Proceedings of DL4KG2019 - Workshop on Deep Learning for Knowledge Graphs

Co-located with ESWC 2019
16th European Semantic Web Conference
Portoroz, Slovenia
2nd June 2019

Edited by

Mehwish Alam *
Davide Buscaldi +
Michael Cochez †
Francesco Osborne ⋄
Diego Reforgiato Recupero ⊕
Harald Sack *

∗ FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany
+ Labortoire d'Informatique Paris Nord (LIPN), Paris, France
† Fraunhofer Institute for Applied Information Technology FIT, Germany
⋄ Knowledge Media Institute (KMI), The Open University, UK
⊕ University of Cagliari, Cagliari, Italy

Organizing Committee

- Mehwish Alam, FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany
- Davide Buscaldi, Universitè Paris 13, USPC, Paris, France
- Michael Cochez, Fraunhofer Institute for Applied Information Technology FIT, Germany
- Francesco Osborne, Knowledge Media Institute (KMi), The Open University, UK
- Diego Reforgiato Recupero, University of Cagliari, Cagliari, Italy
- Harald Sack, FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, Germany

Program Committee

- Danilo Dessi, University of Cagliari, Italy
- Stefan Dietze, L3S Hannover, Germany
- Mauro Dragoni, Fondazione Bruno Kessler, Italy
- Aldo Gangemi, University of Bologna, Italy
- Pascal Hitzler, Wright State University, USA
- Gerard de Melo, Rutgers University, USA
- Amedeo Napoli, LORIA, CNRS, France
- Finn Arup Nielsen, Technical University of Denmark, Denmark
- Andrea Nuzzolese, , National Council of Research, Italy
- Achim Rettinger, AIFB-KIT, Germany
- Petar Ristoski, IBM research, USA
- Thiviyan Thanapalasingam, The Open University, UK
- Veronika Thost, IBM Research, USA
- Volker Tresp, Siemens AG, Germany

Preface

Over the past years there has been a rapid growth in the use and the importance of Knowledge Graphs (KGs) along with their application to many important tasks. KGs are large networks of real-world entities described in terms of their semantic types and their relationships to each other. On the other hand, Deep Learning methods have also become an important area of research, achieving some important breakthrough in various research fields, especially Natural Language Processing (NLP) and Image Recognition.

In order to pursue more advanced methodologies, it has become critical that the communities related to Deep Learning, Knowledge Graphs, and NLP join their forces in order to develop more effective algorithms and applications. This workshop, in the wake of other similar efforts at previous Semantic Web conferences such as ESWC2018 as DL4KGs and ISWC2018, aimed to reinforce the relationships between these communities and foster inter-disciplinary research in the areas of KG, Deep Learning, and Natural Language Processing.

Contents

Loss Functions in Knowledge Graph Embedding Models

Sameh K. Mohamed[1], Vít Nováček[1], Pierre-Yves Vandenbussche[2], and Emir Muñoz[1]

[1] Data Science Institute at National University of Ireland, Galway, Ireland
[2] Fujitsu Ireland Ltd, Galway, Ireland
sameh.kamal@insight-centre.org

Abstract. Knowledge graph embedding (KGE) models have become popular for their efficient and scalable discoveries in knowledge graphs. The models learn low-rank vector representations from the knowledge graph entities and relations. Despite the rapid development of KGE models, state-of-the-art approaches have mostly focused on new ways to represent embeddings interaction functions (*i.e.*, scoring functions). However, we argue that the choice of a training loss function can have a substantial impact on a model's efficiency, which has been rather neglected by the state of the art so far. In this paper, we provide a thorough analysis of different loss functions that can help with the procedure of embedding learning, providing a reduction of the evaluation metric based error. We experiment with the most common loss functions for KGE models and also suggest a new loss for representing training error in KGE models. Our results show that a loss based on training error can enhance the performance of current models on multiple datasets.

1 Introduction

The recent advent of knowledge graph embedding (KGE) models has allowed for scalable and efficient manipulation of large knowledge graphs (KGs), improving the results of a wide range of tasks such as link prediction [3,21], entity resolution [15,2] and entity classification [16]. KGE models operate by learning embeddings in a low-dimensional continuous space from the relational information contained in the KG while preserving its inherent structure. Specifically, their objective is to rank knowledge facts—relational triples (s, p, o) connecting subject and object entities s and o by a relation type p—based on their relevance. Various interactions between their entity and relation embeddings are used for computing the knowledge fact ranking. These interactions are typically reflected in a model-specific scoring function. For instance, TransE [3] uses a scoring function defined as the distance between the o embedding and the translation of the embedding associated to s by the relation type p embedding. DistMult [22], ComplEx [19] and HolE [14] use multiplicative composition of the entity embeddings and the relation type embeddings. This leads to a better reflection of the

relational semantics and to state-of-the-art performance results (see [20] for a review).

Although there is a growing body of literature proposing different KG models (mostly focusing on the design of new scoring functions), the study of loss functions—a core part of the learning process—has not received much attention to date. This has already been shown to influence the behaviour of the KGE models. For instance, [9] observed that despite the different motivations behind HolE and CompleEx models, they have equivalent scoring functions. Yet their performance still differs. In [18], the authors conclude that this difference is caused by the fact that HolE uses a max-margin loss while ComplEx uses a log-likelihood loss, showing that loss functions are important for thorough understanding, and even improvement of the performance of different KGE models. However, a comprehensive study is still missing.

In this paper, we focus on comparing different loss functions when applied to several representative KGE models. By performing a systematic analysis of the performance (in terms of Mean Reciprocal Rank, MRR) of different models using different loss functions, we hope to contribute towards improving the understanding of how loss functions influence the behaviour of KGE models across different benchmark datasets.

The summary of our contributions is as follows:

(a) We provide a comprehensive analysis of training loss functions as used in several state-of-the-art KGE models (Section 2);
(b) We preform an empirical evaluation of different KGE models with different loss functions, and show the effect of different losses on the KGE models predictive accuracy (Section 3);
(c) We propose a new formulation for a KGE loss that can provide enhancements to the performance of KGE models. Section 3 demonstrates experimentally that the proposed loss function can enhance performance of state-of-the-art KGE models over multiple datasets.

2 Loss Functions in KGE Models

Generally, KGE models are cast as learning to rank problems. They employ multiple training loss functions that comply with the ranking loss approaches. In the state-of-the-art KGE models, loss functions were designed according to various pointwise and pairwise approaches that we review next.

2.1 KGE Pointwise Losses

First, we discuss existing pointwise loss functions for KGE models, namely, square error (SE), hinge, and logistic losses. Let $\mathbf{x} \in X$ be one fact of the KG, f a scoring function, and l a labelling function.

Pointwise Square Error Loss (SE). SE is the ranking loss function used in RESCAL [15]. It models training losses with the objective of minimising the

squared difference between model scores and labels (expected output):

$$\underset{\mathrm{SE}_{Pt}}{\mathcal{L}} = \frac{1}{2}\sum_{\mathbf{x}\in X}(f(\mathbf{x}) - l(\mathbf{x}))^2.$$

The optimal score for true and false facts is 1 and 0, respectively. A nice to have characteristic of SE loss is that it does not require configurable training parameters, shrinking the search space of hyper parameters compared to other losses (*e.g.*, the margin parameter of the hinge loss).

Pointwise Hinge Loss. Hinge loss can be interpreted as a pointwise loss, where the objective is to generally minimise the scores of negative facts and maximise the scores of positive facts to a specific configurable value. This approach is used in HolE [14], and it is defined as:

$$\underset{\mathrm{hinge}_{Pt}}{\mathcal{L}} = \sum_{\mathbf{x}\in X}[\lambda - l(\mathbf{x})\cdot f(\mathbf{x})]_+,$$

where $l(\mathbf{x}) = 1$ if $\mathbf{x}$ is true and -1 otherwise, and $[x]_+$ denotes $max(x, 0)$. This effectively generates two different loss slopes for positive and negative scores. Thus, the objective resembles a pointwise loss that minimises negative scores to reach $-\lambda$, and maximises positives scores to reach λ.

Pointwise Logistic Loss. The ComplEx [19] model uses a logistic loss, which is a smoother version of pointwise hinge loss without the margin parameter. Logistic loss uses a logistic function to minimise the negative triples score and maximise the positive triples score. This is similar to hinge loss, but uses a smoother linear loss slope defined as:

$$\underset{\mathrm{logistic}_{Pt}}{\mathcal{L}} = \sum_{\mathbf{x}\in X}\log(1 + \exp(-l(\mathbf{x})\cdot f(\mathbf{x}))),$$

where $l(\mathbf{x})$ is the true label of fact $\mathbf{x}$ where it is equal to 1 for positive facts and is equal to -1 otherwise.

Taking the best of the previous loss functions, we propose a new pointwise loss, called the Pointwise Square Loss (PSL), which combines the square growth of SE and the configurable margin of hinge loss.

Pointwise Square Loss (PSL). In the SE loss, the objective is to set the scores of negative and positive instances to 0 and 1, respectively. As a result, the scores of negative instances that are less than 0, and the scores of positive instances that are greater than 1 are penalised despite their actual compliance with the main training objective: $\forall_{\mathbf{x}\in X^+}\forall_{\mathbf{x}'\in X^-} f(\mathbf{x}) > f(\mathbf{x}')$, where X^+ and X^- are the sets of positive and negative facts, respectively. Therefore, we propose a new loss, PSL, that allows scores of positive instances to grow and scores of negative instances to decrease without boundaries. We also use a configurable value λ instead of 0 and 1 to allow for more search configurations as in hinge and logistic losses. PSL is defined as:

$$\underset{\mathrm{PSL}_{Pt}}{\mathcal{L}} = \frac{1}{2}\sum_{\mathbf{x}\in X}([\lambda - l(\mathbf{x})\cdot f(\mathbf{x})]_+)^2,$$

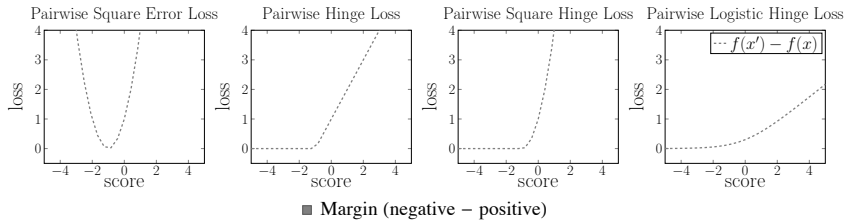


Fig. 1. Plots of growth of pairwise margin-based losses: compared to its margin with default $\lambda = 1$ and $\alpha = 0.5$.

where $l(\mathbf{x}) = 1$ if $\mathbf{x}$ is true and -1 otherwise. We can see that PSL can be made equivalent to squared hinge loss by defining it as $\mathcal{L}_{\text{PSL}_{Pt}}(f; X, l) = \mathcal{L}_{\text{hinge}_{Pt}}(f; X, l)^2$.

2.2 KGE Pairwise Losses

Here, we discuss established pairwise loss functions in KGE models, and present two new proposed loss functions, namely *tanh* and *softsign* losses. Fig. 1 shows the set of pairwise loss functions to be discussed in this subsection.

Pairwise Hinge Loss. Hinge loss is a linear learning-to-rank loss that it is used for maximum-margin classification and can be implemented in both pointwise or pairwise settings. TransE [3] and DistMult [22] models use the pairwise margin based hinge loss. It is defined as:

$$\underset{\text{hinge}_{Pr}}{\mathcal{L}} = \sum_{\mathbf{x} \in X^+} \sum_{\mathbf{x}' \in X^-} [\lambda + f(\mathbf{x}') - f(\mathbf{x})]_+,$$

where X^+ is the set of true facts, X^- is the set of false facts, and λ is a configurable margin. In this case, the objective is to maximise the difference between the scores of negative and positive instances by a good margin. This approach optimises towards having embeddings that satisfy $\forall_{\mathbf{x} \in X^+} \forall_{\mathbf{x}' \in X^-} f(\mathbf{x}) > f(\mathbf{x}')$ as in Fig. 1.

Pairwise Logistic Loss. Logistic loss can also be interpreted as pairwise margin based loss following the same approach as in hinge loss. The loss is defined as:

$$\underset{\text{logistic}_{Pr}}{\mathcal{L}} = \sum_{\mathbf{x} \in X^+} \sum_{\mathbf{x}' \in X^-} \log(1 + \exp(f(\mathbf{x}') - f(\mathbf{x}))),$$

where the objective is to minimise marginal difference between negative and positive scores with a smoother linear slope than hinge loss as shown in Fig. 1.

2.3 KGE Multi-Class Losses

Recent KGE approaches have addressed the ranking problem as a multi-class classification. Next, we discuss how this is done.

Binary Cross Entropy Loss. ConvE model [5] proposed a new binary cross entropy multi-class loss to model its training error. In this setting, the whole vocabulary of entities is used to train each positive fact such that for a triple

(s, p, o), all facts (s, p, o') with $o' \in E$ and $o' \neq o$ are considered false. Despite the extra computational cost of this approach, it allowed ConvE to generalise over a larger sample of negative instances and outperform other approaches [5].

Negative-Log Softmax Loss. In a recent work, Lacroix et. al. [10] introduced a softmax regression loss to model training error of the ComplEx model as a multi-class problem. In this approach, the objective for each triple $\mathbf{x} = (s, p, o)$ is to minimise the following loss:

$$\begin{aligned} \mathcal{L}_{spo}^{\text{softmax}} &= \mathcal{L}_{spo}^{o'} + \mathcal{L}_{spo}^{s'} \text{ , s.t.} \\ \mathcal{L}_{spo}^{o'} &= -f_{spo} + \log(\sum\nolimits_{o'} \exp(f_{spo'})) \\ \mathcal{L}_{spo}^{s'} &= -f_{spo} + \log(\sum\nolimits_{s'} \exp(f_{s'po})) \end{aligned} \tag{1}$$

where $s' \in E$, $s' \neq s$, $o' \in E$ and $o' \neq o$. This resembles a log-loss of the softmax value of the positive triple compared to all possible object and subject corruptions, where the objective is to maximise positive facts scores and minimise all other scores. This approach achieved significant improvement to the prediction accuracy of ComplEx model over all benchmark datasets when used with the 3-nuclear norm regularisation of embeddings [10].

2.4 Negative Sampling for KGE Losses

In learning to rank approaches, models use a ranking loss, *e.g.*, pointwise or pairwise loss to rank a set of true and negative instances [4], where negative instances are generated by corrupting true training facts with a ratio of negative to positive instances [3]. This corruption happens by changing either the subject or object of the true triple instance. In this configuration, the ratio of negative to positive instances is traditionally learnt using a grid search, where models compromise between the accuracy achieved by increasing the ratio and the runtime required for training.

On the other hand, multi-class based models train to rank positive triples against their all possible corruptions as a multi-class problem where the range of classes is the set of all entities. For example, training on a triple (s, p, o) is achieved by learning the right classes "s" and "o" for the pairs $(?, p, o)$ and $(s, p, ?)$, where the set of possible classes is E of size N_e. Despite the enhancements of predictions accuracy achieved by such approaches [5,10], such negative sampling procedure is exhaustive and require high space complexity due to the usage of the entire entity vocabulary for each triple.

3 Experiments

In this section, we describe the experiments conducted on three state-of-the-art KGE models, namely, TransE [3], DistMult [22] and ComplEx [19] (equivalent to HolE [9]), using the previously discussed loss functions. TransE is a distance-based scoring function, while DistMult and ComplEx are multiplicative scoring functions.

Table 1. Characteristics of the datasets.

Dataset	# Entities	# Relations	|Train|	|Valid|	|Test|
WN18	41k	18	141k	5k	5k
WN18RR	41k	11	87k	3k	3k
NELL50k	50k	497	159k	5k	5k
NELL239	48k	239	74k	3k	3k
FB15k-237	15k	237	272k	18k	20k

We present the benchmarking datasets, experiments setup, and implementation details including software and hardware configurations.

Benchmarking Datasets. In our experiments we use six knowledge graph benchmark datasets:

- WN18 & WN18RR: subsets of Wordnet dataset [11] that contain lexical information of the English language [3,5].
- NELL50k & NELL239: subsets of NELL dataset [6,7] that we have created to test our model, which contains general knowledge about people, places, teams, universities, etc.
- FB15k-237: a subset of the Freebase dataset [1] that contains information about general human knowledge [17].

Table 1 contains the characteristics of our benchmark datasets[3].

Evaluation Protocol. The three KGE models are evaluated using a unified protocol that assesses their performance in the task of link prediction. Let X be the set of facts (triples), Θ_E be the embeddings of entities E, and Θ_R be the embeddings of relations R. The KGE evaluation protocol works in three steps:

(1) *Corruption*: For each $\mathbf{x} = (s, p, o) \in X$, $\mathbf{x}$ is corrupted $2|E| - 1$ times by replacing its subject and object entities with all the other entities in E. The corrupted triples can be defined as:

$$\mathbf{x}_{\text{corr}} = \bigcup_{s' \in E} (s', p, o) \cup \bigcup_{o' \in E} (s, p, o')$$

where $s' \neq s$ and $o' \neq o$. These corruptions are considered effectively negative examples for the supervised training and testing process under the Local Closed World Assumption [13].

(2) *Scoring*: Both original triples and corrupted instances are evaluated using a model-dependent scoring function. This process involves looking up embeddings of entities and relations, and computing scores depending on these embeddings.

(3) *Evaluation*: Each triple and its corresponding corruption triples are evaluated using the reciprocal ranking metric as a separate query, where the original triples represent true objects and their corruptions false ones. It is possible that corruptions of triples may contain positive instances that exist

[3] All the benchmark datasets and experimental results are available for download in the following url: `https://figshare.com/s/8c2f1e1f98aff44b5b71`

Table 2. Link prediction results for KGE models with different loss functions on standard benchmark datasets. (*) represents the models' default loss function. In the ranking losses, best results are computed per model: bold results represent the model's best result and underlined results represent the best result in a loss approach. In multi-class losses, best results are computed across all models.

	Model	Approach	Loss	WN18		WN18RR		NELL50k		NELL239		Fb15k-237	
				MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Ranking Loss	TransE	Pairwise	Hinge *	0.52	**0.95**	0.20	0.47	**0.76**	**0.91**	**0.28**	**0.43**	**0.27**	**0.43**
			Logistic	**0.53**	0.92	**0.21**	**0.48**	0.71	0.86	0.27	**0.43**	0.26	**0.43**
		Pointwise	Hinge	0.15	0.38	0.12	0.34	0.28	0.40	0.19	0.32	0.12	0.25
			Logistic	0.14	0.36	0.11	0.31	0.26	0.38	0.17	0.31	0.01	0.23
			SE	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.01	0.01
			PSL	0.20	0.49	0.12	0.33	0.38	0.54	0.19	0.33	0.11	0.24
	DistMult	Pairwise	Hinge *	0.77	0.89	0.40	0.45	0.45	0.60	0.20	0.32	0.10	0.16
			Logistic	0.79	0.93	0.39	0.45	0.68	0.83	0.26	0.40	0.19	0.36
		Pointwise	Hinge	**0.85**	0.95	**0.43**	0.49	0.81	0.92	0.25	0.41	0.21	0.39
			Logistic	0.77	0.93	**0.43**	**0.50**	0.70	0.84	0.28	0.43	0.20	0.39
			SE	0.81	**0.95**	**0.43**	**0.50**	0.81	**0.94**	**0.31**	**0.48**	**0.22**	**0.40**
			PSL	**0.85**	**0.95**	0.40	0.46	**0.85**	**0.94**	0.24	0.40	0.20	0.38
	ComplEx	Pairwise	Hinge	0.94	0.95	0.39	0.45	0.86	0.94	0.24	0.38	0.20	0.35
			Logistic	0.91	0.95	0.41	0.47	0.72	0.86	0.27	0.43	0.19	0.35
		Pointwise	Hinge	0.91	0.95	0.41	0.47	0.86	0.95	0.21	0.36	0.20	0.39
			Logistic *	0.94	0.95	0.36	0.39	0.85	0.94	0.14	0.24	0.13	0.28
			SE	**0.95**	**0.96**	**0.47**	**0.53**	0.82	0.94	**0.35**	**0.52**	0.22	0.41
			PSL	0.94	0.95	0.41	0.45	**0.90**	**0.96**	0.24	0.40	**0.24**	**0.43**
Multi-class loss	CP	-	BCE	-	-	-	-	-	-	-	-	-	-
		-	Softmax	0.12	0.18	0.08	0.12	-	-	-	-	0.22	0.42
	DistMult	-	BCE	0.82	0.94	0.43	0.49	-	-	-	-	0.24	0.42
		-	Softmax	0.81	0.95	0.43	0.50	0.91	0.96	0.39	0.55	0.34	**0.53**
	ComplEx	-	BCE	**0.94**	**0.95**	**0.44**	0.51	-	-	-	-	0.25	0.43
		-	Softmax	0.92	0.95	**0.44**	**0.52**	**0.94**	**0.97**	**0.40**	**0.58**	**0.35**	**0.53**

among training or validation triples. In our experiments, we alleviate this problem by filtering out positive instances in the triple corruptions. Therefore, MRR and Hits@k are computed using the knowledge graph original triples and non-positive corruptions [3].

Implementation. We use TensorFlow framework (GPU) along with Python 3.5 to implement the KGE models. Experiments were executed on a Linux machine with processor Intel(R) Core(TM) i70.4790K CPU @ 4.00GHz, 32 GB RAM, and an nVidia Titan X GPU.

Experimental Setup. In the experiments, we use state-of-the-art KGE models TransE, DistMult, and ComplEx to analyse the impact of various loss functions. We run these models over the previously mentioned benchmark datasets. A grid search was performed to obtain the best hyperparameters for each model[4]. In all our experiments, the set of investigated parameters are: embeddings size $K \in \{50, 100, 150, 200\}$ and margin $\lambda \in \{1, 2, 3, 4, 5\}$. We use a fixed learning rate of 0.1 and generate two corruptions per triple during training. All embeddings

[4] Detailed results and best hyperparameters can be found at: `https://figshare.com/s/8c2f1e1f98aff44b5b71`

vectors of our models are initialised using the uniform Xavier random initialiser [8]. We use 10 mini-batches per epoch, with a maximum of 1,000 epochs for training. We implemented early stopping for the training with a target MRR metric that is checked every 50 epochs (*i.e.*, training stops if the filtered MRR decreases).

4 Results and Discussion

Table 2 shows evaluation results for KGE models using different loss functions on standard benchmark datasets.

4.1 Ranking Losses

The results clearly show that changing the models' default loss functions can improve the reported performance of the KGE models. Moreover, the loss function we have proposed, PSL, enhances models' performance on multiple datasets. For example, the DistMult model uses pairwise hinge loss by default, but its version with the PSL function achieve 1.2% and 7.1% better MRR scores on WN18 and NELL50k datasets, and its version with SE loss provides best result on the other datasets. On the other hand, ComplEx originally uses pointwise logistic loss, but its version with SE loss results in better MRR score on WN18, WN18RR and NELL239 datasets. ComplEx using PSL version achieves the best results in terms of MRR on the NELL50k and FB15k-237 datasets.

In addition to confirming our main assumption, the results provide for an interesting observation. The versions of the TransE model with pairwise loss functions consistently achieve better results in terms of mean rank, MRR, and Hits@k when compared to the versions with pointwise losses. Conversely, the DistMult and ComplEx models achieve the best MRR and Hits@k scores when pointwise losses are used. This behaviour is likely caused by the fact that the models use different scoring approaches: TransE scores triples using distances in the embedding space, but DistMult and ComplEx use a multiplicative approach. This observation may be used for designing optimal combinations of scoring and cost functions in future KGE models. However, for a truly comprehensive recommendation, more thorough analysis of other distance-based and multiplicative scoring functions is required.

In terms of the type of cost function, the results show that the models with our proposed pointwise square loss (PSL) function outperform their versions with other pointwise losses (the MRR score is better in 7 out of 15 experiment configurations of TransE, DistMult, and ComplEx models on all datasets).

An important technical observation is that the number of configurable parameters of a loss function has a significant impact on the time required for training the corresponding model. The training time grows exponentially with respect to the number of hyperparameters used in training. Pointwise SE and both pointwise and pairwise logistic losses do not have configurable parameters, therefore they require minimal training time when compared to other losses with additional configurable parameters. Even the margin based loss functions that require only one parameter, λ, have significantly slower training time than SE

and logistic losses. In our experiments, models with configurable margin losses required 5 times more training time than model using losses with no configurable parameters as we searched for best margin λ in a set of five elements.

In ranking loss functions, the differences in evaluation accuracy of models using different loss functions can be sometimes relatively small. In real world large scale knowledge graph applications, the choice of training loss function for a KGE model will therefore always involve a compromise between both evaluation accuracy and training time efficiency.

4.2 Multi-Class Losses

Results of multi-class loss shows that models' versions with negative-log softmax loss outperform their versions with BCE loss over all datasets. Also, it shows that multi-class loss can provide significant improvement in terms of MRR over ranking losses as on NELL239 and FB15k-237 datasets.

Despite the enhancements of predictions accuracy achieved by multi-class loss approaches [5,10], they can have scalability issues in real-world knowledge graphs with a large number of entities as they use the full entities vocabulary as negative instances [12].

5 Conclusions and Future Work

In summary, our results clearly confirm all our key assumptions. First of all, the choice of a loss function does have a considerable impact on the performance of KGE models. Secondly, loss functions can be consciously selected in a way that can optimise particular evaluation metrics. This marks a big improvement over state-of-the-art approaches where the cost functions have been selected in a rather non-systematic way. Last but not least, we have brought up several interesting observations that can inform more rational and efficient design of future KGE models.

For future work, we intend to experiment with models that use a sampled multi-class approach, *i.e.*, they sample negatives as a portion of the vocabulary rather that the whole vocabulary. We also aim to study the different properties of KGs and their effects on the performance of KGE models.

References

1. Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250. ACM, 2008.
2. Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation. *Machine Learning*, 94(2):233–259, 2014.
3. Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

4. Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In *NIPS*, pages 315–323. Curran Associates, Inc., 2009.
5. Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*. AAAI Press, 2018.
6. Tom M. Mitchell et. al. Never-ending learning. *Commun. ACM*, 61(5):103–115, 2018.
7. Matt Gardner and Tom M. Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*, pages 1488–1498. The Association for Computational Linguistics, 2015.
8. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.
9. Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. In *ACL (2)*, pages 554–559. Association for Computational Linguistics, 2017.
10. Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, volume 80 of *JMLR Workshop and Conference Proceedings*, pages 2869–2878. JMLR.org, 2018.
11. George A. Miller. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
12. Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, pages 2265–2273, 2013.
13. Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.
14. Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961. AAAI Press, 2016.
15. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, pages 809–816. Omnipress, 2011.
16. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *WWW*, pages 271–280. ACM, 2012.
17. Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509. ACL, 2015.
18. Théo Trouillon and Maximilian Nickel. Complex and holographic embeddings of knowledge graphs: A comparison. *CoRR*, abs/1707.01475, 2017.
19. Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.
20. Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743, 2017.
21. Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. AAAI Press, 2014.
22. Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

Graph-Convolution-Based Classification for Ontology Alignment Change Prediction

Matthias Jurisch, Bodo Igler

RheinMain University of Applied Sciences
Department of Design – Computer Science – Media
Unter den Eichen 5
65195 Wiesbaden, Germany
matthias.jurisch@hs-rm.de, bodo.igler@hs-rm.de

Abstract. Finding alignments between ontologies is a challenging and time-consuming task. When the aligned ontologies change, these alignments need to be changed as well. A recent approach to this problem proposes using embeddings as a representation for classifying changes. In this work, we compare embedding-based approaches to a neural network architecture built for node classification in knowledge graphs, namely relational graph convolutional networks. In our evaluation on two datasets from the biomedical domain, the best-performing embedding-based methods are RDF2Vec and TransE. The Graph convolution approach achieves similar results as the best-performing embedding based methods on a smaller dataset but outperforms all other approaches in standard classification metrics on a bigger dataset.

Keywords: Ontology Alignment · Alignment Adaption · Graph Embedding · Graph Neural Network.

1 Introduction

Ontologies that cover overlapping topics are often connected by so-called ontology alignments, that describe the relation of concepts in different ontologies. Finding these alignments is challenging and requires some degree of manual work, which can be supported by approaches from the area of ontology matching. Ontology matching has been an important area of semantic web research for years [17]. However, finding these alignments is only a part of the puzzle. As ontologies should change with the knowledge they represent, not only the ontologies need to be adapted, but the alignments as well. As with finding alignments, adapting them is often done manually and is very time-consuming. This is especially the case in the area of biomedical ontologies, given the changes required in this area as well as the size of the ontologies. Hence, automation of this task is desirable.

Rule-based approaches like [7] and [14] can be used to automate this task. These methods are based on a set of hand-crafted rules, that need to be adapted and maintained as ontology evolution itself may change over time. To automatically classify changes, we proposed a learning based method that first learns

graph embeddings as a change representation and then applies established classification approaches [10].

In this work, we examine the application of relational graph convolutional networks [16] to the same problem. This approach can be applied directly to the graph and does not require a separate pre-training to generate a meaningful graph representation. Also, we compare its performance regarding established classification methods to approaches with intermediate representation learning. To achieve a fair comparison, we evaluate several embedding-based methods on an established dataset for mapping adaption and compare the results to results obtained by applying a graph convolution.

The remainder of this paper is structured as follows: Section 2 presents foundations and related work and identifies a research gap. Our approach is discussed in Section 3. In Section 4, an evaluation of different classification approaches is presented. Results of this evaluation are discussed in Section 5. Section 6 closes this paper with a conclusion and an outlook.

2 Foundations and Related Work

Ontology Alignments (sometimes referred to as Ontology Mappings) are used to connect concepts in different ontologies. When these ontologies change, an adaption of these alignments is usually done manually. The problem of adapting these alignments is referred to as the mapping adaption problem [7]. Work on this topic is usually divided into the area of compositional mapping adaption and incremental adaption [4]. To create a new alignment $A'_{\mathcal{O}'_1,\mathcal{O}'_2}$ when the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ evolve to $\mathcal{O}'_1$ and $\mathcal{O}'_2$, the compositional approach creates a *composition of the alignment* $A_{\mathcal{O}_1,\mathcal{O}_2}$ and $A^{+}_{\mathcal{O}_2,\mathcal{O}'_2}$, the alignment between $\mathcal{O}_2$ and $\mathcal{O}'_2$. The incremental approach applies a *set of rules* that determine how alignments should be adapted to ontology changes. While these approaches stem from the area of database and XML schema adaption [19,22], works by Gross [6] and Dos Reis [5] have shown that both ideas can be applied to ontologies.

The mentioned approaches either rely on ontology mapping approaches between versions or a set of hand-crafted rules. Therefore, adaptions are either dependent on the quality of automatic ontology matching techniques or rely on manual work by an expert for creating rules. To automate incremental approaches using machine learning, a representation of nodes in the ontologies is required. An approach with manual feature engineering for predicting changes in ontologies in general was proposed by [2]. Features considered included background information from other ontologies on the same topic and records from publication databases as well as simple structural features such as the number of siblings and sub- or superclasses. [10] proposed using graph embeddings, specifically RDF2Vec-Embeddings [15] as a node representation, which provides a more detailed representation of the graph structure. This representation is then used to train established classification approaches for the mapping adaption problem. However, only one kind of graph embedding, namely RDF2Vec, is regarded.

Over the last ten years, several approaches for embedding knowledge graphs have been proposed. Some of these approaches are inspired by language modeling techniques such as Word2Vec [12]. The aforementioned RDF2Vec [15] and an approach based on global embeddings [3] are methods from this category. Another category of techniques is based on knowledge base completion approaches, where entity and relation embeddings are learned to provide some kind of scoring function to predict whether a triple should be part of the graph. The scoring function is mostly based on some kind of translation [1,20] or multiplication [21,18]. Embeddings from both categories can theoretically be useful when predicting alignment changes, however only one approach from the first category has been evaluated in [10].

The aforementioned idea focuses on first creating embeddings and then learning the actual task at hand, namely, predicting which part of an ontology alignment should change as a consequence of an ontology change. [16] has presented a graph network architecture that can be used for end-to-end learning on RDF graphs. However, this kind of approach has not yet been evaluated for predicting ontology alignment changes.

To our knowledge, the state of the art on ontology mapping adaption lacks an evaluation of graph network approaches and a comparison of different knowledge base embedding methods as a foundation for change classification tasks. This aspect is at the core of the research presented in this paper.

3 Approach

The classification task we try to solve in this work is the same as in [10]: For each changed entity c that is near an alignment statement, we predict whether an alignment statement near c needs to be changed. To do so, we train a classifier on data extracted from a version history. The classes we extract represent whether in a given version change, for a changed entity c, alignment statements in the neighbourhood of c have also been changed. For the classification itself we use two approaches in several variants: (1) the *approach with intermediate representation learning*, where we train a model for embedding all entities in the knowledge graph and subsequently apply established classification techniques and (2) the *graph-network-based approach*, where we use a relational graph convolutional network [16] for end-to-end classification.

For the approach with intermediate representation learning, an embedding is learned by creating a single graph out of both ontologies and the mapping and applying a knowledge graph embedding approach to this graph. As embedding approaches we compare the established knowledge base completion approaches TransE [1], TransH [20], Distmult [21] and Complex [18] and RDF2Vec [15] from the area of language-modeling based approaches. These approaches have been chosen because of their easy accessibility in the OpenKE framework and their general popularity. The classification approaches we apply on top of these embeddings are from the area of Regression, Naive Bayes, Tree-Based Algorithms as well as Support Vector Machines and Multilayer Perceptrons.

For the graph-network-based approach we apply relational graph convolutional networks (RGCNs) [16]. RGCNs are an extension of graph convolutional networks [11] to relational graphs. The core idea of RGCNs is that the propagation between RGCN-layers is based on the direct connections of a node in the relational graph. At each layer of the network, each neuron represents the activation at a graph node. In detail, the propagation function for a forward pass is

$$h_i^{(l+1)} = \sigma(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)})$$

where $h_i^{(l)}$ is the activation in node i at layer l of the neural network, R is the set of all relations, N_i^r is the set of all nodes connected to i by relation r, σ is an activation function, $c_{i,r}$ is a normalization constant and W_r^l is a weight matrix in layer l for relation r. Hence, neural network activation travels through the graph. The weight matrices determine what kind of relations at each layer transport what kind of information. The depth of the network determines how many steps in the graph the activation is propagated. To create the sets R and N_i^r for all relations, we use a graph constructed from both ontologies and the alignment prior to the changes we want to classify.

4 Evaluation

The main research question of our evaluation is how a graph-convolution-based approach compares to a two-step approach with separated representation learning and classification. To evaluate this question, the impact of the graph embedding choice in relation to the performance of the approach when separating classification and representation learning also needs to be examined. To address these issues, we conducted a series of experiments that are described in the following subsections.

4.1 Dataset

The dataset we use has been extracted from biomedical ontologies by [9] and has been made publically available by the authors of [6] on the web[1]. This dataset consist of three biomedical ontologies – SNOMED, FMA and the NCI Thesaurus, with version from 2009-2012 and mappings between all ontologies for each of those versions.

These ontology versions are used as a silver standard, since ontology and mapping versions are not necessarily perfect but contain errors. For each changed entity close to an alignment statement in a new version of the ontology, we examine, if an alignment statement has been changed. If this is the case, we

[1] https://dbs.uni-leipzig.de/de/research/projects/evolution_of_ontologies_and_mappings/ontology_mapping_adaption

assign the changed entity the class C_{change}, else, we assign the class $C_{nochange}$. Table 1 gives an overview of the datasets we use. For each pair of ontologies, we use two sets of changes: one set consisting of changes from 2009-2011, which is always used as the training set, and one set of changes from 2011-2012, which is used as the test set. In general, C_{change} is always smaller than $C_{nochange}$. This effect is present to a larger extent in SNOMED-FMA than in FMA-NCI.

Table 1. Datasets

	Version	*Entities*	*Triples*	$\#C_{change}$	$\#C_{nochange}$
FMA-NCI	2009-2011	2M	7M	725	984
	2011-2012			352	421
SNOMED-FMA	2009-2011	4M	15M	1526	13435
	2011-2012			177	6925

4.2 Approach with Intermediate Representation Learning

For comparing different embedding methods as a part of the representation learning process, we first train embeddings and then compare the performance of classifiers that use these embeddings as features on our dataset. To train embeddings from the knowledge graph completion area, we use the OpenKE-Framework [8]. To train RDF2Vec-embeddings, we use the implementation[2] provided by the authors of [15]. Hyperparameters where chosen based on recommendations in the documentation.

The classification itself was implemented using scikit-learn[13]. We used classifiers from several areas, including classic regression, naive bayes and nearest neighbour approaches as well as tree-based algorithms, SVMs and a feed-forward neural network. A list of all classifiers used is shown in Table 2. In order to find the optimal embedding-classifier-combination all possible combinations were evaluated, yielding a total of 40 combinations.

As evaluation metrics we use standard classification metrics, namely precision, recall, f1-measure, accuracy, roc-auc score and average precision. Precision, recall and f1-score are measured regarding the class C_{change}, since classification performance regarding this aspect is the most important for this task. We evaluated each classification method on each embedding on both datasets, FMA-NCI and SNOMED-FMA. The only exception is that we did not use RDF2Vec-embeddings on SNOMED-FMA. RDF2Vec uses a two-step approach: first, random walks through the graph are created (typically around 200 random walks per entity). With the SNOMED-FMA graph containing 4 million entities and more than 15 million triples, creating random walks for all entities would have taken too long to be feasible given the hardware we had available. The main issue here was the size and number of random walks.

[2] http://data.dws.informatik.uni-mannheim.de/rdf2vec/code/

Table 2. Classifiers

Category	Method
Regression	Logistic Regression (LR)
Naive Bayes	Gaussian Naive Bayes (NB)
Nearest Neighbour	KNN
Tree-Based Algorithms	CART, Random Forest (RF)
Support Vector Machines	RBF-Kernel, Linear Kernel
Multilayer Perceptron	MLP

4.3 Graph Network Based Approach

For the graph network based approach, we used the RGCN implementation written by the authors of [16] that is available on GitHub[3]. For trainining

the model we used 5 hidden layers, a l2 penalty of 0.005, a dropout rate of 0.05 and a learning rate of 0.01 with 50 training epochs. We choose only 5 hidden layers, since otherwise the model would consume too much memory on the SNOMED-FMA dataset. The other parameters were determined by a grid search over the hyperparameters. This approach was evaluated on the same dataset with the same metrics as the embedding-based approach.

4.4 Results

Results of this evaluation procedure for the dataset FMA-NCI are shown in Table 3. Underlined entries represent the best values for each metric. For readability purposes, we only show the best and the second-best results for every metric of the combinations of embedding methods and classification approaches. For each embedding method, we also show the two best combinations of classification method and embedding regarding f-measure. The graph embedding is nondeterministic. To account for graph embedding stability, we repeated the embedding step and evaluation. Since the differences between different runs of the embedding models were insignificant, we only report results of the first experiment.

Of the embedding-based methods, RDF2Vec with Naive Bayes achieves the best performance comparing all metrics except precision. At least one of the other classifier/embedding combination achieves a similar performance given one specific metric. The RGCN approach achieves very similar results to the RDF2Vec-based classifier.

Results for SNOMED-FMA are shown in Table 4. When observing the metrics for this dataset, it is important to reconsider the distribution of classes in the test set: only 2.6% of changes in the test set are in class C_{change}. As already mentioned, RDF2Vec is missing from this evaluation, as creating the embeddings would have taken to much time on this dataset. On this dataset, no embedding based method is better than the other methods in nearly all metrics. The RGCN approach clearly outperforms the embedding-based methods on this

[3] https://github.com/tkipf/relational-gcn

Table 3. FMA-NCI Results

embedding	classifier	f1	acc	prec	rec	roc_auc	avg. prec
TransE	KNN	0.587	0.659	0.642	0.541	0.648	0.553
	RF	0.551	0.659	0.672	0.467	0.641	0.553
	SVM(RBF)	0.265	0.602	0.771	0.160	0.561	0.500
	MLP	0.550	0.683	0.756	0.432	0.659	0.582
TransH	MLP	0.555	0.655	0.659	0.479	0.638	0.549
	RF	0.567	0.657	0.655	0.500	0.643	0.552
DistMult	RF	0.516	0.639	0.647	0.429	0.619	0.534
	MLP	0.560	0.656	0.657	0.488	0.640	0.551
Complex	MLP	0.572	0.611	0.565	0.580	0.608	0.516
	LR	0.398	0.542	0.485	0.337	0.523	0.461
RDF2Vec	NB	0.657	0.548	0.701	0.619	0.501	0.701
	LR	<u>0.735</u>	0.647	<u>0.774</u>	<u>0.700</u>	0.611	<u>0.752</u>
RGCN		0.624	<u>0.778</u>	0.706	0.561	<u>0.723</u>	0.540

dataset. Applying RGCNs results in significantly higher values in nearly all metrics except for accuracy, where the combination of TransE and linear regression obtains similar results, which is not a remarkable score given the distribution of classes in the test set.

To adapt our approach to the unbalanced dataset, we conducted two further experiments: We repeated our experiments on the SNOMED/FMA-Dataset using (1) oversampling and (2) undersampling of training data so that the training set was balanced in both experiments. Table 5 shows an excerpt of the results. We only show the results with highest f-measure for each embedding method out of both experiments. All best results of embedding-based methods stem from the experiment with oversampling, whereas the best RGCN-result stems from the undersampling experiment. Using oversampling, the RGCN overfitted despite regularization and dropout. While the f-measure values look similar to the best results without oversampling or undersampling, as expected, recall is higher, whereas precision is lower.

5 Discussion

When comparing the approaches that combine embeddings with traditional classification methods, we can observe that RDF2Vec in combination with Naive Bayes seems to show the best results where the computational effort allows it. However, since generating random walks is very costly, using this approach is not possible for large knowledge graphs. Another embedding method that seems

Table 4. SNOMED-FMA Results

embedding	classifier	f1	acc	prec	rec	roc_auc	avg. prec
TransE	LR	0.070	0.974	0.318	0.040	0.519	0.037
	NB	0.155	0.857	0.091	0.525	0.695	0.060
	KNN	0.155	0.965	0.192	0.130	0.558	0.047
	RF	0.189	0.944	0.148	0.260	0.611	0.057
TransH	MLP	0.248	0.966	0.276	0.226	0.605	0.082
	RF	0.191	0.944	0.149	0.266	0.613	0.058
Distmult	MLP	0.193	0.943	0.150	0.271	0.616	0.059
	RF	0.221	0.943	0.168	0.322	0.641	0.071
Complex	RF	0.243	0.944	0.183	0.362	0.660	0.082
	MLP	0.221	0.937	0.160	0.356	0.654	0.073
RGCN		0.542	0.978	0.508	0.581	0.784	0.305

Table 5. SNOMED-FMA Over/Undersampling

embedding	classifier	f1	acc	prec	rec	roc_auc	avg. prec
TransE	RF	0.222	0.896	0.137	0.598	0.751	0.091
TransH	CART	0.224	0.896	0.138	0.605	0.754	0.093
Distmult	MLP	0.223	0.895	0.136	0.605	0.753	0.092
Complex	MLP	0.222	0.895	0.136	0.599	0.751	0.091
RGCN		0.461	0.965	0.355	0.656	0.814	0.241

very promising is TransE. TransE-based classification is present in all top two performers for every metric. Hence, the choice of embedding makes a difference regarding classification performance. RDF2Vec and TransE as representations perform best on the presented datasets.

Given the performance of the RGCN approach on both datasets we can see that RGCN can achieve similar or better results than a combination of embedding and classification approaches. On the first dataset, its performance was similar to the best combined approach, on the second dataset it was significantly better. Another advantage of this end-to-end learning approach is that it is significantly faster than first training the embeddings, especially for large databases. Training embeddings using OpenKE-Embeddings or RDF2Vec at this scale was slower then the complete RGCN classification by a factor of 20-50. On the larger dataset, embedding the graph using RDF2Vec even took to long to be actually usable. Therefore the answer to the research question is that a graph-network-based approach can achieve similar or superior performance than a separated representation learning and classification approach.

6 Conclusion and Outlook

In this paper, we presented two approaches to predicting, whether alignment statements need to change after an ontology update: a two-step approach that consists of representation learning as a first step and established classification methods as a second step and an end-to-end approach that uses a neural network architecture specialized on node classification in relational graphs. In our evaluation, we could show that on the dataset we used, the best-performing representation learning approaches where RDF2Vec and TransE. The end-to-end learning approach was able to achieve similar results on one dataset and outperform the other approaches on a second, much larger dataset while in both cases needing much less computing time.

As future work, the integration of node and change features seems promising, since the current approach only uses the graph structure to reason about possible changes. Naturally, this can not be sufficient information to determine everything about the nature of a change and how to react to it. To evaluate the capabilities of the presented approach in other domains besides biomedical ontologies, another dataset is required. To our knowledge, the dataset used in this paper is the only dataset currently available for ontology mapping adaption. Hence, a new dataset needs to be built that contains knowledge from a different domain.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
2. Cardoso, S.D., Pruski, C., Silveira, M.D.: Supporting biomedical ontology evolution by identifying outdated concepts and the required type of change. Journal of Biomedical Informatics **87**, 1 – 11 (2018). https://doi.org/10.1016/j.jbi.2018.08.013
3. Cochez, M., Ristoski, P., Ponzetto, S.P., Paulheim, H.: Global RDF vector space embeddings. In: International Semantic Web Conference. pp. 190–207. Springer (2017). https://doi.org/10.1007/978-3-319-68288-4_12
4. dos Reis, J.C., Pruski, C., Reynaud-Delaître, C.: State-of-the-art on mapping maintenance and challenges towards a fully automatic approach. Expert Systems with Applications **42**(3), 1465–1478 (2015). https://doi.org/10.1016/j.eswa.2014.08.047
5. dos Reis, J.C., Pruski, C., Silveira, M.D., Reynaud-Delaître, C.: Dykosmap: A framework for mapping adaptation between biomedical knowledge organization systems. Journal of Biomedical Informatics **55**, 153 – 173 (2015). https://doi.org/10.1016/j.jbi.2015.04.001
6. Groß, A., dos Reis, J.C., Hartung, M., Pruski, C., Rahm, E.: Semi-automatic adaptation of mappings between life science ontologies. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7970 LNBI**, 90–104 (2013). https://doi.org/10.1007/978-3-642-39437-9_8
7. Groß, A., Pruski, C., Rahm, E.: Evolution of Biomedical Ontologies and Mappings: Overview of Recent Approaches. Computational and Structural Biotechnology Journal pp. 1–8 (2016). https://doi.org/10.1016/j.csbj.2016.08.002

8. Han, X., Cao, S., Xin, L., Lin, Y., Liu, Z., Sun, M., Li, J.: OpenKE: An open toolkit for knowledge embedding. In: Proceedings of EMNLP (2018)
9. Jiménez-Ruiz, E., Grau, B.C., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. In: JOURNAL OF BIOMEDICAL SEMANTICS (2010). https://doi.org/10.1186/2041-1480-2-S1-S2
10. Jurisch, M., Igler, B.: RDF2Vec-based classification of ontology alignment changes. In: Proceedings of the First Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies (DL4KGS) co-located with the 15th Extended Semantic Web Conerence (ESWC 2018) (2018)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013), http://arxiv.org/abs/1301.3781
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
14. dos Reis, J.C., Pruski, C., Da Silveira, M., Reynaud-Delaître, C.: The DyKOSMap approach for analyzing and supporting the mapping maintenance problem in biomedical knowledge organization systems. In: Simperl, E., Norton, B., Mladenic, D., Della Valle, E., Fundulaki, I., Passant, A., Troncy, R. (eds.) The Semantic Web: ESWC 2012 Satellite Events. pp. 163–175. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46641-4_12
15. Ristoski, P., Paulheim, H.: RDF2Vec: RDF graph embeddings for data mining. In: The Semantic Web - ISWC 20162016. pp. 498–514 (2016). https://doi.org/10.1007/978-3-319-46523-4_30
16. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., Navigli, R., Vidal, M.E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., Alam, M. (eds.) The Semantic Web. pp. 593–607. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
17. Shvaiko, P., Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. IEEE Transactions on Knowledge and Data Engineering **25**(10), 158–176 (2013). https://doi.org/10.1109/TKDE.2011.253
18. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on Machine Learning - Volume 48. pp. 2071–2080. ICML'16, JMLR.org (2016)
19. Velegrakis, Y., Miller, R.J., Popa, L.: Mapping adaptation under evolving schemas. VLDB '03 Proceedings of the 29th international conference on Very large data bases - Volume 29 pp. 584–595 (2003). https://doi.org/10.1016/B978-012722442-8/50058-6
20. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Twenty-Eighth AAAI conference on artificial intelligence (2014)
21. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. CoRR **abs/1412.6575** (2014), http://arxiv.org/abs/1412.6575
22. Yu, C., Popa, L.: Semantic Adaptation of Schema Mappings when Schemas Evolve. Very Large Data Bases pp. 1006 – 1017 (2005). https://doi.org/10.1.1.72.2410

Mining Scholarly Data for Fine-Grained Knowledge Graph Construction

Davide Buscaldi[1], Danilo Dessì[2], Enrico Motta[3], Francesco Osborne[3], and Diego Reforgiato Recupero[2]

[1] LIPN, CNRS (UMR 7030), University Paris 13, Villetaneuse, France
davide.buscaldi@lipn.univ-paris13.fr
[2] Computer Science Department, University of Cagliari, Cagliari (Italy)
{danilo_dessi, diego.reforgiato}@unica.it
[3] Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
{enrico.motta, francesco.osborne}@open.ac.uk

Abstract. Knowledge graphs (KG) are large networks of entities and relationships, typically expressed as RDF triples, relevant to a specific domain or an organization. Scientific Knowledge Graphs (SKGs) focus on the scholarly domain and typically contain metadata describing research publications such as authors, venues, organizations, research topics, and citations. The next big challenge in this field regards the generation of SKGs that also contain an explicit representation of the knowledge presented in research publications. In this paper, we present a preliminary approach that uses a set of NLP and Deep Learning methods for extracting entities and relationships from research publications, and then integrates them in a KG. More specifically, we i) tackle the challenge of knowledge extraction by employing several state-of-the-art Natural Language Processing and Text Mining tools, ii) describe an approach for integrating entities and relationships generated by these tools, iii) analyze an automatically generated Knowledge Graph including 10 425 entities and 25 655 relationships derived from 12 007 publications in the field of Semantic Web, and iv) discuss some open problems that have not been solved yet.

Keywords: Knowledge Graph · Semantic Web · Knowledge Extraction · Scholarly Data · Natural Language Processing

1 Introduction

Knowledge graphs (KG) are large networks of entities and relationships, usually expressed as RDF triples, relevant to a specific domain or an organization [6]. Many state-of-the-art projects such as DBPedia [9], Google Knowledge Graph, BabelNet, and YAGO build KGs by harvesting entities and relations from textual resources, such as Wikipedia pages. The creation of such KGs is a complex process that typically requires to extract and integrate various information from structured and unstructured sources.

Scientific Knowledge Graphs (SKGs) focus on the scholarly domain and typically contain metadata describing research publications such as authors, venues,

organizations, research topics, and citations. Good examples are Open Academic Graph[4], Scholarlydata.org [15], and OpenCitations [17]. These resources provide substantial benefits to researchers, companies, and policy makers by powering several data-driven services for navigating, analyzing, and making sense of research dynamics. One of their main limitations is that the content of scientific papers is represented by unstructured texts (e.g., title and abstract). Therefore, a significant open challenge in this field regards the generation of SKGs that contain also an explicit representation of the knowledge presented in scientific publications [2], and potentially describe entities such as approaches, claims, applications, data, and so on. The resulting KG would be able to support a new generation of content-aware services for exploring the research environment at a much more granular level.

Most of the relevant information for populating such a KG might be derived from the text of research publications. In the last year, we saw the emergence of several excellent Natural Language Processing (NLP) and Deep Learning methods for entity linking and relationship extraction [12, 2, 8, 11, 10]. However, integrating the outputs of these tools in a coherent KG is still an open challenge.

In this paper, we present a preliminary approach that uses a set of NLP and Deep Learning methods for extracting entities and relationships from research publications and then integrates them in a KG. Within our work, we refer to an entity as a linguistic expression that refers to an object (e.g., topics, tools names, a well-know algorithm, etc.). We define a relation between two entities when they are syntactically or semantically connected. As an example, if a tool T adopts an algorithm A, we may build the relation (T, *adopt*, A).

The main contributions of this paper are: i) a preliminary approach that combines different tools for extracting entities and relations from research publications ii) an approach for integrating these entities and relationships, iii) a qualitative analysis of a generated SKG in the field of Semantic Web, and iv) a discussion of some open problems that have not been solved yet.

2 Related Work

In textual resources there are both syntactical and semantic peculiarities that make hard the identification of entities and relations.

In previous works, entities in textual resources were detected by studying Part-Of-Speech (POS) tags. An example is constituted by [14], where authors provided a graph based approach for Word Sense Disambiguation (WSD) and Entity Linking (EL) named Babelfly. Later, some approaches started to exploit various resources (e.g., context information and existing KGs) for developing ensemble methodologies [11]. Following this idea, we exploited an ensemble of tools to mine scientific publications and get the input data. Subsequently, we have developed our methodology on top of the ensemble result. Relations extraction is an important task in order to connect entities of a KG.

For doing so, authors in [8] developed a machine reader called FRED which exploits Boxer [4] and links elements to various ontologies in order to represent

[4] https://www.openacademic.ai/oag/

the content of a text in a RDF representation. Among its features FRED extracts relations between frames, events, concepts and entities[5]. One more project that enables the extraction of RDF triples from text is [3], where a framework called PIKES has been designed to exploit the frame analysis to detect entities and their relations. These works consider a single text at a time and do not consider the type of text they parse. In contrast with them, our approach aims at parsing specific type of textual data and, moreover, at combining information from various textual resources. We decided to rely on open domain information extraction tool results refined by contextual information of our data, adapting open domain results on Scholarly Data. In addition, we combined entities and relations coming from different scientific papers instead of mining a single text at a time. With our approach the resulting KG represents the overall knowledge presented in the input scientific publications.

Recently, extraction of relations from scientific papers has also raised interest within the SemEval 2018 Task 7 *Semantic Relation Extraction and Classification in Scientific Papers* challenge [7], where participants had to face the problem of detecting and classifying domain-specific semantic relations. An attempt to build KGs from scholarly data was also performed by [10], as an evolution of their work at SemEval 2018 Task 7. Authors proposed both a Deep Learning approach to extract entities and relations, and then built a KG on a dataset of 110,000 papers. Our work finds inspiration from it, but we used different strategies to address open issues for combining entities and relations. In fact, authors of [10] considered clusters of co-referenced entities to come up with a representative entity in the cluster and solving ambiguity issues. On the contrary, we adopted textual and statistics similarity to solve it.

3 The Proposed Approach

In this section, we describe the preliminary approach that we applied to produce a KG of research entities. We used an input dataset composed by 12 007 abstracts of scientific publications about the Semantic Web domain. It was retrieved by selecting all publications from the Microsoft Academic Graph dataset which contains in the string "Semantic Web" in the "field of science" heading.

3.1 Extraction of Entities and Relations

For extracting entities and relations, we exploited the following resources:

- An extractor framework designed by [10], which is based on Deep Learning models and provides tools for detecting entities and relations from scientific literature. It detects six types of entities (*Task, Method, Metric, Material, Other-Scientific-Term, and Generic*) and seven types of relations among a list of predefined choices (*Compare, Part-of, Conjunction, Evaluate-for, Feature-of, Used-for, Hyponym-Of*).

[5] http://wit.istc.cnr.it/stlab-tools/fred/

- OpenIE [1] provided with Stanford Core NLP[6]. It detects general entities and relations among them, especially those which can be derived by verbs.
- The CSO Classifier [18], a tool for automatically classifying research papers according to the Computer Science Ontology (CSO)[7] [19] which is a comprehensive automatically generated ontology of research areas in the field of Computer Science.

We processed each sentence from the abstract and used the three tools to assign to each sentence s_i a list of entities E_i and a list of relations R_i. For each sentence s_i, we firstly extracted entities and relations with the extractor framework, and saved them in two lists (E_i and R_i, respectively). We discarded all relations with type *CONJUNCTION* because they were too generic. Then, we used CSO to extract all Computer Science topics from the sentence, further expanding E_i. Finally, we processed each sentence s_i with OpenIE, and retrieved all triples composed by subject, verb, and object in which both subject and object were in the set of entities E_i.

3.2 Entities Refinement

Different entities in E_i may actually refer to the same concept with alternative forms (e.g., machine learning, machine learning methods, machine-learning).

In this section, we report the methodology used to merge these entities when they appeared together in the same abstract.

Cleaning up entities. First, we removed punctuation (e.g., dots and apostrophes) and stop-words (e.g., pronouns). Then we merged singular and plural forms by using the *WordNet Lemmatizer* available in the NLTK[8] library.

Splitting entities. Some entities actually contained multiple compound expressions, e.g., *machine learning and data mining*. Therefore, we split entities when they contained the conjunction *and*. Referring to our example, we obtained the two entities *machine learning* and *data mining*.

Handling Acronyms. Acronyms are usually defined, appearing the first time near their extended form (e.g., *Computer Science Ontology (CSO)*) and then by themselves in the rest of the abstract (e.g., *CSO*). In order to map acronyms with their extended form in a specific abstract we use a regular expression. We then substituted every acronym (e.g., *CSO*) in the abstract with their extended form (e.g., *Computer Science Ontology*).

3.3 Graph Generation

In order to generate the graph, we need to integrate all triples extracted from the abstracts. In this phase we have to address three main issues. First, multiple entities derived from different abstracts may refer to the same concept.

[6] https://stanfordnlp.github.io/CoreNLP/
[7] http://cso.kmi.open.ac.uk
[8] https://www.nltk.org/

Secondly, multiple relationships derived from the verbs in the abstract may be redundant (e.g., {*emphasize*, *highlight*, *underline*}), Finally, some entities may be too generic (e.g., paper, approach) and thus useless for a SKG.

Entity Merging For the entity merging task we exploit two data structures. The first one, labelled $W2LE$, maps each word to a list of entities that share the last token (e.g., *medical ontology*, *biomedical ontology*, *pervasive agent ontology*, and so on.). With $W2LE$ we avoided comparing those entities that syntactically could not refer to the same entity (e.g., the entities *ontology generation* and *ontology adoption* were not compared). The second one, labelled $E2E$, maps each original entity to the entity that will represent it in the KG.

Given an entity e and the list of its tokens $\{t_0, ..., t_n\}$, we took t_n. If t_n was not present in $W2LE$, a new entry key t_n was added to $W2LE$ and its value is a list with e as its unique element. If t_n was in $W2LE$, then we compute the Levenshtein string similarity[9] between the entity e and all other entities $e'_0, ..., e'_m \in W2LE[t_n]$. If the resulting score met a given threshold t_L, the entity e was mapped as e'_i in $E2E$. Otherwise e was mapped to itself in $E2E$. At the end, the entity e was added to $W2LE[t_n]$. Finally, the map $E2E$ was used to select the entities for the graph. For each entry key e_x, if its corresponding entity $e_y = E2E[e_x]$ was not in the graph, a new entity with label e_y was added.

Relationship Merging After selecting a unique set of entities, we need to take care the relationships among them. First we cluster all verbs labels in order to reduce their number. For such a reason, we exploited WordNet [13] and a set of Word2Vec word embeddings trained on a set of 9 milion research papers from Microsoft Academic Graph[10]. In details, given the set of all verbs $V = \{v_0, ..., v_n\}$, we built a distance matrix M considering as a distance between two verbs v_i and v_j the $1 - WuPalmer$[11] similarity between their synsets. Then, we apply a hierarchical clustering algorithm, cutting the dendrogram where the number of clusters had the highest value of overall silhouette-width [5]. Subsequently, clusters were refined as follows. Given a cluster c, we assigned each verb $v_{i_c} \in c$ with the word embedding w_i in the Word2Vec model, and computed the centroid ce of the cluster as the average of word embeddings of its elements. Then, we ordered verbs in ascending order by the distance from ce. All verbs with a distance over a threshold t were discarded. All the other verbs were mapped on the verb nearest to the centroid ce in a map $V2V$.

Finally, given each pair of entities $p = (e_1, e_2)$ and their relations $\{r_0, ..., r_n\}$, we took every relation label $l_{r_i} \forall r_i \in \{r_0, ..., r_n\}$. All relations label coming from the extractor framework were directly merged into a single label L. All verb labels were firstly mapped through $V2V$ and then merged.

[9] https://pypi.org/project/python-Levenshtein/
[10] Avaliable at http://tiny.cc/w0u43y
[11] http://www.nltk.org/howto/wordnet.html

3.4 Detection of Generic Entities

The resulting graph may contain several generic entities (e.g., content, time, study, article, input, and so on.) In order to discard them we used a frequency-based filter which detects generic terms by comparing the frequency of the entities in three set of publications:

1. the set of 12 007 publications about the *Semantic Web*;
2. a set of the same size covering *Computer Science* but not the *Semantic Web*;
3. a set of the same size containing generic papers not about the *Semantic Web* nor the *Computer Science*.

For each entity e, we computed the number of times it appeared in the above datasets, so that we had three different counts c', c'', c'''. Then we computed the ratios $r' = \frac{c'}{c''}$ and $r'' = \frac{c'}{c'''}$. If the ratio r' met a threshold t', and the ratio r'' met a threshold t'' the entity e was included in the graph.

In addition, we automatically preserved all entities within a whitelist composed by CSO topics and all the paper keywords in the initial dataset.

4 The Knowledge Graph

In this section, we report our preliminary results about the KG produced from 12 007 papers about the Semantic Web. We used the following parameters $t_L = 0.9$, $t' = 2$, and $t'' = 3$, which have been determined empirically. The resulting KG has 10 425 entities and 25 655 relationships.

Table 1. Examples of relationships in the KG.

Subject Entity	Relation	Object Entity
content integration	help	linked data
context reasoning	support	web ontology language
machine readable information	PART-OF	semantic web
semantic wikis	USED-FOR	query interpretation
semantic relationship	establish	semantic link network
semantic relationship	determine	wordnet

Table 1 reports as example some relationships extracted by our framework. The KG contains both verb-based relations (from OpenEI, in lowercase) and default relations (from the Extractor Framework, in uppercase). Verbs are usually more informative, but also harder to extract. Conversely, the Extractor Framework is more flexible and it is able to extract a large number of relationships, but these are usually less specific. Using both systems allows us obtaining a good balance between coverage and specificity. Naturally, this set of relationships could also be expanded by reasoning methods. For instance, the last two relationships in Table 1 could be used to infer that *wordnet* is most likely a *semantic link network*.

Table 2. Contribution of Extractor Framework and CSO to the KG entities.

Tools Entities Contribution	Count	Percentage
CSO	1 034	9.92%
Extractor Framework	8 668	83.15%
Exclusive CSO	117	1.12%
Exclusive Extractor Framework	7 751	74.35%
Entities where both tools contribute	917	8.8%
Derived Entities	1 640	15.73%

4.1 Graph Statistics

In this section, we report some statistics about entities and relations of our KG. Table 2 reports statistics about entities. To weigh the actual contribution of each tool, we counted the number of entities that were detected by applying each tool. With the label *Exclusive* we indicate the number of entities detected only by that underlying tool. The row *Derived Entities* refers to the additional entities that were obtained by merging or splitting the original entities. The majority of entities that are present in the resulting KG comes from the Extractor Framework tool which contributes to the 83.15% of all entities, and exclusively contributes to 74.35% of them. The CSO Classifier contributes with 9.92%, but only a minority are exclusive. This was expected, since CSO contains fairly established research topics that appeared in a minimum of 50 papers in the dataset from which it was generated [16]. Conversely, the Extractor Framework is able to identify many long tail entities [12] that may only appear in few research papers. It is worth nothing that in the final KG, 15.73% of all entities are different from the original ones due to the transformations we applied. On average, each entity was extracted 3.69 times by one of the tools, with a maximum of 52 and a minimum of 1.

Table 3. Contribution of Extractor Framework and OpenIE to the KG relations.

Tools Relations Contribution	Count	Percentage
Extractor Framework	23 624	92.09%
OpenIE	3 116	12.15%
Exclusive Extractor Framework	22 539	87.85%
Exclusive OpenIE	2 031	7.92%
Contribution of both tools	1 085	4.23%

Similarly to entities, the Extractor Framework produced also the majority of the relations with a coverage of 92.09%, 87.85% of which exclusive to this tool. However, the 12.15% of relations extracted by OpenIE are usually more informative since they are mapped to specific verbs. On average, each relationship was extracted 1.32 times, with a maximum of 54 and a minimum of 1.

4.2 Limitations

In this section, we analyze some key entities of the Semantic Web and highlight some issues that still need to be solved to automatically produce high quality

SKGs. In order to focus on specific subsections of the KG, we extracted three subgraphs containing all the entities directly linked to *ontology*, *natural language processing*, and *artificial intelligence*. For the sake of space, in the following figures we display only the most representative relationships between each pair of entities, considering the following priority order: any verb extracted from OpenEI, *Used-for, Part-of, Feature-of, Hyponym-Of, Evaluate-for, Compare*

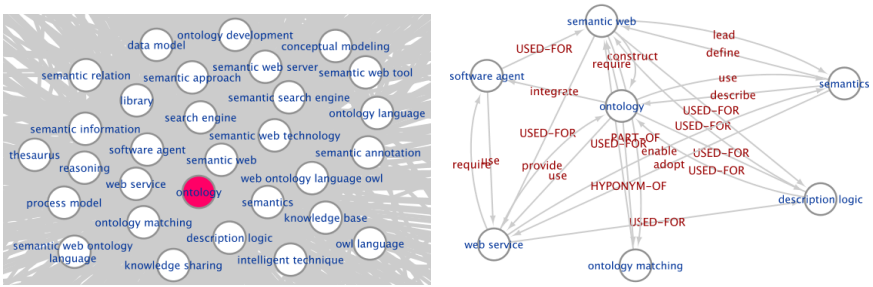
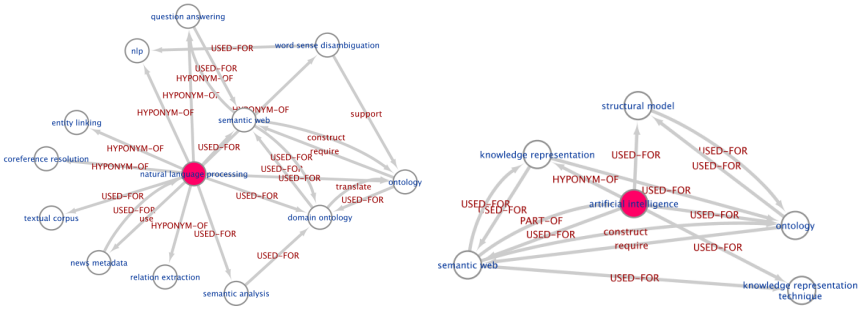


Fig. 1. The subgraph of *ontology*. (a) A snippet where many entities related to *ontology* are showed. (b) A snippet where relations between its nearest entities are showed.

Figure 1 shows the subgraph of *ontology*, which is very dense since this concept is very well represented in the input dataset. The *ontology* entity was correctly connected to several relevant entities as *semantics*, *knowledge base*, *ontology language*, *description logic* and so on.

The subgraph of the *natural language processing* entity is showed in Figure 2a. It is less dense than that in Figure 1, since the *natural language processing* entity is less represented in the input dataset. The subgraph highlights an important issue that needs to be addressed. The entities *natural language processing* and *nlp* were not merged. This problem is due to the fact that acronyms are managed at abstract level, but not at graph level. Another issue regards the distinctive lack of verb-based relations, which are often useful to better specify a relationship between two entities.

Similar considerations also apply to Figure 2b which shows the subgraph of the *artificial intelligence* entity. Some relationships between significant entities appear to be missing. For instance, *machine learning* and *artificial intelligence* are not connected here because they were originally linked by the *CONJUNCTION* relations, which was able to detect entities listed together, but we discarded since it is too generic. Another reason can be identified in textual forms that our approach may not be able to detect. We thus need to improve our pipeline to be able to handle similar instances and infer more specific relationships.



(a) (b)

Fig. 2. The subgraph of (a) *natural language processing* and (b) *artificial intelligence*

5 Conclusion and Future Work

In this paper we described a preliminary workflow for producing a Scientific Knowledge Graph from the text of research publication. We analysed a SKG derived from a set of 12 007 publication in the field of Semantic Web, with the aim of gaining a better understanding of the open problems that need to be solved when addressing this task. In summary, the analysis presented in this paper highlighted two main challenges. The first regards the disambiguation of entities that need to be further improved by also considering their semantic similarity. We also need to be able to resolve acronyms at a graph level by inferring to which extended form a certain acronym refers to in a specific publication. This may be addressed by representing entities according to word embedding learned from the input data or relevant textual resources. However, this representations would not consider long-tail entities that appear in few research papers. The second challenge regards the specificity of the relationships. While the Extractor Framework is quite good at extracting a large number of relationships, many of them are too generic. We thus intend to experiment with other techniques that combine Deep Learning and NLP for deriving specific predicates from research publications. Furthermore, we aim at validating the SKGs by human experts through a precision-recall analysis.

Acknowledgments

Danilo Dessì acknowledges Sardinia Regional Government for the financial support of his PhD scholarship (P.O.R. Sardegna F.S.E. 2014-2020).

References

1. Angeli, G., Premkumar, M.J.J., Manning, C.D.: Leveraging linguistic structure for open domain information extraction. In: Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP. vol. 1, pp. 344–354 (2015)

2. Auer, S., Kovtun, V., Prinz, M., Kasprzik, A., Stocker, M., Vidal, M.E.: Towards a knowledge graph for science. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics. p. 1. ACM (2018)
3. Corcoglioniti, F., Rospocher, M., Aprosio, A.P.: A 2-phase frame-based knowledge extraction framework. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 354–361. ACM (2016)
4. Curran, J.R., Clark, S., Bos, J.: Linguistically motivated large-scale nlp with c&c and boxer. In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions. pp. 33–36 (2007)
5. Dessì, D., Recupero, D.R., Fenu, G., Consoli, S.: A recommender system of medical reports leveraging cognitive computing and frame semantics. In: Machine Learning Paradigms, pp. 7–30. Springer (2019)
6. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. SEMANTiCS (Posters, Demos, SuCCESS) **48** (2016)
7. Gábor, K., Buscaldi, D., Schumann, A.K., QasemiZadeh, B., Zargayouna, H., Charnois, T.: Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In: Proceedings of The 12th International Workshop on Semantic Evaluation. pp. 679–688 (2018)
8. Gangemi, A., Presutti, V., Recupero, D.R., Nuzzolese, A., et al.: Semantic Web Machine Reading with FRED. Semantic Web **8**(6), 873–893 (2017)
9. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., et al.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**(2), 167–195 (2015)
10. Luan, Y., He, L., Ostendorf, M., Hajishirzi, H.: Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In: Proceedings of the EMNLP 2018 Conference. pp. 3219–3232 (2018)
11. Martinez-Rodriguez, J.L., Lopez-Arevalo, I., Rios-Alvarado, A.B.: Openie-based approach for knowledge graph construction from text. Expert Systems with Applications **113**, 339–355 (2018)
12. Mesbah, S., Lofi, C., Torre, M.V., Bozzon, A., Houben, G.J.: Tse-ner: An iterative approach for long-tail entity extraction in scientific publications. In: ISWC. pp. 127–143. Springer (2018)
13. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
14. Moro, A., Raganato, A., Navigli, R.: Entity linking meets word sense disambiguation: a unified approach. Transactions of the Association for Computational Linguistics **2**, 231–244 (2014)
15. Nuzzolese, A.G., Gentile, A.L., Presutti, V., Gangemi, A.: Conference linked data: the scholarlydata project. In: ISWC. pp. 150–158. Springer (2016)
16. Osborne, F., Motta, E.: Klink-2: integrating multiple web sources to generate semantic topic networks. In: ISWC. pp. 408–424. Springer (2015)
17. Peroni, S., Shotton, D., Vitali, F.: One year of the opencitations corpus. In: ISWC. pp. 184–192. Springer (2017)
18. Salatino, A.A., Thanapalasingam, T., Mannocci, A., Osborne, F., Motta, E.: Classifying research papers with the computer science ontology. In: ISWC (P&D/Industry/BlueSky). CEUR Workshop Proceedings. vol. 2180
19. Salatino, A.A., Thanapalasingam, T., Mannocci, A., Osborne, F., Motta, E.: The computer science ontology: a large-scale taxonomy of research areas. In: ISWC. pp. 187–205 (2018)

A Comprehensive Survey of Knowledge Graph Embeddings with Literals: Techniques and Applications

Genet Asefa Gesese[1,2], Russa Biswas[1,2], and Harald Sack[1,2]

[1] FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany
[2] Karlsruhe Institute of Technology, Institute AIFB, Germany
firstname.lastname@kit.edu

Abstract. Knowledge Graphs are organized to describe entities from any discipline and the interrelations between them. Apart from facilitating the inter-connectivity of datasets in the LOD cloud, KGs have been used in a variety of applications such as Web search or entity linking, and recently are part of popular search systems and Q&A applications etc. However, the KG applications suffer from high computational and storage cost. Hence, there arises the necessity of having a representation learning of the high dimensional KGs into low dimensional spaces preserving structural as well as relational information. In this study, we conduct a comprehensive survey based on techniques of KG embedding models which consider the structured information of the graph as well as the unstructured information in form of literals such as text, numerical values etc. Furthermore, we address the challenges in their embedding models followed by a discussion on different application scenarios.

Keywords: Knowledge Graph · Embedding · Literals · Knowledge Graph embedding survey.

1 Introduction

Various Knowledge Graphs (KGs) have been published for the purpose of sharing linked data. Some of the most popular general purpose KGs are DBpedia [14], Freebase [1], Wikidata [23], and YAGO[16]. KGs have become quite invaluable for various applications mainly in the area of artificial intelligence. For instance, in a more general sense, KGs can be used to support decision making process and to improve different machine learning applications. Spam detection, movie recommendation, and market basket analysis are some of the ML applications which can benefit from KGs [25]. General purpose KGs as e.g., Wikidata, often comprise millions of entities, represented as nodes, with hundreds of millions of facts, represented as edges connecting those nodes. However, a significant number of important graph algorithms needed for the efficient manipulation and analysis of graphs have proven to be NP-complete [11]. Although KGs are effective in representing structured data, the underlying symbolic nature of the way data is encoded as triples (i.e. $< subject, predicate, object >$) usually makes KGs hard

to manipulate [24]. In order to address these issues and use a KG efficiently, it is recommended to convert it into a low dimensional vector space while preserving the graph properties. To this end, various attempts have been made so far to learn vector representation (embeddings) for KGs. However, most of these approaches, including the state of the art TransE [2], are structure based embeddings which do not include any literal information. This is a major disadvantage because a lot of information encoded in the literals will be left unused when capturing the semantics of a certain entity.

Literals can bring advantages to the process of learning KG embeddings in two major ways. The first is in learning embeddings for novel entities i.e., entities which are not linked to any other entity in the KG but have some literal values associated with them. In most existing structure based embedding models, it is not possible to learn embeddings for such novel entities. However, this can be addressed by utilizing the information held in literals to learn embeddings. The other advantage of literals is improving the representation of entities in structure based embedding models where an entity is required to appear in at least minimum number of relational triples. Some approaches have been proposed to make use of literals for KG embeddings. The focus of this paper is to discuss these different embedding approaches and their advantages and drawbacks in the light of different application scenarios. Our contributions include:

- A detailed analysis of the existing literal enriched KG embedding models and their approaches. In addition, a method is proposed to categorize them into different groups.
- The research gaps in the area of KG embeddings in using literals are indicated as directions for further future works.

The rest of this paper is organized as follows. Sect. 2 presents a brief overview of related work. In Sect. 3, the problem formulation is provided. In Sect. 4, the analysis of the different KG embedding techniques with literals is discussed. In Sect. 5, various tasks used to evaluate the embedding models discussed in Sect. 4 are explained. The survey is concluded in Sect. 6 by providing directions for future work for KG embedding with literals.

2 Related Work

Few attempts have been made to conduct surveys on the techniques and applications of KG embeddings [12, 3, 24]. However, none of these surveys include all the existing KG embedding models which make use of literals. The first survey [12] is conducted with focus on network embedding models. The second [3] and the third [24] surveys discuss only RESCAL [17] and KREAR [15] as methods which use attributes of entities for KG embeddings, and focuses mostly on the structure based embedding methods.

However, RESCAL is a matrix-factorization method for relational learning which encodes each object/data property as a slice of the tensor ending up increasing the dimensionality of the tensor automatically. Thus, this method is

not efficient to utilize literals in KG embedding. Similarly, KREAR is not a proper KG embedding model with literals since it takes only those data properties which have categorical values and ignores those which take any random literals as values. This shows that there is a gap in the KG embedding surveys. Taking this into consideration, in this paper, a survey on KG embedding models, which make use of literals is provided.

3 Problem Formulation

In this section, a brief introduction is provided on fundamental KG and its embeddings followed by a formal definition of KG embedding with literals.

3.1 Definitions and preliminaries

Relations (or Properties). Based on the nature of the objects, relations are classified into two main categories:

- **Object Relations** – relations that link entities to entities
- **Data Type Relations** – relations that link entities to data values (literals).The triples consisting of literals as objects are often referred to as attributive triples.

3.2 Types of literals

Literals in a KG encode information that is not captured by the relations or links between the entities. There are different types of literals present in the KGs:

- **Text** – Wide variety of different information can be stored in KG in the form of free text such as names, labels, titles, descriptions, comments, etc. In most of the KG embedding models with literals, text information has been further categorized into ***Short text*** and ***Long text*** for better capture of the semantics in the model. The literals which are fairly short such as for relations like names, titles, etc. are considered as *Short text*. On the other hand, for strings that are much longer such as descriptions of entities, comments, etc. are considered as *Long text*.
- **Numeric** – Information encoded in the form of real numbers, decimal numbers such as height, year or date, population, etc. also provide useful insight. It is worth considering the numbers as distinct entities in the embeddings models, as it has its own semantics to be covered which cannot be covered by string distance metrics. For e. g. 777 is more similar to 788 than 77.
- **Units of Measurement** – (Numeric) literals often denote units of measurements to a definite magnitude. For e. g. Wikidata property wdt:P2048 takes values in mm, cm, m, km, inch, foot, and pixel. Hence, discarding the units and considering only the numeric values without normalization results in loss of semantics, especially in the case if units are not comparable, as e.g. units of length and units of weight.

- **Images** – Images also provide latent useful information for modelling of the entities. For example, a person's details such as age, gender etc. can be deduced via visual analysis of an image depicting the person.
- **Others** – Useful information encoded in the form of other literals such as external URIs which could lead to an image, text, audio or video files.

Since the information present in the KGs is diverse, modelling of the entities is a challenging task.

- **RQ1** – *How to combine the structured (triples with object relations) and unstructured information (attributive triples) in the KGs into the representation learning?*
- **RQ2** – *How to capture and combine the heterogeneity of the types of literals present in the KGs into representation learning?*

4 Knowledge Graph Embeddings with Literals

In this study, the investigated KG embedding models with literals are divided into the following different categories based on the literals utilized: (i) Text, (ii) Numeric, (iii) Image , and (iv) Multi-modal. A KG embedding model which utilizes at least two types of literals is considered as multi-modal. This section consists of an analysis of the models in each category, with their similarities and differences, followed by a discussion of potential drawbacks.

4.1 Text Literals

Subsequently, four KG models considering text literals are discussed, namely, Extended RESCAL [18], DKRL [28], KDCoE [4], and KGloVe with literals [5].

Extended RESCAL improves the original RESCAL approach by processing literal values more efficiently and deal with the sparsity nature of the tensors. In this method, attributive triples are handled in a separate matrix factorization, which is performed jointly with the tensor factorization of the non-attributive triples. Attributive triples containing only text literals are encoded in an entity-attributes matrix in such a way that given a triple, one or more $< data\ type\ relation, value >$ pairs are created by tokenizing and stemming the object literal. Despite the advantage that this approach handles multi-valued literals, it does not consider the sequence of words in the literal values.

DKRL generates embeddings of entities and relations of a KG by combining structure-based and description-based representations. The structure based representation of entities and relations are obtained via TransE [2], in which the relation in each triple *(head, relation, tail)*, is regarded as the translation from head entity to tail entity. On the other hand, continuous bag of words (CBOW) and a deep convolutional neural network model (CNN) have been used to generate the description based representations of the head and tail entities. In case of CBOW, short text is generated from the description based on keywords and

their corresponding word embeddings are summed up to generate the entity embedding. In the CNN model, after preprocessing of the description, pre-trained word vectors from Wikipedia are provided as input. The CNN has five layers and after every convolutional layer pooling is applied to decrease the parameter space of CNN and filter noises. Max-pooling is applied for the first pooling and mean pooling for the last one. CNN model works better than CBOW because it preserves the sequence of words.

KDCoE focuses on the creation of an alignment between entities of multilingual KGs by creating new inter-lingual links (ILLs). The model leverages a weakly aligned multilingual KG for semi-supervised cross-lingual learning using entity descriptions. It performs co-training of a multilingual KG embedding model (KGEM) and a multilingual literal description embedding model (DEM) iteratively in order for each model to propose a new ILL alternately. KGEM adopts TransE whereas DEM uses an attentive gated recurrent unit encoder (AGRU) to encode the multilingual entity descriptions.

KGloVe with literals works by first creating a cooccurrence matrix from the underlying graph by performing Personalized PageRank (PPR) on the (weighted) graph followed by the same optimisation used in the GloVe [19] approach. Two cooccurrence matrices are generated independently and merged in the end. The first matrix is generated using KGloVe [6] technique and Named Entity Recognition is performed prior to the creation of the second matrix.

The basic differences between these models lie in the methods used to exploit the information given in the text literals and combine them with structure-based representation. One major advantage of KDCoE over text literal based embedding models is that it considers descriptions present in multilingual KGs. Also, both DKRL and KDCoE embedding models are designed to perform well for the novel entities which have only attributive triples in the KGs. Other types of text literals are not widely considered.

4.2 Numeric literals

In this section, four models which make use of numeric literals, namely, MT-KGNN [21], KBLRN [10], LiteralE [13], and TransEA [26] are discussed.

MT-KGNN trains a relational network (RelNet) for triple classification and an attribute network (AttrNet) for attribute value regression in order to learn embeddings for entities, object properties, and data properties. Only data properties with non-discrete literal values are considered in this approach. RelNet is a simple binary (pointwise) classification whereas the AttrNet is a regression task. In RelNet, a concatenated triple is passed through a nonlinear transform and then a sigmoid function is applied to get a linear transform. In the case of AttrNet, two regression tasks are performed for head and tail data properties respectively. Finally, the two networks are trained in a multi-task fashion using a shared embedding space.

KBLRN combines the relational, latent (learned by adapting TransE), and numerical features together. It uses a probabilistic PoE (Product of Experts) method to combine these feature types and train them jointly end to end. Each

relational feature is formulated by adopting the rule mining approach AMIE +[9], to be evaluated in the KG to compute the value of the features. Numerical features are used with the assumption that, for some relation types, the differences between the head and tail is seen as characteristics for the relation itself. In PoE, one expert is trained for each (relation type, feature type) pair. The parameters of the entity embedding model are shared by all the experts in order to create dependencies between them. For numerical features, a radial basis function is applied as activation function if the difference of values is in a specific range.

LiteralE is designed in order to incorporate literals into existing latent feature models, which are designed for link prediction. Given a base model, for instance *Distmult*, LiteralE modifies the scoring function f used in *Distmult* by replacing the vector representations of the entities e_i in f with literal enriched representations e_i^{lit}. In order to generate e_i^{lit}, LiteralE uses a learnable transformation function g which takes e_i and its corresponding literal vectors l_i as inputs and maps them to a new vector. For g, linear transformations, non-linear transformations, simple multi-layer NNs, and non-linear transformations with gating mechanisms are proposed. The modified scoring function f is trained following the same procedure as in the base model.

TransEA has two component models; a newly proposed attribute embedding model and a directly adopted translation-based structure embedding model, TransE. For the attribute embedding, it uses all attributive triples containing numeric values as input and applies a linear regression model to learn embeddings of entities and attributes. The loss function for TransE is defined by taking the sum of the respective loss functions of the component models with a hyperparameter to assign a weight for each of the models. Finally, the two models are jointly optimized in the training process by sharing the embeddings of entities.

Despite their support for numerical literals, all the embedding methods discussed fail to interpret the semantics behind data types of literals and units. For e. g., '1999€' and 'the year 1999' could be considered same because type semantics are discarded. Moreover, none of the models apply normalization for literal values, hence the semantic similarity between two literal values such as, 200 mm and 2 cm is not captured. Also, most of the models do not have proper mechanism to handle multi-valued literals.

4.3 Image

IKRL [27] learns embeddings by jointly training a structure-based (by adapting TransE) with an image-based representation. For the image-based representation, an image encoder is applied to generate embedding for each instance of a multi-valued image relation. Attention-based multi-instance learning is used to integrate the representations learned for each image instance by automatically calculating the attention that should be given to each instance. Given a triple, the overall energy function is defined by combining four energy functions which are based on two kinds of entity representations. The first energy function is same as TransE and the second uses their corresponding image-based representations

for both head and tail entities. The third function is based on the structure-based representation of the head entity and the image-based representation of the tail entity whereas the fourth function is the exact opposite.

4.4 Multi-modal

Numeric literals and text: LiteralE with blocking [8] proposes to improve the effectiveness of the data linking task by combining LiteralE with a CER blocking[7] strategy. Unlike LiteralE, it also considers literals from URI infixes of the head entities and data relations of attributive triples. The CER blocking is based on a two-pass indexing scheme. In the first pass, Levenshtein distance metric is used to process literal objects and URI infixes whereas in the second pass semantic similarity computation with Wordnet is applied to process object/data relations. All the extracted literals are tokenized into word lists so as to create the indices.

EAKGAE [22] jointly learns entity embeddings of two KGs using structure embedding (by adapting TransE) and attribute character embedding. Given a triple (h, r, a), the data property r is interpreted as a translation from the head entity h to the literal value a i.e. $h + r = f_a(a)$ where $f_a(a)$ is a compositional function. Three different compositional functions SUM, LSTM, and N-gram-based functions have been proposed. SUM is defined as a summation of all character embeddings of the attribute value. In LSTM, the final hidden state is taken as a vector representation of the attribute value. The N-gram-based function, which shows better performance than the others, uses the summation of n-gram combination of the attribute value.

The common drawback with both methods is that text and numeric literals are treated in the same way. They also do not consider literal data type semantics or multi-valued literals in their approach. Furthermore, since EAKGAE is using character-based attribute embedding, it fails to capture the semantics behind the cooccurrence of syllables.

Numeric literals, Text, and Images: MKBE [20] is a a multi-modal knowledge graph embedding, in which the text, numeric and image literals are modelled together. It extends DistMult, which creates embedding for entities and relations, by adding neural encoders for different data types. For image triples, a fixed-length vector is encoded using CNN. On the other hand, textual attributes are encoded using sequential embedding approaches like LSTMs. Given the vectors representations of the entities, relations and attributes, the same scoring function from DistMult is used to determine the correctness probability of triples.

5 Applications

In this section, the different KG application scenarios used by the techniques discussed in Sect. 4 are presented.

Link prediction. Link prediction aims to predict new links for a KG given the existing links among the KG entities. The models Extended RESCAL, LiteralE, TransEA, KBLRN, DKRL, KDCoE, EAKGAE, IKRL, and MKBE have

	FB15K	FB15K-237	YAGO-10
KBLN	0.739	0.301	0.487
MTKGNN(DistMult+MultiTask)	0.669	0.285	0.481
DistMult+LiteralE	0.583	**0.314**	0.504
DistMult+LiteralE-Gate	0.723	0.300	-
ComplEx+LiteralE	**0.765**	0.299	**0.509**
ConvE+LiteralE	0.66	**0.314**	0.506

(a)

Model	MRR
Numeric	
KBLN	0.503
S+N	**0.549**
Image	
IKRL	0.509
S+I	**0.566**

(b)

Table 1: (a) MRR results on link prediction taken from LiteralE [13], and (b) MRR results on link prediction task on YAGO-10 taken from MKBE [20].

been evaluated on the link prediction task. However, it is not possible to compare the obtained evaluation results because the experiments have been carried out on different datasets. The authors of the LiteralE and MKBE models conducted some experiments to compare their proposed models/submodels with already existing ones. LiteralE has been compared with KBLN, which is a submodel of KBLRN designed without taking into consideration the relational information of graph feature methods. Besides KBLN, LiteralE has been compared with a new modified version of MTKGNN, where its ER-MLP part is replaced with DistMult to make it compatible with their specific implementation environment. The results taken from LiteralE are shown in Table 1a. From the result, it can be seen that DistMult+LiteralE delivers better MRR values when it is compared with both KBLN and MTKGNN on the datasets FB15k-237 and YAGO-10. The authors of LiteralE argue that the performance of DistMult+LiteralE is lower than the others on the FB15K dataset because this dataset includes a lot of inverse relations and hence claim that it is not an appropriate dataset for link prediction. The other experiments conducted are in MKBE where the submodels, structures along with numeric and image literals respectively arecompared with KBLN and IKRL respectively as shown in Table 1b. Thereby, it can be inferred that the two submodels of MKBE perform better than their counterparts.

Triple Classification. A potential triple is classified as 0 (false) or 1 (true). MTKGNN, KGlove with literals, and IKRL have been evaluated on this task. Since they do not use a common evaluation dataset, it is not possible to compare the reported results directly.

Entity Classification. Given a KG and an entity, the entity type is predicted using a multilabel classification algorithm with KG entity types as given classes. DKRL has been evaluated on this task.

Entity Alignment. Semantically similar entities are determined from multiple KGs using specific similarity metrics. EAKGAE has been evaluated on an entity alignment task. In addition, KDCoE has also been evaluated on a cross-lingual entity alignment task which determines similar entities in different languages. Despite the fact that both these models use the same task for evaluation, their experimental results cannot be compared since they are based on different datasets.

Other Machine Learning problems. Attribute value prediction, nearest neighbor analysis, data linking, and document classification are other applications scenarios used for the evaluation of the models discussed in Sect. 4. In MTKGNN, attribute value prediction is applied using an attribute-specific Linear Regression classifier for evaluation. Nearest neighbor analysis has been performed in LiteralE to compare DistMult+LiteralE with the base model DistMult. Data linking and document classification tasks have been used in LiteralE with blocking and KGlove with literals respectively.

6 Conclusion and Future Directions

To sum up, in this paper, a comprehensive survey of KG embedding models with literals is presented. The survey provides a detailed analysis and categorization of the embedding techniques of these models along with their application scenarios and limitations. As mentioned in Section 4, these embedding models have different drawbacks. None of them consider the effect that data types and units have on the semantics of literals. Most of them also do not have a proper mechanism to handle multi-valued literals. Thus, filling these gaps will be taken as a direction for future work.

Moreover, only few approaches have been proposed for multi-modal KG embeddings and none of them take into consideration literals with URIs linking to items such as audio, video, or pdf files. This clearly indicates that more work has to be invested to address different types of literals. Regarding the comparison of the quality of the models, as discussed in Section 5, it was only possible to use the experimental results conducted for some of the models as most use different datasets and application scenarios. However, as a future work, experiments for all of the models on different applications will be performed to enable better comparability.

References

1. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: ACM SIGMOD international conference on Management of data (2008)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-Relational Data. In: NIPS (2013)
3. Cai, H., Zheng, V.W., Chang, K.C.C.: A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. TKDE (2018)
4. Chen, M., Tian, Y., Chang, K.W., Skiena, S., Zaniolo, C.: Co-training Embeddings of Knowledge Graphs and Entity Descriptions for Cross-Lingual Entity Alignment. arXiv preprint arXiv:1806.06478 (2018)
5. Cochez, M., Garofalo, M., Lenßen, J., Pellegrino, M.A.: A First Experiment on Including Text Literals in KGloVe. arXiv preprint arXiv:1807.11761 (2018)
6. Cochez, M., Ristoski, P., Ponzetto, S.P., Paulheim, H.: Global rdf Vector Space Embeddings. In: International Semantic Web Conference. Springer (2017)

7. de Assis Costa, G., de Oliveira, J.M.P.: A Blocking Scheme for Entity Resolution in the Semantic Web. In: AINA (2016)
8. de Assis Costa, G., de Oliveira, J.M.P.: Towards Exploring Literals to Enrich Data Linking in Knowledge Graphs. In: AIKE (2018)
9. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast Rule Mining in Ontological Knowledge Bases with AMIE+. VLDB (2015)
10. García-Durán, A., Niepert, M.: Kblrn: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. In: UAI (2018)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1990)
12. Goyal, P., Ferrara, E.: Graph Embedding Techniques, Applications, and Performance: A Survey. Knowl.-Based Syst. (2018)
13. Kristiadi, A., Khan, M.A., Lukovnikov, D., Lehmann, J., Fischer, A.: Incorporating Literals into Knowledge Graph Embeddings. CoRR (2018)
14. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al.: Dbpedia–A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web (2015)
15. Lin, Y., Liu, Z., Sun, M.: Knowledge Representation Learning with Entities, Attributes and Relations. ethnicity (2016)
16. Mahdisoltani, F., Biega, J., Suchanek, F.M.: Yago3: A Knowledge Base from Multilingual Wikipedias. In: CIDR (2013)
17. Nickel, M., Tresp, V., Kriegel, H.P.: A Three-Way Model for Collective Learning on Multi-Relational Data. In: ICML (2011)
18. Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing Yago: Scalable Machine Learning for Linked Data. In: Proceedings of the 21st international conference on World Wide Web. ACM (2012)
19. Pennington, J., Socher, R., Manning, C.: Glove: Gobal Vectors for Word Representation. In: EMNLP (2014)
20. Pezeshkpour, P., Chen, L., Singh, S.: Embedding multimodal relational data for knowledge base completion. arXiv preprint arXiv:1809.01341 (2018)
21. Tay, Y., Luu, A.T., Phan, M.C., Hui, S.C.: Multi-task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs. CoRR (2017)
22. Trsedya, B.D., Qi, J., Zhang, R.: Entity Alignment between Knowledge Graphs Using Attribute Embeddings. In: AAAI (2019)
23. Vrandečić, D., Krötzsch, M.: Wikidata: A Free Collaborative Knowledge Base (2014)
24. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge Graph Embedding: A Survey of Approaches and Applications. TKDE (2017)
25. Wilcke, X., Bloem, P., De Boer, V.: The Knowledge Graph as the Default Data Model for Learning on Heterogeneous Knowledge. Data Science (2017)
26. Wu, Y., Wang, Z.: Knowledge Graph Embedding with Numeric Attributes of Entities. In: Rep4NLP@ACL (2018)
27. Xie, R., Liu, Z., Chua, T.S., Luan, H.B., Sun, M.: Image-embodied knowledge representation learning. In: IJCAI (2017)
28. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation Learning of Knowledge Graphs with Entity Descriptions. In: AAAI (2016)

Iterative Entity Alignment with Improved Neural Attribute Embedding

Ning Pang[1], Weixin Zeng[1], Jiuyang Tang[1,2], Zhen Tan[1], and Xiang Zhao[1,2] ✉

[1] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China
[2] Collaborative Innovation Center of Geospatial Technology, Wuhan, China
xiangzhao@nudt.edu.cn

Abstract. Entity alignment (EA) aims to detect equivalent entities in different knowledge graphs (KGs), which can facilitate the integration of knowledge from multiple sources. Current EA methods usually harness KG embeddings to project entities in various KGs into the same low-dimensional space, where equivalent entities are placed close to each other. Nevertheless, most methods fail to take fully advantage of other sources of information, e.g., attribute information, and overlook the negative impact brought by lack of labelled data. To overcome these deficiencies, in this paper, we propose to generate neural attribute representation by considering both local and global signals. Besides, entity representations are refined via an iterative training process on the neural network. We evaluate our proposal on real-life datasets against state-of-the-art methods, and the results demonstrate the effectiveness of our solution.

Keywords: Entity alignment · Attribute information · Iterative training.

1 Introduction

Knowledge graphs (KGs) are becoming increasingly important for many downstream applications such as question answering [1] and sentence generation [2]. A large number of KGs, e.g., YAGO and DBpedia, have been constructed. However, in reality, these KGs are far from complete. To tackle this problem, various methods have been proposed, among which KG alignment attracts growing attention since it can incorporate complementary knowledge from multiple external KGs. Unfortunately, KGs are usually built in different natural languages or with various ontology systems, resulting in the obstacle of integrating knowledge from external KGs to refine the target KG. As thus, many research works have been devoted to improving the performance of KG alignment.

Current KG alignment approaches lay emphasis on entity alignment (EA), as entities are the pivots connecting different KGs. The task of EA aims to identify equivalent entities in different KGs. State-of-the-art methods [3, 4] normally harness translation-based KG embeddings to project entities and relations into a low-dimensional embedding space. The separated embedding spaces are then unified by harnessing seed entity pairs. Eventually given a target entity, its counterparts in other KGs can be determined in accordance to the distance in the unified embedding space. Nevertheless, Wang et al. [5] argued that KG embedding might fail to fully mine the structural information and instead they utilize graph convolutional network (GCN) [6] to generate entity embeddings.

Additionally, they proposed to incorporate attribute information to serve as additional signals for EA. Due to the limitation of dataset, attribute names are considered instead of attribute values. Their method has also achieved superior results on existing EA benchmarks.

In [5], attribute names are represented as one-hot embeddings of the most frequent attributes. However, the most frequent attributes appear with the majority of entities and are not able to help identify a specific entity. Additionally, the neighbourhood attribute information is completely ignored. For instance, to determine the equivalent entity of entity `Michael_Jordan`, optional attribute *hasSpouse* would be more useful than obligatory attribute *birthDate* since every person has a birthday while not necessarily a spouse. Besides, the attributes of `Michael_Jordan`'s neighbouring entities, e.g., *hasNBAChampionship* for `Chicago_Bulls`, can also be harnessed for representing `Michael_Jordan`. Also, the shortage of labelled data (seed entity pairs) is largely overlooked by previous works, which will restrain the quality of entity embeddings, and hence, the performance of EA.

In this paper, to handle these drawbacks, we devise an iterative entity alignment method with improved neural attribute embedding, Inga, which enhances EA performance by harnessing neural network, i.e., GCN, iterative training strategy and refined attribute information to generate entity representations. In specific, by incorporating the neighbouring attributes of an entity (local attribute information) and the frequency of an attribute (global attribute information) to form the improved attribute feature vector, more comprehensive signals can be captured in comparison to the one-hot representation [5]. To deal with the second limitation, an iterative training strategy is utilized to train GCN, which keeps labelling unlabelled instances and select high-quality ones to retrain itself so as to generate better entity embeddings.

The main contributions of this work are:

- Attribute representation is improved by considering both local and global information.
- We apply an iterative training mechanism on GCN to generate more accurate structure and attribute representations.
- We evaluate Inga against state-of-the-art methods on three cross-lingual EA datasets, and the results demonstrate the effectiveness of our proposal.

Related Works. The task of KG alignment can be traced back to traditional ontology matching task [7]. With the emergence and prevalence of embedding techniques, most KG alignment solutions resort to KG embedding for determining equivalent elements in different KGs. Chen et al. [3] (MTransE) are the first to utilize TransE to embed entities in each KG into separated embedding spaces, which are then unified by different alignment models using seed entities pairs. The distance in the unified embedding space is used to determine entity pairs. JAPE [4] introduces attribute type information for refining structure representation captured by KG embedding. GCN [5], on the other hand, harnesses GCN, instead of KG embedding, to generate entity representation. Attribute information, represented as one-hot vectors of most frequent attributes, is also utilized to complement structure information.

2 Methodology

Task Definition. A KG is usually represented as $G = (E, R, A, V)$, where E, R, A, V denotes entities, relations, attributes and attribute values respectively. Given two KGs, G_1 and G_2, EA aims to automatically mine new aligned entity pairs based on existing seed entity pairs $S = \{(e_{i1}, e_{i2})|e_{i1} \in E_1, e_{i2} \in E_2\}_{i=1}^{m}$.

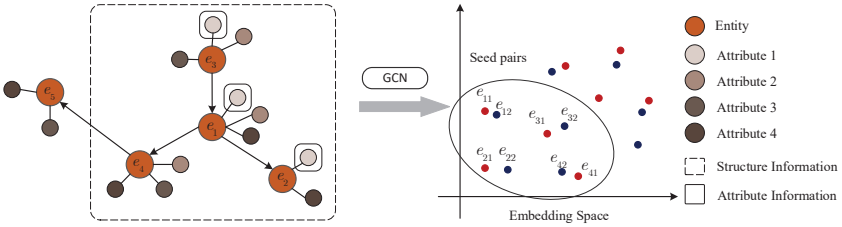


Fig. 1. The framework of our model. Dashed-line rectangle represents the structure information of e_1. The solid-line rectangle represents the attribute information of e_1 with respect to attribute 4. GCN embeds entities into a unified embedding space.

Structure and Attribute Embedding. Equivalent entities in multiple KGs are assumed to have similar neighbours (structure information) and attribute names (attribute information). To capture these information, GCN is utilized to operate on KGs and produce node-level embeddings for all entities. An entity's structure information can be represented by $\mathbf{x}_s$, as thus, the matrix encoding structure information of all entities is denoted by $\mathbf{X}_s$, which is randomly initialized and updated during model training in our setting. Similarly, the attribute feature of an entity can be represented by a vector $\mathbf{x}_a$, and the corresponding attribute feature matrix for all entities is $\mathbf{X}_a$. The initial attribute matrix is pre-computed, as detailed in the following. Note that following previous works, here we focus on attribute names, instead of attribute values.

In [5], attribute information is converted into a k-dimension one-hot vector encoding k most frequent attributes. Nonetheless, this setting fails to differentiate entities or consider the neighboring information. In our model, the most frequent attributes (which we define as attributes appearing with more than 80% of entities) are discarded for representing an entity since they appear with many entities and are not discriminative. Among the rest of attributes, we select k most frequent ones as they can better distinguish entities and are not too long-tail (which might result in very sparse attribute matrix). For an entity, its attribute feature vector can be denoted by $\mathbf{x}_a = [x_a^1, x_a^2, ..., x_a^k]$, where

$$x_a^i = \frac{n_i}{\sum_{j=1}^{k} n_j}. \tag{1}$$

n_i is the total times of i-th attribute appearing among the attributes of an entity and its one-hop neighbours, which is harnessed to capture local attribute information. As thus, both local and global attribute information can be encoded.

The inputs of GCN model include $\mathbf{X}_s$, $\mathbf{X}_a$, and adjacency matrix $\mathbf{A}$. By feeding the inputs into GCN model, the output entity embedding matrix is:

$$[\mathbf{C}_s; \mathbf{C}_a] = GCN(\mathbf{A}, [\mathbf{X}_s; \mathbf{X}_a]), \tag{2}$$

where $[;]$ denotes the concatenation of two matrices, $\mathbf{C}_s \in \mathbb{R}^{N \times d_s}$ is the final structure embedding matrix and $\mathbf{C}_a \in \mathbb{R}^{N \times d_a}$ represents the final attribute embedding matrix. In our model, we harness two 2-layer GCNs to generate embeddings for entities in two KGs respectively. The dimensionalities of structure and attribute feature vectors are set to d_s and d_a for all layers in respective models.

Distance Function. A weighted distance function, which combines structure embedding and attribute embedding, is designed for entity alignment prediction. Concretely, for $e_{i1} \in G_1$ and $e_{i2} \in G_2$, the distance can be calculated by:

$$Dis(e_{i1}, e_{i2}) = \theta Dis_s(e_{i1}, e_{i2}) + (1-\theta) Dis_a(e_{i1}, e_{i2}), \tag{3}$$

where θ is a hyper-parameter balancing the importance of structure embedding distance and attribute embedding distance. The structure (attribute) embedding distance is defined as the vector norm of $\mathbf{c}_s^{i1} - \mathbf{c}_s^{i2}(\mathbf{c}_a^{i1} - \mathbf{c}_a^{i2})$ divided by the dimensionality $d_s(d_a)$. The distance between equivalent entities is expected to be as small as possible. As thus, the entity in G_2 with the smallest distance from a specific entity $e_{i1} \in G_1$ can be regraded as the counterpart of e_{i1}.

Loss Function. We use pre-aligned entity pairs S to train GCN models. The training objectives for learning structure embedding and attribute embedding are to minimize the following margin-based ranking loss functions,

$$\mathcal{J}_s = \sum_{(e_1,e_2)\in S} \sum_{(v_1,v_2)\in S^-} d_s \cdot [Dis_s(e_1, e_2) - Dis_s(v_1, v_2) + \gamma_s]_+, \tag{4}$$

$$\mathcal{J}_a = \sum_{(e_1,e_2)\in S} \sum_{(v_1,v_2)\in S^-} d_a \cdot [Dis_a(e_1, e_2) - Dis_a(v_1, v_2) + \gamma_a]_+, \tag{5}$$

where $[x]_+ = max\{0, x\}$, S^- denotes the set of negative aligned entity pairs; γ_s and γ_a are two positive margins separating positive and negative aligned entity pairs. Loss functions $\mathcal{J}_s$ and $\mathcal{J}_a$ are optimized by stochastic gradient descent (SGD) separately.

Iterative Training. Considering the lack of labelled data, inspired by [8], we adopt semi-supervised training strategy to enlarge the training set iteratively by including aligned pairs with high confidence during training process.

Once newly-aligned entity pairs are added into S, they are considered as valid training data. However, some false positive pairs may be included, which will hurt the following training process. Consequently, the key challenge is how to choose highly confident samples from newly-aligned entity pairs to enlarge S. In consequence, we consider candidate entity pairs $\{(e_{i1}, e_{j2}) | e_{i1} \in G_1 \backslash S_1, e_{j2} \in G_2 \backslash S_2\}$, which satisfy $e_{i1} = \arg\min Dis(\_, e_{j2}), e_{j2} = \arg\min Dis(e_{i1}, \_)$, as reliable aligned pairs for iterative training, where S_1 and S_2 are the set of pre-aligned entities in G_1 and G_2 respectively.

3 Experiment

Datasets. We adopt the widely used DBP15K datasets in the experiments, which were developed by [4]. The datasets were constructed from subsets of DBpedia, which has multiple versions in different languages. DBP15K consists of three datasets, Chinese-English (Zh-En), Japanese-English (Ja-En), and French-English (Fr-En). In each dataset, there are 15 thousand already-known equivalent entity pairs, 30% of which are used for training and 70% of which are for testing.

Parameter Settings. In our GCN models, the dimensionality of structure embedding and attribute embedding in all layers were set to $d_s = 300$ and $d_a = 600$ respectively. The number of top attributes k is set to 1000. The iterative training processing would not stop until the size of the newly-included set $|C|$ is under a threshold $\alpha = 100$. The margins γ_s and γ_a are set to 3. The hyper-parameter θ in weighted distance function is set to 0.9.

Competing Approaches and Evaluation Metric. Three approaches are utilized for comparison, including MTransE [3], JAPE [4], and GCN [5]. The evaluation metric, $Hits@k$, measures the proportion of correctly aligned entities in top k ranked candidates. We report the results of $Hits@1$ (accuracy), $Hits@10$, and $Hits@50$ in the experiment.

Table 1. Experimental Results

	$Zh-En$			$En-Zh$		
	$Hits@1$	$Hits@10$	$Hits@50$	$Hits@1$	$Hits@10$	$Hits@50$
MTransE	30.83	61.41	79.12	24.78	52.42	70.45
JAPE	41.18	74.46	88.9	40.15	71.05	86.18
GCN	41.25	74.38	86.23	36.49	69.94	82.45
Inga	**50.45**	**79.42**	**89.79**	**49.36**	**76.05**	**86.38**
	$Ja-En$			$En-Ja$		
	$Hits@1$	$Hits@10$	$Hits@50$	$Hits@1$	$Hits@10$	$Hits@50$
MTransE	27.86	57.45	75.94	23.72	49.92	67.93
JAPE	36.25	68.5	85.35	38.37	67.27	82.65
GCN	39.91	74.46	86.1	38.42	71.81	83.72
Inga	**51.46**	**79.46**	**88.25**	**51.05**	**77.04**	**86.27**
	$Fr-En$			$En-Fr$		
	$Hits@1$	$Hits@10$	$Hits@50$	$Hits@1$	$Hits@10$	$Hits@50$
MTransE	24.41	55.55	74.41	21.26	50.6	69.93
JAPE	32.39	66.68	83.19	32.97	65.91	82.38
GCN	37.29	74.49	86.73	36.77	73.06	86.39
Inga	**50.45**	**79.42**	**87.79**	**49.36**	**76.05**	**86.48**

Experiment Results. The experimental results of Inga and three competitors on DBP15K datasets are shown in Table 1. It can be easily observed that

Inga achieves the best performance among most settings on three bi-directional datasets.

Among the four approaches, MTransE achieves relatively worse results. The $Hits@1$ values of MTransE on all datasets are between 20% and 30%, indicating that translation-based KG embeddings can capture structure information and serve as useful signals for EA. Another KG embedding based method, JAPE, outperforms MTransE significantly by over 10% in most cases due to its ability to incorporate attribute information for refining entity structure embeddings. GCN attains slightly better results than JAPE on $Ja-En$ and $Fr-En$ language pairs, indicating the effectiveness of GCN model for generating structure representation. Inga is built on the architecture of GCN, whereas it improves the results by a large margin. In both alignment directions, Inga outperforms GCN and JAPE by about $3\% - 12\%$ regarding all $Hits@k$ metrics. This demonstrates the usefulness of the improved attribute feature representation and iterative training strategy.

Noteworthy is that the gap between Inga and the rest approaches is much larger on $Hits@1$ (accuracy) than other metrics. This reveals that Inga can align more *accurate* entity pairs, which is critical to EA task.

4 Conclusion

In this paper, we propose a GCN-based model to align entities in different KGs by projecting entities into a unified embedding space, where equivalent entities are placed close to each other. Attribute representation is improved by capturing more informative attribute features. Furthermore, we devise an iterative training strategy to enlarge training set and generate better entity embeddings via neural network. Our proposal is then evaluated on real-life datasets and the results demonstrate that our model outperforms three state-of-the-art competitors by a large margin. For further work, to take more information especially attribute values as guidance for our model is also necessary.

Acknowledgements. This work was partially supported by NSFC under grants Nos. 61872446, 61876193 and 71690233.

References

1. J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li, and X. Li. Neural generative question answering. In *Proceedings of IJCAI*, pages 2972–2978, 2016.
2. B. D. Trisedya, J. Qi, R. Zhang, and W. Wang. GTR-LSTM: A triple encoder for sentence generation from RDF data. In *Proceedings of ACL*, pages 1627–1637, 2018.
3. M. Chen, Y. Tian, M. Yang, and C. Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of IJCAI*, pages 1511–1517, 2017.
4. Z. Sun, W. Hu, and C. Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In *Proceedings of ISWC, Part I*, pages 628–644, 2017.
5. Z. Wang, Q. Lv, X. Lan, and Y. Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of EMNLP*, pages 349–357, 2018.
6. T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.
7. F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2011.
8. H. Zhu, R. Xie, Z. Liu, and M. Sun. Iterative entity alignment via joint knowledge embeddings. In *Proceedings of IJCAI*, pages 4258–4264, 2017.

Knowledge Reconciliation with Graph Convolutional Networks: Preliminary Results*

Pierre Monnin[1], Chedy Raïssi[1,2], Amedeo Napoli[1], and Adrien Coulet[1,3]

[1] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
{pierre.monnin, chedy.raissi, amedeo.napoli, adrien.coulet}@loria.fr
[2] Ubisoft, Singapore
[3] Stanford Center for Biomedical Informatics Research, Stanford University, 94305 Stanford, California, USA

Abstract. In this article, we investigate the task of identifying nodes that are identical, more general, or similar within and across knowledge graphs. This task can be seen as an extension of instance matching or entity resolution and is here named knowledge reconciliation. In particular, we explore how Graph Convolutional Networks (GCNs), previously defined in the literature, can be used for this task and evaluate their performance on a real world use case in the domain of pharmacogenomics (PGx), which studies how gene variations impact drug responses. PGx knowledge is represented in the form of n-ary relationships between one or more genomic variations, drugs, and phenotypes. In a knowledge graph named PGxLOD, such relationships are available, coming from three distinct provenances (a reference database, the biomedical literature and Electronic Health Records). We present and discuss our preliminary attempt to generate graph embeddings with GCNs and to use a simple distance between embeddings to assess the similarity between relationships. By experimenting on the 68,686 PGx relationships of PGxLOD, we found that this approach raises several research questions. For example, we discuss the use of the semantics associated with knowledge graphs within GCNs, which is of interest in the considered use case.

Keywords: Knowledge Reconciliation · N-ary relationships · Graph Embeddings · Graph Convolutional Networks.

1 Introduction

Data and knowledge can be accessed extensively on the Web and interpreted by both human and software agents. Because these elements of knowledge are of various provenances, spread in various places and published following distinct standards, it is challenging to compare and conjointly use their content. Semantic Web and Linked Open Data (LOD) [2] provide standards and technologies to

* Supported by the *PractiKPharma* project, founded by the French National Research Agency (ANR) under Grant ANR15-CE23-0028, by the IDEX "Lorraine Université dExcellence" (15-IDEX-0004) and by the *Snowball* Inria Associate Team.

facilitate the interoperability of knowledge spread over the Web, such as Uniform Resource Identifiers (URIs) and the Resource Description Framework (RDF) format. URIs identify nodes that can represent entities of the real world (*e.g.*, places, persons, drugs), while RDF statements represent edges, using predicates to link entities to each others or to literals (*e.g.*, strings, integers). Such predicates express the semantics of the relationship that connects two nodes or a node and a literal (*e.g.*, *is-born-in*, *has-firstname*). Therefore, URIs and RDF statements enable to represent knowledge in the form of a directed and labeled multigraph, loosely called a *knowledge graph*.

Because datasets are independently published on the Web, possibly with some overlap, it happens that different URIs are used to identify the same resource. For example, `dbpedia:Warfarin` and `wikidata:Q407431` are two URIs representing the chemical compound *Warfarin* in DBpedia and Wikidata. As a consequence, identifying different URIs possibly referring to the same resource is necessary to use various and initially independent datasets together. This task, called *instance matching* [6] or *entity resolution*, can be extended to identify not only identical resources but also more general or somehow similar ones, a task we call *knowledge reconciliation* (by analogy with reconciliation in databases [1]).

In this work, we illustrate this task with a real world application in the field of pharmacogenomics (abbreviated PGx). This field studies the influence of genomic variations in drug response phenotypes. Knowledge in PGx is typically composed of n-ary relationships between one or more genomic variations, drugs and phenotypes, stating that a patient having the specified genomic variations, and being treated with the specified drugs will be more likely to experience the given phenotypes. PGx relationships can be found in different sources: reference databases, biomedical literature, or by mining Electronic Health Records (EHRs). Therefore, there is a need to reconcile these PGx relationships from different sources, for example to confirm state-of-the-art knowledge found in the literature with clinical counterpart found in EHRs [5].

Several existing works use Semantic Web technologies to represent PGx knowledge, as they allow to easily relate resources to other nodes in the knowledge graph that can enrich their semantics (*e.g.*, *partOf* resources, classes of ontologies). For example, we built PGxLOD [10], a large knowledge graph containing 68,686 PGx relationships from the three aforementioned sources. As Semantic Web technologies only allow binary predicates, to be represented, PGx relationships are reified: the relationship itself is a node, linked by predicates to its components. For example, Figure 1 depicts the reification as the node `pgx_rel_1` of a ternary relationship between gene `CYP2C9`, drug `warfarin` and phenotype `cardiovascular_diseases`. A PGx relationship is fully defined by its components and, accordingly, two relationships involving the same sets of components are identical. Hence, reconciliation techniques based on the relational structure of nodes [6] are well-suited to reconcile PGx relationships represented using Semantic Web technologies.

In this paper, we investigate how the task of knowledge reconciliation can be achieved using graph embeddings [4], *i.e.*, low-dimensional vectors representing

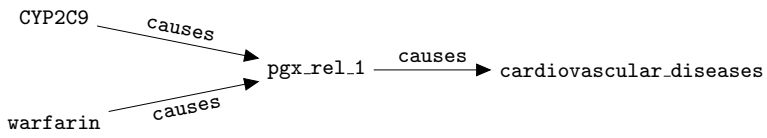


Fig. 1. Representation of a reified ternary PGx relationship between gene `CYP2C9`, drug `warfarin` and phenotype `cardiovascular_diseases`. The relationship is reified as the node `pgx_rel_1`, which is connected to its components by the `causes` predicate.

graph structures (*e.g.*, nodes, edges, subgraphs) while preserving variously the properties of the graph. Particularly, we present the preliminary results of an original experiment using Graph Convolutional Networks (GCNs) [8, 15] that have already been successfully used for link prediction, a task somehow similar to knowledge reconciliation. GCNs compute an embedding for each node considering its neighbors, and, thus, are well adapted to our task in which the relational structure is of prime importance. Similarity between n-ary relationships could be represented by ensuring a low distance between their respective embeddings. Inspired by recent works [13, 16], we use definitions of inverses of predicates to illustrate how semantics of knowledge graphs could be used in GCNs. We experimented by reconciling the 68,686 PGx relationships from PGxLOD [10]. The remainder of this article is organized as follows. Section 2 presents related works. Section 3 details GCNs and the proposed general setting for knowledge reconciliation of n-ary relationships. Section 4 describes our experiment with the biomedical knowledge graph PGxLOD. Finally, we discuss our results and future directions in Section 5.

2 Related Works

Numerous works exist on ontology matching. The interested reader could refer to [6] for a detailed presentation of approaches. In the following, we focus on graph embeddings techniques, that have been investigated in multiple works and successfully applied on knowledge graphs for tasks such as node classification or link prediction [8, 14, 15]. Works differ in the considered type of graphs (*e.g.*, homogeneous graphs, heterogeneous graphs such as knowledge graphs) or in the graph embedding techniques used (*e.g.*, matrix factorization, deep learning with or without random walk), as listed in the taxonomies of problems and techniques in Cai *et al.* survey [4]. In the following, few specific examples are detailed but a more thorough overview can be found in some of the existing surveys [4, 11].

A first example is TransE [3], which computes for each triple $\langle s, p, o \rangle$ of a knowledge graph, embeddings h_s, h_p, h_o, such that $h_s + h_p \approx h_o$, *i.e.*, the translation vector from the subject to the object of a triple corresponds to the embedding of the predicate. This approach is adapted for link prediction but, according to the authors, it is unclear if it can model adequately relationships

of distinct arities, such as *1-to-Many*, or *Many-to-Many*. Another example is RDF2Vec [14], which first extracts, for each node, a set of sequences of graph sub-structures starting from this node. Elements in these sequences can be edges, nodes or even subtrees. Then, sequences feed algorithms from the word2vec neural language model that compute embeddings for each element in a sequence by either maximizing the probability of an element given the other elements of the sequence (Continuous Bag of World architecture) or maximizing the probability of the other elements given the considered element (Skip-gram architecture). A third approach, adopted in this article, is GCNs that have been introduced in [8] for semi-supervised classification on graphs and extended in [15] for entity classification and link prediction in knowledge graphs. Contrasting TransE and RDF2Vec that respectively work at the triple and sequence levels, GCNs compute the embedding of a node by considering its neighborhood. Therefore, GCNs seem more suited to the task of reconciling n-ary relationships, that are entirely defined by their neighboring nodes representing their components.

However, previous methods do not consider the semantics associated with predicates and nodes. Alternatively, Logic Tensor Networks [16] are used to learn groundings. The grounding of a logical term is a vector of real numbers (*i.e.*, an embedding) and the grounding of a logical clause is a real number in the interval $[0, 1]$ (*i.e.*, the confidence in the truth of the clause). The learning process tries to minimize the satisfiability error of a set of clauses, while trying to ensure the logical reasoning. This work can interestingly be compared to graph embeddings if knowledge graphs are considered in their logical form, *i.e.*, considering nodes as logical terms and edges linking two nodes as logical formulae. We adopted such consideration by exploring in this work a first manner to include (limited) semantics within GCNs, for the knowledge reconciliation task.

3 Knowledge Reconciliation with GCNs

3.1 Learning Task

We consider that we have at our disposal a knowledge graph, with specific nodes representing reified n-ary relationships. Our task consists in learning embeddings for these relationships such as their distance reflects their similarity or dissimilarity. The learning task relies on two elements: the learning of embeddings associated with each node of the graph and the assessment of the similarity between nodes representing n-ary relationships by computing the distance between their respective embeddings.

To train our GCN model, we constitute a training set and a test set made of a balanced number of positive and negative examples. Regarding positive examples, we assume that some n-ary relationships are already labeled as similar from a manual labeling or from the execution of another method, such as the similarity rules validated by an expert we use in Section 4. This similarity labels may have different levels (*e.g.*, very high, high) or may reflect different semantics (*e.g.*, identical relationships, more general ones). However in this preliminary setting, we do not take into account such detailed semantics and only consider a

coarse-grained similarity: similar or not labeled. Indeed, as knowledge graphs are built under the *Open World Assumption*, absent statements from a knowledge graph are only unknown and not false. For this reason, we consider that we have at our disposal only positive similarity labeling for n-ary relationships. To generate "negative" examples, an approach similar to the one used in TransE [3] is considered: for each pair of similar n-ary relationships (i, j), another n-ary relationship k is found such as it is not labeled as similar either to i or to j. The triple (i, j, k) representing a training example is then added to the training set S. The same method is used to generate the test set.

3.2 Using GCNs to generate graph embeddings

In the following, we adopt the notations defined in [15]. As such, $\mathcal{R}$ denotes the set of predicates in the considered knowledge graph. Considering a node i and a predicate $r \in \mathcal{R}$, we denote $\mathcal{N}_i^r$ the set of nodes reachable from i by r. Only nodes and predicates linking nodes are considered. Literals and predicates linking nodes to literals are discarded.

GCNs can be seen as a message-passing framework of multiple layers, in which the embedding $h_i^{(l+1)}$ of a node i at layer $(l+1)$ depends on the embeddings of its neighbors at level (l), as stated in Equation (1).

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \tag{1}$$

The convolution over the neighboring nodes of i is computed with a specific weight matrix $W_r^{(l)}$ for each predicate $r \in \mathcal{R}$ and each layer (l). This convolution is regularized by a constant $c_{i,r}$, that can be set for each node and each predicate. Additionally, to ensure that the embedding of i at layer $(l+1)$ also depends on its embedding at layer (l), a self connection is allowed, with the weight matrix $W_0^{(l)}$. σ is a non-linear function such as ReLU, used in our experiment (Section 4).

Authors in [15] consider that every predicate $r \in \mathcal{R}$ has an inverse $r_{inv} \in \mathcal{R}$. As this is not always true in knowledge graphs, and to illustrate how the semantics of predicates could be used in GCNs, we leverage potentially defined inverse predicates. We consider the three following cases for a predicate r:

(i) If r is defined as symmetric, we do not consider an inverse but ensure that the adjacency matrix for r is symmetric;
(ii) If r has a defined inverse r^{-1}, we use r and r^{-1} and ensure their adjacency matrices are indeed representing inverse relations;
(iii) Otherwise, we generate an inverse r_{inv} such as its adjacency matrix represent the inverse of r.

By doing so, we avoid generating an abstract inverse r_{inv} for predicates r having a defined inverse r^{-1} or being symmetric, which would add unnecessary messages. Indeed, consider a predicate r, its defined inverse r^{-1}, and two nodes i and j such that edges $i \xrightarrow{r} j$ and $j \xrightarrow{r^{-1}} i$ are in the knowledge graph. By always

generating an abstract inverse, two edges would be added, $j \xrightarrow{r_{inv}} i$ and $i \xrightarrow{r_{inv}^{-1}} j$, duplicating the existing edges and adding two unnecessary messages.

To train GCNs, we minimize the loss function presented in Equation (2), inspired from the one used in TransE [3].

$$\mathcal{L} = \sum_{(i,j,k) \in S} \max\Big(||h_i - h_j||_2 + \gamma - ||h_i - h_k||_2,\ 0 \Big) \tag{2}$$

Given a training example (i, j, k) from the training set S, minimizing the loss function aims at minimizing the distance between h_i and h_j, *i.e.*, ensuring their similarity, while maximizing the distance between h_i and h_k. The constant γ is a margin hyperparameter aiming at increasing the difference between the two distances.

4 Experimentation on PGx Knowledge

4.1 Input Knowledge Graph: PGxLOD

We experimented our approach on PGxLOD [10], a large knowledge graph containing PGx relationships from three distinct sources: PharmGKB (a reference database), the biomedical literature and results from studies on Electronic Health Records. Main statistics of PGxLOD are presented in Table 1.

Table 1. Main statistics of PGxLOD. The line "Predicates" only counts predicates used to link two nodes together, excluding literals. As our experiment uses a 3-layer network, only the 3-hop neighborhood of PGx relationships is considered to compute their embeddings.

Triples	59,136,400
Nodes and literals	25,226,599
Predicates	378
PGx relationships	68,686
↳ Nodes in their 3-hop neighborhood	2,943,613
↳ Edges in their 3-hop neighborhood	32,773,429
Similarity links between PGx relationships	283,248
↳ `owl:sameAs` links	109,226
↳ `skos:broadMatch` links	136,264
↳ `skos:relatedMatch` links	37,758

In PGxLOD, some similar PGx relationships are already labeled by being linked together with one of the three following predicates: `owl:sameAs` expressing identical relationships, `skos:broadMatch` expressing more general ones and `skos:relatedMatch` expressing related ones to some extent. Such labels result from the application of logical reconciliation rules that are described in [10].

They are used to constitute the training and test sets, in a "knowledge graph as silver standard" perspective [12]. Even if `owl:sameAs`, `skos:broadMatch` and `skos:relatedMatch` links express different similarity semantics, here we indifferently consider the three predicates as expressing a coarse-grained similarity. Thus, two PGx relationships are labeled as similar if they are linked by one of these three predicates. Additionally, we consider their adjacencies in an undirected perspective, *i.e.*, having (i, j) as a similarity edge is equivalent to having (j, i). As `owl:sameAs` and `skos:relatedMatch` are symmetric, numbers of available links for training and test sets are consequently half of those presented in Table 1. For each predicate, the training set is constituted by $\frac{2}{3}$ of the links and the test set by $\frac{1}{3}$. To form triples (i, j, k) in these sets, k is chosen such as it is not directly linked via a similarity predicate either to i or to j.

4.2 Experimental Setting

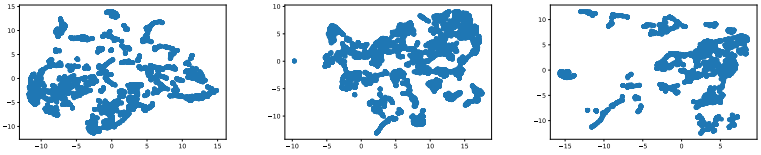
For our preliminary experiment, we use a standard architecture and hyperparameters previously reported in the literature with successful uses of GCNs. We consider a 3-layer network where each layer uses ReLU as the activation function. In such a 3-layer architecture, only neighboring nodes up to 3 hops of PGx relationships will have an impact on their embeddings, output at layer 3 (Equation (1)). The input layer consists in a featureless approach as in [8, 15], *i.e.*, the input is just a one-hot vector for each node of the graph. Both the input layer and the hidden layer have an output dimension of 16 while the output layer has an output dimension of 10. Therefore, embeddings for all nodes in the knowledge graph are in $\mathbb{R}^{10}$. Only the embeddings for nodes representing the reified PGx relationships are of interest in our reconciliation task and are considered in the loss function. As in [15], the constant $c_{i,r}$ is set to $|\mathcal{N}_i^r|$ and we use the basis-decomposition with 10 basis to avoid the growth in number of parameters. For the learning process, we use the Adam optimizer [7] with a starting learning rate of 0.01 and a L2-regularization coefficient of 0.0005. The margin hyperparameter γ is set to 2. Our experiment was implemented using PyTorch and the Deep Graph Library.

4.3 Results

We trained our model during 60 epochs. The last layer outputs embeddings for all nodes in the graph but we only consider the ones representing reified PGx relationships. The mean and standard deviation of distances between embeddings of similar relationships in the training set were respectively $\mu_{train} = 1.93$ and $\sigma_{train} = 4.18$. As a simple evaluation, for each example (i, j, k) from the test set, (i, j) or (i, k) were considered as similar if their embeddings were distant of less than $\mu_{train} + \sigma_{train}$. These results were compared with existing similarity links, obtaining a precision of 0.92, a recall of 0.94 and a F1-score of 0.93.

Then, we investigated differences between the three similarity predicates. 2D projections of embeddings using UMAP [9] are depicted in Figure 2. We can see

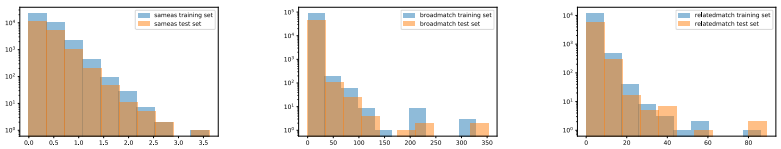
that clusters of nodes are appearing but seem still close. More epochs or a wider neighborhood may allow to emphasize the difference between such clusters.



(a) `sameAs` links (b) `broadMatch` links (c) `relatedMatch` links

Fig. 2. 2D projections using UMAP [9] of embeddings of PGx relationships involved in the given similarity predicates.

Figure 3 depicts the distributions of distances between embeddings of similar PGx relationships depending on their similarity link. The low means for the three predicates illustrate that similar relationships have indeed embeddings with low distances. We notice that the mean distances for `owl:sameAs` are the lowest while the ones for `skos:relatedMatch` are the greatest. This could indicate the ability of the network to learn the close similarity expressed by `owl:sameAs` links and the more fuzzy one expressed by `skos:relatedMatch` links. Regarding `skos:broadMatch` links, the distance ranges and variances are more important than for the two other predicates. This could also illustrate the ability to learn the semantics of the predicate. Additionally, `skos:broadMatch` links are directed, and thus, could be more difficult to fit correctly, mixed with symmetric similarity predicates. Also, only `skos:broadMatch` links connect PGx relationships across the three considered sources [10]. Therefore, they link together relationships that may have more diversity in the semantics and vocabularies of their components, potentially explaining the higher distance ranges and variances.



(a) `sameAs` links
$\mu_{train} = 0.33$
$\sigma_{train} = 0.26$
$\mu_{test} = 0.33$
$\sigma_{test} = 0.26$

(b) `broadMatch` links
$\mu_{train} = 2.27$
$\sigma_{train} = 4.90$
$\mu_{test} = 2.30$
$\sigma_{test} = 5.12$

(c) `relatedMatch` links
$\mu_{train} = 4.16$
$\sigma_{train} = 2.77$
$\mu_{test} = 4.35$
$\sigma_{test} = 3.31$

Fig. 3. Distributions of distances between embeddings of similar PGx relationships linked by the given similarity predicates. μ and σ respectively denote the mean and the standard deviation.

5 Discussion and Conclusion

In this paper, we investigated the task of generating graph embeddings for knowledge reconciliation using Graph Convolutional Networks and ensuring that embeddings associated with similar reified n-ary relationships have a low distance. We experimented our approach on the real world use case of reconciling PGx n-ary relationships from three distinct sources. Our preliminary results found this approach to be suitable and to raise several research questions.

First, the network output different distances for the three considered predicates: `owl:sameAs`, `skos:broadMatch` and `skos:relatedMatch`. This could indicate that it was able to learn their different similarity semantics or had difficulties to adequately fit some of them. Possible improvements would be to increase the number of epochs or test other values for hyperparameters. The loss function could integrate the different semantics of the predicates linking similar relationships i and j, for example by considering three different weighted sums. Separate models could be learned: one per predicate or one for `owl:sameAs` and `skos:relatedMatch` and another for `skos:broadMatch` which is not symmetric, which could make easier interpreting the semantics of the output similarity.

Because we used a 3-layer network, only nodes in the 3-hop neighborhood of PGx relationships were considered for the computation of their embeddings. However, nodes in further neighborhoods may bring additional semantics. In particular, phenotypes extracted from the biomedical literature are frequently complex and formed by several simpler phenotypes, indicated by *dependsOn* links. Therefore, we could benefit from using a network with more layers.

We also illustrated how semantics associated with a knowledge graph can be used in GCNs by considering the definitions of inverses of predicates. This could be improved, for example by considering the semantics of `owl:sameAs` links between nodes. Indeed, these links indicate identical nodes that are currently considered as neighboring nodes and used as such in the embeddings computation. Thus, a pre-processing step could consist in mapping nodes linked by `owl:sameAs` links into a unique node. Additionally, the generation of negative examples could be improved by considering ontologies. In such case, PGx relationships whose components instantiate classes in different parts of an ontology could be more interesting negative examples.

Our model was evaluated using a manually-defined threshold on distances between embeddings of relationships to assess their similarity. Other methods such as (multi-)classification machine learning models could also be investigated. Advanced performance results could also be computed on knowledge graphs from other domains as well as be compared with other state-of-the-art methods presented in Section 2. Finally, we should manually check on a few examples whether relationships considered as close given the distance between their embeddings but not linked by any of the similarity predicates are indeed similar.

To conclude, these results constitute solely an initial attempt to use graph embeddings for the non-trivial task of reconciling n-ary relationships. Several future directions are considered, among which is the further integration of the semantics associated with knowledge graphs in GCNs.

References

1. Abiteboul, S., Manolescu, I., Rigaux, P., Rousset, M., Senellart, P.: Web Data Management. Cambridge University Press (2011)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. **5**(3), 1–22 (2009)
3. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 2787–2795 (2013)
4. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
5. Coulet, A., Smaïl-Tabbone, M.: Mining electronic health records to validate knowledge in pharmacogenomics. ERCIM News **2016**(104) (2016)
6. Euzenat, J., Shvaiko, P.: Ontology Matching, Second Edition. Springer (2013)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014), http://arxiv.org/abs/1412.6980
8. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR **abs/1609.02907** (2016), http://arxiv.org/abs/1609.02907
9. McInnes, L., Healy, J., Saul, N., Grossberger, L.: Umap: Uniform manifold approximation and projection. The Journal of Open Source Software **3**(29), 861 (2018)
10. Monnin, P., Legrand, J., Husson, G., Ringot, P., Tchechmedjiev, A., Jonquet, C., Napoli, A., Coulet, A.: PGxO and PGxLOD: a reconciliation of pharmacogenomic knowledge of various provenances, enabling further comparison. BMC Bioinformatics **20-S**(4), 139:1–139:16 (2019)
11. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. Proceedings of the IEEE **104**(1), 11–33 (2016)
12. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017)
13. Paulheim, H.: Make embeddings semantic again! In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018. (2018)
14. Ristoski, P., Paulheim, H.: Rdf2vec: RDF graph embeddings for data mining. In: The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I. pp. 498–514 (2016)
15. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings. pp. 593–607 (2018)
16. Serafini, L., d'Avila Garcez, A.S.: Learning and reasoning with logic tensor networks. In: AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings. pp. 334–348 (2016)

End-to-End Learning for Answering Structured Queries Directly over Text

Paul Groth[1], Antony Scerri[2], Ron Daniel, Jr.[2], and Bradley P. Allen[2]

[1] University of Amsterdam p.groth@springer.com
[2] Elsevier Labs
{a.scerri,r.danel,b.allen}@elsevier.com

Abstract. Structured queries expressed in languages (such as SQL, SPARQL, or XQuery) offer a convenient and explicit way for users to express their information needs for a number of tasks. In this work, we present an approach to answer these directly over text data without storing results in a database. We specifically look at the case of knowledge bases where queries are over entities and the relations between them. Our approach combines distributed query answering (e.g. Triple Pattern Fragments) with models built for extractive question answering. Importantly, by applying distributed querying answering we are able to simplify the model learning problem. We train models for a large portion (572) of the relations within Wikidata and achieve an average 0.70 F1 measure across all models. We describe both a method to construct the necessary training data for this task from knowledge graphs as well as a prototype implementation.

1 Introduction

Database query languages (e.g. SQL, SPARQL, XQuery) offer a convenient and explicit way for users to express their information needs for a number of tasks including populating a dataframe for statistical analysis, selecting data for display on a website, defining an aggregation of two datasets, or generating reports.

However, much of the information that a user might wish to access using a structured query may not be available in a database and instead available only in an unstructured form (e.g. text documents). To overcome this gap, the area of *information extraction* (IE) specifically investigates the creation of structured data from unstructured content [15]. Typically, IE systems are organized as pipelines taking in documents and generating various forms of structured data from it. This includes the extraction of relations, the recognition of entities, and even the complete construction of databases. The goal then of IE is not to answer queries directly but first to generate a database that queries can be subsequently executed over.

In the mid-2000s, with the rise of large scale web text, the notion of combining information extraction techniques with relational database management systems emerged [4, 10] resulting in what are termed *text databases*. Systems like Deep Dive [22] InstaRead [9], or Indrex [11], use database optimizations

within tasks such as query planning to help decide when to perform extractions. While, in some cases, extraction of data can be performed at runtime, data is still extracted to an intermediate database before the query is answered. Thus, all these approaches still require the existence of a structured database to answer the query.

In this paper, we present an approach that **eliminates the need to have an intermediate database in order to answer structured database queries over text**. This is essentially the same as treating the text itself as the store of structured data. Using text as the database has a number of potential benefits, including being able to run structured queries over new text without the need for a-priori extraction; removing the need to maintain two stores for the same information (i.e. a database and a search index); eliminating synchronization issues; and reducing the need for up-front schema modeling. [3] provides additional rationale for not pre-indexing "raw data", although they focus on structured data in the form of CSV files.

Our approach builds upon three foundations: 1. the existence of large scale publicly available knowledge bases (Wikidata) derived from text data (Wikipedia); 2. recent advances in end-to-end learning for extractive question answering (e.g. [21]); 3. the availability of layered query processing engines designed for distributed data (e.g. SPARQL query processes that work over Triple Pattern Fragment [25] servers).

A high-level summary of our approach is as follows. We use a publicly-available knowledge base to construct a parallel corpus consisting of tuples each which is made up of a structured slot filling query, the expected answer drawn from the knowledge base, and a corresponding text document in which we know the answer is contained. Using this corpus, we train neural models that learn to answer the given structured query given a text document. This is done on a per relation basis. These models are trained end-to-end with no specific tuning for each query. These models are integrated into a system that answers queries expressed in a graph query language directly over text with no relational or graph database intermediary.

The contributions of this paper are:

- an approach, including training data generation, for the task of answering structured queries over text;
- models that can answer slot filling queries for over 570 relations with no relation or type specific tuning. These models obtain on average a 0.70 F1 measure for query answering.
- a prototype system that answers structured queries using triple pattern fragments over a large corpus of text (Wikipedia).

The rest of this paper is organized as follows. We begin with an overview of the approach. This is followed by a description of the training data. Subsequently, we describe the model training and discuss the experimental results. After which, we present our prototype system. We end the paper with a discussion of related and future work.

2 Overview

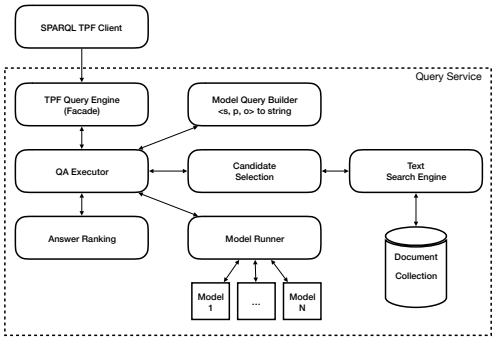


Fig. 1: Components of the overall system for structured query answering over text.

Our overall approach consists of several components as illustrated in Figure 1. First, structured queries as expressed by SPARQL [7] are executed using a Triple Pattern Client (SPARQL TPF Client). Such a client breaks down a more complex SPARQL query into a series of triple patterns that are then issued to a service. Triple patterns are queries of the form subject, predicate, object, where each portion can be bound to an identifier (i.e. URI) or a variable.[3] Within the service, the execution engine (QA Executor) first lexicalizes the given identifiers into strings using the Model Query Builder component. For example, this component would translate an identifier like https://www.wikidata.org/wiki/Q727 into the string form "Amsterdam". These queries are then issued to a candidate selection component. This component queries a standard text search engine to find potential documents that could contain the answer to the specified query.

The candidate documents along with the lexicalized queries are provided to a model runner which issues these to models trained specifically to bind the variable that is missing. That is given a query of the form $< s, p, ?o >$ where s and o are bound and o is the variable, there would be specific models trained to extract $?o$ from the provided document. For example, given the query (:Amsterdam :capital_of $?o$) we would have models that know how to answer queries of where the type of the subject is City and the property is capital_of. Likewise, there would be models of that are able to answer queries of the form $<?s, p, o >$ and so on. Each model is then asked to generate a binding of the variable. Note that the bindings generated by the models are strings. The results of each model are then ranked (Answer Ranking). Using a cut-off, the results are then translated back into identifier space and returned to the client.

[3] Objects can also be bound to a literal.

A key insight of our approach is that by breaking down complex queries into triple patterns we can simplify the queries that need to be answered by the learned models.

Our approach relies on the construction of models that are able to extract potential candidate answers from text. Following from [5] and [13], we cast the problem in terms of a question answering task, where the input is a question (e.g. entity type + relation) and a document and the output is answer span within the document that binds the output. To learn these sorts of models we construct training data from knowledge graphs that have a corresponding representation in text. In the next section, we go into detail about the construction of the necessary training data.

3 Training Data Construction

Our training data is based on the combination of Wikidata and Wikipedia. Wikidata is a publicly accessible and maintained knowledge base of encyclopedic information [27]. It is a graph structured knowledge base (i.e. a knowledge graph) describing entities and the relations between them. Every entity has a globally unique identifier. Entities may also have attributes which have specific datatypes. Entities have may have more than one type. Relations between entities may hold between differing entity types.

Wikidata has a number of properties that make it useful for building a corpus to learn how to answer structured queries over text. First, and perhaps most importantly, entities have a parallel description with Wikipedia. By our count, Wikidata references 7.7 million articles in the English language Wikipedia. Thus, we have body of text which will also most likely contain answers that we retrieve from Wikidata. Second, every entity and relation in Wikidata has a human readable label in multiple languages. This enables us to build a direct connection between the database and text. Third, Wikidata is large enough to provide for adequate training data in order to build models. Finally, Wikidata provides access to their data in a number of ways including as a SPARQL endpoint, a triple patterns fragment endpoint and as a bulk RDF download. While we use Wikidata, we believe that our approach can be extended to any knowledge graph that has textual sources.

Using this input data we generate datasets of the form: [QUERY; ANSWER; TEXT IN WHICH THE QUERY IS ANSWERED]. As previously mentioned, complex queries can be expressed as a series of graph patterns. Thus, the queries we consider are graph patterns in which two of the variables are bound (e.g. :New_England_Patriots :play $?x$). We term these *slot filling* queries as the aim is to bind one slot in the relation (i.e. the subject or the object). While we do not test graph patterns where the predicate is the variable, the same approach is also applicable. In some sense, one can think of this as generating data that can be used to build models that act as substitute indexes of a database.

Our construction method loops through all of the predicates (i.e. relations) in the dataset. It determines the frequency with which a predicate connects different

types of entities. This is essential as large knowledge graphs can connect many different types using the same predicate. Thus, examples from different types of subjects and objects are needed to capture the semantics of that predicate. Using the most frequently occurring pairs of entity types for a predicate, the algorithm then retrieves as many example triples as possible where the subject and object of the triple are instances of the connected types - up to a given maximum threshold. Thresholding is used to help control the size of the training data.

Each triple is then used to generate a row of training data for learning how to answer graph pattern queries that contain the given predicate. To connect the graph pattern queries, which are expressed using entity IRIs to the plain text over which it should be answered, each of the components of the triple is lexicalized. The lexicalized subject and predicate of each triple are concatenated together to form a textual query and use the lexicalized object as the answer. (Note, this is trivally modified for the (?s, p, o case). We then retrieve the text describing the subject. We assume that the text contains some reference to the object under consideration.

The location of that reference which we term an anchor is computed by the given anchor function. For simplicity, in our implementation, we locate the first instance of the answer in the text. This may not always represent an instance of the answer's lexical form which is located in an expression which answers the specific question form. More complex implementations could use different heuristics or could return all possible anchor locations.

We apply the algorithm to the combination of Wikipedia and Wikidata dumps[4]. We attempted to obtain training data for all 1150 predicates in Wikidata that associate two entities together. At this time, we do not treat predicates that connect entities to literals. This is left for future work. We limited the extraction to the top 20 entity type pairs per predicate, and limited each type pair to 300 examples). Thus, there is a maximum yield of 6000 examples per predicate. We then apply the following cleaning/validation to the retrieved examples. First, we drop examples where there is no Wikipedia page. Second, we ensure that the answer is present in the Wikipedia page text. Finally, in order to ensure adequate training data we filter out all models with less than 30 examples. Note that this means that we have differing amounts of training data per predicate. After cleaning, we are able to obtain training data for 572 predicate for the setting in which the object is the variable/answer. We term this the SP setting. On average we have 929 examples per predicate with a maximum number of examples of 5477 and a minimum of 30 examples. The median number of examples is 312. In the setting in which the subject is the variable / answer we are trying to extract, enough data for 717 predicates is obtained. This is because the subject answer is more likely to appear in the Wikipedia page text. We term this the PO setting.

[4] Specifically we used Wikipedia 2018-08-20 (enwiki-20180820-pages-articles-multistream.xml.bz2) and Wikidata 2018-08-29.

4 Models

Based on the above training data, we individual train models for all predicates using the Jack the Reader framework [5]. We use two state-of-the-art deep learning architectures for extractive question answering, namely, FastQA [28] and the implementation provided by the framework, JackQA. Both architectures are interesting in that while they perform well on reading comprehension tasks (e.g. SQuAD [20]) both architectures try to eliminate complex additional layers and thus have the potential for being modified in the future to suit this task. Instead of describing the architectures in detail here, we refer the reader to corresponding papers cited above. We also note that the Jack the Reader configuration files provide succinct descriptions of the architectures, which are useful for understanding their construction.

To improve performance both in terms of reducing training time and to reduce the amount of additional text the model training has to cope with, we applied a windowing scheme. This is because longer text is normally associated with greater issues when dealing with sequence models. Our scheme takes a portion of the text around the answer location chosen from the Wikipedia content. We now describe the following parameters for each architecture.

FastQA All text is embedded using pre-trained GloVe word embeddings [17] (6 billion tokens, and 50 dimensions). We train for 10 epochs using a batch size of 20. We constrain answers to be a maximum of 10 tokens and use a window size of 1000 characters. The answer layer is configured to be bilinear. We use the ADAM optimizer with a learning rate of 0.11 and decay of 1.0.

JackQA Here we embed the text using pre-trained GloVe word embeddings (840 billion tokens and 300 dimensions). We use the default JackQA settings. We use a window size of 3000 characters. The batch sizes were 128/96/64 for three iterative runs. The subsequent runs with smaller batch sizes were only run if the prior iteration failed. We specified a maximum number of 20 epochs.

Baseline In addition to the models based on neural networks, we also implemented a baseline. The baseline consisted of finding the closest noun phrase to the property within the Wikipedia page and checking whether the answer is contained within that noun phrase.

Note, we attempted to find functional settings that worked within our available computational constraints. For example, FastQA requires more resources than JackQA in relation to batch size , thus, we chose to use smaller embeddings and window size in order to maintain a "good" batch size.

We use 2/3 of the training data for model building and 1/3 for testing. Data is divided randomly. Training was performed using an Amazon EC2 p2.xlarge[5] box. It took 23 hours for training of FastQA models, which included all models for all predicates even when there were too few training samples. For JackQA, the window was increased to 3000 characters, and multiple training sessions were required, reducing the batch size each time to complete the models which not

[5] 1 virtual GPU - NVIDIA K80, 4 virtual CPUs, 61 GiB RAM

finish from earlier runs, in all three passes were required with 128, 96 and 64 batch size respectively. Total training time was 81 hours.

Note that we train models for the setting where the subject and predicate are bound but the object is not. We also use the FastQA architecture to build models for the setting where the subject is treated as the variable to be bound.

5 Experimental Results and Analysis

Table 1 one reports the average F1 measure across all models as well as the baseline. This measure takes into account the overlap of the identified set of tokens with the gold standard answer controlling for the length of the extracted token. By definition, the baseline only generates such overlap scores.

Model	Model Count	mean	std	min	max	25%	50%	75%
JackQA - SP	572	0.70	0.24	0.0	1.0	0.54	0.77	0.89
FastQA - SP	572	0.62	0.24	0.0	1.0	0.43	0.65	0.80
FastQA - PO	717	0.89	0.10	0.4	1.0	0.85	0.92	0.96
Baseline	407	0.15	0.17	0.0	0.86	0.03	0.08	0.20

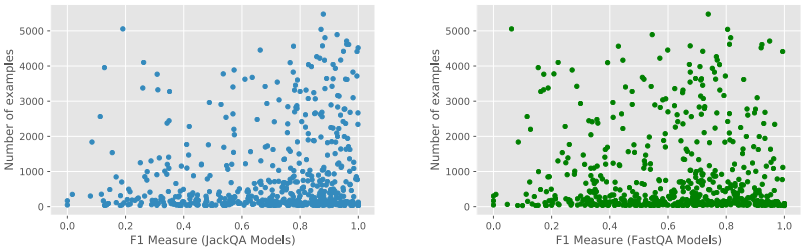
Table 1: F1 results across all models and the baseline

Table 2 reports the average exact match score over all models. This score measures whether the model extracts the exact same string as in the gold standard. For reference, both tables also reports the total number of models trained (Model Count), which is equivalent to the training data provided. The Model Count for the baseline is equivalent to the number of predicates for which the baseline method could find an answer for.

Model	Model Count	mean	std	min	max	25%	50%	75%
JackQA - SP	572	0.64	0.26	0.0	1.0	0.44	0.71	0.86
FastQA - SP	572	0.55	0.25	0.0	1.0	0.36	0.57	0.74
FastQA - PO	717	0.83	0.14	0.1	1.0	0.75	0.86	0.94

Table 2: Exact results

Figure 3 plots individual model performance against the size of the training data given. Overall, models based on deep learning notably outperform the baseline models on average. Additionally, using these deep learning based approaches we are able to create models that answer queries for 160 additional properties over the baseline. In terms of analysis, first, we wanted to see if there was a correlation between the amount of training data and the performance of a model. Using the data presented in Figure 3, we fit a linear regression to it. We found no statistically significant correlation ($R^2 = 0.37$). The model architectures show



Models trained using Jack QA Models trained using Fast QA.

Fig. 3: Plot of individual model performance vs. training data size. All 572 models are shown for the SP setting.

strong correlation in performance. The R^2 value being 0.97 in the case of the F1 measure and 0.96 for the Exact measure. This suggests that the performance is primarily a factor of the underlying kind of data. More details are provided in Appendix A

6 Prototype

To understand whether this approach is feasible in practice, we implemented a prototype of the system outlined in Figure 1. For the triple pattern fragment facade we modify Piccolo, an open source triple pattern fragments server to replace its in-memory based system with functions for calling out to our QA answering component. The facade also implements a simple lexicalization routine. The query answering component is implemented as a Python service and calls out to an Elasticsearch search index where documents are stored. The query answering component also pre-loads the models and runs each model across candidate documents retrieved by querying elastic search. We also specify a max number of candidate documents to run the models over. Currently, we execute each model sequentially over all candidate documents. We then chose the top set of ranked answers given the score produced by the model. Note that we can return multiple bindings for the same ranked results. We made some preliminary timing estimates of a query. It takes on the order of 10 seconds to provide results for a single triple pattern query. This is surprisingly good given the fact that we execute models sequentially instead of in parallel. Furthermore, we execute the models over the entirety of the Wikipedia article. Our own anecdotal experience shows that question answering models are both faster and produce more accurate results when supplied with smaller amounts of text. Thus, there is significant room for optimizing query performance with some simple approaches including parallelizing models, chunking text into smaller blocks, and limiting the number of models executed to those that are specific for the triple pattern. Furthermore, it is straightforward to issue triple pattern fragment query requests

over multiple running instances [25]. One could also implement more complex sharding mechanisms designed for triple tables [1]. Overall, the prototype gives us confidence that this sort of system could be implemented practically.[6]

7 Related Work

Our work builds upon and connects to a number of existing bodies of literature. The work on information extraction is closely related. [15] provides a recent survey of the literature in this area specifically targeted to the the problems of extracting and linking of entities, concepts and relations. One can view the models that we build as similar to distantly supervised relation extraction approaches [16, 23], where two mentions of entities are found in text and the context around those mentions is used to learn evidence for that relation. Recent approaches have extended the notion of context [18] and applied neural networks to extract relations [30, 6].

The closest work to ours in the information extraction space is [14] where they apply machine comprehension techniques to extract relations. Specifically, they translate relations into templated questions - a process they term querification. For example, for the relation spouse(x,y) they created a series of corresponding question templates such as "Who is x married to?". These templates are constructed using crowdsourcing, where the workers are provided a relation, example sentence and asked to produce a question template. This dataset is used to train a BiDAF-based model [21] and similar to our approach they address slot filling queries where the aim is to populate one side of the relation. While we apply a similar technique, our approach differs in a two key aspects. First, we target a different task, namely, answering structured queries. Second, we do not generate questions through question templates but instead build the questions out of the knowledge base itself.

Like much of the work in this space our approach is based on a large scale parallel corpus. Of particular relevance to our task are the WikiSQL and WikiReading corpora. WikiSQL [31] provides a parallel corpus that binds SQL queries to a natural language representation. The task the dataset is used for is to answer natural language questions over SQL unlike ours which is to answer SQL-like queries over text. SQLWikiReading [8] like our approach extracts a corpus from Wikidata and Wikipedia in order to predict the value of particular properties. Another corpus of note is ComplexWebQuestions [24], which pairs complex SPARQL queries with natural language queries. Importantly, it looks at the compositionality of queries from smaller units. Like WikiSQL, it looks at answering natural language queries over databases. In general, we think our approach in also specifying an extraction procedure is a helpful addition for applying corpus construction in different domains.

As mentioned in the introduction, text databases, where information extraction is combined with databases are also relevant. Our system architecture was

[6] We also integrated the prototype with Slack.

inspired by the pioneering work of [10]. In that work, a search index is used to first locate potential documents and then information extraction techniques are applied to the selected documents to populate a database. Our approach differs in two key aspects. First, instead of populating a database our system substitutes the indexes of the database with models. Second, we use distributed query techniques in order to process complex queries on the client side. Recent work [12] uses deep learning based approaches to perform information extraction during database query execution specifically for entity disambiguation. Similar to other work in this area, and unlike ours, they integrate the information extraction within the database engine itself.

Finally, there is a long history of mixing information retrieval and database style queries together. For example, for the purposes of querying over semistructured data [2]. [19] provides an accessible introduction to that history. While our system is designed to answer database queries one can imagine easily extending to the semistructured setting.

8 Conclusion & Future Work

In this work, we have explored the notion of answering database queries over text absent the need for a traditional database intermediary. We have shown that this approach is feasible in practice by combining machine comprehension based models with distributed query techniques.

There are a number of avenues for future work. In the short term, the developed models could be expanded to include extracting properties as well as subjects and objects. We also think that joint models for all triple pattern predictions is worth exploring. One would also want to extend the supported queries to consider not only relationships between entities but also to the attributes of entities. Our current lexicalization approach is also quite simple and could be improved by considering it as the inverse of the entity linking problem and applying those techniques or applying summarization approaches [26]. In this work, we used model architectures that are designed for answering verbalized questions and not database queries. Modifying these architectures may also be a direction to obtain even better performance. Obviously more extensive experimental evaluations would be of interest, in particular, extending the approach to other knowledge bases and looking more deeply at query result quality.

In the long term, the ability to query over all types of data whether images, structured data or text has proven useful for knowledge bases [29]. Extending our concept to deal with these other datatypes could be powerful -making it easy to perform structured queries over unstructured data while minimizing information extraction overhead. In general, we believe that structured queries will continue to be a useful mechanism for data professionals to both work with data and integrate information into existing data pipelines. Hence, focusing on automated knowledge base construction from the query vantage point is an important perspective.

References

1. Abdelaziz, I., Harbi, R., Khayyat, Z., Kalnis, P.: A survey and experimental comparison of distributed sparql engines for very large rdf data. Proceedings of the VLDB Endowment **10**(13), 2049–2060 (2017)
2. Abiteboul, S.: Querying semi-structured data. In: International Conference on Database Theory. pp. 1–18. Springer (1997)
3. Alagiannis, I., Borovica, R., Branco, M., Idreos, S., Ailamaki, A.: Nodb: efficient query execution on raw data files. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 241–252. ACM (2012)
4. Cafarella, M.J., Re, C., Suciu, D., Etzioni, O., Banko, M.: Structured querying of web text. In: 3rd Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA (2007)
5. Dirk Weissenborn, Pasquale Minervini, T.D.I.A.J.W.T.R.M.B.J.M.T.D.P.S.S.R.: Jack the Reader A Machine Reading Framework. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) System Demonstrations (July 2018), https://arxiv.org/abs/1806.08727
6. Glass, M., Gliozzo, A., Hassanzadeh, O., Mihindukulasooriya, N., Rossiello, G.: Inducing implicit relations from text using distantly supervised deep nets. In: International Semantic Web Conference. pp. 38–55. Springer (2018)
7. Harris, S., Seaborne, A., Prudhommeaux, E.: Sparql 1.1 query language. W3C recommendation **21**(10) (2013)
8. Hewlett, D., Lacoste, A., Jones, L., Polosukhin, I., Fandrianto, A., Han, J., Kelcey, M., Berthelot, D.: WIKIREADING: A novel large-scale language understanding task over Wikipedia. In: Proceedings of the The 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016) (2016)
9. Hoffmann, R., Zettlemoyer, L., Weld, D.S.: Extreme extraction: Only one hour per relation. arXiv preprint arXiv:1506.06418 (2015)
10. Jain, A., Doan, A., Gravano, L.: Sql queries over unstructured text databases. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. pp. 1255–1257. IEEE (2007)
11. Kilias, T., Löser, A., Andritsos, P.: Indrex: In-database relation extraction. Information Systems **53**, 124–144 (2015)
12. Kilias, T., Löser, A., Gers, F.A., Koopmanschap, R., Zhang, Y., Kersten, M.: Idel: In-database entity linking with neural embeddings. arXiv preprint arXiv:1803.04884 (2018)
13. Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., Socher, R.: Ask me anything: Dynamic memory networks for natural language processing. In: International Conference on Machine Learning. pp. 1378–1387 (2016)
14. Levy, O., Seo, M., Choi, E., Zettlemoyer, L.: Zero-shot relation extraction via reading comprehension. In: Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). pp. 333–342 (2017)
15. Martinez-Rodriguez, J., Hogan, A., Lopez-Arevalo, I.: Information extraction meets the semantic web: A survey. Semantic Web Journal (2018)
16. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2. pp. 1003–1011. Association for Computational Linguistics (2009)

17. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162
18. Quirk, C., Poon, H.: Distant supervision for relation extraction beyond the sentence boundary. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. vol. 1, pp. 1171–1182 (2017)
19. Raghavan, S., Garcia-Molina, H.: Integrating diverse information management systems: A brief survey. Bulletin of the Technical Committee on Data Engineering p. 44 (2001)
20. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 2383–2392 (2016)
21. Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603 (2016)
22. Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C., Ré, C.: Incremental knowledge base construction using deepdive. Proceedings of the VLDB Endowment **8**(11), 1310–1321 (2015)
23. Surdeanu, M., Tibshirani, J., Nallapati, R., Manning, C.D.: Multi-instance multi-label learning for relation extraction. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. pp. 455–465. Association for Computational Linguistics (2012)
24. Talmor, A., Berant, J.: The web as a knowledge-base for answering complex questions. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). vol. 1, pp. 641–651 (2018)
25. Verborgh, R., Sande, M.V., Hartig, O., Herwegen, J.V., Vocht, L.D., Meester, B.D., Haesendonck, G., Colpaert, P.: Triple pattern fragments: A low-cost knowledge graph interface for the web. Web Semantics: Science, Services and Agents on the World Wide Web **37-38**, 184 – 206 (2016). https://doi.org/https://doi.org/10.1016/j.websem.2016.03.003, http://www.sciencedirect.com/science/article/pii/S1570826816000214
26. Vougiouklis, P., Elsahar, H., Kaffee, L.A., Gravier, C., Laforest, F., Hare, J., Simperl, E.: Neural wikipedian: Generating textual summaries from knowledge base triples. Journal of Web Semantics (2018). https://doi.org/https://doi.org/10.1016/j.websem.2018.07.002, http://www.sciencedirect.com/science/article/pii/S1570826818300313
27. Vrandečić, D.: Wikidata: A new platform for collaborative data collection. In: Proceedings of the 21st International Conference on World Wide Web. pp. 1063–1064. ACM (2012)
28. Weissenborn, D., Wiese, G., Seiffe, L.: Making neural qa as simple as possible but not simpler. In: Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). pp. 271–280 (2017)
29. Wu, S., Hsiao, L., Cheng, X., Hancock, B., Rekatsinas, T., Levis, P., Ré, C.: Fonduer: Knowledge base construction from richly formatted data. In: Proceedings of the 2018 International Conference on Management of Data. pp. 1301–1316. ACM (2018)
30. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J.: Relation classification via convolutional deep neural network. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 2335–2344 (2014)

31. Zhong, V., Xiong, C., Socher, R.: Seq2sql: Generating structured queries from natural language using reinforcement learning. CoRR **abs/1709.00103** (2017)

A Individual Model and Error Analysis

We looked more deeply at performance for individual models for a given property. Table 3 shows the highest performing models. We find some consistent patterns. First, properties that have specific value constraints within Wikidata generate good results. For example, the "crystal system" property needs to have one of 10 values (e.g cubic crystal system, quasicrystal, amorphous solid). Likewise, the "coolant" property needs to be assigned one of fourteen different values (e.g. water, oil, air). This is also true of "discovery method", which oddly enough is actually defined as the the method by which an exoplanet is discovered. This is also a feature of properties whose values come from classification systems (e.g. "Kppen climate classification" and "military casualty classification").

A second feature that seems to generate high performing models are those that refer to common simple words. For example, the "source of energy" property takes values such as "wind" or "human energy".

Lastly, simple syntactic patterns seem to be learned well. For example, the property "birthday", which links to entities describing a month, day combination (e.g. November 8) which is thus restricted to a something that looks like a month string followed by one or two numerical characters. Likewise, the expected value for the property "flag" often appears directly in text itself. That is the correct answer for the query "Japan flag" is "flag of Japan", which will appear directly in text.

Property	Fast QA F1	Fast QA Exact	Jack QA F1	Jack QA Exact	Training Data Size
birthday	0.95	0.91	1.0	1.0	32
flag	0.98	0.88	1.0	1.0	50
league points system	1.00	1.00	1.0	1.0	90
discovery method	0.98	0.91	1.0	1.0	69
source of energy	0.94	0.94	1.0	1.0	50
military casualty classification	1.00	1.00	1.0	1.0	92
topic's main category	0.99	0.91	1.0	1.0	31
Kppen climate classification	1.00	1.00	1.0	1.0	34
coolant	0.98	0.98	1.0	1.0	128
crystal system	0.96	0.87	1.0	1.0	43

Table 3: Highest 10 performing models in the SP setting as determined by F1 measures from models trained using the Jack QA architecture.

We also look at the lowest performing models, shown in Table 4 to see what is difficult to learn. Ratings for films (e.g. Australian Classification, RTC film rating, EIRIN film rating) seem extremely difficult to learn. Each of these properties

expect values of two or three letters (e.g. PG, R15+, M). The property "blood type" also has the same form. It seem that using character level embeddings may worked better in these cases.

The property "contains administrative territorial entity " is an interesting case as there are numerous examples. This property is used within Wikidata to express the containment relation in geography. For example, that county contains a village or a country contains a city. We conjecture that this might be difficult to learn because the sheer variety of linkages that this can express making it difficult to find consistencies in the space. A similar issue could be present for properties such as "voice actor" and "cast member" where the values can be essentially any person entity. Similarly, "polymer of" and "species kept" both can take values that come from very large sets (e.g. all chemical compounds and all species). It might be useful for the model to be provided specific hints about types (i.e. actors, chemicals, locations) that may allow it to find indicative features.

Property	Fast QA F1	Fast QA Exact	Jack QA F1	Jack QA Exact	Training Data Size
Australian Classification	0.00	0.00	0.00	0.00	48
RTC film rating	0.00	0.00	0.00	0.00	167
EIRIN film rating	0.01	0.01	0.02	0.02	349
blood type	0.00	0.00	0.08	0.08	302
contains administrative territorial entity	0.09	0.06	0.08	0.07	1838
voice actor	0.12	0.11	0.11	0.09	2562
species kept	0.11	0.03	0.12	0.03	354
best sprinter classification	0.19	0.18	0.12	0.11	165
cast member	0.15	0.14	0.13	0.11	3955
polymer of	0.18	0.08	0.13	0.08	38

Table 4: Lowest 10 performing models in the SP setting as determined by F1 measures from models trained using the Jack QA architecture.

Can Knowledge Graphs and Deep Learning Approaches help in Representing, Detecting and Interpreting Metaphors?

Mehwish Alam

FIZ Karlsruhe - Leibniz Institute for Information Infrastructure, AIFB Institute, KIT, Karlsruhe, Germany

Abstract. This paper gives an introduction to Conceptual Metaphor Theory (CMT) as introduced by George Lakoff and discusses the possible research problems that can open in the context of Knowledge Graphs and Deep Learning Methods and Metaphors in different mediums.

1 Proposal

Typically when a human mind thinks of a metaphor, the mind tries to map one concept to the another concept based on their properties or functionality etc. In Conceptual Metaphor Theory (CMT) [9], George Lakoff discusses that in the presence of a metaphor there are cross-domain mappings, i.e., a mapping between a source domain and a target domain. For example, in

Corruption is infecting our society.

, the source domain is infection (i.e., a `Disease`) which is mapped to the target domain *Corruption* (i.e., a `Criminal Activity`).

A published resource is available on-line called MetaNet [1], which defines a list of such metaphors and each metaphor consists of a source and a target domain. In case of the above example, it evokes the metaphor `Crime is a disease`. Each of these domains are represented as a linguistic frame called as source frame and target frame respectively. These frames resembles the frames as introduced in FrameNet [2], however, there are only few exact matches between the frames in both the resources, meaning that MetaNet contains its own specific frames. For the running example, the source frame is a `Disease` and the target frame is a `Criminal Activity`, where each of the roles of source frame i.e., `disease` and `patient` map to the roles in the target frame `criminal activtiy` and `victim` respectively.

Can Knowledge Graphs Capture such kind of Semantics. While thinking in terms of Knowledge Graphs, can this information about cross-domain mapping be represented in the form of a Knowledge Graph? Amnestic Forgery [5, 6] is one of the attempts to integrate the metaphors from MetaNet to the existing linguistic linked data cloud based on Frame Semantics, Framester [4]. In

this resource each metaphor is represented following the theory of Description & Situation (D&S) [7]. According to this, a metaphor is a description and its occurrence in the text is a situation. One of the drawbacks of this resource is that it keeps very general metaphors. There is a need to find a middle ground between the cross-domain mappings as represented by frames and mappings occurring in the text. In order to find such kind of mappings we need to process the textual resources rich in metaphors such as poems or corpora specifically created for metaphors.

One of the solutions is to use previously designed deep learning methods [8] for distinguishing between metaphoric and literal expressions. Then finally learning from these metaphoric expressions their specific domains and enrich the Knowledge Graph with this kind of information. Another solution would be to create such kind of mappings in the existing Knowledge Graphs such as DBpedia which contain those domains and are represented based on their literal meanings but are not connected to the other domains based on their possible metaphoric relation. This can help in better Identification/interpretation of metaphors or generation of new metaphors.

Metaphors in Different Mediums Metaphors not only occur in language but they also occur in different mediums such as visual metaphors (occurring in images which can be related to political comics, advertisement or art work). A metaphor can also be expressed in multiple mediums such as text with image or gestures which can be found in videos. The last kind of metaphors are referred to as multi-modal metaphors [3]. Tensors can help in dealing with multi-dimensionality in such kind of metaphors. Following these lines many other tasks come into play such as: (i) Metaphor identification along with their interpretation by combining the information present in different mediums, (ii) Capturing/Modeling cultural biases, meaning that the metaphor is interpreted differently based on cultural background.

References

1. E. Dodge, J. Hong, and E. Stickles. Metanet: Deep semantic automatic metaphor analysis. In *Proceedings of the Third Workshop on Metaphor in NLP*. Association for Computational Linguistics, 2015.
2. C. J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1), 1976.
3. C. J. Forceville and Eduardo Urios-Aparisi. *Multimodal Metaphor*. De Gruyter Mouton, Berlin, Boston, 2009.
4. A. Gangemi, M. Alam, L. Asprino, V. Presutti, and D. R. Recupero. Framester: A wide coverage linguistic linked data hub. In *EKAW*, 2016.
5. A. Gangemi, M. Alam, and V. Presutti. Amnestic forgery: An ontology of conceptual metaphors. In *FOIS*, 2018.
6. A. Gangemi, M. Alam, and V. Presutti. Linked metaphors. In *In: ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks with (ISWC 2018)*, 2018.

7. A. Gangemi and P. Mika. Understanding the semantic web through descriptions and situations. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE*, 2003.
8. G. Gao, E. Choi, Y. Choi, and L. Zettlemoyer. Neural metaphor detection in context. In *EMNLP*, 2018.
9. G. Lakoff and M. Johnson. *Metaphors we Live by.* University of Chicago Press, Chicago, 1980.