MASTER'S DEGREE PROJECT

Deep embeddings and Graph Neural Networks: can context improve domain-independent predictions?

Produced by
Fernando Luis Sola Espinosa

For the completion of
Máster en Ingeniería del Software: Cloud, Datos y Gestión TI

Supervised by
Inmaculada Concepción Hernández Salmerón
Daniel Ayala Hernández

June call, academic year 2021/22

"The Python panda is used to work with data structure in efficient manner."

Rupal Snehkunj and Khushboo Vachiyatwala, 2022

Acknowledgements

I would like to thank the DEAL research group for all the support, help and feedback they have given me at all times. They have been, without a doubt, an essential part of the realisation of this project.

I would also like to thank my parents, my brother and my whole family for encouraging me, taking care of me and putting up with me on a daily basis.

I would also like to thank my grandmother Dolores, who was unable to see me finish my Master's degree, but I am sure she is proud of me. Here's to you, grandma.

Resumen

Las redes neuronales de grafos (GNN) son arquitecturas de aprendizaje profundo que aplican convoluciones de grafos mediante procesos de "message passing" entre nodos, representados como embeddings. Las GNNs se han hecho populares recientemente por la forma en que obtienen una representación contextual de cada nodo que tiene en cuenta la información de los nodos circundantes. Los trabajos existentes en la literatura se han centrado en el desarrollo de arquitecturas GNN, utilizando información básica específica del dominio sobre los nodos para calcular los embedding.

En el contexto de los grafos de conocimiento, se han realizado muchos esfuerzos para desarrollar técnicas de aprendizaje profundo que permitan obtener embeddings de nodos que preserven la información sobre las relaciones y la estructura sin depender de los datos específicos del dominio. El potencial de la aplicación de las redes neuronales de grafos con embeddings profundos de grafos de conocimiento sigue inexplorado en su mayoría.

En este proyecto, llevamos a cabo una serie de experimentos para responder a preguntas de investigación abiertas sobre el rendimiento de dichos embeddings cuando se utiliza una red neuronal de grafos. Probamos 7 técnicas de embeddings profundos diferentes en varias tareas de predicción de atributos en dos conjuntos de datos ricos en atributos. Llegamos a la conclusión de que hay una mejora significativa en el rendimiento, pero varía mucho dependiendo de la tarea y de la técnica de embedding empleada.

Dado el interés de los resultados obtenidos, enviamos un artículo a la conferencia CIKM'22, y hemos definido algunas tareas y trabajos futuros para continuar este estudio en forma de doctorado.

Palabras clave: Grafos de Conocimiento, Redes Neuronales de Grafos, Embeddings Atributivos, Embeddings profundos de grafos, Machine Learning

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 how they obtain a contextual representation of each node that takes into account the information from surrounding nodes. Existing work has focused on the development of GNN architectures, using basic domain-specific information about the nodes to compute embeddings.

In the context 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 of the application of graph neural networks with deep embeddings of knowledge graphs remains largely unexplored.

In this project, we carry out a number of experiments to answer open research questions about how said embeddings perform when using a graph neural network. We test 7 different deep embeddings across several attribute prediction tasks in two attribute-rich datasets. We conclude that there is a significant performance improvement but it varies heavily depending on the task and deep embedding technique.

Given the interest of the results obtained, we have submitted an article to the conference CIKM'22, and we have defined some tasks and future work in order to continue this study in the form of a PhD.

Keywords: Knowledge Graphs, Graph Neural Networks, Attributive embeddings, Deep graph embeddings, Machine Learning

Contents

List of Figures

List of Algorithms

1. Introduction

1.1. Introduction

Graph Neural Network (GNN) architectures seek to leverage the connections of a graph when it comes to making predictions about the elements of the graph [33]. To achieve this, the nodes of the graph are represented as numeric vectors called embeddings that can capture and summarize the implicit information present in them. For example, graph convolutional layers [20] 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 overall picture when it comes to, for example, making predictions about nodes. Research in this area so far has mainly focused on developing new 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 created using any domain-specific information available about the nodes; for example, when representing data from the academic research domain in which papers are nodes, the embeddings can be bag-of-words representations of the text in the papers [14]. Another example can be found in the genetic information domain, using gene positional sequences as embeddings [10].

Knowledge Graphs (KGs) have become a popular research topic as many companies such as Google, Facebook, and Amazon [21] are increasingly relying on the integrated and curated information in knowledge graphs. A considerable amount of research effort focuses on developing deep learning techniques that are able to obtain embeddings in a domain-independent way. A well-produced embedding space contains latent information about the elements it represents, and therefore can be fed to algorithms for a variety of tasks [38, 12, 13, 6, 31], such as question answering, KG completion, link prediction, or clustering, to mention a few. These techniques usually train the embeddings so that they contain the necessary information to predict the presence of edges in the graph. More recent approaches have introduced attributive embeddings [7] to take into account the information about node properties, which tend to be numeric values such as years or ages. This may be beneficial for prediction tasks in which the value of a property or the similarity with that of another node 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. While domain-specific embeddings benefit from GNNs, it is interesting to study the effect of using deep KG embeddings with GNNs because of their different nature.

This motivated us to carry out an experiment to shed some light on the feasibility of such combination. 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 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 datasets that are rich in attributes: YAGO [27] and FB15K237 [29]. 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.

All of this has made it possible to send an article to the Conference on Information and Knowledge Management'22 (CIKM 2022), Appendix A.1.

1.2. Context of this TFM

In this section, the following points are developed in a reasoned manner:

— **Degree of difficulty or complexity of the subject**

GNN is a technology that has been on the rise in the last two or three years and on which there is a lot of work to be done today. It is another deep learning approach, which means that the mathematical and algorithmic components are very present, so the theoretical difficulty is high, although the convolution principle characteristic of GNNs is not really that complicated.

However, as far as implementation and experimentation are concerned, the degree of difficulty is considerable, especially in my case since I opted for the use of a baseline network, and given that frameworks such as PyTorch or, in my case, Tensorflow with Keras, have a steep learning curve.

Talking about the issue addressed, that is, the study of the performance of GNNs when employing deep/attributive embeddings, it is worth noting that no work has been done on this research line, as showed by our state of art analysis, so the interest and novelty of this project is high.

— **Degree of compliance with the objectives set**

Initially, all of the defined goals and sub-goals have been fulfilled. The completion of the goal G1.1 about the state of art can be found in Section 3, goal G1.2 about the implementation of the technical infrastructure employed in experiments can be found in Section 4.3 and Section 4.4, goal G1.3 about the selection of adecuate datasets can be found in Section 4.4, goal G1.4 about experimentation and the research questions posed can be found in Section 4.2 and Section 4.4, goal G1.5 about the analysis of the results obtained can be found in Section 4.5 and Section 4.6; and goal G1.6 about the dissemination of the results can be found on the paper presented to CIKM'22 which is offered as an annex in Section A.1.

— **Research component of the work**

Through a deep study of the state of the art, we identified 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. That is why this project focuses on shedding some light on the feasibility of this combination by performing many experiments with different setups, datasets and deep embedding techniques. To do so, we developed both a baseline feedforward classical neural network and a standard graph neural network with a highly configurable setup in order to automate this experimentation. As a result we were able to give an answer to some open research questions and extract some valuable conclusions that may define the direction of further research.

— **Potential application of this work in an academic or industrial environment**

This work has generated interesting results and have opened new lines of research as collecting a large set of attribute-rich datasets for the evaluation of GNNs and deep embeddings in an automated way, or catalogue datasets according to their topology and other characteristics that should affect how useful the information-passing mechanisms in GNNs are.

So much so, in fact, that it has led to an article sent to the conference Conference on Information and Knowledge Management'22 (CIKM 2022), Appendix A.1.

1.3. Document structure

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

2. Preliminary study

2.1. Introduction

This chapter is dedicated to management aspects and prior to the execution of the project, such as the study of the technologies used, the objectives set, the methodology followed and the time and cost planning estimated at the beginning, as well as the real values obtained at the end of the project.

2.2. Goals

The main objective of this project is:

Goal G1: Study the influence of deep embeddings on graph neural networks performance when combined.

To do so, we have defined a set of sub-goals:

Goal G1.1: Study the state of art by understanding graph neural networks architecture, variations, mode of operation, applicability and most popular approaches.

Goal G1.2: Implement a baseline feedforward neural network and a standard graph neural network script with a highly configurable execution pipeline, in order to automate subsequent experiments as much as possible.

Goal G1.3: Select a variety of benchmark knowledge graphs/datasets, including some of the most widely used in the state of the art, and others that are suitable for the specific experimentation we are proposing.

Goal G1.4: Define and execute a set of experiments that can provide answers to the research questions posed in this project.

Goal G1.5: Analyse experiments results by means of visual elements such as charts and tables in order to extract final conclusions.

Goal G1.6: Disseminate the results of this work by means of a research paper to be presented in a prestigious international forum or journal.

2.3. Methodology

The methodology used has been an adaptation of the classic software development methodologies with different phases executed sequentially to the research context and within the limitations and particularities of a Master's thesis. Therefore, work has been carried out based on the objectives defined for the project, which are very similar to its different phases in a similar way to the stages of the research process and the typical structure of a scientific paper in the field; starting with a study of the current literature, the implementation of an architecture for the pertinent experimentation, the selection of suitable data sets, experimentation based on defined research questions, the analysis of the results obtained and their dissemination.

It is worth highlighting the use, in this process, of different management support tools such as Toggl to control the dedication or GitHub for the versioning of the developed code [1]. In addition, the weekly monitoring of the project's progress through meetings with the project manager and the consequent acquisition of feedback on a regular basis is also noteworthy.

2.4. Planning

The time required to complete the three hundred hours stipulated for a Master's thesis was divided between the different parts or phases worked on in this project, mainly related to the goals defined in Section 2.2. An initial time distribution planning is presented, Table 2.1.

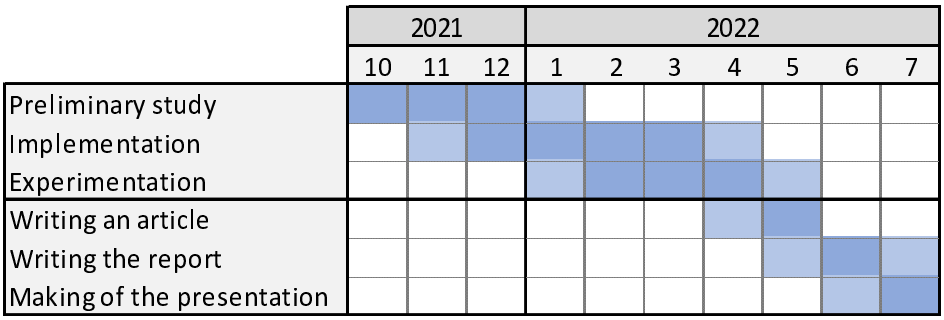

Table 2.1: Initial planning

The estimation of each of these phases and the actual time spent on the completion of the project is therefore presented below:

Phase	**Estimated dedication (hours)**	**Real dedication (hours)**
Preliminary study	70	60
Implementation	100	126.73
Experimentation	70	40
Writing an article	30	59.82
Writing the report	20	18.03
Making of the presentation	10	9.24
Total	**300**	**313.82**

In addition to the classical research phases, those of writing a scientific article based on the study carried out, this project report and the making of the presentation have been included.

As can be seen, the estimates made are fairly accurate with respect to the hours finally invested. The most pronounced deviations are to be found in the phases of experimentation and writing a research paper, mainly due to my lack of experience on these tasks and my lack of first-hand knowledge of their typical scope.

[1] https://github.eii.us.es/DEAL/fsola-CIKM-22-code

This meant that the 300 hours planned finally amounted to a total of 313.82 hours spent.

2.5. Budget

The costing of this project will take into account several items, the amount of which will depend to a greater or lesser extent on the amount of time spent on the project.

Firstly, personnel costs, for a full-time computer engineer with an annual salary of around 32,000 euros [2], plus income tax and social security costs, amount to a total annual cost of 41,568 euros for 1800 hours a year, i.e. 23.09 euros per hour worked.

On the other hand, within the costs of equipment and supplies, the depreciation of the equipment used should be included, a laptop with an initial acquisition cost of 680 euros, and a high-performance server with an i9 9900k, 64gb of RAM and an RTX 2080ti graphics card; with an initial cost of 2,903.70 euros and with an estimated useful life of four years for both computers, i.e. 48 months, and eight months of use for the project; the electricity supply for the computers, with an estimated combined consumption of around 1000W/hour and an average cost per kilowatt per hour of 0.40 euro cents; and the internet connection supply, with a monthly fee of 55 euros per month.

Finally, a risk reserve cost of ten percent of the total costs is established to cover any incident that may occur during the development of the project.

This brings the estimated costs to a total of 8,433.90 euros:

Item	**Calculation**	**Total**
Personnel cost	23.09€/h x 300 hours	6.927€
Amortización del equipo	3,583.70€/48 months x 8 months of project	597.28€
Electricity supply	1kWh x 0,40€ x 300 hours	120€
Internet connection supply	55€ / 720hours/month x 300 hours	22,91€
	Subtotal sum	**7,667.19€**
	Reserve costs (10%)	**766.72€**
	Total	**8,433.90€**

The costs finally incurred, after the execution of the project, were:

[2] Value on 19 June 2022 on https://www.glassdoor.es/Sueldos/ingeniero-inform%C3%A1tico-sueldo-SRCH_KO0,21.htm

Item	Calculation	Total
Personnel cost	23.09 €/h x 313.82 hours	7,246.10 €
Amortización del equipo	3,583.70 €/48 months x 8 months of project	597.28 €
Electricity supply	1kWh x 0,40 € x 313.82 hours	125.53 €
Internet connection supply	55 € / 720hours/month x 313.82 hours	23.97 €
	Total spending	**7,992.88 €**
	Total planned with reserve	**8,433.90 €**
	Surplus	**441.02 €**

Thanks to the risk reserve cost, the deviations produced by the extra temporal dedication were covered and the total spending was into budget.

2.6. Conclusions

In this section, different aspects of the management of the project itself have been presented and reviewed.

3. State of art

3.1. Introduction

In this section we present the existing related work and offer a more deep context of the domain and the problem.

3.2. Graph neural networks

Graphs are a mathematical tool that compose a whole widely studied field known as graph theory. Nevertheless, in short, graphs are no more than structures made of nodes, which may have a set of features that describe and characterise them, and edges between them, which, again, may have some kind of weight or feature vector associated. These principles are now applied to many domains where data is represented in form of graphs leveraging its great expressive power, but which has associated challenges involving the complexity of the application of algorithms on these kind of data.

It has been a huge recent progress on neural networks, deep learning and the treatment of Euclidean data, that is, images, text or video, with convolutional neural networks (CNN) that leverage the local features of elements and their inter-connectivity to extract much richer information. In that way, there is an increasing interest in applying these deep learning methods to non-Euclidean domains, generally known as geometric deep learning and, specifically, when talking about graphs, this architectures are known as graph neural networks (GNNs).

Back in the 1990s, 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) [33]. Some of their main drawbacks were that they are computationally costly and lack 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 [3]. 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 [37]. 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.

Talking about CGNNs and its two approaches, spectral-based ones works on the spectral graph theory, with a solid mathematical foundation in graph signal processing and defining graph convolutions as removing noises from graph signals. Spatial-based approaches, on their side, define convolutions as RGNNs do, an information passing process, and have been developed quickly due to its efficiency and flexibility.

Graph neural networks performance might be influenced by knowledge graphs size and type, so they should be taken into consideration. KGs can be classified as following [39]:

- **Directed/undirected**: directed edges provide more information than undirected ones, which can also be seen as double-directed edges.
- **Homogeneous/heterogeneous**: heterogeneous graphs provides a type for each node and edge, adding an additional value to them.
- **Spatiotemporal/static**: on dynamic graphs, also known as spatial-temporal ones, topology or features change over time, a characteristic that needs to be properly addressed.
- **Small/large**: there is not a clear defined criteria to distinguish between a small or large graph as it is 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 [14], link or edge strength prediction [9, 26, 19], and graph level tasks such as graph classification [36, 34, 22, 23]. With all that, GNNs can work on a wide variety of domains: on computer vision by parsing images into graphs consisting of objects and their relationships or vice versa, natural language processing by leveraging words or documents relations to label them, for example; traffic flows prediction, recommender systems modelled as a graph with users and items as nodes or in biology and chemistry fields by leveraging the graph-like structure of molecules, compounds or proteins interactions.

Nonetheless, there are some challenges about GNNs that are still to be addressed like the depth of these models, that is, how many graph convolutional layers, and thus convolutions, do we have to use in order to get good results taking into consideration that with message passing strategy and a considerable number of convolutions, nodes information will converge and be equal; or scalability issues, as these techniques usually require having the graph loaded in memory in order to perform the convolutions and doing sampling or clustering may end up on losing neighbourhood information [11].

3.3. Node embedding techniques

Knowledge graphs embeddings techniques have been widely studied recently because of its numerous possibilities. 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, such as GNNs. Consequently, It should be noted that the performance of GNNs, as deep learning algorithms, can therefore be influenced by the type of node embedding it is provided with.

Typical knowledge graph embedding approaches use distance-based scoring functions to learn embeddings, 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 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 discarded.

The challenge lies in learning embeddings taking these literals into account, which can be addressed in two ways [7]. The first option is to handle literals separately, that is, 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 about each node and this may be highly leveraged by GNNs and their message-passing step.

3.4. GNN popular approaches

With all these considerations in mind, we summarise here, and in Table 3.1, the most popular GNN architectures approaches, as well as the tasks that they carry out, 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:

- **PATCHY-SAN (2016)** [20]: It focuses on learning substructures from graph data before convolution and was applied to graph classification task on bio-chemical domain datasets as MUTAG, PCT, NCI1, NCI109, PROTEIN and D&D. They do not mention how nodes are represented and performed a 10-fold cross-validation with 9 folds for training and one for testing.

- **GraphSage (2017)** [10]: It puts the focus on using feature information to train a model to produce embeddings for unseen nodes. They apply it to node and graph classification tasks on a citation dataset derived from Thomson Reuters Web of Science Core Collection, Reddit dataset and protein interactions PPI dataset. In the citation dataset, with 302,424 nodes, they are represented as 300-dimensional word vectors obtained from papers abstracts by GenSim word2vec implementation and nodes degrees, and use 2000-2004 data for training and 2005 for testing with 30% for validation. For Reddit dataset, with 232,965 nodes, they took a 300-dimensional GloVe CommonCrawl word vector for each node, which represents a post, concatenating the average embedding of the title, the average embedding of the post's comments, the post's score and the number of its comments; and use the posts of the first 20 days of the month of September 2014 for training and the remaining days of the month for testing with 30% for validation. In PPI dataset nodes are represented by positional gene sets, motif gene sets and immunological signatures features and there are 20 graphs for training, two for test and two for validation.

- **GCN (2017)** [14]: It is a spectral-based approach with some simplifications to improve scalability and applicability in large-scale graphs. It is tested in node classification with Cora, Citeseer, Pubmed and NELL datasets. Nodes are represented as sparse bag-of-words feature vectors in the three first ones, and as sparse feature vectors in the last one. They use 20 instances per class as training data, 1000 instances as test data and 500 labeled examples for hyperparameter optimization, the rest are used as unlabeled data.

- **MoNet (2017)** [18]: It focuses on a generic spatial-domain framework for deep learning on non-Euclidean data such as graphs and manifolds. It is tested in node classification with Cora and Pubmed datasets, with nodes represented as sparse bag-of-words feature vectors and with the same setup as in GCN, 20 instances per class for training, 1000 for test and 500 instances for validation.

- **GAT (2018)** [30]: It uses attentional layers, enabling the specification of different weights to different nodes in a neighbourhood, and test it on node classification task

with Cora, Citeseer and Pubmed datasets, and graph classification on PPI dataset. Nodes are represented as sparse bag-of-words feature vectors in the three first ones, and as a 50 features vector in PPI dataset. They use, for node classification, 20 instances per class as training data, that is 140, 120 and 60 respectively, 1000 instances as test data and 500 labeled examples for hyperparameter optimization; while for graph classification task, they use 20 graphs for training, 2 for validation and 2 for testing.

- **GAAN (2018)** [35]: It proposes a new attention model by gating attention heads to control its importance. It was applied to graph classification on PPI dataset and node classification on Reddit dataset, without specifying in any case, in what form the nodes are represented. For PPI they use 20 sub-graphs as training set, 2 in the validation set and 2 in the testing set. For Reddit dataset, they used the first 20 days posts for training and the rest for testing (leaving 30% for validation). They also address traffic speed forecasting where information is represented as a spatiotemporal graph, and use METR-LA dataset with 70% training, 10% validation and 20% testing.

- **FastGCN (2018)** [4]: It concentrates on dealing with memory issue on GNN training through batching using probability measures and test it on node classification on Cora, Pubmed and Reddit datasets. They do not specify nodes representations and take a 45/18/37 (in percentage) training/validation/test split for Cora, 92/3/5 for Pubmed and 65/10/25 for Reddit dataset.

- **ClusterGCN (2019)** [5]: It also focuses on memory issue and addresses it by, on each step, using a batch of nodes that form a dense subgraph and which is identified by clustering. It was applied to node classification task on PPI dataset, Reddit dataset and a self-made Amazon2M dataset. They do not specify PPI and Reddit datasets nodes representations but do specify that Amazon2M dataset nodes are represented as a bag-of-words features vector extracted from products descriptions which dimension was then reduced to 100 by applying PCA. They use a training/test/validation split of 82/7/11, 66/24/10, 28/72/- and 70/30/- for PPI, Reddit, Amazon and Amazon2M, respectively.

As can be seen, the most popular approaches focus on node or graph classification tasks, and do not give much importance to the type of representation of the nodes with which the algorithms are fed. In terms of benchmark datasets, we usually found Reddit [10], PPI [10], Cora [25], and Citeseer [8], which are domain-specific, while other general purpose graphs datasets, which are rich in attributes as YAGO [24] or FreeBase [1], are not usually considered. Additionally, there is a big heterogeneity in the train/test/validation splits, since they are very dependent on the characteristics of each dataset.

3.5. Conclusions

In this chapter we have reviewed graph neural networks and node embeddings previous work in order to show the context of the topic and point out the limitations we have found.

Table 3.1: GNN architecture popular approaches (a dash means not specified)

Approach	Task	Datasets	Node embeddings	Train/test/val. %
PATCHY-SAN (2016) [20]	Graph classification	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) [10]	Node classification	Citation dataset from Thomson Reuters Web of Science Core Collection	GenSim word2vec	80/14/6
		Reddit	GloVe CommonCrawl	67/23/10
	Graph classification	PPI	Gene sets and features	84/8/8
GCN (2017) [14]	Node classification	Cora	Sparse bag-of-words feature vector	9/61/30
		Citeseer	Sparse bag-of-words feature vector	7/62/31
		Pubmed	Sparse bag-of-words feature vector	4/64/32
		NELL	Sparse feature vector	12/58/30
MoNet (2017) [18]	Node classification	Cora	Sparse bag-of-words feature vector	9/61/30
		Pubmed	Sparse bag-of-words feature vector	4/64/32
GAT (2018) [30]	Node classification	Cora	Sparse bag-of-words feature vector	9/61/30
		Citeseer	Sparse bag-of-words feature vector	7/62/31
		Pubmed	Sparse bag-of-words feature vector	4/64/32
	Graph classification	PPI	50 features vector	84/8/8
GAAN (2018) [35]	Node classification	Reddit	-	67/23/10
	Graph classification	PPI	-	84/8/8
	Traffic speed forecasting (spatiotemporal graph)	METR-LA	-	70/20/10
FastGCN (2018) [4]	Node classification	Cora	-	45/37/18
		Pubmed	-	92/5/3
		Reddit	-	65/25/10
ClusterGCN (2019) [5]	Node classification	PPI	-	82/7/11
		Reddit	-	66/24/10
		Amazon	-	28/72/-
		Amazon2M	Reduced bag-of-words feature vector	70/30/-

4. Experimentation

4.1. Introduction

In this chapter we present the contributions made: in Section 4.2 we describe the research goals and questions we defined, in Section 4.3, we analyse the architectures we used to answer the previous questions, in Section 4.4 the datasets that we have used, the different embedding techniques under evaluation and the tasks that were performed. Subsequently, in Section 4.5 we display and comment on our experimental results, and in Section 4.6 we provide the answer to the research questions.

4.2. Research questions

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 any report on how this type of 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 being 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?
- **Q6**: Are the performance differences between GNNs and regular neural networks affected by the selected optimizer technique?

4.3. Architecture of the neural networks

In order to answer the previous questions, we used a baseline feedforward neural network and a standard graph neural network. The feedforward one, Figure 4.1, is composed by a series of five "dense blocks" with skip connections, except the first one; and a final dense layer which outputs logits, that is, the set of probabilities of the prediction classes in classification

and the predicted value in regression. Each "dense block" is composed by two groups of batch normalisation, dropout and dense layers. The input of this network is the matrix of the embeddings of the nodes.

The standard graph neural network, shown in Figure 4.2, is composed of two dense blocks that perform preprocessing and postprocessing functions and between which we have placed two graph convolution layers. These are made up of a preparation "dense block", a message aggregation layer, which can be performed by the sum, mean or max of the embeddings to aggregate; and a node embedding updating layer that can be configured to add, concatenate or employ a GRU layer; as well as a skip connection over them. Finally, the output is given by a dense layer as logits. The input of this network is a matrix of the embeddings of the nodes and the list of graph edges.

Note that, since the specific architectures are not the focus of this project, we have relied on some basic state-of-the-art architectures in both cases. We should also highlight that the technologies employed to implement these networks were Tensorflow and Keras in Python, leveraging, as far as possible, the provided layers by the framework, as explained before.

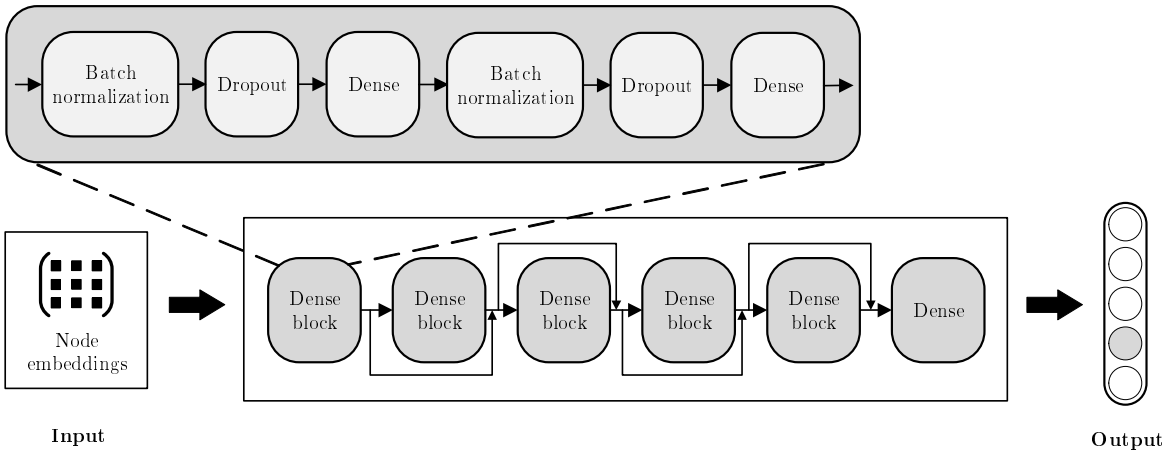

Figure 4.1: Baseline neural network architecture

4.4. Experimental setup

The attribute-rich datasets taken into consideration for experiments were FB15K237 [29] and a reduced version of YAGO [27] which only contains nodes with at least one attribute and have at least ten connected edges. Additionally, we also included Cora [25], a citations dataset that is very commonly used in experiments involving GNNs, as shown in Table 3.1. 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 these results to those obtained by domain-specific embeddings.

This dataset preprocessing consisted on a script that takes the various dataset files containing node embeddings, relations and tasks information, and outputs the necessary file hierarchy to be given to the script, Algorithm 1, explained later on this section. This file hierarchy was defined to facilitate its management and avoid repeated information, as node embeddings, which usually have a considerable size in disk. For example, for dataset YAGO,

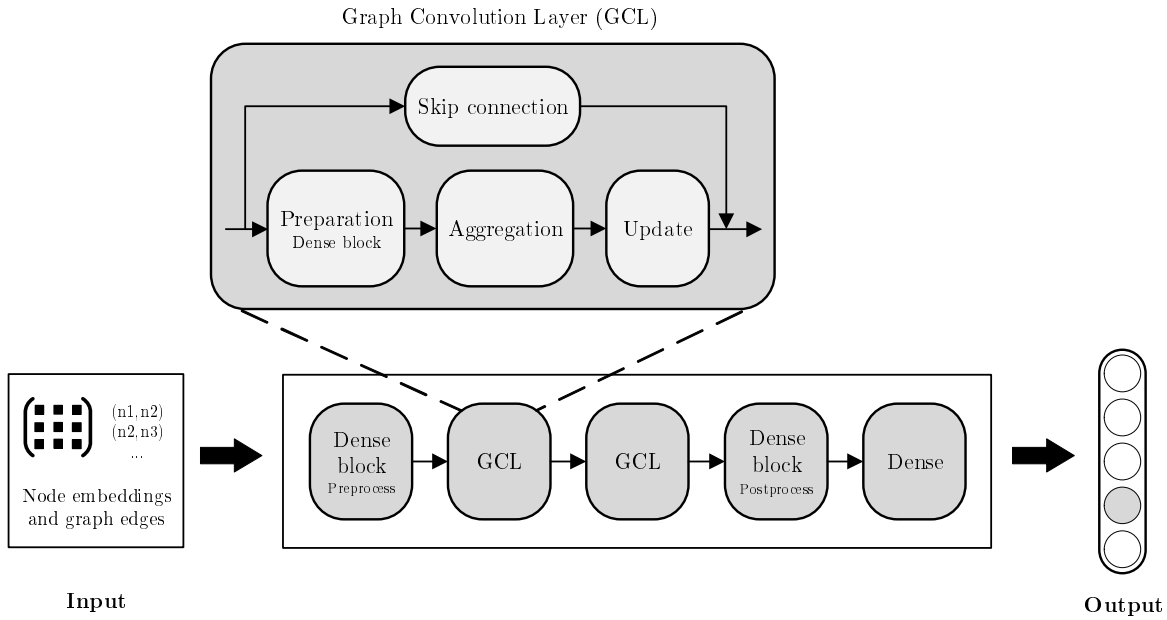

Figure 4.2: Standard graph neural network architecture

Figure 4.3, we have a folder *embeddings* which contains sub-folders for every embedding technique calculated and its correspondent *.nodes* file containing the embeddings. Additionally, each defined task has a folder to contain the prediction value for each node in a *.preds* file and the results to be generated subsequently. Finally, we have a *.edges* file on the same level as the other folders that contains the edges between nodes.

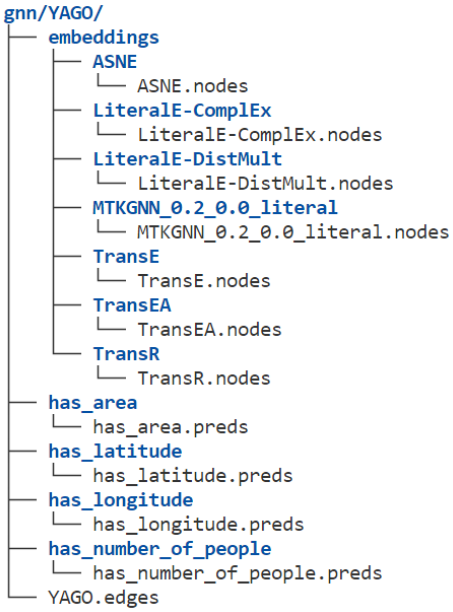

Figure 4.3: Folder hierarchy

In terms of embedding techniques, we selected the ones proposed by Gesese et al. [7], which employ textual or/and numerical attributes and have an accessible implementation: ASNE [16], LiteralE-ComplEx [15], LiteralE-DistMult [15], MTKGNN [28] and TransEA [32]. In addition, we included two well-known non -attributive embeddings as baselines: Trans-E [2]

Table 4.1: Prediction tasks

KG	Task	Nodes	Number of different values	Values range
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]

and Trans-R [17].

The node attribute prediction tasks we selected, shown in Table 4.1, include a variety of attributes to predict. Except for the dataset Cora, which have only one available task, the other ones were selected, as far as possible, taking into consideration the proportion between different possible values and the number of instances in order to select those with few possible values and lots of examples to facilitate the neural network's task of learning, e.g. $filmRating$, Table 4.1.

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 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 the 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%, 50% and 20% of examples for training. Additionally, we executed every task ten times in order to better assess the overall obtained performance.

To perform the experiments, we designed a script, 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 graph neural network. The datasets, tasks, embedding techniques and optimizers selected are given in a dictionary and data itself is contained in different files, as explained before in this section. It is also possible to perform automated binning, if indicated, in classification tasks with numerical values, that is, gather prediction values in different groups and assign a new prediction label to them. The output consists on the prediction evaluation metrics values for each setup, that is, MAE, MSE and RMSE for regression tasks and accuracy and cross entropy for classification tasks. However, in subsequent sections, we consider only MAE and accuracy for more simplicity.

Algorithm 1: Experimental script

Input: TS: List<Double> Train splits
EN: Integer Executions number
D: Dictionary with datasets, tasks and its types, embedding techniques and optimizers

Output: PM: Dictionary with prediction evaluation metrics

```
foreach dataset in D do
    foreach task in dataset do
        foreach train_size in TS do
            foreach optimizer in task do
                foreach embedding_technique in task do
                    foreach iteration in EN do
                        test_nn(task, train_size, optimizer, embedding_technique,
                        iteration)
                        test_gnn(task, train_size, optimizer, embedding_technique,
                        iteration)
```

Table 4.2: GNN accuracy on Cora

	NN	GNN	ΔMAE%
20%	64.7431	67.5637	4.36
50%	73.9569	81.0633	9.61
80%	76.6745	84.8474	10.66

4.5. Experimental results

Tables 4.2 and 4.3 show the results of the experiments that we conducted.

Table 4.2 collects the reference results of the Cora dataset, in terms of accuracy. Figure 4.4 shows in a more visual way the MAE difference between GNNs and NNs depending on the performed task on FB15K237 and YAGO, and the train set size, for deep embedding technique LiteralE-DistMult, in which the effect of using different train set sizes is particularly significant. On the other hand, Figure 4.5 offers a performance difference comparison between GNNs and regular networks in terms of the selected optimizer technique, on $film\_rating$ task of dataset FB15K237 and for LiteralE-DistMult deep embedding technique.

Table 4.3 contains the mean absolute error (MAE) 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.

4.6. Research questions

Next, we provide the answers to the questions posed in Section 4.2, according to the former experimental results.

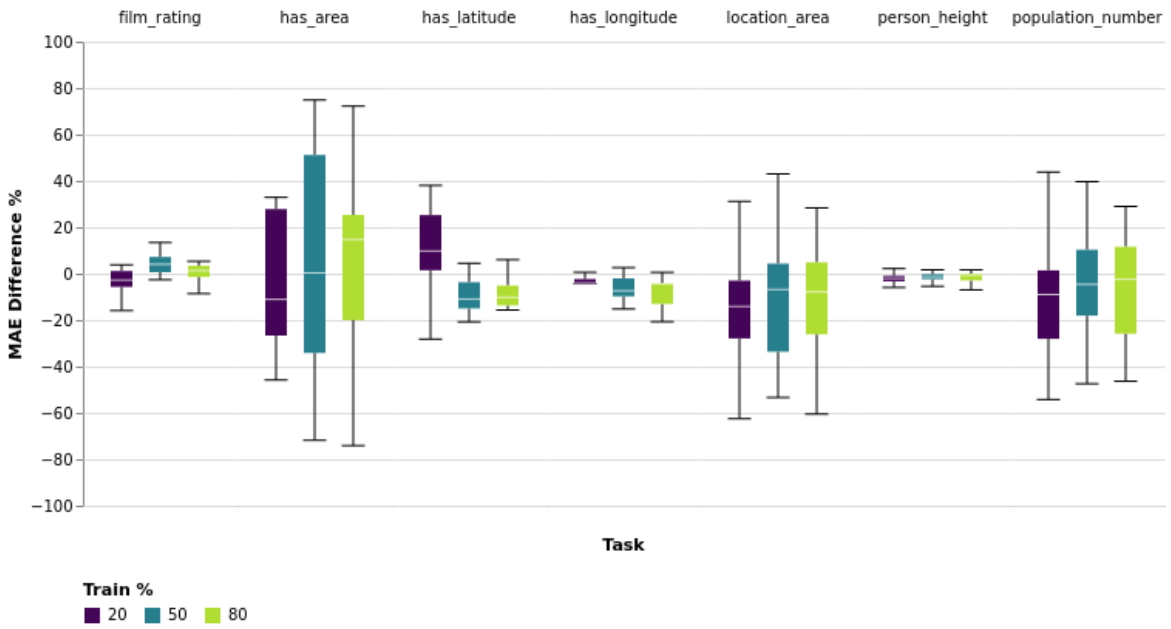

Figure 4.4: GNN vs. NN MAE percentage difference

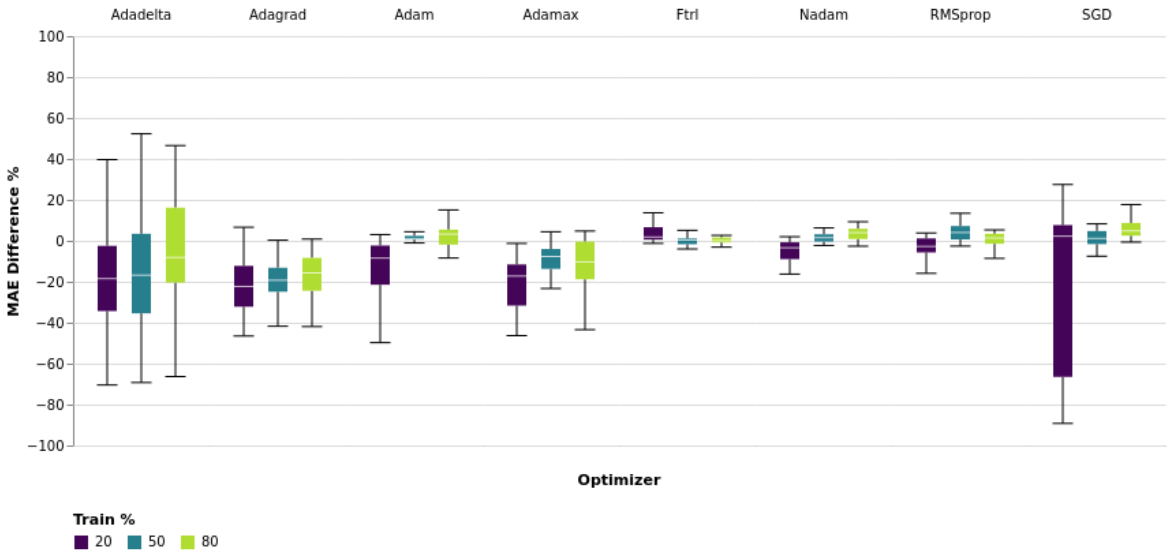

Figure 4.5: GNN vs. NN MAE percentage difference on different optimizer techniques

Q1: When using a GNN, does the kind of deep embedding being 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 in GNN column in Table 4.3. However, we haven't 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.

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 4.3, showing the percentage difference between GNNs and NN with bold numbers when GNN outperforms the regular NN in a majority of cases. However, some tasks tend to leverage context information and thus, the improvement is greater in them, like *locationArea* or *hasLongitude*, see Figure 4.4.

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 for attributive embeddings than for non-attributive ones with the exception of TransEA. As can be seen in Table 4.3, GNNs lead to an improvement in roughly 50% of cases when using the TransR, TransE, or TransEA embeddings, but when using ASNE, LiteralE-ComplEx, LiteralE-DistMult, and MTKGNN there is an improvement in 66% to 75% of cases.

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 Cora experiments in Table 4.2 show an accuracy improvement between 5% and 10%. The results in Table 4.3, however, vary significantly between tasks.

It is clear than in some tasks, like $filmRating$ or $personHeight$, there is little information in the context of a node that could help improve the prediction, while for example, the area of a country could be easier to predict based on contextual information. In this case, GNNs lead to 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 3.1, and reach stunning performances. Other tasks such as $populationNumber$ seem to benefit more from the contextual information provided by the GNNs, reaching improvements of more than 25%.

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

As can be seen in Figure 4.4, 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.

Q6: Are the performance differences between GNNs and regular neural networks affected by the selected optimizer technique?

As can be seen in Figure 4.5, it is clear that some optimizer techniques do benefit GNNs performance in contrast to regular neural networks, such as Adagrad or Adamax. In terms of the amount of training data, there is not an appreciable difference as the performance keeps generally the same despite training data size changes.

4.7. Conclusions

In this section we have we have defined a series of open research questions, presented the experimental setup we employed and the results obtained, and as a consequence, we provide the answers to the questions posed.

Table 4.3: Embedding techniques MAE comparison

			ASNE			LiteralE-ComplEx			LiteralE-DistMult			MTKGNN			TransEA			TransE			TransR		
		Train size	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%	NN	GNN	ΔMAE%
FB15K237	filmRating	20%	9.5737	9.1583	**-4.34**	16.4734	15.8074	**-4.04**	16.5041	15.7854	**-4.35**	16.2449	15.5837	**-4.07**	15.9838	15.9753	**-0.05**	16.4583	16.3101	**-0.90**	16.1969	16.0243	**-1.07**
		50%	5.9916	6.2949	5.06	15.4682	15.1325	**-2.17**	14.8965	15.5231	4.21	15.2264	14.9973	**-1.50**	15.4585	15.7934	2.17	15.5508	15.6815	0.84	14.7528	15.2734	3.53
		80%	5.5274	5.0774	**-8.14**	14.5767	14.6457	0.47	14.7380	14.7621	0.16	14.8746	14.6467	**-1.53**	15.4188	15.7456	2.12	15.2660	15.3452	0.52	14.5562	14.8972	2.34
	locationArea ($x10^5$)	20%	12.6237	9.8694	**-21.82**	9.8040	10.8875	11.05	10.8566	8.8964	**-18.06**	10.1552	9.0006	**-11.37**	9.5080	9.0846	**-4.45**	9.0743	6.7642	**-25.46**	7.0607	6.8754	**-2.62**
		50%	9.9147	7.4608	**-24.75**	9.4036	8.0787	**-14.09**	9.8365	8.2094	**-16.54**	8.7479	8.2774	**-5.38**	11.7681	8.5035	**-27.74**	9.0598	9.0088	**-0.56**	7.8769	10.2782	30.49
		80%	9.2169	8.3831	**-9.05**	10.0565	8.4622	**-15.85**	8.9004	8.5203	**-4.27**	9.4371	8.6090	**-8.77**	9.4937	8.0584	**-15.12**	8.5825	8.1500	**-5.04**	9.2707	8.6887	**-6.28**
	personHeight	20%	0.0527	0.0577	9.46	0.0709	0.0687	**-3.06**	0.0704	0.0681	**-3.17**	0.0697	0.0713	2.23	0.0816	0.0819	0.38	0.0811	0.0816	0.63	0.0785	0.0794	1.17
		50%	0.0455	0.0481	5.63	0.0668	0.0661	**-0.94**	0.0665	0.0654	**-1.58**	0.0653	0.0656	0.41	0.0797	0.0808	1.39	0.0782	0.0802	2.53	0.0718	0.0726	1.07
		80%	0.0429	0.0456	6.37	0.0655	0.0653	**-0.35**	0.0651	0.0639	**-1.70**	0.0641	0.0632	**-1.46**	0.0790	0.0799	1.24	0.0778	0.0796	2.27	0.0711	0.0707	**-0.51**
	populationNumber ($x10^6$)	20%	8.2767	6.5413	**-20.97**	8.7521	6.8001	**-22.30**	7.7478	6.3342	**-18.25**	7.3405	6.8819	**-6.25**	6.8641	6.7753	**-1.29**	7.6700	6.4430	**-16.00**	6.3780	6.3402	**-0.59**
		50%	8.3747	7.1926	**-14.11**	8.8367	6.9774	**-21.04**	7.1711	7.0500	**-1.69**	9.2356	6.9539	**-24.71**	6.8944	6.7403	**-2.24**	7.7979	7.0100	**-10.10**	6.7265	7.0913	5.42
		80%	7.8286	6.2857	**-19.71**	8.8838	6.4304	**-27.62**	7.2562	6.4460	**-11.17**	7.0696	6.4988	**-8.07**	6.2354	6.1064	**-2.07**	6.6006	5.9568	**-9.75**	6.3400	5.9621	**-5.96**
YAGO	hasLatitude	20%	15.8762	16.1874	1.96	8.0253	8.5256	6.23	9.6542	10.7848	11.71	9.6613	10.8258	12.05	7.3028	7.8766	7.86	8.5174	10.7656	26.40	9.6053	11.7178	21.99
		50%	15.1523	15.4819	2.17	5.9231	5.8723	**-0.86**	7.7242	6.9875	**-9.54**	7.6592	7.6600	0.01	5.0203	4.6103	**-8.17**	5.4106	4.6542	**-13.98**	6.1409	6.0703	**-1.15**
		80%	15.1493	15.4370	1.90	5.7398	5.4619	**-4.84**	7.3297	6.6112	**-9.80**	6.9609	6.6623	**-4.29**	5.0551	4.4951	**-11.08**	5.2888	4.3586	**-17.59**	6.0585	4.8657	**-19.69**
	hasLongitude	20%	39.3299	37.9247	**-3.57**	20.1994	20.8493	3.22	23.8878	22.4796	**-5.90**	23.0715	22.6942	**-1.64**	15.9415	16.0364	0.60	20.9329	23.1130	10.41	21.4411	28.7336	34.01
		50%	36.3181	26.9464	**-25.80**	15.7095	17.0096	8.28	18.7299	17.6885	**-5.56**	18.8015	17.9712	**-4.42**	13.5979	13.6642	0.49	14.3521	13.2712	**-7.53**	15.9453	16.5460	3.77
		80%	36.5138	24.9506	**-31.67**	15.8009	15.5438	**-1.63**	18.7531	17.2356	**-8.09**	17.7571	17.5213	**-1.33**	13.6332	13.0070	**-4.59**	15.1582	14.1030	**-6.96**	16.2791	14.0856	**-13.47**
	hasArea ($x10^4$)	20%	13.6660	12.3963	**-9.29**	7.5554	7.5238	**-0.42**	7.9465	8.6081	8.33	9.1942	7.0575	**-23.24**	8.6687	9.2323	6.50	10.6573	10.4682	**-1.77**	12.1870	8.9230	**-26.78**
		50%	12.7663	12.3170	**-3.52**	6.1890	7.1071	14.83	6.5280	6.9359	6.25	6.8200	7.1586	4.96	6.1964	7.3155	18.06	5.9629	7.1403	19.75	7.8738	6.6341	**-15.74**
		80%	12.4752	11.9690	**-4.06**	4.6004	5.6009	21.75	6.4699	6.7573	4.44	5.9645	5.5691	**-6.63**	6.2036	6.6932	7.89	5.9180	7.6781	29.74	6.5564	7.6224	16.26

5. Conclusions and future work

In this work we have presented a much needed comprehensive study about how GNNs perform when applied together with deep embeddings. Deep embeddings have the appeal of being domain-independent and potentially able to capture latent information about the content of the graph, which has led to their extended use in a variety of tasks, including prediction of graph elements by feeding them to neural networks. Graph neural networks, 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 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 new 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.

Personally, I consider this project to be the culmination of my academic stage and the beginning of my research career, as it is the first step on my way to obtaining a doctoral degree, so much so that I have applied for a FPU grant based on this project. It has not been an easy subject to understand and work on, especially on the technical side, but I reckon that the results and conclusions obtained are of great interest, even meaning a research paper sent to an international prestigious forum as CIKM, and there is still much more work to be done. Also, having the support of the DEAL[1] research group and the fact that this work aligns with their objectives makes me look forward to continuing with it.

[1] https://deal.us.es/

6. Bibliography

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, NY, USA, 2008. ACM New York. doi: 10.1145/1376616.1376746.

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *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*, pages 2787–2795, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017. doi: 10.1109/MSP.2017.2693418. URL https://doi.org/10.1109/MSP.2017.2693418.

[4] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR 2018*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rytstxWAW.

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *ACM SIGKDD 2019*, pages 257–266, NY, USA, 2019. ACM. doi: 10.1145/3292500.3330925. URL https://doi.org/10.1145/3292500.3330925.

[6] Marc Franco-Salvador, Paolo Rosso, and Manuel Montes-y Gómez. A systematic study of knowledge graph analysis for cross-language plagiarism detection. *Information Processing & Management*, 52(4):550–570, 2016.

[7] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. A survey on knowledge graph embeddings with literals: Which model links better literal-ly? *Semantic Web*, 12(4):617–647, 2021. doi: 10.3233/SW-200404. URL https://doi.org/10.3233/SW-200404.

[8] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA*, pages 89–98. ACM, 1998. doi: 10.1145/276675.276685. URL https://doi.org/10.1145/276675.276685.

[9] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *IJCAI*, pages 1802–1808, 2017. doi: 10.24963/ijcai.2017/250. URL https://doi.org/10.24963/ijcai.2017/250.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*,

pages 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

[11] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data, 2015. URL https://arxiv.org/abs/1506.05163.

[12] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113, 2019.

[13] Fadhela Kerdjoudj and Olivier Curé. Rdf knowledge graph visualization from a knowledge extraction system. *arXiv preprint arXiv:1510.00244*, 2015.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[15] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. Incorporating literals into knowledge graph embeddings. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2019. doi: 10.1007/978-3-030-30793-6_20. URL https://doi.org/10.1007/978-3-030-30793-6_20.

[16] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *IEEE Trans. Knowl. Data Eng.*, 30(12):2257–2270, 2018. doi: 10.1109/TKDE.2018.2819980. URL https://doi.org/10.1109/TKDE.2018.2819980.

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571.

[18] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 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*, pages 5425–5434. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.576. URL https://doi.org/10.1109/CVPR.2017.576.

[19] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *ACL*, pages 4710–4723, 2019.

[20] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2014–2023. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/niepert16.html.

[21] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Communications of the ACM*, 62(8):36–43, 2019.

[22] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Philip S. Yu. Joint structure feature exploration and regularization for multi-task graph classification. *IEEE Trans. Knowl. Data Eng.*, 28(3):715–728, 2016. doi: 10.1109/TKDE.2015.2492567. URL https://doi.org/10.1109/TKDE.2015.2492567.

[23] Shirui Pan, Jia Wu, Xingquan Zhu, Guodong Long, and Chengqi Zhang. Task sensitive feature exploration and learning for multitask graph classification. *IEEE Trans. Cybern.*, 47(3):744–758, 2017. doi: 10.1109/TCYB.2016.2526058. URL https://doi.org/10.1109/TCYB.2016.2526058.

[24] Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. YAGO: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *ISWC*, volume 9982, pages 177–185, 2016. doi: 10.1007/978-3-319-46547-0_19. URL https://doi.org/10.1007/978-3-319-46547-0_19.

[25] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008. doi: 10.1609/aimag.v29i3.2157. URL https://doi.org/10.1609/aimag.v29i3.2157.

[26] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI*, volume 33, pages 3060–3067, 2019. doi: 10.1609/aaai.v33i01.33013060. URL https://ojs.aaai.org/index.php/AAAI/article/view/4164.

[27] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706. ACM, 2007.

[28] Yi Tay, Luu Anh Tuan, Minh C. Phan, and Siu Cheung Hui. Multi-task neural network for non-discrete attribute prediction in knowledge graphs. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1029–1038. ACM, 2017. doi: 10.1145/3132847.3132937. URL https://doi.org/10.1145/3132847.3132937.

[29] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66, 2015.

[30] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[31] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[32] Yanrong Wu and Zhichun Wang. Knowledge graph embedding with numeric attributes of entities. In Isabelle Augenstein, Kris Cao, He He, Felix Hill, Spandana Gella, Jamie Kiros, Hongyuan Mei, and Dipendra Misra, editors, *Proceedings of The Third Workshop on Representation Learning for NLP, Rep4NLP@ACL 2018, Melbourne, Australia, July 20, 2018*, pages 132–136. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-3017. URL https://doi.org/10.18653/v1/w18-3017.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[34] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 4805–4815, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html.

[35] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In Amir Globerson and Ricardo Silva, editors, *UAI*, pages 339–349. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/139.pdf.

[36] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *AAAI*, pages 4438–4445. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17146.

[37] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: Algorithms, applications and open challenges. In *International Conference on Computational Social Networks*, pages 79–91. Springer, 2018.

[38] Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. Question answering over knowledge graphs: question understanding via template decomposition. *Proc. of the VLDB Endowment*, 11(11):1373–1386, 2018.

[39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. doi: 10.1016/j.aiopen.2021.01.001. URL https://doi.org/10.1016/j.aiopen.2021.01.001.

A. Annexed documents

A.1. Article presented to CIKM'22

Deep embeddings and Graph Neural Networks: can context improve domain-independent predictions?

Anonymous Author(s)

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 how they obtain a contextual representation of each node that takes into account the information from surrounding nodes. However, existing work has focused on the development of GNN architectures, using basic domain-specific information about the nodes to compute embeddings. In the context 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 of the application of graph neural networks with deep embeddings of knowledge graphs remains largely unexplored. In this paper, we carry out a number of experiments to answer open research questions about how said embeddings perform when using a graph neural network. We test 7 different deep embeddings across several attribute prediction tasks in two attribute-rich datasets. We conclude that there is a significant performance improvement but it varies heavily depending on the task and deep embedding technique.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Knowledge representation and reasoning**.

KEYWORDS

Knowledge Graphs, Graph Neural Networks, Attributive embeddings, Deep graph embeddings, Machine Learning

ACM Reference Format:

Anonymous Author(s). 2022. Deep embeddings and Graph Neural Networks: can context improve domain-independent predictions?. In *Proceedings of 31st ACM International Conference on Information and Knowledge Management (CIKM'22).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph Neural Network (GNN) architectures seek to leverage the connections of a graph when it comes to making predictions about the elements of the graph [33]. To achieve this, the nodes of the graph are represented as numeric vectors called embeddings that can capture and summarize the implicit information present in them. For example, graph convolutional layers [20] 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 overall picture when it comes to, for example, making predictions about nodes. Research in this area so far has mainly focused on developing new 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 created using any domain-specific information available about the nodes; for example, when representing data from the academic research domain in which papers are nodes, the embeddings can be bag-of-words representations of the text in the papers [14]. Another example can be found in the genetic information domain, using gene positional sequences as embeddings [10].

Knowledge Graphs (KGs) have become a popular research topic as many companies such as Google, Facebook, and Amazon [21] are increasingly relying on the integrated and curated information in knowledge graphs. A considerable amount of research effort focuses on developing deep learning techniques that are able to obtain embeddings in a domain-independent way. A well-produced embedding space contains latent information about the elements it represents, and therefore can be fed to algorithms for a variety of tasks [6, 12, 13, 31, 38], such as question answering, KG completion, link prediction, or clustering, to mention a few. These techniques usually train the embeddings so that they contain the necessary information to predict the presence of edges in the graph. More recent approaches have introduced attributive embeddings [7] to take into account the information about node properties, which tend to be numeric values such as years or ages. This may be beneficial for prediction tasks in which the value of a property or the similarity with that of another node 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. While domain-specific embeddings benefit from GNNs, it is interesting to study the effect of using deep KG embeddings with GNNs because of their different nature.

This motivated us to carry out an experiment to shed some light on the feasibility of such combination. 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 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 datasets that are rich in attributes: YAGO [27] and FB15K237 [29]. The results obtained by these configurations contribute towards the state of the art by thoroughly

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM'22, October 17–22, 2022, Atlanta, GA

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

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: Section 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, Section 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 graph neural networks and node embedding techniques.

2.1 Graph Neural Networks

Back in the 1990s, 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) [33]. Some of their main drawbacks were that they are computationally costly and lack 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 [3]. 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 [37]. 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.

Graph neural networks performance might be influenced by knowledge graphs size and type, so they should be taken into consideration. KGs can be classified as following [39]:

- **Directed/undirected**: directed edges provide more information than undirected ones, which can also be seen as double-directed edges.
- **Homogeneous/heterogeneous**: heterogeneous graphs provide a type for each node and edge, adding an additional value to them.
- **Spatiotemporal/static**: on dynamic graphs, also known as spatial-temporal ones, topology or features change over time, a characteristic that needs to be properly addressed.
- **Small/large**: there is not a clear defined criteria to distinguish between a small or large graph as it is 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 [14], link or edge strength prediction [9, 19, 26], and graph level tasks such as graph classification [22, 23, 34, 36]. 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 and doing sampling or clustering may end up on losing neighbourhood information [11]. Other challenges that remain unsolved are a method for systematically finding the optimal architecture for each network, or the use of advanced embeddings that maximise the network performance.

2.2 Node embedding techniques

Knowledge graphs embeddings techniques have been widely studied recently because of its numerous possibilities. 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, such as GNNs. Consequently, It should be noted that the performance of GNNs, as deep learning algorithms, can therefore be influenced by the type of node embedding it is provided with.

Typical knowledge graph embedding approaches use distance-based scoring functions to learn embeddings, 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 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 discarded.

The challenge lies in learning embeddings taking these literals into account, which can be addressed in two ways [7]. The first option is to handle literals separately, that is, 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 about each node and this may be highly leveraged by GNNs and their message-passing step.

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 tasks that they carry out, 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.

This summary shows that all of the proposals focus on node or graph classification tasks, while the technique used to compute node embeddings is usually not a specialized one. In terms of datasets, we find that Reddit [10], PPI [10], Cora [25], and Citeseer [8], which are domain-specific, are commonly used. Other general purpose KGs that are rich in node attributes, such as YAGO [24] or FreeBase [1], are not usually considered. Additionally, there is a big heterogeneity

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

Approach	Task	Datasets	Node embeddings	Train/test/val. %
PATCHY-SAN (2016) [20]	Graph classification	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) [10]	Node classification	Citation dataset from Thomson Reuters Web of Science Core Collection	GenSim word2vec	80/14/6
		Reddit	GloVe CommonCrawl	67/23/10
	Graph classification	PPI	Gene sets and features	84/8/8
GCN (2017) [14]	Node classification	Cora	Sparse bag-of-words feature vector	9/61/30
		Citeseer	Sparse bag-of-words feature vector	7/62/31
		Pubmed	Sparse bag-of-words feature vector	4/64/32
		NELL	Sparse feature vector	12/58/30
MoNet (2017) [18]	Node classification	Cora	Sparse bag-of-words feature vector	9/61/30
		Pubmed	Sparse bag-of-words feature vector	4/64/32
GAT (2018) [30]	Node classification	Cora	Sparse bag-of-words feature vector	9/61/30
		Citeseer	Sparse bag-of-words feature vector	7/62/31
		Pubmed	Sparse bag-of-words feature vector	4/64/32
	Graph classification	PPI	50 features vector	84/8/8
GAAN (2018) [35]	Node classification	Reddit	-	67/23/10
	Graph classification	PPI	-	84/8/8
	Traffic speed forecasting (spatiotemporal graph)	METR-LA	-	70/20/10
FastGCN (2018) [4]	Node classification	Cora	-	45/37/18
		Pubmed	-	92/5/3
		Reddit	-	65/25/10
ClusterGCN (2019) [5]	Node classification	PPI	-	82/7/11
		Reddit	-	66/24/10
		Amazon	-	28/72/-
		Amazon2M	Reduced bag-of-words feature vector	70/30/-

in the train/test/validation splits, since they are very dependent on the characteristics of each dataset.

3 OUR WORK

In this section we describe our contributions: in Section 3.1 we define the goals of our work and our research questions, while Section 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 any report on how this type of 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 being 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, we used a baseline feedforward neural network and a standard graph neural network. The feedforward one, Figure 1, is composed by a series of five "dense blocks" with skip connections, except the first one; and a final dense layer which outputs logits. Each "dense block" is composed by two groups of batch normalisation, dropout and dense layers. The input of this network is the matrix of the embeddings of the nodes.

The standard graph neural network, shown in Figure 2, is composed of two dense blocks that perform preprocessing and postprocessing functions and between which we have placed two graph convolution layers. These are made up of a preparation "dense block", a message aggregation layer and a node embedding updating layer, as well as a skip connection over them. Finally, the output is given by a dense layer. The input of this network is a matrix of the embeddings of the nodes and the list of graph edges.

Note that, since the specific architectures are not the focus of this paper, we have relied on some basic state-of-the-art architectures in both cases.

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 [29] and a reduced version of YAGO [27] which only contains nodes with at least one attribute and have at least ten connected edges. Additionally, we also included Cora [25], a citations 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 these results to those obtained by domain-specific embeddings.

Algorithm 1: Experimental script

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)
```

In terms of embedding techniques, we selected the ones proposed by Gesese et al. [7], which employ textual or/and numerical attributes and have an accessible implementation: ASNE [16], LiteralE-ComplEx [15], LiteralE-DistMult [15], MTKGNN [28] and TransEA [32]. In addition, we included two well-known non - attributive embeddings as baselines: Trans-E [2] and Trans-R [17].

The node attribute prediction tasks we selected, shown in Table 2, include a variety of 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 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 the 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%, 50% and 20% of examples for training. Additionally, we executed every task ten times in order to better assess the overall obtained performance.

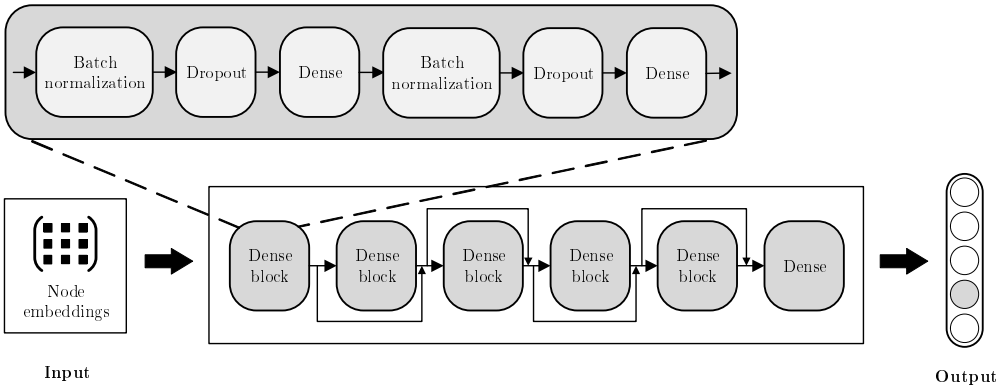

Figure 1: Baseline neural network architecture

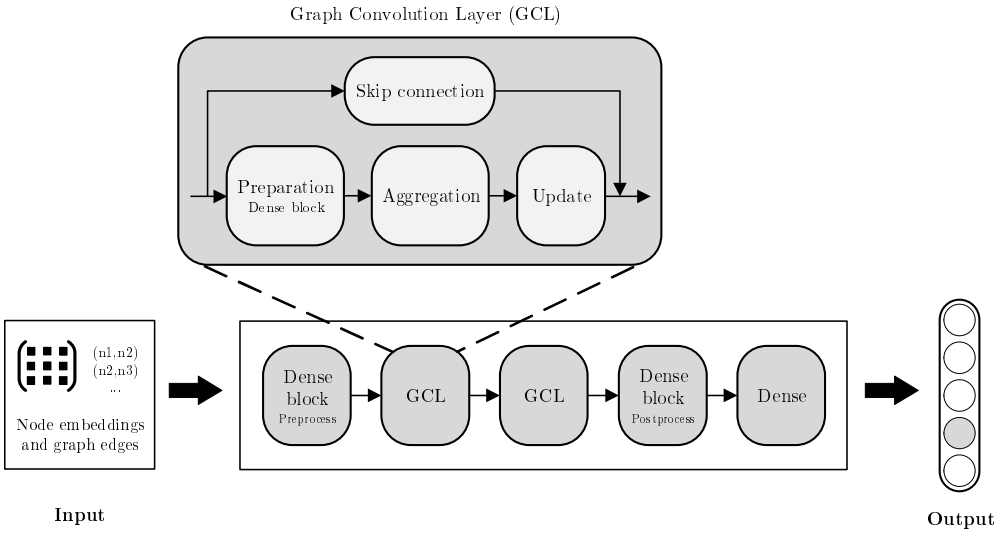

Figure 2: Standard graph neural network architecture

To perform the experiments, we designed a script, 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 graph neural network. The datasets, tasks and embedding techniques information are given in a dictionary and data itself is contained in different files. The output consists of the prediction evaluation metrics values for each setup.

4.2 Experimental Results

Tables 3 and 4 show the results of the experiments that we conducted.

Table 3 collects the reference results of the Cora dataset, in terms of accuracy. Figure 3 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 4 contains the mean absolute error (MAE) 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.

4.3 Research questions

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

Table 2: Prediction tasks

KG	Task	Nodes	Number of different values	Values range
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 3: GNN accuracy on Cora

	NN	GNN	ΔMAE%
20%	64.7431	67.5637	4.36
50%	73.9569	81.0633	9.61
80%	76.6745	84.8474	10.66

Q1: When using a GNN, does the kind of deep embedding being 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 in GNN column in Table 4. However, we haven't 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.

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 4, showing the percentage difference between GNNs and NN with bold numbers when GNN outperforms the regular NN in a majority of cases. However, some tasks tend to leverage context information and thus, the improvement is greater in them, like *locationArea* or *hasLongitude*, see Figure 3.

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 for attributive embeddings than for non-attributive ones with the exception of TransEA. As can be seen in Table 4, GNNs lead to an improvement in roughly 50% of cases when using the TransR, TransE, or TransEA embeddings, but when using ASNE, LiteralE-ComplEx, LiteralE-DistMult, and MTKGNN there is an improvement in 66% to 75% of cases.

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 Cora experiments in Table 3 show an accuracy improvement between 5% and 10%. The results in Table 4, however, vary significantly between tasks.

It is clear than in some tasks, like $filmRating$ or $personHeight$, there is little information in the context of a node that could help improve the prediction, while for example, the area of a country could be easier to predict based on contextual information. In this case, GNNs lead to 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 stunning performances. Other tasks such as $populationNumber$ seem to benefit more from the contextual information provided by the GNNs, reaching improvements of more than 25%.

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

As can be seen in Figure 3, 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.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a much needed comprehensive study about how GNNs perform when applied together with deep embeddings. Deep embeddings have the appeal of being domain-independent and potentially able to capture latent information about the content of the graph, which has led to their extended use in a variety of tasks, including prediction of graph elements by feeding them to neural networks. Graph neural networks, 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

Table 4: Embedding techniques MAE comparison

		Train size	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%	9.5737	9.1583	**-4.34**	16.4734	15.8074	**-4.04**	16.5041	15.7854	**-4.35**	16.2449	15.5837	**-4.07**	15.9838	15.9753	**-0.05**	16.4583	16.3101	**-0.90**	16.1969	16.0243	**-1.07**
		50%	5.9916	6.2949	5.06	15.4682	15.1325	**-2.17**	14.8965	15.5231	4.21	15.2264	14.9973	**-1.50**	15.4585	15.7934	2.17	15.5508	15.6815	0.84	14.7528	15.2734	3.53
		80%	5.5274	5.0774	**-8.14**	14.5767	14.6457	0.47	14.7380	14.7621	0.16	14.8746	14.6467	**-1.53**	15.4188	15.7456	2.12	15.2660	15.3452	0.52	14.5562	14.8972	2.34
	locationArea ($x10^7$)	20%	12.6237	9.8694	**-21.82**	9.8040	10.8875	11.05	10.8566	8.8964	**-18.06**	10.1552	9.0006	**-11.37**	9.5080	9.0846	**-4.45**	9.0743	6.7642	**-25.46**	7.0607	6.8754	**-2.62**
		50%	9.9147	7.4608	**-24.75**	9.4036	8.0787	**-14.09**	9.8365	8.2094	**-16.54**	8.7479	8.2774	**-5.38**	11.7681	8.5035	**-27.74**	9.0598	9.0088	**-0.56**	7.8769	10.2782	30.49
		80%	9.2169	8.3831	**-9.05**	10.0565	8.4622	**-15.85**	8.9004	8.5203	**-4.27**	9.4371	8.6090	**-8.77**	9.4937	8.0584	**-15.12**	8.5825	8.1500	**-5.04**	9.2707	8.6887	**-6.28**
	personHeight	20%	0.0527	0.0577	9.46	0.0709	0.0687	**-3.06**	0.0704	0.0681	**-3.17**	0.0697	0.0713	2.23	0.0816	0.0819	0.38	0.0811	0.0816	0.63	0.0785	0.0794	1.17
		50%	0.0455	0.0481	5.63	0.0668	0.0661	**-0.94**	0.0665	0.0654	**-1.58**	0.0653	0.0656	0.41	0.0797	0.0808	1.39	0.0782	0.0802	2.53	0.0718	0.0726	1.07
		80%	0.0429	0.0456	6.37	0.0655	0.0653	**-0.35**	0.0651	0.0639	**-1.70**	0.0641	0.0632	**-1.46**	0.0790	0.0799	1.24	0.0778	0.0796	2.27	0.0711	0.0707	**-0.51**
	populationNumber ($x10^6$)	20%	8.2767	6.5413	**-20.97**	8.7521	6.8001	**-22.30**	7.7478	6.3342	**-18.25**	7.3405	6.8819	**-6.25**	6.8641	6.7753	**-1.29**	7.6700	6.4430	**-16.00**	6.3780	6.3402	**-0.59**
		50%	8.3747	7.1926	**-14.11**	8.8367	6.9774	**-21.04**	7.1711	7.0500	**-1.69**	9.2356	6.9539	**-24.71**	6.8944	6.7403	**-2.24**	7.7979	7.0100	**-10.10**	6.7265	7.0913	5.42
		80%	7.8286	6.2857	**-19.71**	8.8838	6.4304	**-27.62**	7.2562	6.4460	**-11.17**	7.0696	6.4988	**-8.07**	6.2354	6.1064	**-2.07**	6.6006	5.9568	**-9.75**	6.3400	5.9621	**-5.96**
YAGO	hasLatitude	20%	15.8762	16.1874	1.96	8.0253	8.5256	6.23	9.6542	10.7848	11.71	9.6613	10.8258	12.05	7.3028	7.8766	7.86	8.5174	10.7656	26.40	9.6053	11.7178	21.99
		50%	15.1523	15.4819	2.17	5.9231	5.8723	**-0.86**	7.7242	6.9875	**-9.54**	7.6592	7.6600	0.01	5.0203	4.6103	**-8.17**	5.4106	4.6542	**-13.98**	6.1409	6.0703	**-1.15**
		80%	15.1493	15.4370	1.90	5.7398	5.4619	**-4.84**	7.3297	6.6112	**-9.80**	6.9609	6.6623	**-4.29**	5.0551	4.4951	**-11.08**	5.2888	4.3586	**-17.59**	6.0585	4.8657	**-19.69**
	hasLongitude	20%	39.3299	37.9247	**-3.57**	20.1994	20.8493	3.22	23.8878	22.4796	**-5.90**	23.0715	22.6942	**-1.64**	15.9415	16.0364	0.60	20.9329	23.1130	10.41	21.4411	28.7336	34.01
		50%	36.3181	26.9464	**-25.80**	15.7095	17.0096	8.28	18.7299	17.6885	**-5.56**	18.8015	17.9712	**-4.42**	13.5979	13.6642	0.49	14.3521	13.2712	**-7.53**	15.9453	16.5460	3.77
		80%	36.5138	24.9506	**-31.67**	15.8009	15.5438	**-1.63**	18.7531	17.2356	**-8.09**	17.7571	17.5213	**-1.33**	13.6332	13.0070	**-4.59**	15.1582	14.1030	**-6.96**	16.2791	14.0856	**-13.47**
	hasArea ($x10^4$)	20%	13.6660	12.3963	**-9.29**	7.5554	7.5238	**-0.42**	7.9465	8.6081	8.33	9.1942	7.0575	**-23.24**	8.6687	9.2323	6.50	10.6573	10.4682	**-1.77**	12.1870	8.9230	**-26.78**
		50%	12.7663	12.3170	**-3.52**	6.1890	7.1071	14.83	6.5280	6.9359	6.25	6.8200	7.1586	4.96	6.1964	7.3155	18.06	5.9629	7.1403	19.75	7.8738	6.6341	**-15.74**
		80%	12.4752	11.9690	**-4.06**	4.6004	5.6009	21.75	6.4699	6.7573	4.44	5.9645	5.5691	**-6.63**	6.2036	6.6932	7.89	5.9180	7.6781	29.74	6.5564	7.6224	16.26

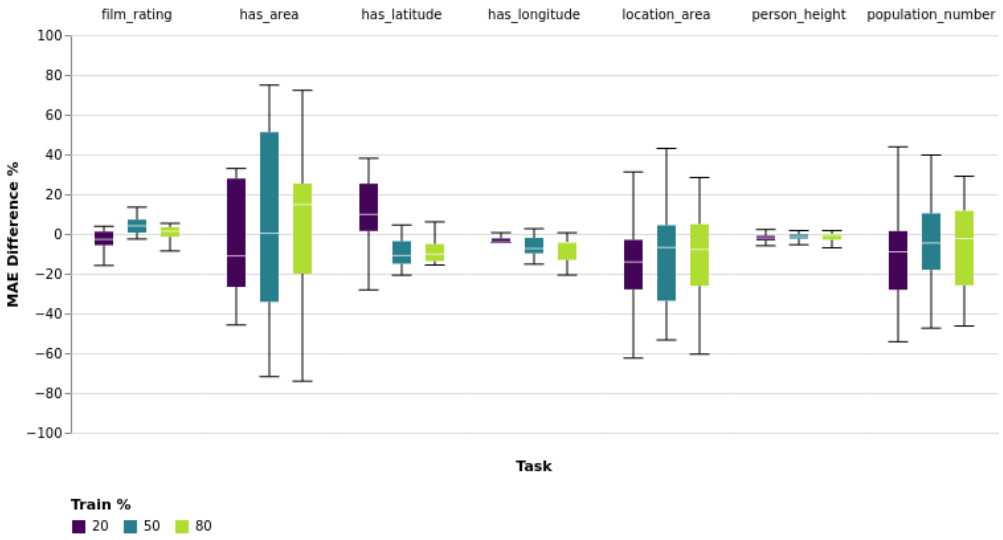

Figure 3: GNN vs. NN MAE percentage difference

performance obtained by deep embeddings has significant potential as 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 new 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.

REFERENCES

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*. ACM New York, NY, USA, 1247–1250. https://doi.org/10.1145/1376616.1376746

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* 34, 4 (2017), 18–42. https://doi.org/10.1109/MSP.2017.2693418

[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR 2018*. OpenReview.net. https://openreview.net/forum?id=rytstxWAW

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *ACM SIGKDD 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, NY, USA, 257–266. https://doi.org/10.1145/3292500.3330925

[6] Marc Franco-Salvador, Paolo Rosso, and Manuel Montes-y Gómez. 2016. A systematic study of knowledge graph analysis for cross-language plagiarism detection. *Information Processing & Management* 52, 4 (2016), 550–570.

[7] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. 2021. A survey on knowledge graph embeddings with literals: Which model links better literal-ly? *Semantic Web* 12, 4 (2021), 617–647. https://doi.org/10.3233/SW-200404

[8] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 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, 89–98. https://doi.org/10.1145/276675.276685

[9] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach. In *IJCAI*. 1802–1808. https://doi.org/10.24963/ijcai.2017/250

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*. 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[11] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. https://doi.org/10.48550/ARXIV.1506.05163

[12] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 105–113.

[13] Fadhela Kerdjoudj and Olivier Curé. 2015. RDF knowledge graph visualization from a knowledge extraction system. *arXiv preprint arXiv:1510.00244* (2015).

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[15] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2019. Incorporating Literals into Knowledge Graph Embeddings. In *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)*, Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon (Eds.). Springer, 347–363. https://doi.org/10.1007/978-3-030-30793-6_20

[16] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed Social Network Embedding. *IEEE Trans. Knowl. Data Eng.* 30, 12 (2018), 2257–2270. https://doi.org/10.1109/TKDE.2018.2819980

[17] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 2181–2187. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571

[18] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 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, 5425–5434. https://doi.org/10.1109/CVPR.2017.576

[19] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In *ACL*. 4710–4723.

[20] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs. In *ICML (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 2014–2023. http://proceedings.mlr.press/v48/niepert16.html

[21] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM* 62, 8 (2019), 36–43.

[22] Shirui Pan, Jia Wu, Xingquan Zhu, Guodong Long, and Chengqi Zhang. 2017. Task Sensitive Feature Exploration and Learning for Multitask Graph Classification. *IEEE Trans. Cybern.* 47, 3 (2017), 744–758. https://doi.org/10.1109/TCYB.2016.2526058

[23] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Philip S. Yu. 2016. Joint Structure Feature Exploration and Regularization for Multi-Task Graph Classification. *IEEE Trans. Knowl. Data Eng.* 28, 3 (2016), 715–728. https://doi.org/10.1109/TKDE.2015.2492567

[24] Thomas Rebele, Fabian M. Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames. In *ISWC*, Vol. 9982. 177–185. https://doi.org/10.1007/978-3-319-46547-0_19

[25] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106. https://doi.org/10.1609/aimag.v29i3.2157

[26] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In *AAAI*, Vol. 33. 3060–3067. https://doi.org/10.1609/aaai.v33i01.33013060

[27] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*. ACM, 697–706.

[28] Yi Tay, Luu Anh Tuan, Minh C. Phan, and Siu Cheung Hui. 2017. Multi-Task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li (Eds.). ACM, 1029–1038. https://doi.org/10.1145/3132847.3132937

[29] Kristina Toutanova and Danqi Chen. 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*. 57–66.

[30] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*. OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[31] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[32] Yanrong Wu and Zhichun Wang. 2018. Knowledge Graph Embedding with Numeric Attributes of Entities. In *Proceedings of The Third Workshop on Representation Learning for NLP, Rep4NLP@ACL 2018, Melbourne, Australia, July 20, 2018*, Isabelle Augenstein, Kris Cao, He He, Felix Hill, Spandana Gella, Jamie Kiros, Hongyuan Mei, and Dipendra Misra (Eds.). Association for Computational Linguistics, 132–136. https://doi.org/10.18653/v1/w18-3017

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[34] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *NeurIPS*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 4805–4815. https://proceedings.neurips.cc/paper/2018/hash/e77dbaf6759253c7c6d0efc5690369c7-Abstract.html

[35] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *UAI*, Amir Globerson and Ricardo Silva (Eds.). AUAI Press, 339–349. http://auai.org/uai2018/proceedings/papers/139.pdf

[36] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 4438–4445. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17146

[37] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. 2018. Graph convolutional networks: Algorithms, applications and open challenges. In *International Conference on Computational Social Networks*. Springer, 79–91.

[38] Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. 2018. Question answering over knowledge graphs: question understanding via template decomposition. *Proc. of the VLDB Endowment* 11, 11 (2018), 1373–1386.

[39] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001