A Method to Predict Semantic Relations on Artificial Intelligence Papers

1st Francisco Andrades
Department of Informatics
Federico Santa María Technical University
Santiago, Chile
francisco.andrades@sansano.usm.cl

2nd Ricardo Ñanculef
Department of Informatics
Federico Santa María Technical University
Santiago, Chile
jnancu@inf.utfsm.cl

***Abstract*—Predicting the emergence of links in large evolving networks is a difficult task with many practical applications. Recently, the Science4cast competition has illustrated this challenge presenting a network of 64.000 AI concepts and asking the participants to predict which topics are going to be researched together in the future. In this paper, we present a solution to this problem based on a new family of deep learning approaches, namely Graph Neural Networks.**

The results of the challenge show that our solution is competitive even if we had to impose severe restrictions to obtain a computationally efficient and parsimonious model: ignoring the intrinsic dynamics of the graph and using only a small subset of the nodes surrounding a target link. Preliminary experiments presented in this paper suggest the model is learning two related, but different patterns: the absorption of a node by a sub-graph and union of more dense sub-graphs. The model seems to excel at recognizing the first type of pattern.

***Index Terms*—Link Prediction, Graph Learning, Deep Learning, Graph Neural Networks, Science4cast.**

I. Introduction

In recent years, there has been an increasing interest on learning from graph structured data [1]. As graphs can be used to represent data in a diverse range of disciplines, problems like graph or node classification have a wide range of applications.

One of the main topics in graph learning is the problem of *Link Prediction*: given a graph at some time t, the goal is to predict the formation of new links in future states of the graph. Many approaches have been developed throughout the years. Graph Kernels [2], Node embeddings [3] and traditional Machine Learning methods [4] have all been widely used with good performance. More recently, Deep Learning has allowed to achieve new, improved results [5].

The Science4Cast challenge seeks to encourage research on Link Prediction [4]. It proposes a new benchmark dataset that contains semantic information on scientific papers and can be used to forecast emerging research topics on the Artificial Intelligence field.

The solution presented in this paper is based on a new, specialized family of Neural Networks, namely Graph Neural Networks (GNN) [6]. Experimental results show that the method can achieve competitive performance using only a subset of the nodes surrounding a target link and ignoring valuable information about the dynamics of the network (which is used by other methods). These simplifications lead to a solution with *low memory usage* that can be used without specialized hardware.

Preliminary experiments we present in this paper suggest the model is learning two related, but different patterns. The first is a kind of *cold-start* scenario in which a node is absorbed by a larger sub-graph. The second is the union of two dense sub-graphs. The model seems to excel at recognizing the asymmetric cold-start case.

The rest of this paper is structured as follows. In the next section we provide an overview of the Science4cast challenge. In Section III, we introduce the problem of link prediction providing a brief summary of the background used to obtain our method and the technique used as baseline. In Section IV we present our solution, stressing the restrictions we had to impose to obtain an efficient method. Section V presents numerical results on the test data made available to the participants of the challenge. Section VI concludes the work with some final remarks.

II. The Challenge

The massive amount of scientific papers (and its accelerated growth) allows for a detailed analysis on the behavior and evolution of scientific disciplines with an unprecedented level of sophistication [4] [7].

This is the motivation behind the Science4Cast Challenge, that seeks to predict emerging trends in AI.

To achieve this, a semantic network obtained from an intelligent analysis on a large corpus of AI papers is presented to the community [4]. In the resulting graph, nodes represents semantic concepts relevant to the field and edges represents the co-occurrence of two concepts in the same paper.

The challenge is as follows. Given the graph formed until the year 2017, predict the probability of link formation for different pairs of nodes by the year 2020. This is equivalent to predict which pairs of concepts are going to be researched together during the upcoming 3 years.

From a technical perspective, the challenge can be viewed as a Link Prediction problem on a partially dynamic graph: new edges can be added between existing vertices on a temporal dimension.

A number of useful applications can be derived from this challenge: from an assistant system which helps researchers to choose their subjects, to the meta-study of science dynamics and evolution; passing through novel approaches to the link prediction problem.

A. Dataset

The network given for the challenge contains information about the co-occurrence of concepts in the AI field since 1994 until 2017. It has around 64000 nodes, each representing a concept addressed in a paper of AI. A list of edges and their creation date (discretized in days, since 01/01/1990) is used to represent the network. There are 7652945 edges formed by the end of 2017. A list of $1E6$ vertex pairs without links by 2017 is designated to make the predictions for 2020. These potential links don't have any restrictions and were chosen completely at random.

An older snapshot of the network is given for testing/validation. A list of $1E6$ vertex pairs without links by the end of 2014 is designated as a standardized test set to make the predictions for 2017. The ground-truth for these test pairs is explicitly provided. Most of the results reported in this paper were obtained on this standardized test set.

			# of degree zero nodes	
	# Nodes	# Edges	1	2
2014	64719	2278611	49.4%	30.9%
2017	64719	7652945	31.8%	3.9%

TABLE I
DESCRIPTIVE STATISTICS OF THE STANDARDIZED TEST SET (2014) AND CHALLENGE SET (2017): NUMBER OF NODES, NUMBER OF EDGES, PERCENTAGE OF TEST PAIRS IN WHICH EXACTLY 1 NODE HAS DEGREE 0, PERCENTAGE OF TEST PAIRS IN WHICH BOTH NODES HAVE DEGREE 0

The semantic meaning of each node is hidden for the participants. In fact, it is missing any type of feature information about nodes and edges. Therefore, only the topology can be used to address the challenge. A notorious problem with this is that there is a significant amount of test pairs where at least one of the nodes has degree 0, which means that there is no information about their relationships with other nodes. This is similar to the cold-start problem in collaborative filtering, a problem that could be addressed using additional metadata.

For this paper, potential links will be categorized under 3 types: 1) Type 0: Links where both nodes has a nonzero degree, 2) Type 1: Links where exactly 1 node has a nonzero degree and 3) Type 2: Links where both nodes have a degree of 0.

It is worth noting that some edges can be repeated if the same pair of concepts is found in different papers, which results in a weighted undirected graph.

III. BACKGROUND

A variety of methods have been investigated to learn from graph data. Graph kernels, Node embeddings and traditional Machine Learning methods have been widely used. Recently, Deep Learning has allowed for breakthroughs on different areas of graph learning.

A. Machine Learning based on Feature Engineering

In this approach, a feature vector is first designed using e.g. metrics from graph theory such as degree, CN, and Katz index. Then, a traditional supervised algorithm is trained. The method in [4] is proposed in the Science4cast Challenge as a competitive baseline referred to as *MK's Baseline*. The feature vector was designed using 15 metrics that captures information about a pair of nodes along the temporal dimension: Degree of both nodes in the current year and previous two years, total number of shared neighbours of each node in the current year and previous two years, number of shared neighbours in current year an previous two years. A 2-layer Fully Connected Neural Network is then trained on the representation.

A modified version of the MK's Baseline is used in this paper to compare the results on the standardized test set. All the Type 2 vertex pairs were excluded from the input data and were assigned a prediction 0.

B. Graph Neural Networks

Graph Neural Networks (GNN) are a specialized family of Neural Networks that can receive the graph directly as an input [8] [9] [10] [11]. Based on message passing mechanisms, the main idea is to learn a latent representation for each node aggregating the information about the neighbours using learnable parameters. As GNN grow in popularity, many layers have been proposed to implement this idea. A GCN is characterized by having a layer that resembles a convolution operation.

a) GCN Layer: : This is the standard layer for graph convolution operation [8]. For a given undirected graph G with N nodes, adjacency matrix $A \in R^{N\times N}$, and node features matrix $X \in R^{N\times c}$, the layer computes the following propagation

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}),$$

where $H^{(l)} \in R^{N\times D}$ is the matrix of activations of layer l ($H^0 = X$), $\tilde{A} = A + I_N$ is the adjacency matrix of the graph G with added self connections, I_N is the $N \times N$ identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $W^{(l)}$ is a layer-specific trainable weight matrix, and $\sigma(\cdot)$ is a non linear activation function (usually $\tanh(\cdot)$).

Many specialized architectures based on graph convolutional layers for specific problems have been proposed. In [12], an architecture for graph classification was presented, namely DGCNN. In a DGCNN, a stack of graph convolutional layers is used to extract latent representations of the nodes. A novel SortPooling layer takes as an input the concatenation of every GCN layer output, sorts the nodes representations in a consistent order and standardizes the output to a fixed size $k \times \sum c_i$, where c_i is the number of channels of the graph convolutional layer i. The parameter k filter the number of nodes considered in the layer's output, therefore, should be fine tuned accordingly.

A stack of 1D convolutional layers consumes the flattened output of the SortPooling layer, and a few fully-connected layers completes the model.

b) Link prediction using GNN: For link prediction, [13] proposed the SEAL framework. Let's define:

- The *h-hop enclosing sub-graph*. Given a graph $G = (V, E)$ and two nodes $(x, y) \in V$, the h-hop enclosing sub-graph for (x, y) is the subgraph $G^h_{x,y}$ induced from G by the set of nodes $\{i \mid d(i, x) \leq h \text{ or } d(i, y) \leq h\}$, where $d(a, b)$ denotes the length of the path from $a \in V$ to $b \in V$.
- A *h-order heuristic*. A heuristic score of node similarity which requires knowledge up to h-hop neighborhood of the target nodes.

Using these definitions, its easy to prove the following theorem.

Theorem 1. Any h-order heuristic for (x, y) can be accurately calculated from $G^h_{x,y}$.

Moreover, they propose a novel *γ-decaying heuristic theory* that unifies a wide range of heuristics in a single framework. It can be proved that any high-order γ-decaying heuristic can be approximated by the *h-hop enclosing sub-graph* with an error that, under certain conditions, decreases at least exponentially with h. Lastly, they demonstrate that most well-known heuristics for link prediction can be represented as *γ-decaying heuristics*.

According to these results, the SEAL framework addresses a link prediction problem as the task of classifying the sub-graphs enclosing the target nodes. This requires 3 stages:

- Enclosing sub-graphs extraction.
- Node information matrix construction X.
- GNN training.

Node Information Matrix Construction. This step is often crucial for training an accurate link prediction model. The node information matrix X in SEAL usually accommodate three types of features: explicit node features, node embeddings, and structural labels. Structural labels encode the role of each node in the topology. The center nodes x and y are the target nodes between which the link is located. Nodes with different relative positions to the center have different structural importance to the link.

[13] proposed the *Double-Radius Node Labeling (DRNL)* method $f_l : V \to N$, which assigns an integer label $f_l(i)$ to every node i in the enclosing sub-graph. $f_l(i)$ takes the form

$$f_l(i) = 1 + min(d_x, d_y) + (\frac{d}{2})[(\frac{d}{2}) + (d \,\%\, 2) - 1]\,,$$

where $d_x := d(i, x)$, $d_y := d(i, y)$, $d := d_x + d_y$, $(\frac{d}{2})$ and $(d\%2)$ are the integer quotient and remainder of d divided by 2, respectively.

IV. Proposed Method

There is a variety of decisions to be made on how the challenge should be posed. The method presented in this paper has 2 main stages: 1) Data usage strategy, and 2) SEAL - DGCNN training.

A. Data Usage Strategy

As the challenge's goal is to use a graph formed until a determined year to predict the existence of a potential link 3 years in the future, it makes sense to prepare training data with exactly the same condition. For a given dataset, let's define:

- *Input year: The last year until the graph is formed.*
- *Target year: The year for which is required to make predictions.*

In the challenge, $Target\ year = Input\ year + 3$. For a given pair $(Input\ year, Target\ year)$ the training data will be created using:

- $Input\ year\ training = Input\ year - 3$.
- $Target\ year\ training = Input\ year$.

Therefore, the training inputs for the model will include node pairs (a, b) for which no links exist in the graph by the end of $input\ year\ training$. The training labels/targets will be $+1$ for node pairs for which a link has been observed by $target\ year\ training$ and 0 for node pairs for which no link has been observed by $target\ year\ training$.

The training pairs (a, b) were chosen almost at random. There were only 2 restrictions made:

- The frequencies of the two classes were forced to be roughly the same, that is, an stratified sampling was applied.
- There was a restriction for considering a training pair as eligible: as there is absolutely no information about nodes with degree 0, only type 1 or type 2 pairs (a, b) were selected.

It is worth noting that the proposed method uses only the final state of the graph as an static input for the network, without considering any form of information on the temporal dimension.

B. SEAL - DGCNN Training

The SEAL Framework based on a DGCNN network is proposed to predict link formation, i.e., for each input pair, the *h-hop enclosing sub-graph* is extracted and that graph is classified by a DGCNN. For computational constraints, only *1-hop enclosing sub-graphs* were used and 65% of it's nodes were dropped at random.

For each sub-graph, the feature matrix X is constructed as a one-hot encoding of the DRNL node labeling method described in the previous section.

The DGCNN architecture used for the challenge consists of 4 stacked GCN Layers with 128 channels/filters each, a 1 channel GCN Layer, a SortPooling Layer, 2 1D Convolutional Layers with 64 and 128 channels respectively, a MaxPooling Layer, and 2 1D Convolutional Layers with 128 and 64 channels respectively, followed by 3 fully-connected layers with 256, 128 and 1 neuron each.

The value of k for the Sort Pooling Layer changes according to the dataset. For the standardized test set, $k = 200$ was used. For the challenge dataset, $k = 400$ was used. This value could be fine tuned to improve the results.

Every GCN Layer uses $tanh(\cdot)$ as an activation function. All the other layers uses $relu(\cdot)$, except for the last one which uses $sigmoid(\cdot)$.

The model was trained using *binary cross-entropy* as a loss function and *NAdam* as the optimizer. The learning rate used was $1E-5$ with a batch Size of 1.

The solution was implemented in TensorFlow, using Keras and the Spektral library [14].

The implementation can be found in: https://github.com/franciscoandrades

V. Experiments and Results

The proposed method is compared with the baseline described in Section III using the AUC score on 3 different scenarios: 1) Standardized test set, 2) Only Type 1 links of the standardized test set (Test†) and 3) Challenge Leaderboard. The experimental results are summarized in Table 2.

	AUC		
	Test	Test†	Challenge
MK's Baseline	0.732	0.654	0.879
This work	0.805	0.775	0.877

TABLE II
COMPARISONS OF OUR METHOD AND MK'S BASELINE ON 3 SCENARIOS: 1) TEST: STANDARDIZED TEST SET, 2) TEST†: ONLY TYPE 1 LINKS ON STANDARDIZED TEST SET AND 3) CHALLENGE: CHALLENGE LEADERBOARD

We can see that the method presented in this work is competitive against MK's Baseline. In our opinion, this is an interesting result as the proposed approach does not explicitly use temporal features or a model (e.g. RNN) devised to precisely learn the dynamics of the graph in the temporal dimension. Moreover, for computational reasons, the method employs only *1 hop enclosing sub-graphs* and 35% of the nodes enclosing a node pair, resulting in a low memory solution.

One of the most interesting results is the AUC Score on Type 1 node pairs. These cases correspond to a kind of cold-start scenario, in which one of the nodes does not have neighbours. They may represent the situation in which an emerging concept starts to be researched by the community (positive example) or a concept that is proposed in the literature but fails to be accepted (negative example). As expected, both the proposed method and the baseline reduce their accuracy on these asymmetric cold-start cases. However, our method looks much more robust in this scenario, increasing its advantage over the baseline. This ability is useful in a range of applications.

We hypothesize that the model is learning to solve two different problems, related to the different types of node pairs it can receive as input (please see III). Type 0 examples corresponds indeed to a regular link prediction problem in which we have 2 relatively large sub-graphs around the target nodes. We refer to this prediction as an *union* problem. A Type 1 link prediction instead, takes a large sub-graph and an isolated node as an inputs. As there is no topology information about the isolated node, it can be interpreted as irrelevant to the output. Therefore, in this case, the task of link prediction can be formulated as predicting the probability that has the sub-graph of absorbing a new, arbitrary node in a particular position of its topology. We refer to this prediction as a *node absorption* problem.

To explore this hypothesis, we study whether the activations of our trained network show evidence of two different learned functions. Our experiments where made on the output of the SortPooling layer as it has a fixed size, consistent order and captures the totality of the latent representations computed by the GCN layers. The 100 best predicted training examples for each label where used for the forward pass to extract the activations. Then, five of them were randomly selected for visualization.

Figures 1 and 2 show the activations triggered by Type 0 and Type 1 positive examples (label 1). Figures 3 and 4 show the activations triggered by Type 0 and Type 1 negative examples (label 0). It appears that the model indeed produces similar activations on positive examples of the same type of link prediction problem (*union* vs *node absorption*) but produces very different activations on examples of a different type of link prediction problem. The same can be observed as regards negative examples of the two types of link prediction problem.

In Figures 5 and 6 we use t-SNE embeddings to visualize the SortPooling activations for examples with label 1 and 0, respectively. It appears that the examples of the same type form differentiated clusters even if they have the same label. For example, in Figure 6, a Type 0 link that have both nodes with degree 1 is still properly differentiated. This result suggests that the reason of the differentiation does not regard the complexity of the sub-graphs, but the type of the problem addressed (*union* vs *node absorption*).

Altogether, results in this preliminary study suggest it is worth investigating more how the model deals with two substantially different phenomena of link emergence in semantic networks.

VI. Conclusions and Final Remarks

In this paper we presented a solution to the Science4cast competition. Building on recent advances in Graph Neural Networks (GNN), we approached the task as a graph classification problem. Given a node pair, we compute the sub-graphs enclosing the nodes and use stack of graph convolutional layers to predict the emergence of a link in the future. For the sake of efficiency, the model was trained by drawing input data from a single outdated state of the network (2014). Moreover, to reduce memory requirements, a small subset of nodes in the neighbour of the target link was presented to the neural net.

The results of the challenge show that despite the strong restrictions we imposed to obtain a parsimonious model, the solution is competitive and scores 0.877 AUC in the public leaderboard. In this paper, we also presented preliminary results that suggest that our solution is more robust to an asymmetric cold start situation in which one node does not

Fig. 1. SortPooling activations of 5 training examples of Type 0 with label 1

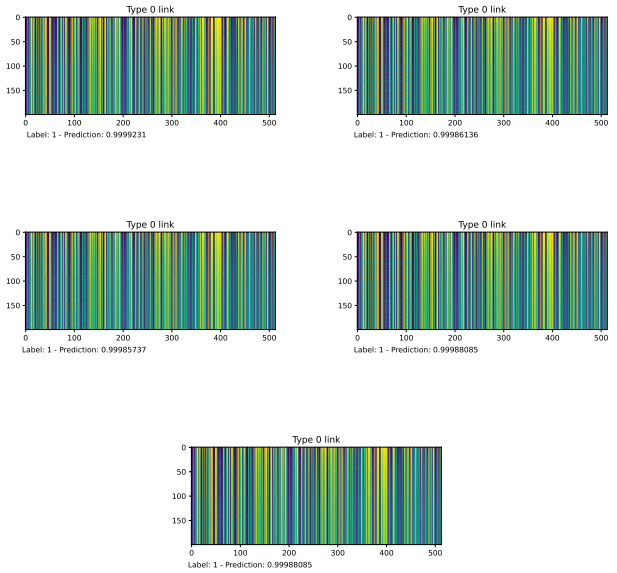
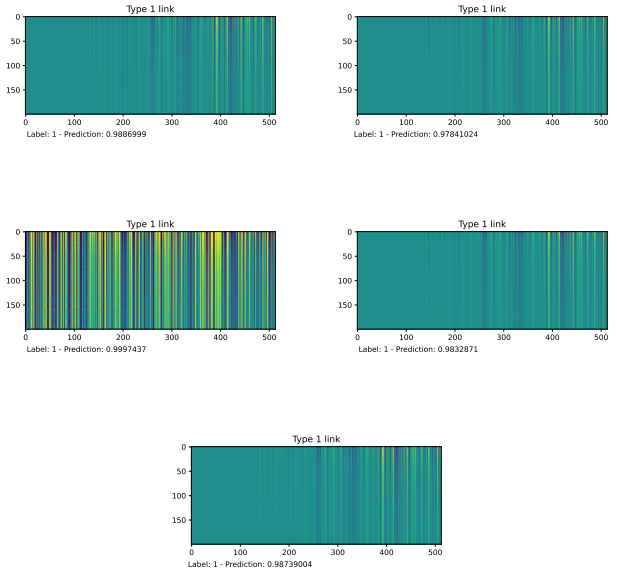


Fig. 2. SortPooling activations of 5 training examples of Type 1 with label 1 (randomly chosen).



have neighbours. In these cases, the solution increased its advantage over the baseline. As this scenario represents the situation in which an emerging concept starts to be researched by the community or fails to be accepted, we believe it is worth conducting more experiments (e.g. visualization strategies) that help to understand the patterns learnt by the model.

In future work, we plan to equip the model with sampling strategies which exploit the edge weights available in the dataset. We also plan to use the GNN trained for the

Fig. 3. SortPooling activations of 5 training examples of Type 0 with label 0 (randomly chosen).

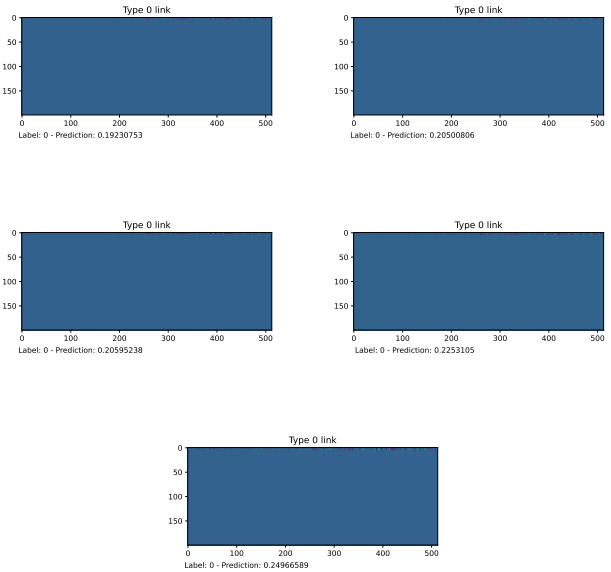


Fig. 4. SortPooling activations of 5 training examples of Type 1 with label 0 (randomly chosen).

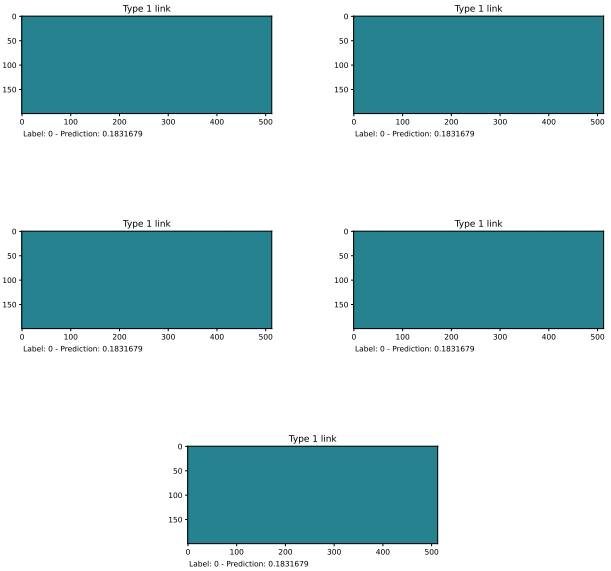


Fig. 5. t-SNE embeddings on training examples with label 1. First row: colors assigned according to node pair Type (see Section III). Second row: colors assigned according to maximum degree of the two nodes.

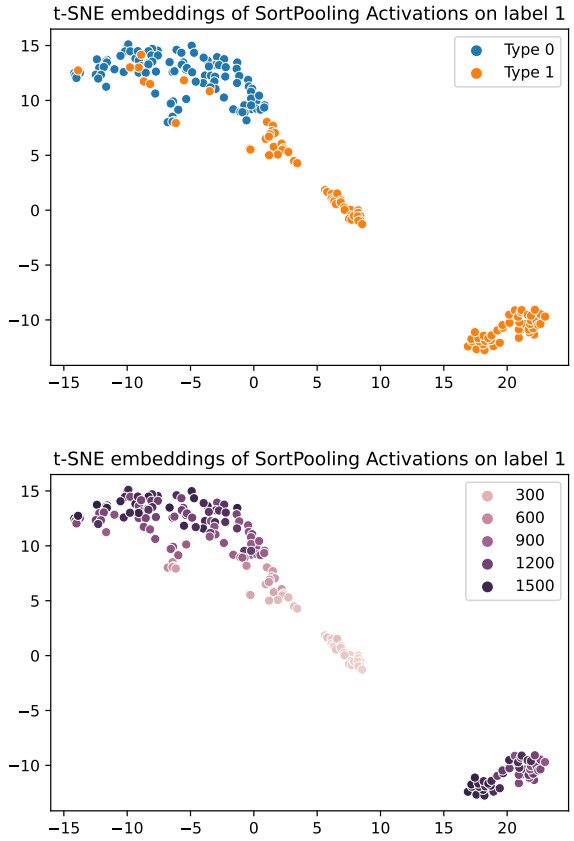
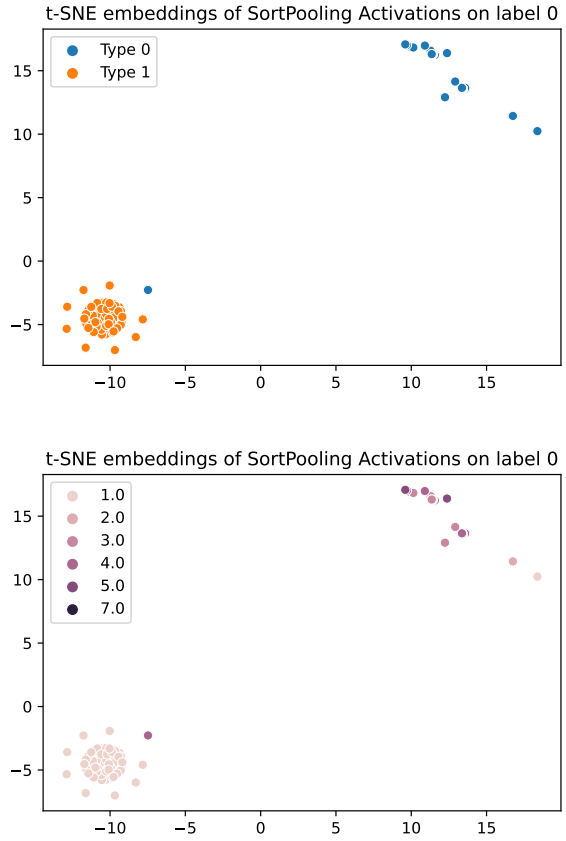


Fig. 6. t-SNE embeddings on training examples with label 0. First row: colors assigned according to node pair type (see Section III). Second row: colors assigned according to maximum degree of the two nodes.



competition to extract features on which a recurrent model could be trained.

Acknowledgment

The first author acknowledges the Scotiabank Centre for Digital Transformation at the Federico Santa Maria University for the funding support to attend the IEEE Big Data Conference 2021.

References

[1] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, p. 78–94, Jul 2018. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2018.03.022

[2] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. 77, pp. 2539–2561, 2011. [Online]. Available: http://jmlr.org/papers/v12/shervashidze11a.html

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016. [Online]. Available: http://arxiv.org/abs/1607.00653

[4] M. Krenn and A. Zeilinger, "Predicting research trends with semantic and neural networks with an application in quantum physics," *Proceedings of the National Academy of Sciences*, vol. 117, no. 4, pp. 1910–1916, 2020. [Online]. Available: https://www.pnas.org/content/117/4/1910

[5] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *CoRR*, vol. abs/1812.04202, 2018. [Online]. Available: http://arxiv.org/abs/1812.04202

[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, p. 4–24, Jan 2021. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2020.2978386

[7] S. Fortunato, C. T. Bergstrom, K. Börner, J. A. Evans, D. Helbing, S. Milojević, A. M. Petersen, F. Radicchi, R. Sinatra, B. Uzzi, A. Vespignani, L. Waltman, D. Wang, and A.-L. Barabási, "Science of science," *Science*, vol. 359, no. 6379, p. eaao0185, 2018. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aao0185

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[9] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2017.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[11] J. You, R. Ying, and J. Leskovec, "Design space for graph neural networks," 2021.

[12] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11782

[13] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in Neural Information Processing Systems*, vol. 31, pp. 5165–5175, 2018.

[14] D. Grattarola and C. Alippi, "Graph neural networks in tensorflow and keras with spektral," 2020.