Applied Intelligence (2023) 53:22415–22428
https://doi.org/10.1007/s10489-023-04685-3

Deep embeddings and Graph Neural Networks: using context to improve domain-independent predictions

Fernando Sola[1] · Daniel Ayala[1] · Inma Hernández[1] · David Ruiz[1]

Accepted: 1 May 2023 / Published online: 28 June 2023

Abstract

Graph neural networks (GNNs) are deep learning architectures that apply graph convolutions through message-passing processes between nodes, represented as embeddings. GNNs have recently become popular because of their ability to obtain a contextual representation of each node taking into account information from its surroundings. However, existing work has focused on the development of GNN architectures, using basic domain-specific information about the nodes to compute embeddings. Meanwhile, in the closely-related area of knowledge graphs, much effort has been put towards developing deep learning techniques to obtain node embeddings that preserve information about relationships and structure without relying on domain-specific data. The potential application of deep embeddings of knowledge graphs in GNNs remains largely unexplored. In this paper, we carry out a number of experiments to answer open research questions about the impact on GNNs performance when combined with deep embeddings. We test 7 different deep embeddings across several attribute prediction tasks in two state-of-art attribute-rich datasets. We conclude that, while there is a significant performance improvement, its magnitude varies heavily depending on the specific task and deep embedding technique considered.

Keywords Knowledge graphs · Graph neural networks · Attributive embeddings · Deep graph embeddings · Machine learning

1 Introduction

Graph Neural Network (GNN) architectures seek to leverage the connections in a graph when it comes to making predictions about the elements of the graph [44]. To achieve this, the nodes of the graph are represented as numeric vectors called embeddings that can capture and summarise the implicit information present in them. For example, graph convolutional layers [27] aggregate the embeddings of each node with those of its neighbours to endow them with contextual information. These architectures allow the networks to have a more complete understanding of the graph that helps it make node predictions for a variety of tasks, such as image and text classification or natural language processing, among others. Research in this area has so far mainly focused on developing new GNN architectures or applying existing ones to new domains. However, the type of embeddings being used has received significantly less attention. In most cases, node embeddings are defined manually, and created using any domain-specific information available; for example, when representing data from the academic research domain in which the nodes are scientific papers, embeddings can be bag-of-words representations of the textual contexts of a paper [21]. Another example can be found in the genetic information domain, using genetical positional sequences as embeddings [18]. Only in a few cases, node embeddings have been considered when dealing with GNN and they just employ popular embedding approaches such as TransE [4] and DistMult [45] as baseline in tasks like link prediction [31, 38].

At the same time, Knowledge Graphs (KGs) have become a popular research topic as companies such as Google, Facebook, and Amazon [28] are increasingly relying on them to integrate and curate information to support their business processes. A considerable amount of research effort

✉ Fernando Sola
fsola@us.es

Daniel Ayala
dayala1@us.es

Inma Hernández
inmahernandez@us.es

David Ruiz
druiz@us.es

1 University of Seville, ETSII, Avda. Reina Mercedes, s/n., Seville, Spain

focuses on developing deep learning techniques that are able to obtain embeddings in a domain-independent way [40]. This is usually done by training a neural network to generate task-specific embeddings, and then using said neural network to carry out some prediction task using the generated embeddings. In addition, it is also possible to leverage these embeddings by using them as input for other deep learning architectures such as GNN, and to tackle prediction tasks different from those they were initially designed for. This is feasible because a well-produced embedding space contains latent information about the elements represented in it [40], and therefore, they can be fed to architectures for a variety of tasks, such as question answering [51], KG completion [5, 19], scientific fact-checking [6], clustering [20], or plagiarism detection [12], to mention a few.

Additionally, deep embedding techniques can automatise the generation of embeddings from heterogeneous datasets, such as those containing nodes with different types and attributes. This can be a significant improvement over handmade embeddings, usually tied to certain entity types and attributes, e.g. the aforementioned bag-of-words representation of nodes symbolising textual documents. More recent approaches have introduced attributive embeddings [14] to be able to leverage the available information about the attributes of a node, which are usually numerical or textual values. This may be beneficial for prediction tasks in which the value of a property or the similarity between node properties can be exploited.

It has caught our attention that, while GNNs and deep KG embeddings are clearly related, there are almost no insights in the existing literature on how well they perform when combined, because most efforts have been put towards the application of GNN on different fields and the development of new GNN models and architectural variants [1]. While domain-specific embeddings benefit from GNNs, we believe that it is interesting to study the effect of using deep KG embeddings with GNNs because of their different nature and aforementioned advantages. This motivated us to carry out an experiment to shed some light on the feasibility of such combination.

In this paper, we present the results of an experimental study to analyse the benefits of combining deep KG embeddings with a baseline GNN architecture. Our focus is far from the specific GNN architectures and deep KG embeddings or their features, which have been thoroughly researched in previous works; instead, we want to analyse the performance improvement that can be expected from a state-of-the-art combination of both approaches. The improvement resulting from GNNs has been thoroughly studied for manually defined embeddings, but remains unexplored in the field of deep embeddings. These may be more challenging to exploit given their unsupervised nature, but can be computed from any graph without homogeneity restrictions or the need to manually define suitable representations. Specifically, our work focuses on comparing the results obtained by a baseline feedforward network and a standard GNN when trying to predict the value of different attributes. We test seven different deep embedding techniques on seven attribute prediction tasks. We focus mainly on the testing of attributive embeddings, since they contain additional information that could increase the benefits of applying a GNN; therefore, we limited ourselves to standard datasets on the deep embedding state-of-the-art that are rich in attributes, namely: YAGO [35] and FB15K237 [37]. The results obtained by these configurations contribute towards the state-of-the-art by thoroughly answering a series of open research questions about how much different types of neural networks can benefit from deep embeddings when performing prediction tasks.

The rest of this paper is structured as follows: Sect. 2 details the state-of-the-art in the fields of GNNs and KG deep embeddings. Section 3 describes the specific research questions we have identified and the neural network architectures used in our experiments. Section 4 describes the experimental setup and discusses the obtained results. Finally, Sect. 5 summarises our contributions and discusses potential future work.

2 Related work

In the following subsections, we summarise the current state-of-the-art both in the fields of GNNs and deep node embedding techniques.

2.1 Graph neural networks

Back in the 1990 s, neural networks were first applied to graphs by propagating states from one node to the others in an iterative way until a stable point was reached, using recurrent graph neural networks (RGNN) [44]. Some of their main drawbacks were that they were computationally costly and lacked representation capabilities and extendability. Later, several approaches that tried to leverage the progress in convolutional techniques emerged and redefined the concept of convolution on graphs by using not only the features of a node, but also those of its neighbours [7]. This type of procedure is common in image processing, in which pixels are updated with the information features of adjacent pixels. The resulting networks are known as convolutional graph neural networks (CGNN), and are further divided in two groups: spectral-based approaches and spatial-based ones [49]. RGNNs and CGNN are significantly related as they are both based on the same node representation update with neighbouring information principle. Their main difference is that RGNNs always use the same recurrent layer, using contractive constraints to ensure convergence, whereas CGNNs use several convolutional layers with different weights in each of them. These characteristics make CGNNs a more flexible, powerful and less costly approach than RGNNs, which are mostly considered, nowadays, pioneer works of GNNs [44] that inspired later research on convolutional networks. Therefore, CGNNs have emerged as the dominant architecture for graph-related tasks, and for that reason, this work focuses on these widespread and state-of-art approaches.

In other matters, the performance of GNNs when applied on a KG might be influenced by the size and type of the graph, so these aspects should be taken into consideration. KGs can be classified as follows [52]:

- **Directed vs. undirected**: directed edges provide more information than undirected ones, which can also be seen as double-directed edges.
- **Homogeneous vs. heterogeneous**: heterogeneous graphs provide a type for each node and edge, adding additional information to them.
- **Spatiotemporal vs. static**: on dynamic graphs, also known as spatial-temporal ones, topology or features change over time, a characteristic that needs to be properly addressed.
- **Large vs. small**: there are not clearly defined criteria to distinguish between a small or large graph; the boundaries are ever changing due to computation capabilities improvements on devices like GPUs.

There are different kinds of tasks that can be carried out using GNNs: node attribute prediction, node classification [21], link or edge strength prediction [17, 26, 33], and graph level tasks such as graph classification [29, 46, 48]. Nonetheless, there are some challenges about GNNs that are still to be addressed. The literature specially reports some scalability issues, as these techniques usually require having the graph loaded in memory in order to perform the convolutions, since doing sampling or clustering may end up losing neighbourhood information [44]. Other challenges than remain unsolved are defining a method to systematically select the optimal GNN architecture for a given network or task, and finding the most suitable knowledge graph embedding technique to maximise the performance of the network.

2.2 Deep node embedding techniques

Knowledge graph deep node embeddings have been widely studied recently because of their numerous possibilities [16]. Such embeddings aim to represent nodes as multi-dimensional vectors while retaining information about the structure of the graph and the attributes of its nodes, so that they can be used as input for other algorithms in subsequent tasks. Consequently, it should be noted that the performance of GNNs, as deep learning architectures, can therefore be influenced by the type of node embedding it is provided with.

Typical deep embedding approaches use distance-based scoring functions to produce embeddings and to maintain information about the relations between nodes. This way, with a triple $<s,r,t>$ where s and t are source and target nodes and r the relation between them, the embedding of s plus the embedding of r should be near the embedding of t in the corresponding multi-dimensional space. In this regard, these approaches only take into consideration the topology of the graph, and so they are called structure-based embeddings.

When using these techniques, literal information contained in nodes such as textual, numeric or even image properties is not leveraged. Since these literals provide useful information, the challenge lies in learning embeddings taking them into account, which can be addressed in two ways [14]. The first option is to handle literals separately, i.e., training the classical structure-based embedding and a node feature-learner one at the same time so that the network uses both data sources in each step to learn the node embeddings. The second option is to combine the classical structure-based embedding with the node literals by adding, multiplying, concatenating, etc. these features in the form of additional embeddings. Intuitively, these attributive embeddings contain much richer information that can be helpful for GNNs and their message-passing nature.

Some of the classical knowledge graph embedding generation techniques are TransE [4] and TransR [24]. TransE trains a vector of embeddings for each entity and relation, so that the sum of the embedding of the source node and the relation embedding results in the embedding of the target node. TransR works in a similar way, but performs the addition with the projection of the embedding vectors into a different space, separate from that of entities and relations.

In terms of attributive embedding techniques, ASNE [23] trains a network that predicts the node connections from the concatenation of an embeddings vector associated to the node itself (structural embedding) and another one associated to the attributes. The vector of attribute embeddings is created from the concatenation of numeric attributes (including their value) and categorical attributes (by one-hot encoding). It does not take into account non-categorical textual attributes, nor does it consider the existence of different types of relations.

LiteralE [22] integrates the embedding of each node with a vector of the node's attributes values. This integration is done by a modular function that enriches the embeddings before providing them to an existing embedding adjustment network, among which the authors propose and provide an implementation of DistMult and ComplEx. The integration function uses dense layers to train the way in which information about literals will be integrated.

MTKGNN [36] incorporates literals by introducing a learning task in addition to the typical separation of positive and negative triples. This task attempts to predict the value of a numerical attribute, so that the embeddings associated with the entities are indirectly affected by them. TransEA [43], similar to MTKGNN, incorporates the attributes in an additional learning task, added to TransE. It only considers numeric attributes.

2.3 State-of-the-art approaches

With all previous considerations in mind, we summarise in Table 1 the most popular GNN architectures approaches, as well as the datasets on which they are applied, the embedding techniques that they use, and the training, testing, and validation splits that are used in their experimental validation. Among them, the most referenced proposals like GCN [21] or

Table 1 GNN architecture popular approaches (a dash means "not specified")

Approach	Datasets	Node embeddings	Train/test/val. %
PATCHY-SAN (2016) [27]	MUTAG	-	10-fold cross-val.
	PCT	-	10-fold cross-val.
	NCI1	-	10-fold cross-val.
	NCI109	-	10-fold cross-val.
	PROTEIN	-	10-fold cross-val.
	D&D	-	10-fold cross-val.
GraphSage (2017) [18]	Thomson Reuters Web of Science Core Collection citations dataset	GenSim word2vec	80/14/6
	Reddit	GloVe CommonCrawl	67/23/10
	PPI	Gene sets and features	84/8/8
GCN (2017) [21]	Cora	Bag-of-words feature vector	9/61/30
	Citeseer	Bag-of-words feature vector	7/62/31
	Pubmed	Bag-of-words feature vector	4/64/32
	NELL	Features vector	12/58/30
MoNet (2017) [25]	Cora	Bag-of-words feature vector	9/61/30
	Pubmed	Bag-of-words feature vector	4/64/32
GAT (2018) [39]	Cora	Bag-of-words feature vector	9/61/30
	Citeseer	Bag-of-words feature vector	7/62/31
	Pubmed	Bag-of-words feature vector	4/64/32
	PPI	50 features vector	84/8/8
GAAN (2018) [47]	Reddit	-	67/23/10
	PPI	-	84/8/8
	METR-LA	-	70/20/10
FastGCN (2018) [9]	Cora	-	45/37/18
	Pubmed	-	92/5/3
	Reddit	-	65/25/10
ClusterGCN (2019) [10]	PPI	-	82/7/11
	Reddit	-	66/24/10
	Amazon	-	28/72/-
	Amazon2M	Bag-of-words feature vector	70/30/-
AP-GCN (2021) [34]	Citeseer	Bag-of-words feature vector	29/29/42
	Cora-ML	Bag-of-words feature vector	23/47/30
	Pubmed	Bag-of-words feature vector	3/92/5
	MS-Academic	Bag-of-words feature vector	1/94/5
	Amazon Computers	Bag-of-words feature vector	-
	Amazon Photos	Bag-of-words feature vector	-

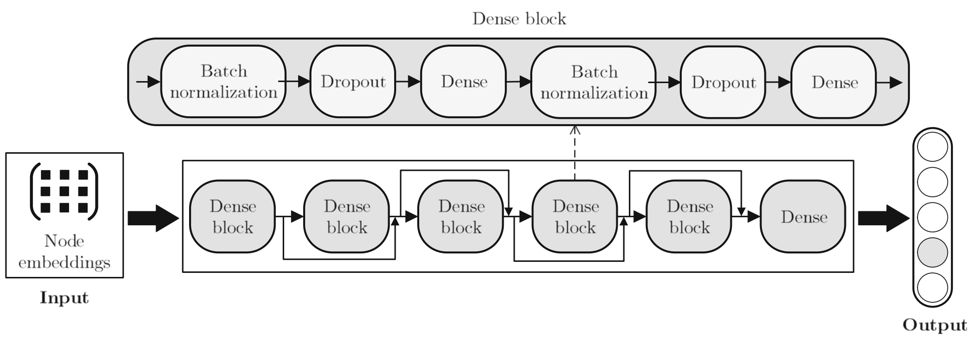

Fig. 1 Baseline neural network architecture

GraphSage [18], represent the classical formulation of GNNs with iterative updates of node features through an aggregation function over the features of neighbouring nodes. Besides these architectures, attention-based GNNs like GAT [39] and GAAN [47], employ attention mechanisms to allow the model to learn neighbouring nodes weights according to their importance.

These GNN architectures offer some advantages like their ability to handle graph-structured data, capture local and global structures, and reach state-of-art performance. However, GNNs also face several challenges such as scalability, interpretability, and robustness [44, 52]. Recent studies have proposed solutions to address these issues, including the use of hierarchical GNNs [8] or adversarial training [42]. Despite these efforts, further research is required to make GNNs more efficient, interpretable, and robust. Another issue to be addressed is the lack of standardisation in the design and evaluation of GNN models, making it difficult to compare results across studies and benchmark datasets; although some efforts have been made in this direction [11].

This is evidenced by the fact that, as shown in Table 1, for all proposals, the technique used to compute node embeddings is usually not specifically designed for the task, or it is even not described at all, not being considered a relevant aspect of the proposal. In terms of datasets, we find that Reddit [18], PPI [18], Cora [32], and Citeseer [15], which are domain-specific ones, are commonly used. Other general purpose KGs that are heterogeneous and rich in node attributes, such as YAGO [30] or Freebase [3], are not usually considered. Additionally, there is a big variability in the train/test/validation splits, since they are very dependent on the characteristics of each dataset. These reasons led us to carry out the current study, in which we focus on the combination of deep embeddings and GNN, abstracting ourselves from state-of-art architectures particularities, as explained later on Sect. 3.2.

3 Aim and scope

In this section we describe our contributions: in Sect. 3.1 we define the goals of our work and our research questions, while Sect. 3.2 describes the architecture of the neural networks that we used to answer the previous research questions.

3.1 Goals

The previous study of the state-of-the-art clearly shows that there has been little consideration for the combination of GNN and KG deep embeddings. More specifically, to the best of our knowledge, it does not exist a report on how this type of

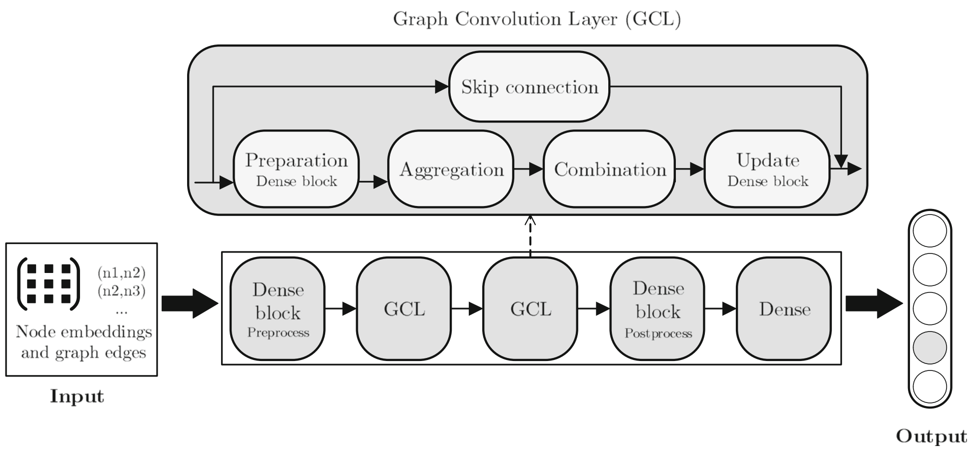

Fig. 2 Standard GNN architecture

Table 2 Datasets summary

Dataset	Nodes	Edges	Edge types	Attributes	Attribute types
FB15K237	14,192	306,773	237	85,894	119
YAGO	29,735	123,630	32	3,691,702	24
CORA	2,708	5,429	1	-	-

neural networks are affected by the use of different kinds of KG deep embeddings. It is currently unclear whether or not more specific research is needed to fully exploit the capabilities of GNNs when applied to deep embeddings. Therefore, we have focused on answering a number of open research questions that will help identify the scenarios in which performance is affected the most by said networks:

- **Q1**: When using a GNN, does the kind of deep embedding that is used have a significant effect on performance? To what extent is the performance of GNNs affected by the use of attributive embeddings?
- **Q2**: When using deep embeddings for attribute prediction tasks, does the use of a GNN instead of a regular neural network result in significant performance differences?
- **Q3**: Are the aforementioned differences affected by the kind of deep embedding being used?
- **Q4**: Is the improvement as pronounced as the reported in the state of art for tasks that use domain-specific embeddings? Does it vary with the prediction task?
- **Q5**: To what extent is the improvement achieved by GNNs affected by the amount of training data?

3.2 Architecture of the neural networks

In order to answer the previous questions and, since the specific architectures being used are not the focus of our work, we used a baseline feedforward neural network and a standard graph neural network. The feedforward one, described in Fig. 1, is composed by a series of five *dense blocks* with skip connections, except the first one; and a final dense layer which provides the output. Each *dense block* is composed by two groups of batch normalisation, dropout and dense layers. The computational modules employed in this architecture are well-known, standard and commonly used together [2, 13, 41] so they conform a representative enough network for our experiments.

The standard GNN, shown in Fig. 2, is based in GraphSage [18], which is a well-known, representative and mature GNN approach of the state-of-art [50], and compared to other well-stablished GNN approaches like GCN [21] and GAT [39], offers a more flexible convolution configuration because of the variable aggregation function. Our representative architecture is composed of two dense blocks that perform preprocessing and postprocessing functions and between which we have placed two graph convolution lay-

Table 3 Datasets used in each state-of-the-art deep embedding technique

Deep embedding technique	Datasets
ASNE	Facebook Oklahoma
	Facebook UNC
	DBLP
	Citeseer
LiteralE-ComplEx	Freebase (FB15K, FB15K237)
	YAGO (YAGO3-10)
LiteralE-DistMult	Freebase (FB15K, FB15K237)
	YAGO (YAGO3-10)
MTKGNN	YAGO (YG24K)
	Freebase (FB28K)
TransEA	YAGO (YG58K)
	Freebase (FB15K)
TransE	WordNet
	Freebase (FB15K, FB1M)
TransR	WordNet (WN18, WN11)
	Freebase (FB15K, FB13, FB40K)

ers. These are made up of a setup *dense block*, a message aggregation layer and a node embedding updating layer, as well as a skip connection over them. Finally, the output is provided by a dense layer.

It is worth noting that all layers in the networks, essentially the dense layers, are configured with 32 nodes or units. The initial hyperparameter setup was the one proposed in GraphSage [18], and was carefully tuned through a set of small, empirical and informal tests until a suitable typical state-of-art configuration was found. This configuration includes a learning rate of 0.01, RMSprop optimization, a dropout rate of 0.5, a batch size of 256 and 300 epochs with early stopping.

In terms of input, the feedforward network only receives a matrix containing the embeddings of the graph nodes and outputs the predicted value for the specific attribute prediction task performed. The input of the GNN architecture is a matrix of the embeddings of the nodes and the list of graph edges, that is, the relations between nodes in a (source, target) format. The GNN architecture also outputs the predicted attribute value for each node. Graph nodes embeddings are computed with the knowledge graph embedding techniques specified later in Sect. 4 and using the available node attributes on each dataset.

4 Evaluation

In this section, we discuss our evaluation setup: the datasets that we have used, the different embedding techniques under evaluation and the tasks that were performed. Subsequently, we display and comment on our experimental results.

4.1 Experimental setup

The attribute-rich datasets we took into consideration for our experiments were FB15K237 [37] and a reduced version of YAGO [35] which only contains nodes with at least one attribute and that have at least ten connected edges. It is worth noting that they were chosen because they are standard attribute-rich datasets on the deep embedding state-of-art, as can be seen in Table 3. Other datasets such as WordNet are also typically used, but they do not contain node attributes, making it impossible to apply attribute-based techniques on them. Additionally, we also included Cora [32], a citation dataset that is very commonly used in experiments involving GNNs, as shown in Table 1. The first two datasets are rich in attributes, which allow us to compute attributed embeddings and perform attribute prediction tasks, while the Cora dataset allows us to compare the improvement in performance with regards to that obtained by domain-specific embeddings. More detailed statistics of each dataset, like nodes, edges and attributes number, are provided in Table 2.

In terms of embedding techniques, we selected the ones proposed by Asefa et al. [14], which employ textual and/or numerical attributes and have an accessible implementation, namely: ASNE [23], LiteralE-ComplEx [22], LiteralE-DistMult [22], MTKGNN [36], and TransEA [43]. In addition, we included two well-known non-attributive embeddings as baselines: TransE [4] and TransR [24].

The prediction tasks we selected, as shown in Table 4, include a variety of node attributes to predict. It is also worth noting that tasks involving FB15K237 and YAGO consist in the regression of numerical attributes, while the Cora task consists in node classification. While the node type can be treated as an attribute, the Cora dataset does not include any actual attributes that can be used to compute attributive embeddings, and thus we limit the deep embeddings experiments to the FB15K237 and YAGO tasks, leaving Cora dataset to the domain-specific embeddings experiment that serves as a representative of how GNNs improve performance when using said embeddings. We tested each task with three different train/test split proportions: 80%/20%, 50%/50% and 20%/80%, respectively. Additionally, we executed every task 10 times in order to better assess the overall obtained performance.

To perform the experiments, we implemented a script, shown in Algorithm 1, to execute all the combinations in terms of train split proportions and embedding techniques for every defined task in the datasets, for both the baseline feedforward neural network and the standard GNN. The output

Table 4 Prediction tasks

KG	Task	Nodes	No. of different values	Range of observed values
FB15K237	filmRating	739	13	[0, 100]
	locationArea	2,150	2,063	[0.004, 165,250,000]
	personHeight	2,870	122	[1.35, 2.18]
	populationNumber	52,704	49,928	[0, 1,205,624,648]
YAGO	hasLatitude	8,671	6,620	[-75, 73]
	hasLongitude	8,671	7,394	[−171.83, 178.44]
	hasArea	11,922	10,075	[0.52, 8,000,036]
Cora	hasSubject	2708	7	[0, 6]

Table 5 GNN accuracy on the Cora dataset (higher is better)

Train%	NN	GNN	ΔAccuracy
20%	64.74	67.56	+2.82
50%	73.96	81.06	+7.11
80%	76.67	84.85	+8.17

of every step consists of the prediction evaluation metrics values for each setup.

Experiments where performed on a computer equipped with an Intel Core i9-9900K CPU, 32 GB of RAM and an Nvidia RTX 2080Ti GPU.

Algorithm 1 Experimental process

```
Input: $TS$: List<Double>                                  Train splits
       $EN$: Integer                                 Executions number
       $D$: Dictionary with datasets, tasks and its types, and
       embedding techniques
Output: $PM$: Dictionary with prediction evaluation metrics
1  foreach $dataset$ in $D$ do
2      foreach $task$ in $dataset$ do
3          foreach $train\_size$ in $TS$ do
4              foreach $embedding\_technique$ in $task$ do
5                  foreach $iteration$ in $EN$ do
6                      $test\_nn(task, train\_size,$
7                      $embedding\_technique, iteration)$
8                      $test\_gnn(task, train\_size,$
9                      $embedding\_technique, iteration)$
10                 end
11             end
12         end
13     end
14 end
```

4.2 Experimental results and analysis

Tables 5 and 6 show the results of the experiments that we conducted (note that a more comprehensive account of the embeddings experimental results can be found in Table 8). Table 7 offers a summarised view of the results.

Table 5 collects the reference results of the Cora dataset, in terms of accuracy. Figure 4 shows in a more visual way the MAE difference between GNNs and NNs depending on the performed task and the train set size, for deep embedding technique LiteralE-DistMult, in which the effect of using different train set sizes is particularly significant.

Table 6 contains the mean absolute error (MAE) difference obtained after applying each embedding technique in combination with the standard neural network and the GNN, to perform different tasks and considering different training set sizes, on two of the datasets (FB15K237 and YAGO). Each execution was repeated ten times to compute average values. We have boldfaced those results where MAE difference is a negative value, that is, GNN outperforms the regular neural network.

Fig. 3 displays a representative picture of the convergence of the networks and early stopping policy, on two representative tasks, employing ASNE embeddings and 80% training set size. Table 9 shows a summary of mean training times for both architectures on every performed task.

Next, we provide the answers to the questions posed in Sect. 3.1, according to the former experimental results.

Q1: When using a GNN, does the kind of deep embedding that is used have a significant effect on performance? To what extent is the performance of GNNs affected by the use of attributive embeddings?

There are clear differences in GNN performance depending on the embedding technique and it is more noticeable in certain tasks, e.g. between TransEA and ASNE in the $hasLatitude$ task, as shown in Table 8. However, we have not identified an embedding technique that consistently obtains better results in most tasks. The same applies to non-attributive embeddings (TransE and TransR) when compared to the rest. We conclude that the kind of embedding has a significant effect in a per-task manner, without a clear overall winner.

Q2: When using deep embeddings for attribute prediction tasks, does the use of a GNN instead of a regular neural network result in significant performance differences?

Generally, it does, as can be seen in the ΔMAE% columns in Table 6, showing the percentage difference between GNNs and NN in bold. The results indicate that the GNN outperforms the regular NN in the 65.7% of cases, Table 7, with GNN taking less but longer-in-time epochs to converge, Fig. 3. However, some tasks tend to leverage context information and thus, the improvement is greater in them, like $locationArea$ or $populationNumber$, see Fig. 4, while other tasks may only need their own information to make a judgement, and the use of convolutions and contextual data may blur the characteristics of the current node, showing GNN then, no clear strength over NN, like in task $hasArea$. It may therefore be of interest to study whether the neighbour-aggregation process of GNN is beneficial for a specific task and whether the improvement in results obtained with a GNN compared to those of a common neural network justify the higher complexity and computational cost of a GNN when dealing with that specific task, which can mean i.e., on average, about double the training time, as shown in Table 9.

Q3: Are the aforementioned differences affected by the kind of deep embedding being used?

Some embeddings seem to benefit more from the application of a GNN with higher, more consistent reductions of the MAE. Overall, there is a higher chance of improvement

Table 6 Differences in Mean Absolute Error of a GNN vs a regular NN using different embeddings. Negative differences, where GNN outperforms NN, are shown in **bold**

KG	Task	ASNE	LiteralE-ComplEx	LiteralE-DistMult	MTKGNN	TransEA	TransE	TransR	Task mean ΔMAE%
FB15K237	filmRating	**-2.47**	**-1.91**	0.01	**-2.37**	1.41	0.15	1.60	**-0.51**
	locationArea	**-18.54**	**-6.30**	**-12.96**	**-8.51**	**-15.77**	**-10.35**	7.19	**-9.32**
	personHeight	7.16	**-1.45**	**-2.15**	0.40	1.00	1.81	0.58	1.05
	populationNumber	**-18.26**	**-23.65**	**-10.37**	**-13.01**	**-1.87**	**-11.95**	**-0.38**	**-11.36**
YAGO	hasLatitude	2.01	0.18	**-2.54**	2.59	**-3.80**	**-1.72**	0.38	**-0.41**
	hasLongitude	**-20.35**	3.29	**-6.52**	**-2.46**	**-1.17**	**-1.36**	8.10	**-2.92**
	hasArea	**-5.62**	12.05	6.34	**-8.3**	10.82	15.90	**-8.76**	3.21
	Embedding mean ΔMAE%	**-8.01**	-2.54	**-4.03**	**-4.52**	**-1.34**	**-1.07**	1.25	

Table 7 Comparison of a regular NN vs GNN, summarized MAE results

	Attributive embeddings	Non-attributive embeddings
% cases with improvement	65.71%	42.86%
Mean improvement	−8.28%	−5.75%
Mean worsening	3.94%	4.47%

Table 8 Regular NN vs GNN MAE extended comparison

KG	Task	Train	ASNE			LiteralE-ComplEx			LiteralE-DistMult			MTKGNN			TransEA			TransE			TransR		
		%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%
FB15K237	filmRating	20%	9.57	9.16	**-4.34**	16.47	15.81	**-4.04**	16.50	15.79	**-4.35**	16.24	15.58	**-4.07**	15.98	15.98	**-0.05**	16.46	16.31	**-0.90**	16.20	16.02	**-1.07**
		50%	5.99	6.29	5.06	15.47	15.13	**-2.17**	14.90	15.52	4.21	15.23	15.00	**-1.50**	15.46	15.79	2.17	15.55	15.68	0.84	14.75	15.27	3.53
		80%	5.53	5.08	**-8.14**	14.58	14.65	0.47	14.74	14.76	0.16	14.87	14.65	**-1.53**	15.42	15.75	2.12	15.27	15.35	0.52	14.56	14.90	2.34
	locationArea ($x10^5$)	20%	12.62	9.87	**-21.82**	9.80	10.89	11.05	10.86	8.90	**-18.06**	10.16	9.00	**-11.37**	9.51	9.08	**-4.45**	9.07	6.76	**-25.46**	7.06	6.88	**-2.62**
		50%	9.91	7.46	**-24.75**	9.40	8.08	**-14.09**	9.84	8.21	**-16.54**	8.75	8.28	**-5.38**	11.77	8.50	**-27.74**	9.06	9.01	**-0.56**	7.88	10.28	30.49
		80%	9.22	8.38	**-9.05**	10.06	8.46	**-15.85**	8.90	8.52	**-4.27**	9.44	8.61	**-8.77**	9.49	8.06	**-15.12**	8.58	8.15	**-5.04**	9.27	8.69	**-6.28**
	personHeight	20%	0.05	0.06	9.46	0.07	0.07	**-3.06**	0.07	0.07	**-3.17**	0.07	0.07	2.23	0.08	0.08	0.38	0.08	0.08	0.63	0.08	0.08	1.17
		50%	0.05	0.05	5.63	0.07	0.07	**-0.94**	0.07	0.07	**-1.58**	0.07	0.07	0.41	0.08	0.08	1.39	0.08	0.08	2.53	0.07	0.07	1.07
		80%	0.04	0.05	6.37	0.07	0.07	**-0.35**	0.07	0.06	**-1.70**	0.06	0.06	**-1.46**	0.08	0.08	1.24	0.08	0.08	2.27	0.07	0.07	**-0.51**
	populationNumber ($x10^6$)	20%	8.28	6.54	**-20.97**	8.75	6.80	**-22.30**	7.75	6.33	**-18.25**	7.34	6.88	**-6.25**	6.86	6.78	**-1.29**	7.67	6.44	**-16.00**	6.38	6.34	**-0.59**
		50%	8.37	7.19	**-14.11**	8.84	6.98	**-21.04**	7.17	7.05	**-1.69**	9.24	6.95	**-24.71**	6.89	6.74	**-2.24**	7.80	7.01	**-10.10**	6.73	7.09	5.42
		80%	7.83	6.29	**-19.71**	8.88	6.43	**-27.62**	7.26	6.45	**-11.17**	7.07	6.50	**-8.07**	6.24	6.11	**-2.07**	6.60	5.96	**-9.75**	6.34	5.96	**-5.96**
YAGO	hasLatitude	20%	15.88	16.19	1.96	8.03	8.53	6.23	9.65	10.78	11.71	9.66	10.83	12.05	7.30	7.88	7.86	8.52	10.77	26.40	9.61	11.72	21.99
		50%	15.15	15.48	2.17	5.92	5.87	**-0.86**	7.72	6.99	**-9.54**	7.66	7.66	0.01	5.02	4.61	**-8.17**	5.41	4.65	**-13.98**	6.14	6.07	**-1.15**
		80%	15.15	15.44	1.90	5.74	5.46	**-4.84**	7.33	6.61	**-9.80**	6.96	6.66	**-4.29**	5.06	4.50	**-11.08**	5.29	4.36	**-17.59**	6.06	4.87	**-19.69**
	hasLongitude	20%	39.33	37.92	**-3.57**	20.20	20.85	3.22	23.89	22.48	**-5.90**	23.07	22.69	**-1.64**	15.94	16.04	0.60	20.93	23.11	10.41	21.44	28.73	34.01
		50%	36.32	26.95	**-25.80**	15.71	17.01	8.28	18.73	17.69	**-5.56**	18.80	17.97	**-4.42**	13.60	13.66	0.49	14.35	13.27	**-7.53**	15.95	16.55	3.77
		80%	36.51	24.95	**-31.67**	15.80	15.54	**-1.63**	18.75	17.24	**-8.09**	17.76	17.52	**-1.33**	13.63	13.01	**-4.59**	15.16	14.10	**-6.96**	16.28	14.09	**-13.47**
	hasArea ($x10^4$)	20%	13.67	12.40	**-9.29**	7.56	7.52	**-0.42**	7.95	8.61	8.33	9.19	7.06	**-23.24**	8.67	9.23	6.50	10.66	10.47	**-1.77**	12.19	8.92	**-26.78**
		50%	12.77	12.32	**-3.52**	6.19	7.11	14.83	6.53	6.94	6.25	6.82	7.16	4.96	6.20	7.32	18.06	5.96	7.14	19.75	7.87	6.63	**-15.74**
		80%	12.48	11.97	**-4.06**	4.60	5.60	21.75	6.47	6.76	4.44	5.96	5.57	**-6.63**	6.20	6.69	7.89	5.92	7.68	29.74	6.56	7.62	16.26

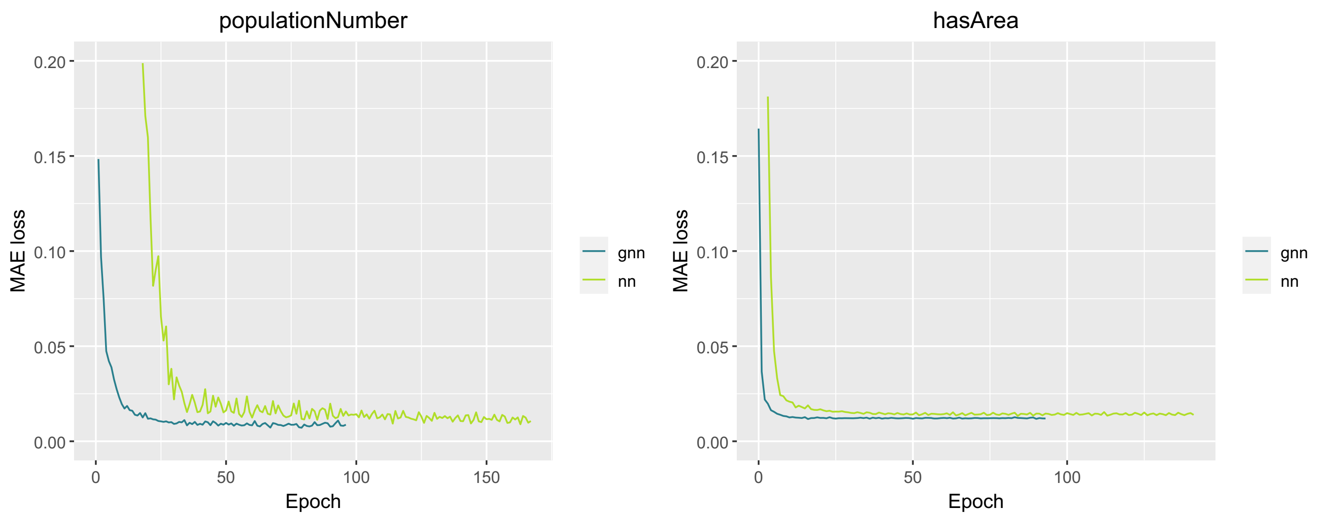

Fig. 3 NN vs GNN convergence on two representative tasks for ASNE embedding technique

for attributive embeddings than for non-attributive ones. As can be seen in Table 7, GNNs lead to an improvement in roughly 43% of cases when using TransR or TransE non-attributive embeddings, but when using the attributive ones ASNE, LiteralE-ComplEx, LiteralE-DistMult, MTKGNN and TransEA there is an improvement in 65% of cases. For these reasons, the deep embedding technique used as GNN input should be carefully selected as it may heavily influence the obtained results.

Q4: Is the improvement as noticeable as the reported in the state of art for tasks that use domain-specific embeddings? Does it vary with the prediction task?

The results for the Cora dataset shown in Table 5 showcase an accuracy improvement between 3% and 8%. However, when talking about more generic and attribute-rich datasets, results vary significantly between tasks, as shown in Table 6.

It is clear that some tasks, like *populationNumber*, seem to benefit more from the contextual information provided by the GNNs, reaching improvements of more than 25% in some cases, see Table 8. In other tasks, such as *hasArea* or *personHeight*, there is little information in the context of a node that could help improve the prediction while, for example, the rating of a film could be easier to predict based on its contextual information. In this case, GNNs lead to

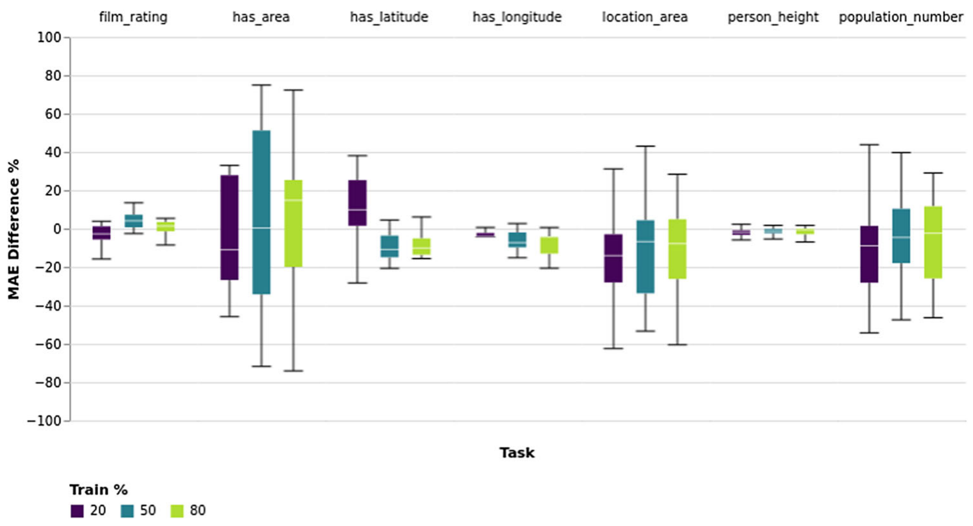

Fig. 4 Mean Absolute Error differences of Regular NN vs GNN

lesser improvements or even worse results, probably as a consequence of the increased architecture complexity. These results are in contrast to the state-of-the-art ones, where GNN experiments are always carried out on citations or domain-specific datasets, as seen in Table 1, and reach better performances.

Q5: To what extent is the improvement achieved by GNNs affected by the amount of training data?

As can be seen in Fig. 4, there is no clear trend when training size is altered. In some tasks a reduced training size leads to a larger improvement, maybe due to how a GNN can help use additional information to compensate for the lack of numerous examples. In others, the opposite happens, which may be caused by the higher complexity of a GNN needing a larger number of examples to reach its potential. Therefore, training set size does not seem to be a determining factor in GNNs performance improvement against regular neural networks.

Table 9 NN and GNN mean training times (in seconds)

KG	Task	NN	GNN
FB15K237	filmRating	9.22	19.20
	locationArea	10.68	24.05
	personHeight	13.37	42.20
	populationNumber	10.32	23.47
YAGO	hasLatitude	17.27	41.37
	hasLongitude	17.86	43.90
	hasArea	24.81	63.08
CORA	hasSubject	8.98	12.99
	Mean training time	**14.06**	**33.78**

5 Conclusions and future work

In this paper we have presented a comprehensive study about how deep embeddings perform when applied together with Graph Neural Networks (GNNs). Deep embeddings have the appeal of being domain-independent, potentially able to capture latent information about the content of the graph and offer universal, automatised embedding generation that can deal with heterogeneous datasets. This has led to their extended use in a variety of tasks, including prediction of graph elements by feeding them to neural networks. GNNs, which intend to endow these networks with contextual information, seem to be a perfect fit, but so far they have only been tested with domain-specific embeddings, which motivated our study.

The novelty and value of our work resides in how we answer several open research questions about the performance of GNNs under several sets of circumstances including seven attribute prediction tasks, seven types of deep embeddings, and three different training sizes. We conclude that the application of GNNs to improve performance obtained by deep embeddings has significant potential, as it can be seen in several tasks in which there is a reduction of error of more than 25%. However, research so far has focused too much on proposing novel network architectures and too little on determining under what circumstances they work best. As our experiments have shown, the same GNN can obtain completely different results depending on the task and embeddings being used.

Future work should focus on collecting a large set of attribute-rich datasets for the evaluation of GNNs and deep embeddings in an automated way. It would be particularly useful to catalogue said datasets according to their topology and other characteristics that should affect how useful the information-passing mechanisms in GNNs are.

Acknowledgements This work was supported by the Spanish Ministry of Science, Innovation and Universities, under Grant PID2019-105471RB-I00; and by the Office for Economy and Knowledge of the Andalusian Regional Government under Grant P18-RT-1060 and Grant US-1380565.

Funding Funding for open access publishing: Universidad de Sevilla/CBUA

Data Availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflicts of interests The authors declare no conflicts of interests.

References

1. Abadal S, Jain A, Guirado R, et al (2022) Computing graph neural networks: A survey from algorithms to accelerators. ACM Comput Surv 54(9):191:1–191:38. https://doi.org/10.1145/3477141
2. Abdani SR, Zulkifley MA, Hussain A (2019) Compact convolutional neural networks for pterygium classification using transfer learning. In: 2019 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2019, Kuala Lumpur, Malaysia, September 17-19, 2019. IEEE, pp 140–143. https://doi.org/10.1109/ICSIPA45851.2019.8977757
3. Bollacker K, Evans C, Paritosh P, et al (2008) Freebase: A collaboratively created graph database for structuring human knowledge. In: SIGMOD. ACM New York, NY, USA, pp 1247–1250. https://doi.org/10.1145/1376616.1376746
4. Bordes A, Usunier N, García-Durán A, et al (2013) Translating embeddings for modeling multi-relational data. In: Burges CJC, Bottou L, Ghahramani Z, et al (eds) 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. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html
5. Borrego A, Ayala D, Hernández I, et al (2021) CAFE: Knowledge graph completion using neighborhood-aware features. Engineering Applications of Artificial Intelligence 103:104302. https://doi.org/10.1016/j.engappai.2021.104302
6. Borrego A, Dessì D, Hernández I et al (2022) Completing scientific facts in knowledge graphs of research concepts. IEEE Access 10:125867–125880. https://doi.org/10.1109/ACCESS.2022.3220241
7. Bronstein MM, Bruna J, LeCun Y et al (2017) Geometric deep learning: Going beyond euclidean data. IEEE Signal Process Mag 34(4):18–42. https://doi.org/10.1109/MSP.2017.2693418
8. Chen C, Li K, Wei W et al (2022) Hierarchical graph neural networks for few-shot learning. IEEE Trans Circuits Syst Video Technol 32(1):240–252. https://doi.org/10.1109/TCSVT.2021.3058098
9. Chen J, Ma T, Xiao C (2018) Fastgcn: Fast learning with graph convolutional networks via importance sampling. In: ICLR 2018. https://openreview.net/forum?id=rytstxWAW
10. Chiang W, Liu X, Si S, et al (2019) Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In: Teredesai A, Kumar V, Li Y, et al (eds) ACM SIGKDD 2019. ACM, NY, USA, pp 257–266. https://doi.org/10.1145/3292500.3330925
11. Dwivedi VP, Joshi CK, Luu AT et al (2023) Benchmarking graph neural networks. Journal of Machine Learning Research 24(43):1–48
12. Franco-Salvador M, Rosso P, Montes-y Gómez M (2016) A systematic study of knowledge graph analysis for cross-language plagiarism detection. Information Processing & Management 52(4):550–570
13. Garbin C, Zhu X, Marques O (2020) Dropout vs. batch normalization: an empirical study of their impact to deep learning. Multim Tools Appl 79(19–20):12777–12815. https://doi.org/10.1007/s11042-019-08453-9
14. Gesese GA, Biswas R, Alam M et al (2021) A survey on knowledge graph embeddings with literals: Which model links better literal-ly? Semantic Web 12(4):617–647. https://doi.org/10.3233/SW-200404
15. Giles CL, Bollacker KD, Lawrence S (1998) Citeseer: An automatic citation indexing system. In: Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA. ACM, pp 89–98. https://doi.org/10.1145/276675.276685
16. Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: A survey. Knowl Based Syst 151:78–94. https://doi.org/10.1016/j.knosys.2018.03.022
17. Hamaguchi T, Oiwa H, Shimbo M, et al (2017) Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In: IJCAI pp 1802–1808. https://doi.org/10.24963/ijcai.2017/250
18. Hamilton WL, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems pp 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html
19. Huang X, Zhang J, Li D, et al (2019) Knowledge graph embedding based question answering. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining pp 105–113
20. Kerdjoudj F, Curé O (2015) RDF knowledge graph visualization from a knowledge extraction system. In: Joint Proceedings of the 1st International Workshop on Summarizing and Presenting Entities and Ontologies and the 3rd International Workshop on Human Semantic Web Interfaces (SumPre 2015, HSWI 2015) co-located with the 12th Extended Semantic Web Conference (ESWC 2015), Portoroz, Slovenia, June 1, 2015, CEUR Workshop Proceedings, vol 1556. URL https://ceur-ws.org/Vol-1556/paper3.pdf
21. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: ICLR. https://openreview.net/forum?id=SJU4ayYgl
22. Kristiadi A, Khan MA, Lukovnikov D, et al (2019) Incorporating literals into knowledge graph embeddings. In: Ghidini C, Hartig O, Maleshkova M, et al (eds) The Semantic Web - ISWC 2019

- 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I, Lecture Notes in Computer Science, vol 11778. Springer, pp 347–363. https://doi.org/10.1007/978-3-030-30793-6_20
23. Liao L, He X, Zhang H et al (2018) Attributed social network embedding. IEEE Trans Knowl Data Eng 30(12):2257–2270. https://doi.org/10.1109/TKDE.2018.2819980
24. Lin Y, Liu Z, Sun M, et al (2015) Learning entity and relation embeddings for knowledge graph completion. In: Bonet B, Koenig S (eds) Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. AAAI Press, pp 2181–2187. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571
25. Monti F, Boscaini D, Masci J, et al (2017) Geometric deep learning on graphs and manifolds using mixture model cnns. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. IEEE Computer Society, pp 5425–5434. https://doi.org/10.1109/CVPR.2017.576
26. Nathani D, Chauhan J, Sharma C, et al (2019) Learning attention-based embeddings for relation prediction in knowledge graphs. In: Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. Association for Computational Linguistics, pp 4710–4723. https://doi.org/10.18653/v1/p19-1466
27. Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: Balcan M, Weinberger KQ (eds) ICML, JMLR Workshop and Conference Proceedings, vol 48. JMLR.org, pp 2014–2023. http://proceedings.mlr.press/v48/niepert16.html
28. Noy N, Gao Y, Jain A et al (2019) Industry-scale knowledge graphs: lessons and challenges. Communications of the ACM 62(8):36–43. https://doi.org/10.1145/3331166
29. Pan S, Wu J, Zhu X et al (2017) Task sensitive feature exploration and learning for multitask graph classification. IEEE Trans Cybern 47(3):744–758. https://doi.org/10.1109/TCYB.2016.2526058
30. Rebele T, Suchanek FM, Hoffart J, et al (2016) YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In: ISWC, pp 177–185. https://doi.org/10.1007/978-3-319-46547-0_19
31. Schlichtkrull MS, Kipf TN, Bloem P, et al (2018) Modeling relational data with graph convolutional networks. In: Gangemi A, Navigli R, Vidal M, et al (eds) The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings, Lecture Notes in Computer Science, vol 10843. Springer, pp 593–607. https://doi.org/10.1007/978-3-319-93417-4_38
32. Sen P, Namata G, Bilgic M et al (2008) Collective classification in network data. AI Mag 29(3):93–106. https://doi.org/10.1609/aimag.v29i3.2157
33. Shang C, Tang Y, Huang J, et al (2019) End-to-end structure-aware convolutional networks for knowledge base completion. In: AAAI, pp 3060–3067. https://doi.org/10.1609/aaai.v33i01.33013060
34. Spinelli I, Scardapane S, Uncini A (2021) Adaptive propagation graph convolutional network. IEEE Trans Neural Networks Learn Syst 32(10):4755–4760. https://doi.org/10.1109/TNNLS.2020.3025110
35. Suchanek FM, Kasneci G, Weikum G (2007) Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007, ACM, pp 697–706. https://doi.org/10.1145/1242572.1242667
36. Tay Y, Tuan LA, Phan MC, et al (2017) Multi-task neural network for non-discrete attribute prediction in knowledge graphs. In: Lim E, Winslett M, Sanderson M, et al (eds) Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017. ACM, pp 1029–1038. https://doi.org/10.1145/3132847.3132937
37. Toutanova K, Chen D (2015) Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality, pp 57–66
38. Vashishth S, Sanyal S, Nitin V, et al (2020) Composition-based multi-relational graph convolutional networks. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. URL https://openreview.net/forum?id=BylA_C4tPr
39. Velickovic P, Cucurull G, Casanova A, et al (2018) Graph attention networks. In: ICLR. https://openreview.net/forum?id=rJXMpikCZ
40. Wang Q, Mao Z, Wang B et al (2017) Knowledge graph embedding: A survey of approaches and applications. IEEE Trans Knowl Data Eng 29(12):2724–2743. https://doi.org/10.1109/TKDE.2017.2754499
41. Wang SH, Tang C, Sun J et al (2018) Multiple sclerosis identification by 14-layer convolutional neural network with batch normalization, dropout, and stochastic pooling. Frontiers in Neuroscience 12. https://doi.org/10.3389/fnins.2018.00818
42. Wei Q, Wang J, Fu X et al (2023) AIC-GNN: adversarial information completion for graph neural networks. Inf Sci 626:166–179. https://doi.org/10.1016/j.ins.2022.12.112
43. Wu Y, Wang Z (2018) Knowledge graph embedding with numeric attributes of entities. In: Augenstein I, Cao K, He H, et al (eds) Proceedings of The Third Workshop on Representation Learning for NLP, Rep4NLP@ACL 2018, Melbourne, Australia, July 20, 2018. Association for Computational Linguistics, pp 132–136. https://doi.org/10.18653/v1/w18-3017
44. Wu Z, Pan S, Chen F et al (2020) A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems 32(1):4–24
45. Yang B, Yih W, He X, et al (2015) Embedding entities and relations for learning and inference in knowledge bases. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings
46. Ying Z, You J, Morris C, et al (2018) Hierarchical graph representation learning with differentiable pooling. In: Bengio S, Wallach HM, Larochelle H, et al (eds) NeurIPS, pp 4805–4815. https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html
47. Zhang J, Shi X, Xie J, et al (2018a) Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In: Globerson A, Silva R (eds) UAI. AUAI Press, pp 339–349. http://auai.org/uai2018/proceedings/papers/139.pdf
48. Zhang M, Cui Z, Neumann M, et al (2018b) An end-to-end deep learning architecture for graph classification. In: McIlraith SA, Weinberger KQ (eds) AAAI. AAAI Press, pp 4438–4445. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17146
49. Zhang S, Tong H, Xu J, et al (2018c) Graph convolutional networks: Algorithms, applications and open challenges. In: International Conference on Computational Social Networks, Springer, pp 79–91
50. Zheng C, Chen H, Cheng Y, et al (2022) Bytegnn: Efficient graph neural network training at large scale. Proc VLDB Endow 15(6):1228–1242. https://www.vldb.org/pvldb/vol15/p1228-zheng.pdf
51. Zheng W, Yu JX, Zou L et al (2018) Question answering over knowledge graphs: question understanding via template decomposition. Proc of the VLDB Endowment 11(11):1373–1386

52. Zhou J, Cui G, Hu S et al (2020) Graph neural networks: A review of methods and applications. AI Open 1:57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

Fernando Sola received his B.S. degree in Software Engineering and M.S. degree in Data Science from the University of Seville in 2021 and 2022, respectively. He is currently a Ph.D. student with the University of Seville and a member of DEAL group, focusing his research work on graph neural networks, deep graph embeddings and, recently, gene databases integration and antibiotic bacterial resistance prediction.

Daniel Ayala is currently a Researcher and Professor with the University of Seville. His research with the DEAL Group, so far, has focused on techniques related to structured data integration and mining, and lately, in the field of knowledge graphs. His current research interests include the use of graph neural networks for attribute completion and generelated predictions regarding antibiotic resistance. He was a recipient of many awards, such as the Best Resource Paper Award in the ESWC 2019 Conference and the Best Paper of the Month awards from the University of Seville-ETSII.

Inma Hernández has been the Coordinator of the master's program in software engineering (cloud, data, and IT management) with the Postgraduate School, University of Seville, since 2020. She is currently an Associate Professor with the University of Seville and a Founding Member of the Data Engineering Applications Laboratory. She is also a Principal Investigator for several projects funded by the Spanish National Research and Development Program. Her current research interests include data engineering and knowledge graphs. She has authored many peer-reviewed publications on these topics in top conferences and journals. She is a very active reviewer and a member of several program committees at major conferences.

David Ruiz is currently a Full Professor of software engineering with the University of Seville. He leads the Data Engineering Applications Laboratory, University of Seville, focusing his research on data engineering, knowledge graphs, and data integration. He has recently started two new related lines of research, focusing on the application of machine learning techniques for the automated retrieval and processing of aviation data, and for the genomic analysis of break multi-resistant bacteria.