Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

INFORMATION ENGINEERING DEPARTMENT

MASTER'S DEGREE IN COMPUTER ENGINEERING

APEROL from Networks: Analyzing Pipeline and Embedding Representations for Optimized Learning (from Networks)

Professor
Fabio Vandin

Students
Marco Annunziata, 2160851
Silvia Mondin, 2141201
Sveva Turola, 2160852

ACADEMIC YEAR 2025-2026

Contents

1 Motivation

Graphs have emerged as a natural model for the representation and analysis of data and complex systems in various domains. For instance, link prediction in social networks can help to understand the spreading process of rumors or epidemics [1], and in biological networks, it is commonly used for predicting novel interactions between proteins [2].
However, most traditional machine learning algorithms are not designed to work directly with graph-structured data, as they require numeric vectors or matrices as inputs. To address this challenge, embedding techniques have been developed to learn low-dimensional vector representations of nodes, edges, or entire graphs while preserving their topological properties.
Consequently, in order to design an effective embedding technique, it is necessary to consider several criteria [3]. Two of these are adaptability and scalability. An embedding method is considered adaptable if it can be utilized for various data and tasks without the need for retraining. The scalability of an embedding technique is determined by its capacity to process large-scale networks within a reasonable amount of time.
The objective of this study is to evaluate known embedding methods from the perspective of these two criteria.
From the perspective of adaptability, the objective is to determine the potential impact of the graph domain on the precision of an embedding method. In other words, the objective is to ascertain whether there exist approaches that are better suited for a particular field, whether there is one embedding technique that consistently outperforms the others, or if the methods are comparable. Eventually, we would like to assess the adaptability of these techniques when they are applied to different tasks.
In the context of scalability, the objective is to identify the embedding methods that are better suited to process large graphs. The objective of this study is to evaluate the tradeoff between the accuracy of the predictions and the time required for training and inference.

2 Datasets

The datasets that will be used in this project are nine and they are divided into three categories: road networks, social networks, and biological networks. For each of these categories, three datasets of increasing size were selected: one small, one medium, and one large.
The datasets, with their main properties (i.e., the number of nodes, the number of edges, and the type of graph), are presented in Table 1.

Network	$\|V\|$	$\|E\|$	**Type**
Pennsylvania [4]	1,088,092	1,541,898	Undirected
Padua (province) [5]	122,680	304,184	Directed
Hong Kong (city) [6]	43,620	91,542	Directed
Italian Covid-19 Retweet Network [7]	221,574	800,000	Directed
Twitch Gamers [8]	168,114	6,797,557	Undirected
GitHub Developers [9]	37,700	289,003	Undirected
Mus Musculus Protein Interactions [10]	20,969	800,000	Undirected
Saccharomyces cerevisiae Protein Interactions [10]	5,786	100,000	Undirected
Bio-grid-fission-yeast [4]	2,000	25,300	Undirected

Table 1: Datasets used in the project

3 Methods

Node embedding is the task of mapping nodes into an embedding space, where each node v is represented by a vector $\mathbf{z_v} \in \mathbb{R}^d$ using the mapping function $ENC : v \rightarrow \mathbf{z}_v$. The encoding of nodes into embeddings has to be such that similarity between each pair of nodes is kept in the correspondent embeddings. In order to achieve this, a similarity score $S(u, v)$ is computed between pairwise nodes and a decoder function $DEC : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is used for computing a similarity score for the embeddings. The discrepancy between the two scores is then compared and minimized by means of a loss function, such as the empirical loss:

$$\mathcal{L} = \sum_{i,j} \ell(DEC(\mathbf{z}_u, \mathbf{z}_v), S(u, v))$$

The node embedding methods are defined by the characteristics of the encoder and decoder. In this project, four of these methods will be employed:

- Node2Vec [11], a shallow encoding method and essentially a tunable version of the DeepWalk algorithm, which is often used in the scientific literature as the baseline for comparing node embedding methods;
- LINE [12], a shallow encoding which is an highly-scalable approximation algorithm;
- Deep Variational Network Embedding in Wasserstein Space [13], a deep encoding method using a Variational Autoencoder. This method has a particular approach of encoding each node to a probability distribution, so in this case the similarity score between embeddings is a distance between probability distributions, which can be measured using the Wasserstein distance;
- GraphSage [14], a deep encoding method making use of a GNN that aggregates the features of each node's neighborhood and learns to generate node embeddings from that.

Given the node embeddings, they are used as input to machine learning models to make predictions, depending on the task to be solved. For this project the main task is link prediction, which aims to predict the existence of a link between two nodes in a network. Since the quality of the embeddings could depend on the model used, we'll try models of different complexity: Support Vector Machines [15], where eventually the kernel trick could be used to handle non-linear relations between embeddings; Random Forest [16], for its really fast computation; Multilayer Perceptron [17], for accurate predictions.

4 Experiments

The link prediction task requires a sample of edges from the graph (positive examples) and a sample of edges from the complement graph (negative example). Consequently, each dataset will be partitioned into training, validation, and test sets, in order to guarantee that each set contains an equal number of positive and negative examples.

Two metrics that are generally used for link prediction are the Area Under the Receiver Operating Characteristic curve (AUROC), and the Area Under the Precision-Recall curve (AUPR). AUROC is historically considered the primary performance metric used to evaluate the performances of link prediction methods [18]. However, AUROC favors accurate classification of positive examples, at the cost of misclassifying the negative ones. In a scenario like the link prediction problem, which is inherently biased towards the negative class, this approach may not be optimal and can overestimate the performance of the methods. Consequently, the AUPR curve was also selected, as it can provide better evaluation of link prediction in the presence of class imbalance.

In regard to the implementation of the embedding techniques mentioned in Sec 3, the following implementations will be adopted: [19] for Node2Vec, [20] for LINE, [21] for DVNE, [22] for GraphSage.

Finally, we are going to use Google Colab. The following are specifics of its basic runtime: Intel(R) Xeon(R) CPU 2-core @ 2.20GHz, 13 Gb, 110 Gb disk, Nvidia Tesla T4 (16 Gb VRAM).

References

[1] Romualdo Pastor-Satorras et al. "Epidemic processes in complex networks". In: *Reviews of modern physics* 87.3 (2015), pp. 925–979.

[2] Björn H Junker and Falk Schreiber. *Analysis of biological networks.* Vol. 2. Wiley Online Library, 2008.

[3] Anthony Baptista et al. "Zoo guide to network embedding". In: *Journal of Physics: Complexity* 4.4 (2023), p. 042001.

[4] Ryan A. Rossi and Nesreen K. Ahmed. "The Network Data Repository with Interactive Graph Analytics and Visualization". In: *AAAI.* 2015. URL: `https://networkrepository.com`.

[5] Marco Annunziata. *Padua_Network_dataset_2025.* 2025. URL: `https://github.com/Remdox/Padua_Network_dataset_2025`.

[6] yzengal. *RoadNetwork-China-City.* 2021. URL: `https://github.com/yzengal/RoadNetwork-China-City/blob/main/Hongkong.road-d.tar.gz`.

[7] V. Mesina et al. *Italian Covid-19 Retweet Network (2020-2022) [Data set].* 13th International Conference on Complex Networks and Their Applications (CNA2024), Istanbul, Turkey. Zenodo. 2024. URL: `https://doi.org/10.5281/zenodo.13909011`.

[8] Benedek Rozemberczki and Rik Sarkar. *Twitch Gamers: a Dataset for Evaluating Proximity Preserving and Structural Role-based Node Embeddings.* 2021. arXiv: `2101.03091 [cs.SI]`.

[9] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. *Multi-scale Attributed Node Embedding.* 2019. arXiv: `1909.13021 [cs.LG]`.

[10] Damian Szklarczyk et al. "The STRING database in 2023: protein–protein association networks and functional enrichment analyses for any sequenced genome of interest". In: *Nucleic Acids Research* 51.D1 (2023), pp. D638–D646. DOI: `10.1093/nar/gkac1000`. URL: `https://doi.org/10.1093/nar/gkac1000`.

[11] Aditya Grover and Jure Leskovec. *node2vec: Scalable Feature Learning for Networks.* 2016. arXiv: `1607.00653 [cs.SI]`. URL: `https://arxiv.org/abs/1607.00653`.

[12] Jian Tang et al. "LINE: Large-scale Information Network Embedding". In: *Proceedings of the 24th International Conference on World Wide Web.* WWW '15. International World Wide Web Conferences Steering Committee, May 2015, pp. 1067–1077. DOI: `10.1145/2736277.2741093`. URL: `http://dx.doi.org/10.1145/2736277.2741093`.

[13] Dingyuan Zhu et al. "Deep Variational Network Embedding in Wasserstein Space". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD '18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2827–2836. ISBN: 9781450355520. DOI: `10.1145/3219819.3220052`. URL: `https://doi.org/10.1145/3219819.3220052`.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs.* 2018. arXiv: `1706.02216 [cs.SI]`. URL: `https://arxiv.org/abs/1706.02216`.

[15] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[16] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[17] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[18] I. Bhargavi Kalyani, A. Rama Prasad Mathi, and Niladri Sett. "Evaluating link prediction: new perspectives and recommendations". In: *International Journal of Data Science and Analytics* 20.7 (2025), pp. 6855–6886. ISSN: 2364-4168. DOI: `10.1007/s41060-025-00858-0`. URL: `https://doi.org/10.1007/s41060-025-00858-0`.

[19] palash1992. 2022. URL: `https://github.com/palash1992/GEM`.

[20] tangjianpku et al. 2019. URL: `https://github.com/tangjianpku/LINE`.

[21] Lakshya-99. 2020. URL: `https://github.com/Lakshya-99/Deep_Variational_Network_Embedding/tree/master`.

[22] William L. Hamilton et al. 2018. URL: `https://github.com/williamleif/GraphSAGE`.

Contribution of Authors and AI Usage

The contributions for this project are:

- Motivation: Marco Annunziata (research), Silvia Mondin (research and writing), Sveva Turola (research);
- Datasets: Marco Annunziata (research), Silvia Mondin (research), Sveva Turola (research and writing);
- Methods: Marco Annunziata (research and writing), Silvia Mondin (research), Sveva Turola (research);
- Experiments: Marco Annunziata (research and writing), Silvia Mondin (research and writing), Sveva Turola (research and writing).

AI tools (DeepL) were used to correct the grammar and syntax of the text.