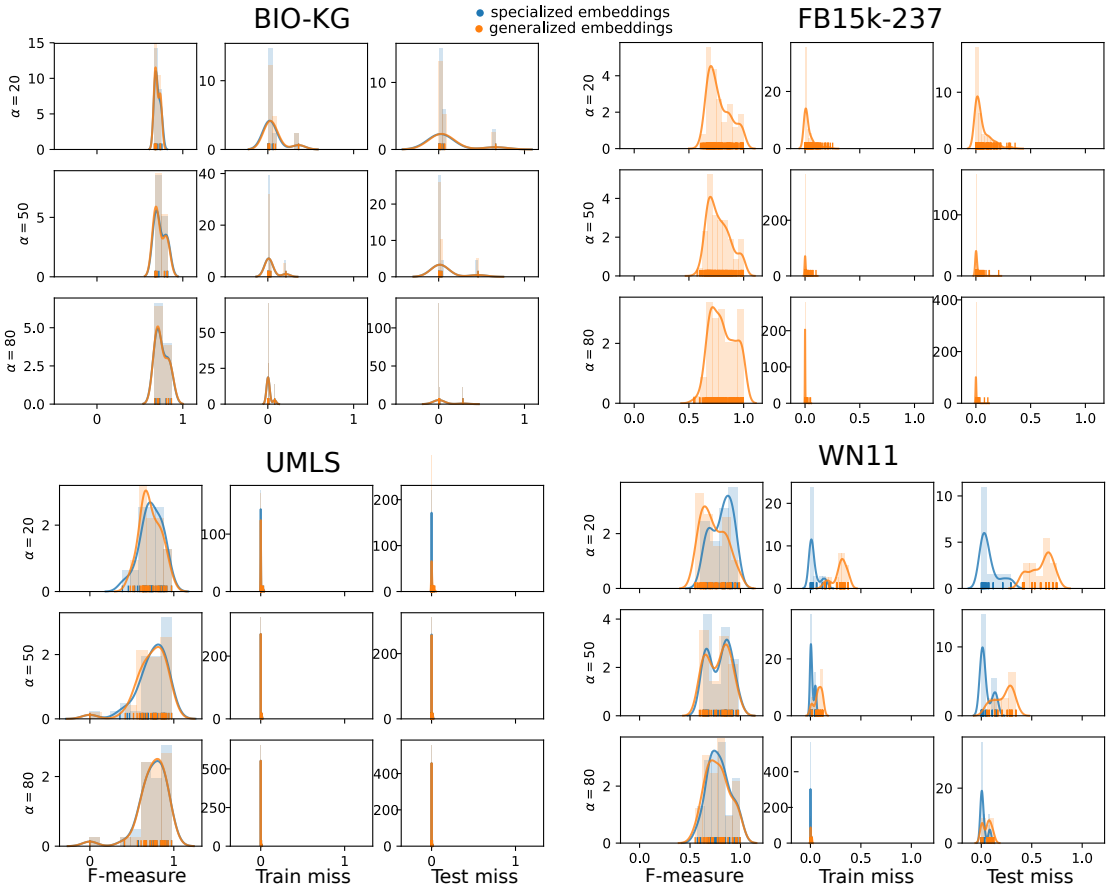

Fig. 7. Averaged distributions of F1 scores, train and test missed examples ratio, for generalized and specialized neural embeddings, for four datasets. These distributions reflect the average performance of f_{r_i} binary classifiers for each relation type r_i that operate on learned embeddings. Robustness of classifiers is measured by varying the amount of positive links available during learning phase of the embeddings (i.e., sampling controlled by α). Due to time constraints we did not compute the generalized distributions for FB15k237. For BIO-KG, UMLS and WN11 the averaged distributions for specialized and generalized embeddings converge.

binary classifier f_{r_i} for a specific relation r_i, and μ_{r_i}. Figure 9 depicts the this dependency for four knowledge graph for the varying parameter α, and the Figure 10 summarizes it with the Pearson correlation scores (we also report Pearson correlation with z_{r_i}). We found out that the performance of binary classifiers evaluated on UMLS knowledge graph exhibits the strongest positive correlation with μ_{r_i} for both specialized and generalized embeddings (Figure 9). The higher the connectivity of the graph G_{r_i} ($\mu \to 1$) the better is the performance of the binary classifier f_{r_i} (F1 $\to$ 1). The dependency F1 vs. μ is positive, yet slightly smaller, for all other datasets. The Pearson coefficients for BIO-KG and WN11 should be interpreted with caution, since the variance of the μ_{r_i} distributions for these two is significantly smaller than that for the other datasets (Figure 5). We also observe positive correlations of the performance of the classifiers with the size of the available positive triples for a given relation z_{r_i} for BIO-KG and UMLS. This trend is however broken by the negative correlations for FB15k-237 and WN11, which leads to believe that μ_{r_i} is a better candidate to explain positive correlations of the classifiers' performance.

4. Discussion

The main insight that gained from our experiments is that as long as the knowledge graph exhibits densely connected structure across all the relations, or contains hundreds of thousands of available triples for each relation type, the performance of the specialized embed-

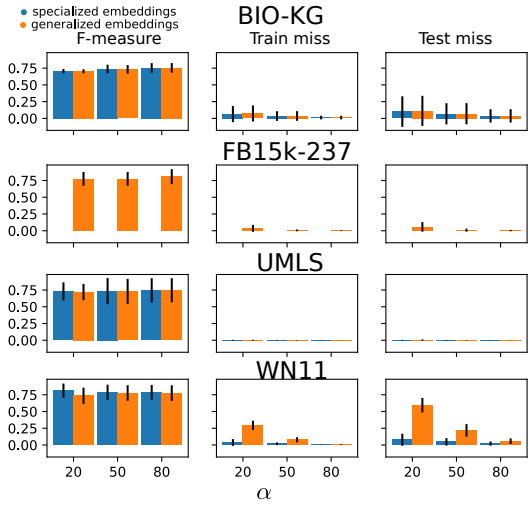

Fig. 8. Averaged F1 and train/test missed example ratio over all relations for four graphs.

dings for link prediction will be quasi-equal to that of the generalized embeddings. In all cases that means $|\text{RELATIONS}(KG)|$-times faster training times for the embeddings and $|\text{RELATIONS}(KG)|$-times less memory to store the embeddings. If researchers work with small or medium-sized, but dense knowledge graphs, and they aim at fast and scalable training of neural embeddings, generalized embeddings approach should be their primary target. We would also recommend StarSpace as the implementation of shallow neural embedding models, as its optimized implementation is often more scalable than an equivalent neural embedding model in [4]Pytorch. Also, we found that the statistical evaluation of the performance of the binary classifiers with repeated sub-sampling yields very small variance for the considered datasets, we thus encourage the researchers to invest their time in preparing less computationally intensive validation schemes (i.e., well prepared hold-out test sets). We would also suggest to invest more time to focus on the structural analysis of the knowledge graphs and their correlation with the classifiers' performance.

In what follows we outline some considerations which might affect link prediction experiments and limit the applicability of our approach.

[4]https://pytorch.org

4.1. Neural embedding model complexity

The focus of our study was not on obtaining best classification performance, but rather supporting statistically and empirically the hypothesis that cheaper and more scalable generalized embeddings can attain performance similar to more computationally expensive specialized embeddings. In doing so, we sought to use a simpler neural embedding model for faster, more scalable and broader evaluation of our results. The neural embedding model we use is a model that is shallow, in the number of model parameters (e.g., number of hidden layers), and first-order, in terms of how much neighborhood information is used. More complex models with deeper and more sophisticated architectures (i.e., involving more fully-connected and/or convolutional hidden layers [5, 18]), and/or models taking into account higher order neighborhood information (i.e., models based on random walks [6, 11], and/or based on graph depth traversal [12]) have not been considered in this work. These models most likely would yield superior classification results. However, we have not investigated the convergence of the overall performance distributions of specialized and generalized embeddings, trained with these models, due to increased complexity (time-wise) for statistical evaluations. Evaluation of performance distributions of binary predictors with generalized and specialized embeddings, trained with complex neural embedding models, would definitely be an interesting extension of this work.

4.2. Link prediction

4.2.1. Link representation or node combination

Based on the embeddings of the nodes of the graph, we can come up with different ways of representing a link between an entity e_i and e_j. This is usually achieved with a binary operator (or a node combination strategy) that transforms entity embeddings representations $\gamma(e_i), \gamma(e_j)$ into one single representation of the link (e_i, r_i, e_j). Popular choices for this operator include operations that preserve the original d dimension of the entity embeddings to represent links (e.g., element-wise sum or mean [12]), as well as the operations that concatenate entity embeddings [15, 16, 18]. In our experiments with 10 cross-validation runs for the WN11 we observed that the concatenation operator outperformed other considered binary operators, and that all the considered node combination strategies resulted in the convergence of performance distributions for spe-

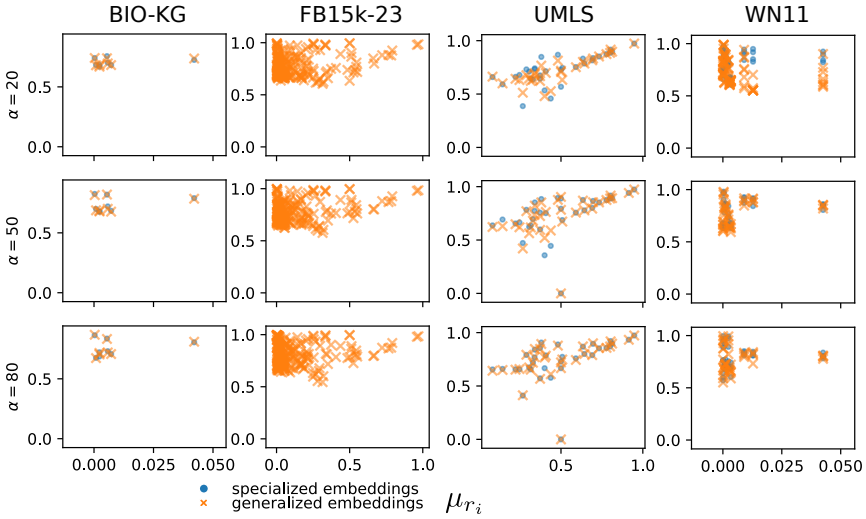

Fig. 9. Dependence of classifier performance (measured with F1) against the connectivity measure μ for each relation type.

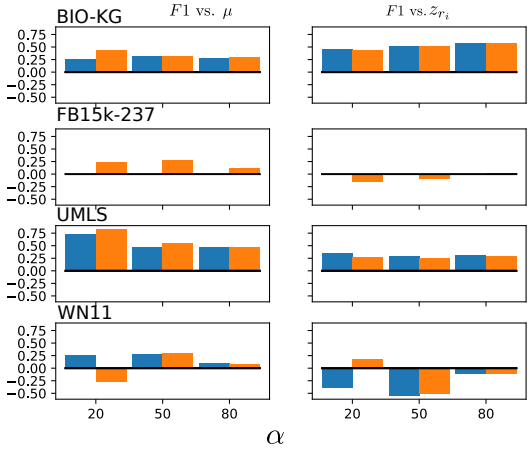

Fig. 10. Pearson correlations to summarize the dependence of classifier performance (measured with F1) against the connectivity measure μ and the normalized number of triples z_{r_i} for each relation type.

cialized and generalized embeddings [15]. In this work we thus stuck with the concatenation as our binary operator for a broader evaluation with additional datasets. Using different node combination strategies on different knowledge graphs, may or may not have significant impact on the performance.

4.2.2. *Negative sampling*

In general, the link prediction problem for knowledge graphs is different from other classification problems where positive and negative examples are well defined, and the *iid* (independent and identically distributed random variables) assumption is valid. Obtaining a representative test set with a prototypical distribution is often not trivial [13], and usually what is done is that we randomly remove some links which we then use as our test positives. Moreover, during the generation of negative links both for train and test sets, we impose that no negative link appears as a training positive or test positive. We therefore implicitly leak information about the test positives when we generate train negatives. In other words, during the generation of negative links $(\bar{e}_i, r_i, \bar{e}_j) \notin Pos_{KG})$ we should account to the possibility that this link might actually turn out to be true, and our binary classifiers should be robust and generalize well to these realistic situations. In our preliminary investigations on the impact of this bias we have not seen big discrepancies in the classifiers' performance, however, a more elaborate analysis into this issue would be very interesting to the community.

4.3. *Properties of knowledge graphs*

In our experiments we considered *relation-centric* metrics $\phi(KG), \mu_{r_i}$ and z_{r_i} to characterize the knowledge graphs. In particular, we used μ_{r_i} and z_{r_i} to identify the correlations of classifier performances, and studied the impact of *multi-relatedness* $\phi(KG)$ on overall classifier performance. While we showed that they help identify the use-cases when the generalized embeddings perform almost identical on average as the specialized embeddings, other characterizations of knowledge graph may better explain other trends. For instance, we have not observed conclusive correlations

of *relation-centric* metrics with the ratio of missing examples at train/test phases. Perhaps *entity-centric* metrics (e.g., incoming and/or outcoming node degree, PageRank centrality) might be more suitable metrics to characterize missing ratios.

5. Conclusions

In this paper we focused on link prediction for knowledge graphs, treated as a binary classification problem on entity embeddings, trained in two different modes: specialized for each relation in the knowledge graph, and generalized for the whole knowledge graph. The advantage of generalized embeddings is that one only needs to train the embeddings once, as opposed to training embeddings for each relation type, as it is usually done in the literature. We used a simple, shallow and first-order neural embedding model to statistically evaluate and demonstrate, that the generalized embeddings provide on average a similar, or even, under certain constraints, quasi-equal performance as the specialized embeddings. The main advantage of the generalized embeddings is that they are much more scalable and require less memory space than the specialized embeddings. In our analysis we compared the distributions of classifiers' performances for all relations in the knowledge graph, and outline the conditions, which strengthen the convergence of performance distributions for the specialized and generalized embeddings. We introduced *relation-centric* connectivity measures for the knowledge graphs to quantify these conditions, and to explain the correlations with the classifiers' performance. In particular, we found that the performance distributions converge even with very limited available information (20% of all positive triples), provided that the knowledge graph is dense and exhibits uniform connectivity over all relation types. However, the performance distributions would need more available information to converge, for the knowledge graphs with sparse connectivity and the low number of positive triples per relation type. Finally, we made our code for the evaluation pipeline for link prediction tasks open source, and hope that it will contribute to the standardization of neural entity embedding evaluation benchmarks.

References

[1] D. Liben-Nowell and J. Kleinberg, The link prediction problem for social networks, in: *Proceedings of the twelfth international conference on Information and knowledge management - CIKM '03*, ACM Press, New York, New York, USA, 2003, p. 556. ISBN 1581137230. doi:10.1145/956863.956972. http://portal.acm.org/citation.cfm?doid=956863.956972.

[2] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proc. IEEE* **104**(1) (2016), 11–33, ISSN 0018-9219. doi:10.1109/JPROC.2015.2483592. http://ieeexplore.ieee.org/document/7358050/.

[3] S.N. Sri Nurdiati and C. Hoede, *25 years development of knowledge graph theory: the results and the challenge*, Memorandum Vol. 2/1876, Discrete Mathematics and Mathematical Programming (DMMP), 2008.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, Distributed Representations of Words and Phrases and their Compositionality, *arXiv* (2013). https://arxiv.org/abs/1310.4546.

[5] P. Goyal and E. Ferrara, Graph embedding techniques, applications, and performance: A survey, *Knowledge-Based Systems* **151** (2018), 78–94, ISSN 0950-7051. doi:https://doi.org/10.1016/j.knosys.2018.03.022. http://www.sciencedirect.com/science/article/pii/S0950705118301540.

[6] M. Alshahrani, M.A. Khan, O. Maddouri, A.R. Kinjo, N. Queralt-Rosinach and R. Hoehndorf, Neuro-symbolic representation learning on biological knowledge graphs., *Bioinformatics* **33**(17) (2017), 2723–2730. doi:10.1093/bioinformatics/btx275. http://dx.doi.org/10.1093/bioinformatics/btx275.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data (2013). https://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.

[8] R. Kadlec, O. Bajgar and J. Kleindienst, Knowledge Base Completion: Baselines Strike Back, *arXiv:1705.10744 [cs]* (2017). http://arxiv.org/abs/1705.10744.

[9] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes and J. Weston, StarSpace: Embed All The Things!, *arXiv* (2017). https://arxiv.org/abs/1709.03856.

[10] M. Nickel and D. Kiela, Poincaré Embeddings for Learning Hierarchical Representations, *arXiv:1705.08039 [cs, stat]* (2017). http://arxiv.org/abs/1705.08039.

[11] B. Perozzi, R. Al-Rfou and S. Skiena, DeepWalk: Online learning of social representations, in: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, ACM Press, New York, New York, USA, 2014, pp. 701–710. ISBN 9781450329569. doi:10.1145/2623330.2623732. http://dl.acm.org/citation.cfm?doid=2623330.2623732.

[12] A. Grover and J. Leskovec, node2vec: Scalable Feature Learning for Networks., *KDD* **2016** (2016), 855–864. doi:10.1145/2939672.2939754. http://dx.doi.org/10.1145/2939672.2939754.

[13] D. Garcia-Gasulla, E. AyguadÃl', J. Labarta and U. CortÃl's, Limitations and Alternatives for the Evaluation of Large-scale Link Prediction, *arXiv* (2016). https://arxiv.org/abs/1611.00547.

[14] B.P. Chamberlain, J. Clough and M.P. Deisenroth, Neural Embeddings of Graphs in Hyperbolic Space, *arXiv:1705.10359 [cs, stat]* (2017). http://arxiv.org/abs/1705.10359.

[15] A. Agibetov and M. Samwald, Global and local evaluation of link prediction tasks with neural embeddings, *arXiv* (2018). https://arxiv.org/abs/1807.10511.

[16] G. Crichton, Y. Guo, S. Pyysalo and A. Korhonen, Neural networks for link prediction in realistic biomedical graphs: a multi-dimensional evaluation of graph embedding-based approaches., *BMC Bioinformatics* **19**(1) (2018), 176. doi:10.1186/s12859-018-2163-9. http://dx.doi.org/10.1186/s12859-018-2163-9.

[17] K. Toutanova, V. Lin, W.-t. Yih, H. Poon and C. Quirk, Compositional learning of embeddings for relation paths in knowledge base and text, in: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2016, pp. 1434–1444. doi:10.18653/v1/P16-1136. http://aclweb.org/anthology/P16-1136.

[18] T. Dettmers, P. Minervini, P. Stenetorp and S. Riedel, Convolutional 2D Knowledge Graph Embeddings (2017). https://arxiv.org/abs/1707.01476.

[19] P. Ristoski, G.K.D. de Vries and H. Paulheim, A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web, in: *The semantic web âĂŞ ISWC 2016*, P. Groth, E. Simperl, A. Gray, M. Sabou, M. KrÃűtzsch, F. Lecue, F. FlÃűck and Y. Gil, eds, Lecture notes in computer science, Vol. 9982, Springer International Publishing, Cham, 2016, pp. 186–194, ISSN 0302-9743. ISBN 978-3-319-46546-3. doi:10.1007/978-3-319-46547-0_20. http://link.springer.com/10.1007/978-3-319-46547-0_20.

[20] M. Cochez, P. Ristoski, S.P. Ponzetto and H. Paulheim, Global RDF vector space embeddings, in: *The semantic web âĂŞ ISWC 2017*, C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. CudrÃł-Mauroux, J. Sequeda, C. Lange and J. Heflin, eds, Lecture notes in computer science, Vol. 10587, Springer International Publishing, Cham, 2017, pp. 190–207, ISSN 0302-9743. ISBN 978-3-319-68287-7. doi:10.1007/978-3-319-68288-4_12. http://link.springer.com/10.1007/978-3-319-68288-4_12.

[21] G.A. Miller, WordNet: a lexical database for English, *Commun ACM* **38**(11) (1995), 39–41, ISSN 00010782. doi:10.1145/219717.219748. http://portal.acm.org/citation.cfm?doid=219717.219748.

[22] K. Bollacker, C. Evans, P. Paritosh, T. Sturge and J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, ACM Press, New York, New York, USA, 2008, p. 1247. ISBN 9781605581026. doi:10.1145/1376616.1376746. http://portal.acm.org/citation.cfm?doid=1376616.1376746.

[23] O. Bodenreider, The Unified Medical Language System (UMLS): integrating biomedical terminology., *Nucleic Acids Res* **32**(Database issue) (2004), 267–70. doi:10.1093/nar/gkh061.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research* (2011). http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

[25] W. McKinney, Data Structures for Statistical Computing in Python, in: *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, eds, 2010, pp. 51–56.