Pretrained Knowledge Base Embeddings for improved Sentential Relation Extraction

Andrea Papaluca[1], **Daniel Krefl**[2], **Hanna Suominen**[1,3], **Artem Lenskiy**[1]
[1] School of Computing, The Australian National University, Canberra, ACT, Australia
[2] Department of Computational Biology, University of Lausanne, Switzerland
[3] Department of Computing, University of Turku, Turku, Finland
{andrea.papaluca, hanna.suominen, artem.lenskiy}@anu.edu.au,
daniel.krefl@unil.ch

Abstract

In this work we put forward to combine pre-trained knowledge base graph embeddings with transformer based language models to improve performance on the sentential Relation Extraction task in natural language processing. Our proposed model is based on a simple variation of existing models to incorporate *off-task* pre-trained graph embeddings with an *on-task* finetuned BERT encoder. We perform a detailed statistical evaluation of the model on standard datasets. We provide evidence that the added graph embeddings improve the performance, making such a simple approach competitive with the state-of-the-art models that perform explicit on-task training of the graph embeddings. Furthermore, we observe for the underlying BERT model an interesting power-law scaling behavior between the variance of the F1 score obtained for a relation class and its support in terms of training examples.

1 Introduction

Besides large quantities of unstructured textual data, also structured data has become widely available to machine learning researchers in recent years. Knowledge Bases (KBs), such as Wikidata (Vrandečić, 2012) (formerly Freebase Bollacker et al., 2007; Pellissier Tanon et al., 2016), Yago (Suchanek et al., 2007) and UMLS (Bodenreider, 2004), organise various kinds of information in structured form and constantly grow in size and richness of included information.

They are represented in terms of relations between entities, forming a graph structure that makes retrieval and processing of the included information easier and finds particular application in various language related tasks, such as question answering (QA), search engine development and knowledge discovery. Distant supervision (Mintz et al., 2009) is another notable example that employs a KB to improve Relation Extraction (RE). For each pair of entities found in a sentence, distantly supervised models check whether a link between the entities in the KB graph exists, and, if there is a match, the sentence is then used as a training example for supervised learning.

Note that the utility of KBs for RE extends beyond being just a useful source of supervision labels. A natural question arises whether one can develop models which combine unstructured textual data with structured information to further improve performance, for general natural language processing tasks.

One particular class of approaches that has been gaining momentum is based on the idea of dynamically learning representations of KB entities simultaneously with word representations (Bastos et al., 2021; Nadgeri et al., 2021; Vashishth et al., 2018). The motivation behind this class of methods is whether such combined representations could improve performance for a downstream NLP task, due to a more representative embedding.

However, although in theory this could result in optimally finetuned word and graph representations for the downstream task, it might be challenging in practice. On-task training of the graph embeddings requires significantly more complex models, and therefore increases the training cost and is more prone to overfitting.

Therefore, instead of training both, graph and word embeddings, we investigate in this work whether combining static pre-trained graph embeddings, such as those provided in Lerer et al. (2019), with *on-task* learned word embeddings already achieves a significant performance boost for downstream tasks. The underlying question being whether a neural model is able to transfer the topological information contained in the pre-trained graph embeddings in a useful manner to the task at hand.

2 Related Work

In the following, we would like to briefly review three particular works in the literature that make use of the information contained in a KB to improve classification performance on the RE task, and explain how our work relates to these previous works.

In Vashishth et al. (2018) the authors propose to use KBs as a supplementary source of information to improve on the multi-instance learning paradigm for RE. Note that in multi-instance RE one aims at identifying the relation between two targeted entities, for a given bag of sentences. In particular, the authors of Vashishth et al. (2018) match the relation predicted by the Stanford OpenIE (Angeli et al., 2015) pipeline with the set of relation aliases found in the KB. Out of this they obtain a *matched relation embedding*, h^{rel}, that is then concatenated to the sentence representation. Similarly, they build an *entity type embedding*, h^{type}, using the entity type found in the KB, which is concatenated as well to the sentence encoding.

In Bastos et al. (2021) the authors make use of the KB information to improve on the sentential RE task. They propose to construct an *Entity Attribute Context* embedding, h^o, by processing several entity properties found in the KB with the help of a BiLSTM (Schuster and Paliwal, 1997) encoder. Additionally, a *triplet context* embedding, h^r, is learned for each relation triplet by imposing the translational property in the embedding space (Bordes et al., 2013) to the triplet and its KB neighbours. Finally, the two different representations obtained are aggregated with the sentence encodings by a GP-GNN (Zhu et al., 2019) and fed to a classifier for relation prediction.

The authors of Nadgeri et al. (2021), however, suggest that statically adding all the available KB information might be counterproductive in some case. They rather propose to dynamically select the useful information. To do so, for each entity they extract and encode several KB properties with a BiLSTM (Schuster and Paliwal, 1997), that are then combined together with the sentence encoding to form a *Heterogenous Information Graph*. The relevant context information is then obtained by pruning this graph with the help of a combination of graph convolutional neural network (Kipf and Welling, 2017), pooling and self-attention layers, and finally aggregated and fed to the relation classifier, similarly to what is done in Bastos et al. (2021).

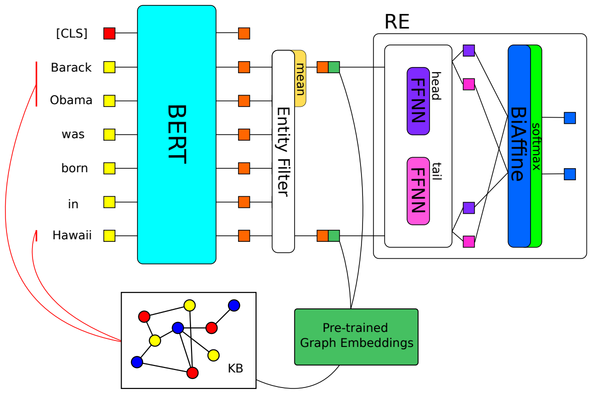

Figure 1: The RE model we make us of, inspired by Giorgi et al. (2019). The initial encodings of the sentences are provided by a pre-trained language model. The encodings corresponding to named entities are extracted, and averaged in case of multi-token entities. For the graph embedding augmented model the graph embeddings are in addition concatenated to the relative entity encodings. Each entity is then decomposed in a *head-tail* representation to form the (h, t) candidate pairs passed to the Biaffine Attention Layer (Dozat and Manning, 2017) for relation classification.

Note that the KB embedding is dynamically learned in all above cited works. Furthermore, in these examples, it derives from a wide variety of entity properties and information retrieved from the KB, but not directly from the graph structure itself. The only exception being the *triplet-context* embedding h^r of Bastos et al. (2021), which comes closest to the graph embedding (Lerer et al., 2019) we are going to make use of. However, their embedding is trained only on the downstream task, but not on the full KB.

Therefore, what is so far missing in the literature, is the basic baseline of simply adding a pre-computed *topology-based* KB embedding in order to improve the RE task performance. In this work we aim to fill this gap. We provide a detailed evaluation of such a model, which takes pre-computed KB embeddings into account. As we will show below, such a simple model extension is in fact competitive with the more advanced models mentioned above on standard benchmark datasets.

3 Model

To perform RE, we make use of the model introduced in Nguyen and Verspoor (2019); Giorgi et al. (2019). The only difference being that we do not perform Named Entity Recognition (NER), but instead train the model using gold entities, i.e. the correct span of the entity mentions is fed to the model as input. Thereby, we are able to evaluate RE performance without contamination with wrongly predicted entities. We chose this model for its relative simplicity, while still providing close to state-of-the-art performance in RE and its high modularity that allows for easy modifications and extensions.

The RE classifier we use resembles the one introduced in Nguyen and Verspoor (2019), except that it is combined with a pre-trained transformer-based (Devlin et al., 2019; Vaswani et al., 2017) encoder, as proposed in Giorgi et al. (2019). For illustration, the complete model is sketched in Figure 1.

The information flows from left to right and the errors are free to backpropagate through all the preceding layers. The input of the model is a sentence of N tokens, $w_1, w_2, ..., w_N$, containing two or more known entities, $e_i = w_j, w_{j+1}, ..., w_{j+k}$. The output is one or more triplets (h, t, r) representing the relations found in the input sentence. h and t represent the head and the tail of the relation respectively, while r is the type of relationship between h and t.

A more detailed description of our model pipeline is given in the remainder of this section. For reproducibility, we made our model source codes available on Github [1].

3.1 Pre-trained Language Model

The pre-trained language model provides the initial encodings for the tokens contained in the sentence. We opted to use variants of the BERT (Devlin et al., 2019; Liu et al., 2019) model (specifically, the implementation of *bert-base-cased* and *roberta-base* by Huggingface (Wolf et al., 2020)), but in general the encoder can be replaced by any other encoder capable of providing context-dependent embeddings of a sentence. We leave the encoder unfrozen during training such that the gradients can propagate through it. Thereby its parameters are fine-tuned to optimize the token representation for the specific task at hand.

In more detail, we start with the encoder completely frozen and gradually unfreeze the last four layers over the first epochs of training, as suggested for instance in Araci (2019). It has been shown that such training procedure does not necessary yield a decrease in accuracy, but it does significantly reduce the computational burden, c.f., Araci (2019).

[1] https://github.com/BrunoLiegiBastonLiegi/Pretrained-KB-Embeddings-for-RE

3.2 Entity Filter

The purpose of the entity filter is to filter out each token that does not compose an entity. Two common choices for multi-token entities are to keep either the last token or the average of tokens as identifier of the complete entity. We tested both cases and did not find any evidence of one being superior to the other in our application. Therefore, we opted to take the average encoding among the tokens composing the entity as identifier.

Note that for the model augmented by a graph-embedding, we concatenate to the average encoding obtained for each entity, $\mathbf{x}_i^{BERT}$, the relative graph embedding $\mathbf{x}_i^{graph}$ of Lerer et al. (2019), such that we obtain for the final input $\mathbf{x}_i$ of the RE module

$$\mathbf{x}_i = [\mathbf{x}_i^{BERT}, \mathbf{x}_i^{graph}] \; . \qquad (1)$$

3.3 RE Module

For details of the RE module we refer to the original works (Dozat and Manning, 2017; Nguyen and Verspoor, 2019; Giorgi et al., 2019). In a nutshell, the RE module consists of two steps. Firstly, the *head-tail* decompositions of the inputs are constructed:

$$\mathbf{x}_i^{h|t} = \mathcal{F}_{h|t}\left([\mathbf{x}_i^{BERT}, \mathbf{x}_i^{graph}]\right) , \qquad (2)$$

where $\mathbf{x}_i^h$ and $\mathbf{x}_i^t$, are the representation of the inputs viewed as the head or the tail of a relation triplet (h, t, r). The projection is performed by two simple feed forward neural networks, $\mathcal{F}_{head}$ and $\mathcal{F}_{tail}$, composed of two linear layers separated by ReLU activation and dropout ($p = 0.1$) for regularization.

Secondly, all pairs $\{(\mathbf{x}_j^{head}, \mathbf{x}_k^{tail})\}_{j \neq k}$ are constructed by combining all the heads with each possible tail (Miwa and Bansal, 2016; Nguyen and Verspoor, 2019; Giorgi et al., 2019), and fed to a biaffine attention layer (Dozat and Manning, 2017; Nguyen and Verspoor, 2019; Giorgi et al., 2019) for relation classification. The biaffine layer $\mathcal{B}(\cdot, \cdot)$ performs a combination of a bilinear transformation and a linear projection:

$$\mathcal{B}(\mathbf{x}_1, \mathbf{x}_2) := \mathbf{x}_1^\top \mathbf{U} \mathbf{x}_2 + \mathbf{W}(\mathbf{x}_1 \| \mathbf{x}_2) + \mathbf{b} \, , \qquad (3)$$

$$\mathbf{x}_i^{RE} = \mathcal{B}(\mathbf{x}_j^{head}, \mathbf{x}_k^{tail}) \, , \qquad (4)$$

where we indicate with $\|$ the column-wise concatenation. A final softmax activation layer provides the scores for each of the relation classes. The RE loss is taken to be the crossentropy,

$$\mathcal{L}_{RE} = \sum_{i=1}^{M} \frac{\exp \mathbf{x}_{i,T}^{RE}}{\sum_{j \neq T} \exp \mathbf{x}_{i,j}^{RE}} , \quad (5)$$

where $\mathbf{x}_{i,T}^{RE}$ represents the score assigned to the correct class.

Note that if a sentence contains n_e entities, the RE module is going to provide $2\binom{n_e}{2} = \frac{n_e!}{(n_e-2)!}$ relation predictions. One for each of the possible entity combinations

$$(e_i^{head}, e_j^{tail})_{i \neq j} \quad i, j = 1, ..., n_e$$

and, therefore, allowing for multiple relations extracted from a single sentence.

4 Results

In order to quantify the benefit of adding pre-trained graph embeddings to a RE model, we have considered two popular RE datasets: the *Wikidata* (Sorokin and Gurevych, 2017) and *NYT* (Riedel et al., 2010) corpora.

Since we make use of pre-trained graph embeddings, we decided to discard all sentences in the training set containing entities not present in the graph embedding (Lerer et al., 2019). Note that in order to be able to fairly compare the models with and without graph embeddings, we also exclude these sentences in training the basic model without graph embeddings. For the test set, we keep all the sentences regardless of the available embeddings. In case no embedding for the entity is available, we simply substitute the graph embedding with a zero tensor.

For each dataset we train the model with a different random initialization ten times with and without the addition of the pre-trained graph embeddings. Hence, in total we train twenty models per experiment. Note that we always use the *AdamW* (Loshchilov and Hutter, 2019) optimizer with learning rate $2 \cdot 10^{-5}$. For the implementation of the optimizer and of the whole model depicted in Figure 1 we rely on the Pytorch Library (Paszke et al., 2019).

For evaluation, we compare the performance of the two models (with and without graph embeddings) using Precision, Recall and F1 score. Both, for each single relation class, and on average with micro- and macro-averaging. In detail we rely on the two following evaluation methods:
At first, for each relation class, we collect the F1 score obtained by the ten different trained models and we plot the complete distribution of the results as a violin plot. The mean of the ten F1 values obtained is also reported as a colored dot inside the violin. The same is done for the global micro- and macro-average. We do this for both, the model with added graph embeddings and the baseline model without graph embeddings.

Then, we generate the micro *Precision-Recall* curve for each of the ten models trained. The ten curves obtained are interpolated and averaged to form a single mean PR curve both, for the model provided with the graph embeddings and the baseline model. At each value of recall we compute the standard deviation of the precision over the ten different curves. The deviation is shown in shaded color around the mean curve.

4.1 Pre-Trained Graph Embeddings

For all examples, we rely on the Wikidata KB (Vrandečić, 2012) with pre-trained graph embeddings of the entities provided by Lerer et al. (2019). In detail, the authors of Lerer et al. (2019) propose a memory efficient and distributed implementation of several popular graph embedding methods, such as RESCAL (Nickel et al., 2011), TransE (Bordes et al., 2013) and DistMult (Yang et al., 2015). This new implementation was specifically developed for dealing with very large graphs while being competitive with the original implementation of the state-of-the-art models aforementioned.

The pre-trained embeddings we use were trained on the so-called "truthy" Wikidata dump dated 2019-03-06, making use of the TransE (Bordes et al., 2013) approach with embedding dimension of size 200.

4.1.1 Wikidata

The *Wikidata* (Sorokin and Gurevych, 2017) dataset is a RE corpus built by distant supervision with entities aligned to the Wikidata Knowledge Base (Vrandečić, 2012). We rely on the dataset version provided by Bastos et al. (2021). Some statistics of the dataset are shown together with our training configuration in Table 1. Note that in the evaluation we ignore the results for the NA relation class (i.e. the "no relation" class), as is usually done in the literature (Bastos et al., 2021; Nadgeri

Dataset	train	train$_{ours}$	test	relations	KB entities	batchsize	epochs
Wikidata	372,059	369,577	360,334	353	464,535	8	1-2
NYT	455,771	∼ 453-455,000	172,448	58	63,601	8	3

Table 1: Statistics and training settings for all the datasets considered. **train**$_{ours}$ indicates the actual number of training sentences used after discarding part of the data, as described in the main text. Note that for the NYT dataset the **train**$_{ours}$ is given as a range, as for each repetition we sampled a different subset of the training and validation set, as described in Section 4. For the Wikidata dateset, we experimented with training for 1 and 2 epochs.

			Micro			Macro		
			P	R	F1	P	R	F1
		RECON (Bastos et al., 2021)	87.24	87.23	87.23	**63.59**	33.91	44.23
		KG-Pool (Nadgeri et al., 2021)	**88.60**	**88.59**	**88.60**	-	-	-
Wikidata	Ours	*bert-base*	85.43	78.37	81.74	51.42	37.24	40.02
		roberta-base	85.50	80.30	82.81	49.33	36.77	39.22
		roberta-base$_{1e}$	83.71	83.01	83.35	46.09	34.52	36.38
	Ours$_{ge}$	*bert-base*	88.34	79.19	83.51	55.72	39.62	43.24
		roberta-base	87.66	82.07	84.77	54.42	**41.64**	**44.33**
		roberta-base$_{1e}$	86.83	83.93	85.36	50.88	38.52	40.57
NYT	Ours	*bert-base*	47.13	75.57	57.98	28.35	45.27	33.05
	Ours$_{ge}$	*bert-base*	**51.13**	**76.46**	**61.24**	**31.66**	**47.31**	**36.20**

Table 2: Summary of the P, R and F1 scores (averaged over ten runs) obtained in our experiments for the three datasets considered. We indicate with the subscript *ge* the results obtained by the model with the graph embeddings added. For the *Wikidata* experiment we even specify with the subscript $1e$ the models that have been trained for 1 single epoch instead of 2. If avaliable, we include the results obtained by others on the same dataset, we leave blank (-) otherwise. In particular, for the *NYT* dataset, we are not aware of other works in the literature measuring the performance under the F1 metric.

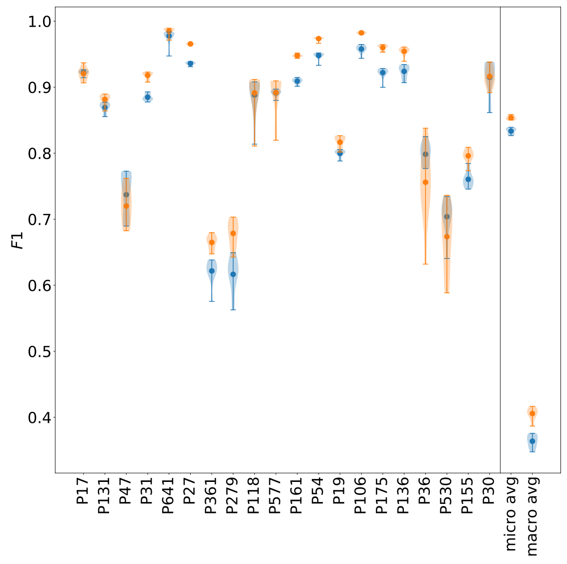

Figure 2: F1 scores on the Wikidata corpus for each relation. The micro- and macro-averages are also shown. Only the top 20 relations are plotted, ordered by decreasing support. The F1 score distribution of the ten different trained models is illustrated via violin plots. The baseline model is shown in blue and the model with added graph embeddings in orange. The means of the distributions are indicated by bullet points inside the violin plots.

	P@10	P@30
RESIDE (Vashishth et al., 2018)	73.6	59.5
RECON (Bastos et al., 2021)	87.5	74.1
KG-Pool (Nadgeri et al., 2021)	92.3	86.7
Ours	96.6	83.1
Ours$_{ge}$	**97.5**	**86.8**

Table 3: Comparison of the P@10 and P@30 for the NYT dataset. We indicate with the subscript *ge* the results obtained by the model with the graph embeddings added.

et al., 2021).

For this dataset, in addition to the *bert-base-cased* (Devlin et al., 2019) model, we also tested the *roberta-base* (Liu et al., 2019) language model. For the latter, we also experimented with training for just one epoch instead of two, obtaining slightly better micro F1, at the cost of lower macro F1.

The F1 score violin plots for the top 20 relation classes in the dataset are shown in Figure 2. The illustrated results were obtained with the *roberta-base* model as sentence encoder, the whole model was trained for 1 epoch. Although some relations do not benefit from the inclusion of graph embeddings, the majority of them show a consistent improvement. On average, we measure a performance boost, both, in micro- and macro-averaged F1 score, of around ~ 2 - 4% depending on the specific language model, c.f. Table 2.

Even so the addition of graph embeddings yields a significant performance boost to the base model, as discussed above, the boosted model does not outperform the current state-of-the-art (Nadgeri et al., 2021) in the micro-averaged metrics. In particular, only the micro precision metric (*bert-base*$_{ge}$: 88.34 ± 0.33), is statistically in range of the state of the art model[2]. However, for macro-averaged scores, our model (*roberta-base*$_{ge}$) surpasses the current state-of-the-art (Bastos et al., 2021), both, in recall (R) and F1.

4.2 NY Times

Another popular RE dataset built by distant supervision is the *NYT* corpus (Riedel et al., 2010), obtained by aligning a set of over 1.8 million articles from the *NY Times* journal to the *Freebase* Knowledge Base. We made use of the *Wikidata SPARQL* query service [3] to obtain the mapping from the old Freebase id scheme to the new Wikidata one. We make use of the dataset version provided by Bastos et al. (2021). In Table 1 are reported some statistics of the dataset together with the settings we used in our experiments.

We train on the validation ($114,317$ sentences) and training sets ($455,771$ sentences) by randomly discarding 20% of the sentences for each one of the ten runs, such that we obtain a number of train sentences comparable to Bastos et al. (2021); Nadgeri et al. (2021); Vashishth et al. (2018). Note that we had to further discard between $1,000$ to $3,000$ train sentences, depending on the run, due to the missing graph embeddings. We train the model with the settings listed in Table 1, and evaluate the performance at P@10 and P@30 (precision at fixed recall 10% and 30%), as proposed in the literature. As before, we ignore the score of the NA relation class in averaging.

Figure 3 illustrates the comparison of the performance of the two models trained. We observe that the model with the added graph embeddings shows a slower decay of the precision with increasing recall and, in particular, an improved precision on average and through the whole curve. The F1 score calculation confirms this trend, as both micro- and macro-F1 exhibit an improvement of about $\sim 3\%$, as reported in Table 2.

[2]For reference, we observed standard deviations in the range ~ 0.1 - 1 for the results reported in Table 2.

[3]https://query.wikidata.org/

The comparison with the results obtained by others in the literature, reported in Table 3, demonstrates the solid performance of our model. In particular, the model is able to surpass the current state-of-the-art (Nadgeri et al., 2021), both, under P@10 and P@30.

4.3 Discussion

To better understand under which circumstances the additional information contained in the graph embeddings is beneficial, some of the results given above are analyzed below. We are going to consider each relation separately, and we will look at four different variables characterizing them:

- $\sigma^2(F1)$: Variance of the F1 score obtained across the ten runs.
- $\overline{F1}$: Mean F1 score obtained across the ten runs.
- S: Support, i.e. the amount of training examples available.
- $\Delta\overline{F1} := \overline{F1}_{ge} - \overline{F1}_b$: Gap between the average scores obtained by the model with the additional graph embeddings and the baseline model.

Note that the *Wikidata* corpus provides the largest number of relation classes and thereby features a wider statistical variety. Therefore, we focus on the results for the *Wikidata* corpus in the remainder of this section, in particular the ones obtained by our *roberta-base* based model trained for 1 epoch. For some relations we have always obtained $\overline{F1} = 0$ with zero variance $\sigma^2(F1) = 0$, and, surprisingly, not all of them had close to zero support S. Those with higher support, however, were usually semantically similar to relations provided with much larger support that were constantly preferred by our model. We exclude all these $\overline{F1} = 0, \sigma^2(F1) = 0$ relations since their identically null variance is an artifact and thus they do not provide a good representation of the $\sigma^2(F1)$ behaviour.

As shown in Figure 2 we observe a large variance of results for several relations across the ten different runs. The relationship between the variance, $\sigma^2(F1)$, and the support S of each relation is plotted in Figure 4 (*left*). Note that we take the log scale for both axis. Both, the baseline, and the graph embedding augmented models, show an approximately linear relationship of $\sigma^2(F1)$ with

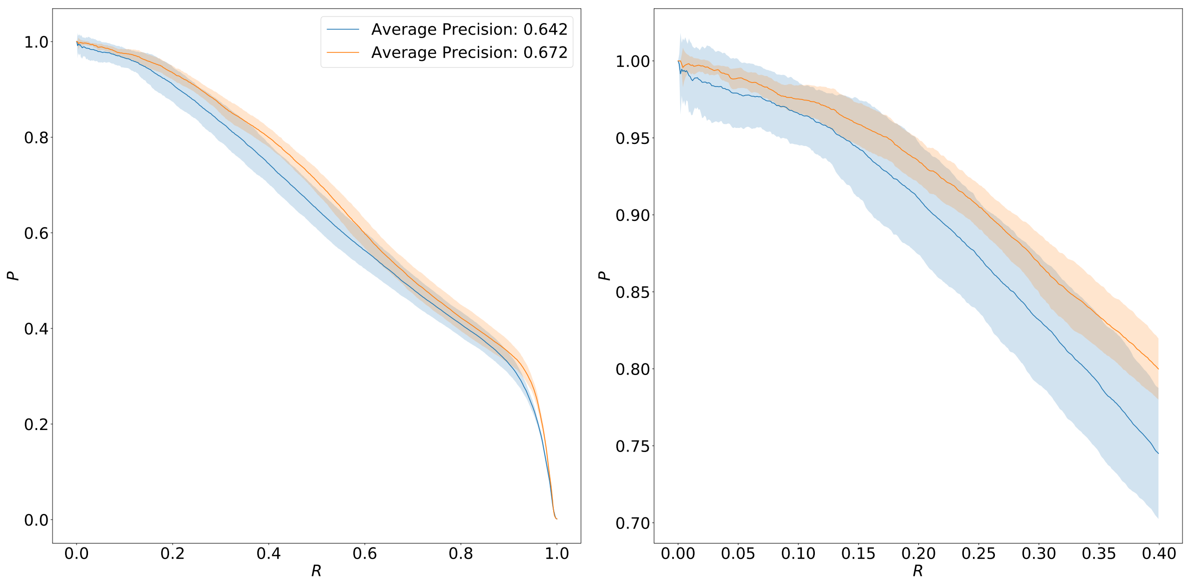

Figure 3: Micro Precision-Recall curves for the baseline model (blue) and the model augmented with graph-embeddings (orange) trained on the *NYT* dataset. The right plot gives a more detailed view into the region with Recall ≤ 0.4. The solid lines represent the average P-R curve for the ten trained models. The shaded regions represent the corresponding standard deviations of the Precision at given Recall.

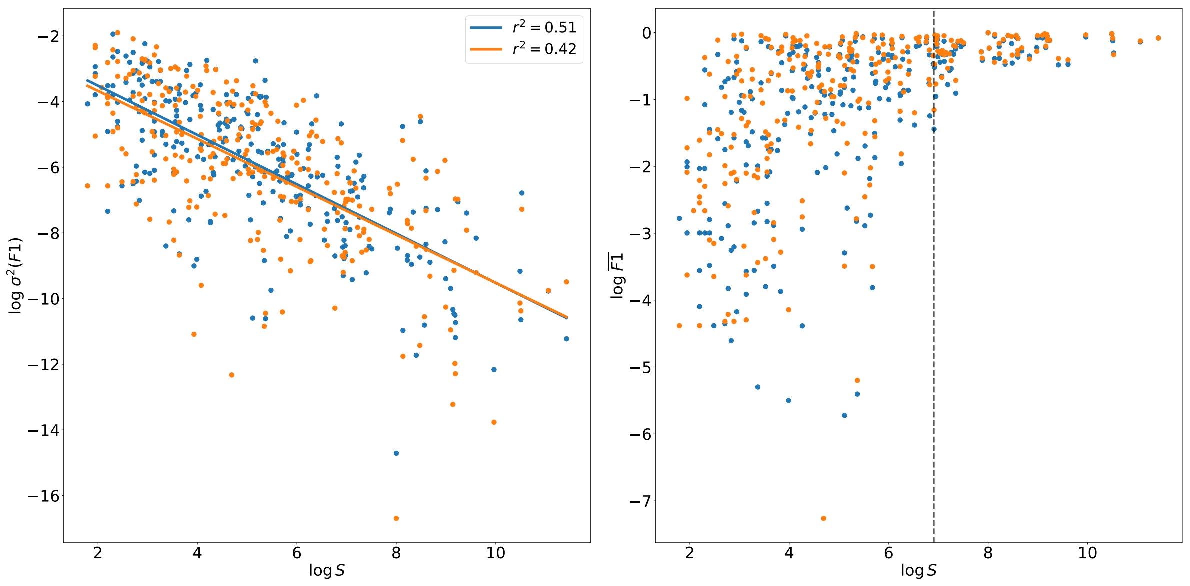

Figure 4: Distribution of the *Wikidata* dataset relations according to their score variance $\sigma^2(F1)$ (*left*) and their average score $\overline{F1}$ (*right*) against relation support S. Note that each point represents a different relation. The color of the points indicates from which model the datapoint originates. Blue for the baseline and orange for the graph embedding augmented model. Both axis are plotted in logarithmic scale. The distributions in the (*left*) plot are fitted with the linear regression (6) with the r^2 value of the fits reported in the legend. The support $S = 1000$ threshold is indicated as a dashed line in the (*right*) plot.

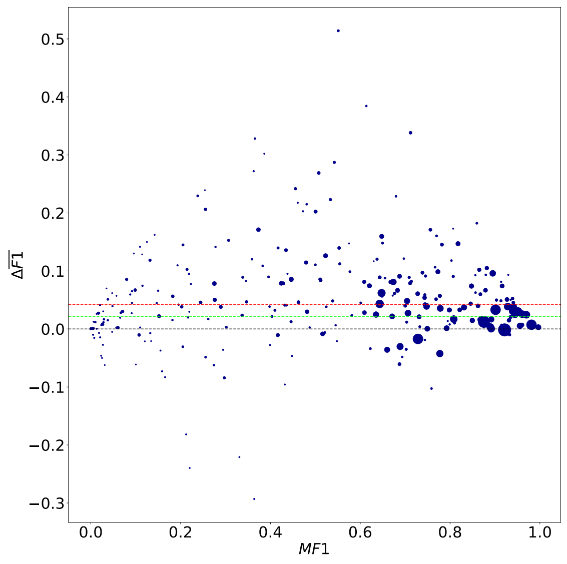

Figure 5: Gap $\Delta\overline{F1}$ for each relation in the *Wikidata* dataset plotted against the average score $MF1$ (8) obtained across the ten runs. The red and green dashed lines indicate the average and the median gap, respectively, whereas the gray dashed line a gap of zero. The size of each dot is given by the squareroot of the support of its corresponding relation.

increasing support, under the $\log - \log$ transformation. We infer the following linear relationships (with and without graph embeddings):[4]

$$\begin{aligned} y &= -0.75x - 2.01\,, \\ y &= -0.73x - 2.21\,, \end{aligned} \tag{6}$$

with standard errors of 0.05 for the slopes and 0.26, respectively, 0.30, for the intercepts (note that we have taken $y = \log \sigma^2(F1)$ and $x = \log S$).

This implies a *power-law* scaling behavior of $\sigma^2(F1)$:

$$\sigma^2(F1) \propto \mathrm{S}^{-0.74}\,, \tag{7}$$

where we took as exponent the mean of the two regression coefficients given above.

In Figure 4 (*right*) the $\overline{F1}$ against the support is plotted using logarithmic scale, as for $\sigma^2(F1)$. However, we do not observe a linear relationship in $\log - \log$ space as before, but rather some non-linear dependence. In particular, it appears that the larger the support of a given relation is, the better the model is able to learn it, as one would naively expect. The plot indicates that a support of at least S $\approx$ 1000 is a sufficient condition for good classification performance with an expected low variance of performance.

The gap $\Delta\overline{F1}$ is plotted against the averaged $\overline{F1}$ score,

$$MF1 = \frac{1}{2}\left(\overline{F1}_{ge} - \overline{F1}_{b}\right), \tag{8}$$

for each relation in Figure 5. Note first that we observe a mean and median gap of $\sim +0.05$, respectively $\sim +0.025$. The performance boost is inline with the micro- and macro- average based observation in the previous section. In the plot also the size of support is indicated for each relation. We clearly observe that relations with large support are clustered at high $MF1$, in accord with the discussion above.

It is interesting to notice that, both, for very high $MF1$ as well as very low $MF1$ the augmentation with graph embeddings only gives mild performance gains with low variance. In contrast, for $MF1 \sim 0.1\text{-}0.9$ we observe a larger variance of $\Delta\overline{F1}$, that leads to gaps in the wider range $\sim -0.3\text{-}0.5$.

5 Conclusion

In both the experiments discussed in Section 4, we measured on average a noticeable improvement over the baseline for the model with included pre-trained graph embeddings. Tables 2 and 3 summarize the numeric outcomes of our experiments, and also include for comparison results of the state-of-the-art methods obtained by others on the same datasets. In particular, our model is able to reach performance close to the current state-of-the-art in the *Wikidata* dataset under the micro-averaged metrics and sets a new state-of-the-art under the macro F1 metric. Similarly, for the *NYT* dataset our model achieves a new state-of-the-art under the P@10 and P@30 metric.

In common with related works in the literature, our model rests on the assumption of having the correct entity identification at hand (gold entities). Identifying entities in a sentence, however, is itself a challenging task, usually referred to as *Named Entity Recognition*. This task is further complicated by the need to map the entities to corresponding nodes in the KB (*Entity Linking*). This currently limits the practical applicability of ours, and models akin to it in the literature, and warrants further research.

We also analyzed in detail the performance of our model for each relation of the *Wikidata* dataset,

[4]The coefficients are rounded off to the first two decimals.

finding an interesting *power-law* scaling of the variance of the F1 score with increasing support of the relation. In particular, this study provided us with an estimate of around $\sim$ 1000 training occurences per relation needed for good prediction performance with small uncertainty. However, further investigation is needed to explore the validity of this finding in more generality.

Therefore, we like to highlight that we were not only able to verify that the inclusion of general pre-trained graph embeddings is helpful for the RE task, but also that such a simple model extension is competitive with other state-of-the-art models that directly perform on-task training of those embeddings. This implies that the inclusion of such pre-trained graph embeddings might be helpful across a wider spectrum of language related tasks to improve performance at a relatively low additional cost of complexity and computational burden.

We see this work as giving further support to the wider adoption of pre-computed graph embeddings in natural language processing tasks. We envisage that their adoption may become comparable to the popular Glove (Pennington et al., 2014) and Word2vec (Mikolov et al., 2013) pre-trained word embeddings.

Acknowledgements

We would like to thank the NCI Australia (National Computational Infrastructure) for providing computing resources, and the anonymous reviewers for their valuable comments and suggestions. AP was supported by an Australian Government Research Training Program International Scholarship. This research was delivered in partnership with, and funded in part by, Our Health in Our Hands (OHIOH), a strategic initiative of the Australian National University, which aims to transform healthcare by developing new personalized health technologies and solutions in collaboration with patients, clinicians, and healthcare providers.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models.

Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Isaiah Onando Mulang', Saeedeh Shekarpour, Johannes Hoffart, and Manohar Kaul. 2021. Recon: Relation extraction using knowledge graph context in a graph neural network.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl_1):D267–D270.

Kurt Bollacker, Robert Cook, and Patrick Tufts. 2007. Freebase: A shared database of structured general human knowledge. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, page 1962–1963.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing.

John Giorgi, Xindi Wang, Nicola Sahar, Won Young Shin, Gary D. Bader, and Bo Wang. 2019. End-to-end named entity recognition and relation extraction using pre-trained language models.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks.

Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. Pytorch-biggraph: A large-scale graph embedding system.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. 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*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Abhishek Nadgeri, Anson Bastos, Kuldeep Singh, Isaiah Onando Mulang', Johannes Hoffart, Saeedeh Shekarpour, and Vijay Saraswat. 2021. Kgpool: Dynamic knowledge graph context selection for relation extraction.

Dat Quoc Nguyen and Karin Verspoor. 2019. End-to-end neural relation extraction using deep biaffine attention. *Advances in Information Retrieval*, page 729–738.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA. Omnipress.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From freebase to wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 1419–1428, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg. Springer Berlin Heidelberg.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1784–1789, Copenhagen, Denmark. Association for Computational Linguistics.

Fabian Suchanek, Gjergji M Kasneci, and Gerhard M Weikum. 2007. Yago: A Core of Semantic KnowledgeUnifying WordNet and Wikipedia. In *16th international conference on World Wide Web*, Proceedings of the 16th international conference on World Wide Web, pages 697 – 697, Banff, Canada.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. RESIDE: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1266, Brussels, Belgium. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, page 1063–1064, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph neural networks with generated parameters for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1331–1339, Florence, Italy. Association for Computational Linguistics.