Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

DATA SCIENCE AND COMPUTATION

Ciclo 34

Settore Concorsuale: 01/B1 - INFORMATICA

Settore Scientifico Disciplinare: INF/01 - INFORMATICA

LEARNING REPRESENTATIONS FOR GRAPH-STRUCTURED SOCIO-TECHNICAL SYSTEMS

Presentata da: Simone Piaggesi

Coordinatore Dottorato

Daniele Bonacorsi

Supervisore

Ciro Cattuto

Co-supervisore

Andrè Panisson

Esame finale anno 2023

Abstract

THE recent widespread use of social media platforms and web services has led to a vast amount of behavioral data that can be used to model socio-technical systems. A significant part of this data can be represented as graphs or networks, which have become the prevalent mathematical framework for studying the structure and the dynamics of complex interacting systems. However, analyzing and understanding these data presents new challenges due to their increasing complexity and diversity. For instance, the characterization of real-world networks includes the need of accounting for their temporal dimension, together with incorporating higher-order interactions beyond the traditional pairwise formalism.

The ongoing growth of AI has led to the integration of traditional graph mining techniques with representation learning and low-dimensional embeddings of networks to address current challenges. These methods capture the underlying similarities and geometry of graph-shaped data, generating latent representations that enable the resolution of various tasks, such as link prediction, node classification, and graph clustering. As these techniques gain popularity, there is even a growing concern about their responsible use. In particular, there has been an increased emphasis on addressing the limitations of interpretability in graph representation learning.

This thesis contributes to the advancement of knowledge in the field of graph representation learning and has potential applications in a wide range of complex systems domains. We initially focus on forecasting problems related to face-to-face contact networks with time-varying graph embeddings. Then, we study hyperedge prediction and reconstruction with simplicial complex embeddings. Finally, we analyze the problem of interpreting latent dimensions in node embeddings for graphs. The proposed models are extensively evaluated in multiple experimental

settings and the results demonstrate their effectiveness and reliability, achieving state-of-the-art performances and providing valuable insights into the properties of the learned representations.

Acknowledgements

THIS thesis would not have been the same without the many inspiring people I have met along my Ph.D. journey. I am extremely grateful for the support and guidance of the many mentors, colleagues, and friends who have been a part of this long path.

First, my sincere thanks go to my supervisors Prof. Ciro Cattuto and Prof. André Panisson, for their unwavering help during the development of my Ph.D. work. Thanks to Ciro, who gave me the rare opportunity to work within the exceptional research environment at ISI Foundation; and thanks to André, who offered me his continuous guidance and expert advice throughout this Ph.D. path.

Then, I would like to thank Prof. Avishek Anand for hosting me during the visiting periods abroad in Hannover and Delft, and for providing me with invaluable resources and support. I am grateful to all the fantastic people I met during those months, both inside and outside the research laboratories.

I am also indebted to the many students, junior researchers, and senior scientists who have shared their time and expertise with me during these years. Their contributions have been instrumental in shaping my research and personal growth.

Finally, I must express my profound gratitude to my family and friends for providing me with unfailing motivation and encouragement throughout these years. This accomplishment would not have been possible without them.

And, last but not least, I am deeply indebted to Simona, whose endless support and love have been a constant source of strength and inspiration.

Thank you.

Contents

List of Figures

List of Tables

List of Tables

CHAPTER 1

Introduction

HUMAN interactions play a central role in the functioning of society and are the driving force behind social connections and relationships. In recent years, the proliferation of online social media and the widespread use of portable devices has led to an explosion of (meta)data on human interactions at various spatial and temporal scales [Kit14]. These digital traces have provided researchers with new insights about human behavior in socio-technical systems [Ves09]. Computational social science [LPA+09] and digital epidemiology [SBB+12] are just a couple of research disciplines that have emerged in response to the increasing availability of data on human interactions. These fields seek to understand human behavior and dynamical phenomena that are influenced by it, such as the spread of diseases, using data-driven approaches.

The ability to access large quantities of relational information has also allowed researchers to examine various kinds of human interactions in techno-social systems, such as proximity contacts [CVdBB+10, ISB+11, SKL+10] and digital communications [LH08, OSH+07, EMS04]. In addition to providing new insights into human activities, the study of behavioral traces also has practical applications. For example, understanding the patterns of human mobility can help policymakers design more effective interventions to address social inequalities [GTP+20]. Similarly, studying diffusion patterns through networks of human interactions can inform the

development of strategies to control the spread of infectious diseases [PSV02].

Given the significance of interactions in shaping the behavior in techno-social systems, several important research directions can advance our understanding of complex systems and inform the design of effective interventions in response to societal issues. Some potential high-impact problems that could be handled include:

- Effectively predicting the spread of diseases through networks of human interactions.
- Understanding the evolution of interacting systems over time and predicting how they may change in the future.
- Identifying key features that play a crucial role in shaping the behavior of the whole system.

To address these and other pressing research challenges, it is more and more necessary to understand the emergent complexity in behavioral data being generated on a continual basis. Specifically, there is a growing need for machine learning techniques, used in combination with traditional approaches, to discover new knowledge from the plethora of data generated by the vast number of digital devices and social media platforms.

1.1 Machine Learning on Graph Structures

Graphs, or networks, are a natural way to represent units connected by pairwise relationships [New03]. Due to the abundance of real-world systems that can be easily described as networks, graph theory [W+01] has been used as the major paradigm to understand organizational principles in complex interacting systems [BGL16, GN02], and contagion phenomena such as epidemics and opinion spreading [PSCVMV15, BBV08]. In recent years, areas of particular interest have been the study of temporal networks [HS12] and higher-order graphs [BCI+20]. Temporal networks capture the changing nature of interactions over time, and higher-order graphs take into account the fact that interactions often take place in groups of individuals. The use of these advanced network representations has allowed researchers to more accurately model the complexity of real-world systems [LRS19].

Besides traditional graph analytics approaches, there has been increasing focus on the use of machine learning methods for uncovering the underlying structure of interactions in complex systems. Representation learning algorithms [Ham20] aim to extract compact representations of graph-shaped data, usually called vector embeddings, which can be used for solving learning tasks such as classification or clustering over networks [XSY+21]. One key advantage of using such embedding

methods is that they are able to encode in a feature space latent relationships between graph units [Xu21]. This is particularly useful for tasks such as node classification and link prediction [LZ11], where the connections between nodes can provide important contextual information in addition to engineered node features [ASK+21]. While most methods have been developed for standard graphs, in the last years we experienced several advancements for representation learning on time-varying [BMVZ21] and higher-order graphs [PSHM23], two central topics of this dissertation.

The use of graph representation learning techniques can greatly enhance our understanding of relationships and interactions within complex systems. However, like the prevalence of machine learning models, there are potential issues that need to be addressed for using these approaches in a responsible manner [HGC20]. In particular, besides important aspects such as unfairness and bias [MMS+21], the key concerns examined in this thesis are the interpretability and explainability of graph representations [SM19, DVK17]. In many cases, models' internal functioning is not transparent and the provided output is not human-understandable [GMR+18]. For instance, low-dimensional embeddings learned by these techniques may be difficult for humans to interpret, making it hard to explain the results or draw meaningful conclusions from the data. This could have serious consequences, particularly when the models are used to make decisions that have real-world impact [WY21].

1.2 Research Questions

In this thesis, *Learning Representations for Graph-Structured Socio-Technical Systems*, we focus on research challenges associated with developing prediction models for complex systems. We will study representation learning algorithms with a focus on their effectiveness, meaning that they can reach state-of-the-art prediction performances, and their reliability, meaning that their output can be easily understood. To achieve these goals, we will examine specific problems related to learning embedding representations for graph-structured data. Ultimately, our purpose is to make a meaningful contribution to the field and to create tools that can be used to tackle real-world problems related to complex systems. Specifically, the contributions of this dissertation are guided by the following research questions:

Research Question 1. *Are representation learning techniques for time-resolved proximity networks effective in predicting the occurrence of contact events and estimating the outcomes of spreading processes?*
Contributions to address this question are described in Chapter 3 and in [PP22]. We designed a feature learning algorithm trained with negative sampling and based on a higher-order generalization of the skip-gram approach [MSC+13], which captures both topological and dynamical properties of time-resolved contact

networks [ZSBB11]. We show that the proposed time-varying graph representations are effective in performing prediction of temporal links in the form of classification tasks, and even that they can be useful to recover crucial information about the outcome of SIR spreading processes [KR08].

Research Question 2. *Are representation learning methods applied to higher-order relational data effective in predicting the occurrence of group-wise interactions?*
Contributions to address this question are described in Chapter 4 and in [PPP22]. There we show a representation learning framework to perform the reconstruction and prediction of group-wise links in the form of classification tasks, for triadic and tetradic interactions. We carefully formalize network reconstruction [CMS21] and link prediction [KSSB20] for polyadic graph structures, and we show that neural representations obtained from simplicial complexes [SCL18] are able to outperform traditional approaches, even when training data is composed only of lower-order interactions.

Research Question 3. *How can we comprehend the latent embedded features, especially in combination with model predictions, extracted by graph representation learning approaches?*
Contributions to address this question are described in Chapter 5, where we propose novel metrics to measure the interpretability [DG18] of node embeddings based on marginal contributions of dimensions to edge prediction. We say an embedding dimension is more interpretable if it can faithfully map to an understandable substructure in the input graph - like community structure [GN02]. We also introduce a novel approach that can retrofit existing node embeddings by making them more interpretable without sacrificing their downstream task performance.

In addition to the main contributions of this thesis, we include a prelude to several background topics in Chapter 2, which are introductory elements to the material presented in the rest of this dissertation. After presenting the core of the work, we will summarize the obtained results and illustrate open directions for future research in Chapter 6.

CHAPTER 2

Background Review

THIS thesis is centered on the idea of *graph representation learning*. Graphs [New03] are ubiquitous data structures, extensively used in a wide range of fields, including physics, computer science, biology and sociology [Bar12]. Many natural, technological and social systems can be modeled as graphs, which capture interactions between individual units. Due to their versatility and usefulness, graphs form the foundation to describe many complex systems [Ves12], and provide convenient means for storing and retrieving relational knowledge about interacting entities [JPC+21].

Representation learning [BCV13] is a key aspect for modern machine learning applications in natural language processing [MSC+13, VSP+17], image recognition [DGE15, MM20], graph mining [GL16, VCC+18] and many others [ZYHD20, MLB+22]. Representation learning is the process of extracting features, typically lower-dimensional and more compact than the raw input, that describe the underlying characteristics of data. These features can be used to facilitate the analysis of data, as well as to build classifiers or other predictors. Traditional methods for obtaining these representations involve manual feature engineering, which is often a time-consuming and expensive procedure. Furthermore, hand-engineered features may not be flexible enough to adapt to different learning tasks. Conversely, representation learning utilizes data-driven approaches to discover representations

that encode hidden patterns and similarities within the data.

Graph representation learning [Ham20] is a collection of techniques that are utilized to incorporate the structural information of a graph into a machine learning model. The aim is to learn a mapping function that projects nodes or (sub)graphs as points of a latent vector space [HYL17b]. The embedding map is usually optimized in an unsupervised manner, so that geometric relationships in the embedding space reflect the structure of the original graph, or directly learned on a target supervised task. The learned representations can be applied to a variety of graph-based learning tasks [XSY+21], such as link prediction, node classification and graph clustering.

In this chapter, we provide necessary background notions and topics that will be extensively used in this thesis.

2.1 Neural Representations from Language Processing

2.1.1 WORD2VEC embeddings

WORD2VEC is one of the most famous representation learning methods and it was designed to compute word embeddings from a textual corpus [MCCD13]. It provided the inspiration for developing the first techniques to learn node embeddings in networks [PARS14, TQW+15, GL16], relying on the parallelism between textual sentences and random walks on a graph. Here we review WORD2VEC's functioning as a paradigmatic example of unsupervised representation learning for relational data.

Given a text corpus $\mathcal{W} = \{w_1, w_2, \dots\}$, where every word $w_t \in \mathcal{V}$ belongs to a given vocabulary $\mathcal{V}$, word embeddings are the results of an encoding function $\mathrm{enc} : \mathcal{V} \to \mathbb{R}^D$ (usually $D << |\mathcal{V}|$) that incorporates syntactic and semantic properties of the training corpus into geometric relationships among word vectors [LG14a]. In particular, WORD2VEC exploits the *distributional hypothesis* [Har54] and computes embeddings for words according to the linguistic context in which they are placed. Essentially, a linguistic context describes the ways in which words relate to other words, according to their relative positions in language, defining a proper relational structure that can be encoded in a latent vector space.

Formally, in WORD2VEC every word token w_t is paired with a contextual window corresponding to the multi-set $\mathcal{C}_T(w_t) = \{w_{t-T}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+T}\}$ of surrounding words, where the hyperparameter T is the maximum number of steps needed to reach any context word. The objective is to minimize the log-probability loss:

$$\min_{\theta} - \sum_{w_t \in \mathcal{W}} \ell\Big(w_t, \mathcal{C}_T(w_t); \theta\Big) \tag{2.1}$$

where each term $\ell(w_t, \mathcal{C}_T(w_t); \theta)$ generally refers to the log-likelihood of observing a target word w_t along with its contextual words $\mathcal{C}_T(w_t)$.

WORD2VEC architecture is a 2-layer dense neural network and model parameters θ refer to two weight matrices $\mathbf{V}, \mathbf{C} \in \mathbb{R}^{|\mathcal{V}| \times D}$ involved in the neural optimization (*input_layer-hidden_layer* and *hidden_layer-output_layer* transformations). Rows of the weight matrices $\{\mathbf{v}_i\}_{i \in \mathcal{V}}$ and $\{\mathbf{c}_i\}_{i \in \mathcal{V}}$ are identifiable as word vectors, respectively called *word embeddings* and *context embeddings*. Thus, the model actually learns two encodings, namely $\mathrm{enc}_{\mathbf{V}}$ and $\mathrm{enc}_{\mathbf{C}}$, but usually only the first one is taken into account for downstream tasks [NMCC16] and for this reason we will use the term enc referring to $\mathrm{enc}_{\mathbf{V}}$. It is even possible to apply *weight tying* [PMH18] to share word and context parameters in the same embedding matrix. Depending on the parametrization of the log-likelihood with respect to embedding parameters, we have two variations of the optimization task known as Skip-Gram (SG) and Continuous-Bag-Of-Words (CBOW).

Skip-Gram model

In the Skip-Gram model, given any input word i, it is optimized the conditional log-probability of predicting the surrounding context words:

$$\ell(i, \mathcal{C}_T(i); \theta) \equiv \log \Pr[\, \mathcal{C}_T(i) \mid i \,; \theta \,] = \log \prod_{j \in \mathcal{C}_T(i)} \Pr[\, j \mid i \,; \theta \,] \tag{2.2}$$

where the multi-word conditional likelihood is factorized because of the independence assumption in observing any neighborhood word from the input word i. The conditional log-probability of a single word-context pair is modeled as a softmax function parametrized by the inner product of input-word and context-word embeddings:

$$\Pr[\, j \mid i \,; \theta \,] = \frac{\exp\left(\mathbf{v}_i \cdot \mathbf{c}_j\right)}{\sum_{u \in \mathcal{V}} \exp\left(\mathbf{v}_i \cdot \mathbf{c}_u\right)} \tag{2.3}$$

Finally, the log-likelihood takes the form:

$$\ell(i, \mathcal{C}_T(i); \theta) = \sum_{j \in \mathcal{C}_T(i)} \left[\mathbf{v}_i \cdot \mathbf{c}_j - \log \sum_{u \in \mathcal{V}} \exp\left(\mathbf{v}_i \cdot \mathbf{c}_u\right) \right] \tag{2.4}$$

Continuous-Bag-Of-Words model

In the CBOW model, given all the input surrounding neighbors $\mathcal{C}_T(i)$ of word i, it is optimized the log-probability of observing the central word given the whole context words:

$$\ell(i, \mathcal{C}_T(i); \theta) \equiv \log \Pr[\, i \mid \mathcal{C}_T(i) \,; \theta \,] \tag{2.5}$$

With the assumption of disregarding the textual ordering of contexts, the conditional log-probability is modeled as a softmax function parametrized by the sum of inner

products between input-word and context-word embeddings:

$$\Pr[\, i \mid \mathcal{C}_T(i)\, ; \theta\,] = \frac{\exp\left(\sum_{j \in \mathcal{C}_T(i)} \mathbf{v}_j \cdot \mathbf{c}_i\right)}{\sum_{u \in \mathcal{V}} \exp\left(\sum_{j \in \mathcal{C}_T(i)} \mathbf{v}_j \cdot \mathbf{c}_u\right)} \tag{2.6}$$

Finally, the log-likelihood takes the form:

$$\ell(i, \mathcal{C}_T(i); \theta) = \sum_{j \in \mathcal{C}_T(i)} \mathbf{v}_j \cdot \mathbf{c}_i - \log \sum_{u \in \mathcal{V}} \exp\big(\sum_{j \in \mathcal{C}_T(i)} \mathbf{v}_j \cdot \mathbf{c}_u\big) \tag{2.7}$$

2.1.2 Skip-gram with negative sampling

When dealing with large vocabularies, a common drawback in CBOW and SG approaches is the expensive computation of the per-word partition function deriving from the softmax:

$$Z_i^{(sg)}(\theta) = \sum_{u \in \mathcal{V}} \exp\big(\mathbf{v}_i \cdot \mathbf{c}_u\big), \quad Z_i^{(cbow)}(\theta) = \sum_{u \in \mathcal{V}} \exp\big(\sum_{j \in \mathcal{C}_T(i)} \mathbf{v}_j \cdot \mathbf{c}_u\big) \tag{2.8}$$

One solution to alleviate this issue consists in using the Negative Sampling technique [MSC+13], a form of Noise Contrastive Estimation (NCE) [MK13] where the problem of fitting the conditional probability is reduced to a binary classification, discriminating between samples from the data distribution and samples from a known noise distribution.

In the next lines, we describe in detail the Skip-Gram with Negative Sampling (SGNS) approach. We firstly denote as $\mathcal{D}$ the multi-set collecting co-occurrences (i, j) between any word i in the text corpus with neighboring words $j \in \mathcal{C}_T(i)$. Also, we denote as $\#(i, j)$ the number of times (i, j) appears in $\mathcal{D}$. Similarly we use $\#i = \sum_j \#(i, j)$ and $\#j = \sum_i \#(i, j)$ as the number of times each word occurs in $\mathcal{D}$, with relative frequencies $P_{\mathcal{D}}(i, j) = \frac{\#(i,j)}{|\mathcal{D}|}$, $P_{\mathcal{D}}(i) = \frac{\#i}{|\mathcal{D}|}$ and $P_{\mathcal{D}}(j) = \frac{\#j}{|\mathcal{D}|}$.

For any word i and its context words $\mathcal{C}_T(i)$, the optimization task consists in minimizing the binary cross-entropy loss:

$$\mathrm{bce}(i, \mathcal{C}_T(i); \theta) = -\sum_{j \in \mathcal{C}_T(i)} \left[\log \sigma(\mathbf{v}_i \cdot \mathbf{c}_j) + \kappa \cdot \mathop{\mathbb{E}}_{j_n \sim P_{\mathcal{N}}^{\alpha}} [\log \sigma(-\mathbf{v}_i \cdot \mathbf{c}_{j_n})]\right] \tag{2.9}$$

where each word-context pair (i, j) encountered in the text is classified against a number κ of negative pairs (i, j_n) randomly sampled. The sigmoid ($\sigma(x) = (1 + e^{-x})^{-1}$) of the inner product of embedding vectors parametrizes the probability of the positive class $\Pr[\, (i, j) \in \mathcal{D} \mid \mathbf{v}_i, \mathbf{c}_j\,] = \sigma(\mathbf{v}_i \cdot \mathbf{c}_j)$, and negative sampling is done according to the noise distribution $P_{\mathcal{N}}^{\alpha}(j) = \frac{(\#j)^{\alpha}}{\sum_u (\#u)^{\alpha}}$. In the original

work [MSC+13], it is observed that setting $\alpha = \frac{3}{4}$ produces superior performance in semantic evaluation experiments, but it has not been proved to have the same effects with graph learning tasks. For this reason, we consider the simpler case $\alpha = 1$ which generates negative contexts sampled with the empirical unigram distribution $P_{\mathcal{N}}(j) = P_{\mathcal{D}}(j)$.

The loss over the entire corpus $\sum_{i \in \mathcal{W}} \mathrm{bce}\Big(i, \mathcal{C}_T(i); \theta\Big)$ gives the SGNS objective function and, following results from Levy and Golberg [LG14b], can be written as:

$$\mathcal{L}^{(sgns)} = -\sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} \Big[P_{\mathcal{D}}(i,j) \log \sigma(\mathbf{v}_i \cdot \mathbf{c}_j) + \kappa \, P_{\mathcal{N}}(i,j) \log \sigma(-\mathbf{v}_i \cdot \mathbf{c}_j) \Big] \quad (2.10)$$

where $P_{\mathcal{N}}(i,j) = P_{\mathcal{D}}(i) \cdot P_{\mathcal{D}}(j)$ is the probability of (i,j) under assumption of statistical independence. Levy and Goldberg [LG14b] established the relation between Skip-Gram trained with negative sampling and traditional matrix decomposition methods [KB09], showing the equivalence of SGNS and low-rank matrix factorization. When D is sufficiently high, optimal SGNS embedding matrices satisfy these relations:

$$(\mathbf{V}\mathbf{C}^{\mathrm{T}})_{ij} \approx \log \left(\frac{P_{\mathcal{D}}(i,j)}{\kappa \, P_{\mathcal{N}}(i,j)} \right) = \mathrm{PMI}(i,j) - \log(\kappa) \equiv \mathrm{SPMI}_{\kappa}(i,j) \quad (2.11)$$

which tell us that SGNS optimization is equivalent to a rank-D matrix decomposition of the word-context pointwise mutual information (PMI) matrix [CH90] shifted by a constant, i.e. the number of negative samples. This equivalence was later retrieved from diverse assumptions [YDZ+20, ABH19, MG17, ALL+16], and exploited to compute closed form expressions approximated in different node embedding models [QDM+18].

2.2 Learning Representations on Graphs

2.2.1 Encoder-decoder perspective

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$ such that entries $\mathbf{A}_{ij} = \omega(i,j)$ are the edge weights, node embeddings are the output of an encoder function:

$$\mathrm{enc} : v \in \mathcal{V} \mapsto \mathbf{v} \in \mathbb{R}^D \quad (2.12)$$

which maps nodes into geometric points of the $D-$dimensional vector space $\mathbb{R}^D$ (usually $D << |\mathcal{V}|$). As in word embeddings, node representations can be collected into the entries of the embedding matrix $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}| \times D}$, that is learned with a reconstruction objective. The encoder enc is jointly optimized with a decoder

function $\mathrm{dec} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ to predict pair-wise node proximities $\mathrm{sim}_{\mathcal{G}}(i,j)$ from node embeddings returned by the encoder:

$$\sum_{(i,j)\in\mathcal{V}\times\mathcal{V}} \ell\Big(\mathrm{dec}\big(\mathbf{v}_i, \mathbf{v}_j\big), \mathrm{sim}_{\mathcal{G}}(i,j)\Big) \tag{2.13}$$

Many embedding methods fit this *encoder-decoder* framework: factorization-based methods like NETMF [QDM+18], HOPE [OCP+16], GRAREP [CLX15]; walk-based like DEEPWALK [PARS14], NODE2VEC [GL16], WALKLETS [PKCS17]; and even deep neural methods like GRAPHSAGE [HYL17a] and GAE [TW16], where the encoder function is given by a graph neural network [SGT+08].

2.2.2 Shallow methods and matrix factorization

A well-known family of approaches is called *shallow embedding* [ZYZZ18] because the encoder function is simply an embedding lookup that selects the corresponding row of the embedding matrix based on the node ID. Despite the apparent heterogeneity of these algorithms, the majority of them can be unified from the perspective of matrix factorization [QDM+18]. In particular, the methods can be described as factorizing a certain node-to-node proximity matrix: explicit decomposition with SVD-like techniques [OCP+16, CLX15], or implicit decomposition by optimizing a proxy task [PARS14, GL16]. Here we mainly describe the second approach, showing the connection with the first one.

DEEPWALK [PARS14] and NODE2VEC [GL16] consist in sampling random walks from the graph and processing node sequences as textual sentences. The Skip-Gram model is used to obtain node embeddings from co-occurrences in random walk realizations, where the empirical objective[1] is derived from Equation (2.9):

$$-\sum_{(i,j)\in\mathcal{D}} \Big[\log\sigma(\mathbf{v}_i\cdot\mathbf{v}_j) + \kappa\cdot \mathop{\mathbb{E}}_{j_n\sim P_{\mathcal{N}}}[\log\sigma(-\mathbf{v}_i\cdot\mathbf{v}_{j_n})]\Big] \tag{2.14}$$

where $\mathcal{D}$ contains training node pairs (i,j) generated from walks co-occurrences. The DEEPWALK approach employs first-order random walks with transition probabilities proportional to edge weights, whereas NODE2VEC adopts tunable second-order transitions to interpolate between breadth-first search and depth-first search. Although the original implementation of DEEPWALK uses hierarchical softmax to compute embeddings, we will refer to the SGNS formulation given by [QDM+18].

Since SGNS can be interpreted as a low-rank factorization of the word-context PMI matrix [LG14b], the closed form of the PMI matrix implicitly decomposed in DEEPWALK and NODE2VEC can be derived [QDM+18]. Given the 1-step random walk matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, whose entries collect the transition probabilities

[1] Here we suppose to learn a unique set of representations $\{\mathbf{v}_i\}_{i\in\mathcal{V}}$, but even separated representations $\{\mathbf{v}_i, \mathbf{c}_i\}_{i\in\mathcal{V}}$ can be computed.

$P(i \to j) \propto \omega(i,j)$, in DEEPWALK the expected PMI for a node-context pair (i,j) occurring in a T-sized window is:

$$\mathrm{PMI}^{\mathrm{DW}}(i,j) = \log\left(\frac{P_{\mathcal{D}}(i,j)}{P_{\mathcal{N}}(i,j)}\right) = \log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left[\frac{d_i}{\mathrm{vol}(\mathcal{G})}(\mathbf{P}^r)_{ij} + \frac{d_j}{\mathrm{vol}(\mathcal{G})}(\mathbf{P}^r)_{ji}\right]}{\frac{d_i}{\mathrm{vol}(\mathcal{G})}\cdot\frac{d_j}{\mathrm{vol}(\mathcal{G})}}\right) \tag{2.15}$$

where $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_{|\mathcal{V}|})$, $d_i = \sum_{j\in\mathcal{V}} \mathbf{A}_{ij}$ is the node degree, and $\mathrm{vol}(\mathcal{G}) = \sum_{(i,j)\in\mathcal{E}} \mathbf{A}_{ij}$.

2.2.3 Graph neural networks and graph autoencoders

A second family of methods consists in learning a parametrized encoder function that returns node embeddings, instead of directly optimizing embedding vectors for each node. Since graphs are complex structures analogous to irregular grids, the class of models called *Graph Neural Networks* (GNNs) [WPC+20] have been proposed as a generalization of convolutional operations for the graph domain.

GNNs work with neural message passing, in which embedding features are exchanged between nodes and updated using customized operations across multiple neural network layers. During each iteration at layer l, the hidden embedding $\mathbf{x}_i^{(l)}$ of node $i \in \mathcal{V}$ is combined with the "message" vector obtained aggregating node embeddings from the neighborhood Γ_i. These operations can be expressed as follows:

$$\mathbf{x}_i^{(l+1)} = \mathrm{UPD}^{(l)}\Big(\mathbf{x}_i^{(l)}, \mathbf{msg}_{\Gamma_i}^{(l)}\Big) \tag{2.16}$$

$$\mathbf{msg}_{\Gamma_i}^{(l)} = \mathrm{AGG}^{(l)}\Big(\{\mathbf{x}_j^{(l)} : j \in \Gamma_i\}\Big) \tag{2.17}$$

The initial embedding $\mathbf{x}_i = \mathbf{x}_i^{(0)}$ may encode input node attributes[2]. Instead, we define the final layer output as the node embedding $\mathbf{v}_i = \mathbf{x}_i^{(L)}$ after L message passing layers.

Despite GNNs can be also trained for node-level and graph-level classification tasks, here we are more interested in considering edge-level pairwise objectives like Equation (2.13) because their optimization is an unsupervised task that does not require the availability of ground-truth labels. For instance, Graph Autoencoders (GAE) [TW16] are unsupervised architectures that utilize a Graph Convolutional Network (GCN) [WK17] to encode the graph structure with the following aggregation rule:

$$\mathbf{x}_i^{(l+1)} = \eta\Big(\mathbf{W}^{(l)} \cdot \sum_{j\in\Gamma_i\cup\{i\}} \frac{\mathbf{x}_j^{(l)}}{\sqrt{d_i d_j}}\Big) \tag{2.18}$$

[2]Attribute features are sometimes unavailable or difficult to be collected. In absence of such features, shallow node embeddings or even one-hot encodings can be used as input.

where η is a nonlinearity like sigmoid or ReLU. GAE trains the embeddings with a pair-wise edges reconstruction loss:

$$\ell(i,j) = -\log \sigma(\mathbf{x}_i^{(L)} \cdot \mathbf{x}_j^{(L)}) - \kappa \cdot \mathop{\mathbb{E}}_{j_n \sim P_{\mathcal{N}}} [\log \sigma(-\mathbf{x}_i^{(L)} \cdot \mathbf{x}_{j_n}^{(L)})]$$

Another GNN model -GRAPHSAGE [HYL17a]- optimizes node embeddings with a random walk-based objective[3], using structural updates defined by:

$$\mathbf{x}_i^{(l+1)} = \eta\Big(\mathbf{W}^{(l)} \cdot \big[\mathbf{x}_i^{(l)}, \frac{1}{d_i} \sum_{j \in \Gamma_i} \mathrm{MLP}(\mathbf{x}_j^{(l)})\big]\Big) \tag{2.19}$$

where $[*,*]$ is the concatenation function and MLP is a multilayer dense neural network.

2.3 Network Reconstruction and Link Prediction

2.3.1 Tasks description

Network reconstruction [GSP09] and link prediction [KSSB20] are common tasks performed over graphs and nowadays are often formulated as classification problems to evaluate the quality of node embeddings. Network reconstruction [CMST21, CMST20] relates to the capability of recovering the training graph structure from latent embeddings. Given a graph $\mathcal{G}$, by *reconstruction* we mean the task of classifying whether a pair of nodes (i,j) is an edge of $\mathcal{G}$, i.e. $(i,j) \in \mathcal{E}$, or not.

Link prediction [ZYR21, CLX19] refers to the generalization capability in identifying out-of-sample edges, i.e. the task of correctly determining the set $\mathcal{E}^*$ of links that have not been observed but are likely to occur. Given any non-edge pair $(i,j) \in \mathcal{U}_\mathcal{E}$ coming from the set of unobserved links $\mathcal{U}_\mathcal{E} = \{(i,j) \in \mathcal{V} \times \mathcal{V} : (i,j) \notin \mathcal{E}\}$, the prediction task is to classify which candidate pairs will connect in the future, and so $(i,j) \in \mathcal{E}^*$, versus those that will remain unconnected, and so $(i,j) \in \mathcal{U}_\mathcal{E} \setminus \mathcal{E}^*$.

2.3.2 Unsupervised setting

In the next paragraphs we summarize different approaches that are part of the unsupervised setting, so called because they do not require to learn additional parameters in order to perform edge classification. Formally, the unsupervised edge classification setting [Sha20] on the input graph $\mathcal{G}$ is composed by: candidate sets for edges and non-edges ($\mathcal{E}^{(+)}$ and $\mathcal{E}^{(-)}$); a similarity function $s : \mathcal{E}^{(+)} \cup \mathcal{E}^{(-)} \to \mathbb{R}$ that assign high scores to candidate edges $\mathcal{E}^{(+)}$ and low scores to candidate non-edges $\mathcal{E}^{(-)}$. From this perspective, in network reconstruction we have $\mathcal{E}^{(+)} = \mathcal{E}$ and $\mathcal{E}^{(-)} = \mathcal{U}_\mathcal{E}$, while for link prediction $\mathcal{E}^{(+)} = \mathcal{E}^*$ and $\mathcal{E}^{(-)} = \mathcal{U}_\mathcal{E} \setminus \mathcal{E}^*$.

[3] Notice that when the window size of co-occurrences is set $T = 1$, the random walk loss is equivalent to the edge reconstruction loss.

Traditional methods are based on similarity heuristics, used to rank pairs of nodes (candidate links), and can be obtained from indices calculated with local or global graph information. These methods, in principle, assign likelihood scores in such a way that presumably occurring connections get higher scores than improbable ones. Given their ability to maintain both local and global characteristics of graphs, node embeddings can be utilized as well to compute likelihood measures.

Local indices These similarity metrics are computed considering first-order and second-order neighbors of a node, and are based on different heuristics. To mention a few: *Common Neighbors* ($CN(i,j) = |\Gamma_i \cap \Gamma_j|$) increases the likelihood of a connection if the candidate nodes have a substantial neighborhoods overlap [New01]; *Adamic-Adar Index* ($AA(i,j) = \sum_{k \in \Gamma_i \cap \Gamma_j} \frac{1}{\log |\Gamma_k|}$) promotes a connection if common neighbors have a low degree [AA03]; and *Preferential Attachment* ($PA(i,j) = |\Gamma_i| \cdot |\Gamma_j|$) assigns high similarity to larger degree nodes [BJN+02].

Global indices These similarity indices are computed using paths statistics from the whole graph: for instance, *Shortest-Path Index* ($SP(i,j) = \frac{1}{d(i,j)}$) and *Katz Index* ($Katz_\beta(i,j) = \sum_{l=1}^{\infty} \beta^l |\mathbf{A}^l|_{ij}, 0 < \beta < 1$) summarize the intuition that highly reachable pair of nodes are likely to form an edge [Kat53]. Since global indices generally outperform local ones, but require higher time for being computed, *quasi-local* indices like *Local Paths* ($LP(i,j) = \mathbf{A}^2 + \beta \mathbf{A}^3$, $0 < \beta < 1$) have been proposed as a trade-off between accuracy and complexity [LJZ09].

Latent feature similarity indices Matrix factorization methods and node embeddings capture structural patterns into low-rank latent representations, allowing their use as topological predictors. Computation of edge likelihoods is possible using pairwise distance scores [RZHW22] like the inner product $\mathbf{v}_i \cdot \mathbf{v}_j$ (the higher the better) and generalized vector distances $||\mathbf{v}_i - \mathbf{v}_j||_p$ (the lower the better) like Manhattan and Euclidean distances.

2.3.3 Supervised setting

Here we summarize the supervised setting for edge classification, so called because an additional classifier is trained on positive and negative example edges. Formally, the supervised edge classification setting [Sha20] on the input graph $\mathcal{G}$ is performed by train-test splits over the candidate pair sets $\mathcal{E}^{(+)}$ and $\mathcal{E}^{(-)}$ of the unsupervised setting, and it involves the following elements: training examples for edges and non-edges ($\mathcal{E}_{tr}^{(+)}$ and $\mathcal{E}_{tr}^{(-)}$); test edges and non-edges ($\mathcal{E}_{ts}^{(+)}$ and $\mathcal{E}_{ts}^{(-)}$); a parametrized score function $s : (\mathcal{E}_{tr}^{(+)} \cup \mathcal{E}_{ts}^{(+)}) \cup (\mathcal{E}_{tr}^{(-)} \cup \mathcal{E}_{ts}^{(-)}) \to \mathbb{R}$ that is trained to assign high

scores to training edges $\mathcal{E}_{tr}^{(+)}$ and low scores to training unconnected pairs $\mathcal{E}_{tr}^{(-)}$. Classifier parameters θ are the solution to the optimization problem:

$$\max_{\theta} \Pr[\, s(e;\theta) > s(\bar{e};\theta) \mid e \in \mathcal{E}_{tr}^{(+)}, \bar{e} \in \mathcal{E}_{tr}^{(-)}] \tag{2.20}$$

For train-test splits we refer to ρ and γ as the fractions of training examples randomly sampled respectively from the candidate edges and non-edges. For instance, in network reconstruction we have $|\mathcal{E}_{tr}^{(+)}| = \rho|\mathcal{E}|$, $|\mathcal{E}_{tr}^{(-)}| = \gamma|\mathcal{U}_{\mathcal{E}}|$ and $\mathcal{E}_{ts}^{(+)} = \mathcal{E} \setminus \mathcal{E}_{tr}^{(+)}$, $\mathcal{E}_{ts}^{(-)} = \mathcal{U}_{\mathcal{E}} \setminus \mathcal{E}_{tr}^{(-)}$. Instead for link predicition $|\mathcal{E}_{tr}^{(+)}| = \rho|\mathcal{E}^*|$, $|\mathcal{E}_{tr}^{(-)}| = \gamma|\mathcal{U}_{\mathcal{E}} \setminus \mathcal{E}^*|$ and $\mathcal{E}_{ts}^{(+)} = \mathcal{E}^* \setminus \mathcal{E}_{tr}^{(+)}$, $\mathcal{E}_{ts}^{(-)} = (\mathcal{U}_{\mathcal{E}} \setminus \mathcal{E}^*) \setminus \mathcal{E}_{tr}^{(-)}$.

Hand-crafted feature construction Feature extraction is a crucial point in supervised edge prediction learning. Generally, similarity indices and other node-based or edge-based topological features can be manually extracted and used to construct hand-engineered edge vectors. For handling domain-specific problems, non-topological features and other attributes could be utilized as well.

Node-level feature aggregation Low-dimensional node embeddings express latent topological features that can be used instead of hand-crafted ones. Arbitrary aggregation of pairwise node embeddings can be implemented to obtain link-level representations, used as input to a machine learning classifier. For example, in [GL16] different binary aggregation functions are used to provide the input to the classifier: average $\frac{\mathbf{v}_i+\mathbf{v}_j}{2}$, Hadamard product $\mathbf{v}_i * \mathbf{v}_j$, absolute difference $|\mathbf{v}_i - \mathbf{v}_j|$, and squared difference $|\mathbf{v}_i - \mathbf{v}_j|^2$.

CHAPTER 3

Representation Learning on Time-Varying Graphs via Higher-Order Skip-Gram with Negative Sampling

A great variety of natural and artificial systems can be represented as networks of elementary structural entities coupled by relations between them. The abstraction of such systems as networks helps us understand, predict and optimize their behaviour [New03, AB02]. In this sense, node and graph embeddings have been established as standard feature representations in many learning tasks [CZC18, GF18]. Node embedding methods map nodes into low-dimensional vectors that can be used to solve downstream tasks such as edge prediction, network reconstruction, and node classification.

Node embeddings have successfully achieved low-dimensional encoding of static network structures, but many real-world graphs are inherently dynamic [HS12, CFQS12]. Time-resolved networks also support important dynamical processes, such as epidemic or rumor spreading, cascading failures, consensus formation, etc. [BBV08]. Time-resolved node embeddings have been shown to yield improved performance for predicting the outcome of dynamical processes over networks, such as information diffusion and disease spreading [SOBC21], providing an estimation

of infection and contagion risk when used with contact tracing data.

Learning meaningful representations of time-resolved proximity networks can be of extreme importance when facing events such as epidemic outbreaks [KBL+20, GSQ+21]. The manual and automatic collection of time-resolved proximity graphs for contact tracing purposes presents an opportunity for quick identification of possible infection clusters and infection chains. Even before the COVID-19 pandemic, the use of wearable proximity sensors for collecting time-resolved proximity networks has been largely discussed in the literature and many approaches have been used to describe patterns of activity and community structure, and to study spreading patterns of infectious diseases [SPW+15, GPC14, GVF+15].

In this chapter and in [PP22] we propose a representation learning model that performs implicit tensor factorization on different higher-order representations of time-varying graphs. The main contributions are as follows:

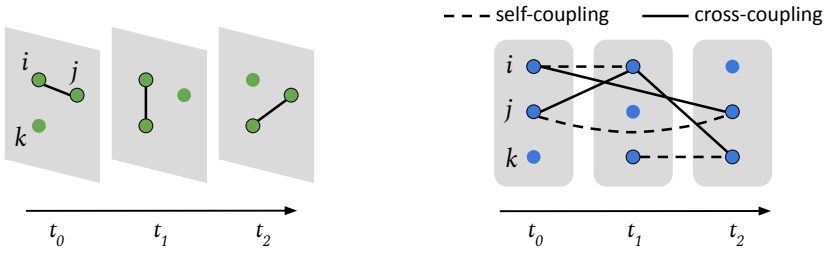
- Given that the skip-gram embedding approach implicitly performs a factorization of the shifted *Pointwise Mutual Information* matrix (PMI) [LG14b], we generalize it to perform implicit factorization of a shifted PMI tensor. We then define the steps to achieve this factorization using Higher-Order Skip-Gram with Negative Sampling (HOSGNS) optimization.
- We show how to apply 3rd-order and 4th-order SGNS on different higher-order representations of time-varying graphs.
- We show that time-varying graph representations learned via HOSGNS outperform state-of-the-art methods when used to solve downstream tasks, even using a fraction of the number of embedding parameters.

We report the results of learning embeddings on empirical time-resolved face-to-face proximity data and using such representations as predictors for solving three different tasks: predicting the outcomes of a SIR spreading process over the time-varying graph, network reconstruction, and link prediction. We compare these results with state-of-the-art methods for time-varying graph representation learning.

3.1 Preliminaries and Related Work

3.1.1 Time-varying graphs and their tensor representations

Time-varying graphs [HS12, CFQS12] are defined as triples $\mathcal{H} = (\mathcal{V}, \mathcal{E}_\mathcal{T}, \mathcal{T})$, i.e. collections of events $(i, j, k) \in \mathcal{E}_\mathcal{T}$, representing undirected pairwise relations among nodes at discrete times ($i, j \in \mathcal{V}$, $k \in \mathcal{T}$). $\mathcal{H}$ can be seen as a temporal sequence of static graphs $\{\mathcal{G}^{(k)}\}_{k\in\mathcal{T}}$, each of those with adjacency matrix $\mathbf{A}^{(k)}$ such that $\mathbf{A}^{(k)}_{ij} = \omega(i, j, k) \in \mathbb{R}$ is the weight of the event $(i, j, k) \in \mathcal{E}$. We can concatenate the list of time-stamped snapshots $[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(|\mathcal{T}|)}]$ to obtain a



(**a**) *time-varying graph* (**b**) *supra-adjacency graph*

Figure 3.1: *A time-varying graph $\mathcal{H}$ with three intervals (a) and its corresponding time-respecting supra-adjacency graph $\mathcal{G}_{\mathcal{H}}$ (b).*

single 3rd-order tensor $\boldsymbol{\mathcal{A}}^{stat}(\mathcal{H}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times|\mathcal{T}|}$, which characterizes the evolution of the graph over time. This representation has been used to discover latent community structures of temporal graphs [GPC14] and to perform temporal link prediction [DKA11].

Indeed, beyond the above stacked graph representation, more exhaustive representations are possible. In particular, the multi-layer approach [DDSRC+13] allows to map the topology of a time-varying graph $\mathcal{H}$ into a static network $\mathcal{G}_{\mathcal{H}} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$ (the *supra-adjacency* graph) such that vertices in $\mathcal{V}_{\mathcal{H}} \equiv \mathcal{V} \times \mathcal{T}$ correspond to *temporal copies* $i^{(k)} \equiv (i,k)$ of original nodes $i \in \mathcal{V}$, and edges in $\mathcal{E}_{\mathcal{H}}$ represent causal connections $(i^{(k)}, j^{(l)})$ among them. This mapping is analogous to the *time-unfolded* representation, which has been exploited in several works to study time-respecting paths and causal patterns in temporal networks [PSG+13]. Since every link can be arranged in a quadruple (i,j,k,l), the connectivity structure is associated with a 4th-order tensor $\boldsymbol{\mathcal{A}}^{dyn}(\mathcal{H}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times|\mathcal{T}|\times|\mathcal{T}|}$ that is equivalent, up to an opportune reshaping, to the adjacency matrix $\mathbf{A}(\mathcal{G}_{\mathcal{H}}) \in \mathbb{R}^{|\mathcal{V}||\mathcal{T}|\times|\mathcal{V}||\mathcal{T}|}$ of $\mathcal{G}_{\mathcal{H}}$. Multi-layer representations for time-varying networks have been used to study time-dependent centrality measures [TPM19] and properties of spreading processes [VFPC15].

In this chapter we consider a particular multi-layer representation [SOBC21] defined by two rules:

1. For each event (i,j,t_0), if i is also active at time $t_1 > t_0$ and in no other time-stamp between the two, we add a *cross-coupling* edge between supra-adjacency nodes $j^{(t_0)}$ and $i^{(t_1)}$. In addition, if the next interaction of j with other nodes happens at $t_2 > t_0$, we add an edge between $i^{(t_0)}$ and $j^{(t_2)}$. The weights of such edges are set to $\omega(i,j,t_0)$.

2. For every case as described above, we also add *self-coupling* edges $(i^{(t_0)}, i^{(t_1)})$ and $(j^{(t_0)}, j^{(t_2)})$, with weights set to 1.

Thus, the supra-adjacency graph has vertex set $\mathcal{V}^{(\mathcal{T})} = \{(i,t) \in \mathcal{V} \times \mathcal{T} : \exists\, (i,j,t) \in \mathcal{E}_{\mathcal{T}}\}$ consisting only in "active" node-time pairs, i.e. nodes with non-null interactions during time-steps. Figure 3.1 shows the differences between a time-varying graph and its time-unfolded supra-adjacency representation, according to the formulation described above.

3.1.2 Representation learning on time-varying graphs

Given a time-varying graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}}, \mathcal{T})$, we define as temporal network embedding the encoding function:

$$f : (v,t) \in \mathcal{V} \times \mathcal{T} \mapsto \mathbf{v}^{(t)} \in \mathbb{R}^D \tag{3.1}$$

which project time-stamped nodes into a latent low-rank vector space that incorporates structural and temporal properties of the original evolving graph [KGJ+20, BMVZ21].

Many existing methods learn node representations from sequences of static snapshots through incremental updates in a streaming scenario: deep autoencoders [GKHL17], SVD [ZCP+18], skip-gram [DWS+18, PLY+20] and random walk sampling [BKPB19, MKA18, YCA+18]. Another class of models learns dynamic node representations by recurrent/attention mechanisms [GCC20, LZP+18, SWG+20, XRK+20] or by imposing temporal stability among adjacent time intervals [ZYR+18, ZGY+16]. Moreover, [KZL19] presented an embedding framework for user-item temporal interactions, and [MUH+21] suggested a tensor-based convolutional architecture for dynamic graphs. In DYANE [SOBC21] and WEG2VEC [TKG20] the dynamic structure is projected into a static graph, in order to compute embeddings with WORD2VEC. Closely related to these ones are [NLR+18] and [ZLM+20], which learn WORD2VEC-based vectors according to time-respecting random walks or spreading trajectory paths. In addition, recently proposed methods learn node embeddings based on higher-order representations that capture non-Markovian properties in temporal networks [SCKC20, BKTK19].

Methods that perform well for predicting outcomes of spreading processes make use of time-respecting supra-adjacency representations such as the one proposed by [VFPC15]. In these graph representations, a random walk corresponds to a temporal path in the original time-varying graph, enconding relevant information about the spreading process into its connectivity structure. This is the case of the concurrent method DYANE, in which vector representations are computed for each time-stamped node $i^{(t)} \in \mathcal{V}^{(\mathcal{T})}$ of the supra-adjacency graph described in Section 3.1.1. Similar models that learn time-resolved node representations require a quantity $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{T}|)$ of embedding parameters to represent the time-varying graph in the latent space. As we will see, compared with these methods, our approach requires a quantity $\mathcal{O}(|\mathcal{V}| + |\mathcal{T}|)$ of embedding parameters for disentangled node and time representations.

3.2 Methods Description

3.2.1 Overview of the proposed method

Given a time-varying graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}}, \mathcal{T})$, we propose a representation learning method that learns disentangled representations for nodes and time slices, namely an encoding function $f^* : (v, t) \in \mathcal{V} \times \mathcal{T} \mapsto \mathbf{v}, \mathbf{t} \in \mathbb{R}^D$. This embedding representation can then be reconciled with the definition in Equation (3.1) by combining $\mathbf{v}$ and $\mathbf{t}$ in a single $\mathbf{v}^{(t)}$ representation using any aggregation function $\text{AGG} : (\mathbf{v}, \mathbf{t}) \in \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbf{v}^{(t)} \in \mathbb{R}^D$. It follows that computing and combining distinct vector embeddings for nodes and time slices needs a quantity $\mathcal{O}(|\mathcal{V}| + |\mathcal{T}|)$ of learnable parameters, leading to a more efficient method to obtain time-aware node representations without requiring to learn a much bigger number $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{T}|)$ of learnable parameters.

The parameters of the embedding representation encoded by f^* are learned through a higher-order generalization of skip-gram with negative sampling (HOSGNS), based on the existing skip-gram framework for node embeddings, as shown in Section 3.2.2. We show that this generalization allows to implicitly factorize higher-order relations that characterize tensor representations of time-varying graphs, in the same way that the classical SGNS decomposes dyadic relations associated with a static graph.

Similar approaches have been applied in NLP for dynamic word embeddings [RB18], and higher-order extensions of the skip-gram model have been proposed to learn context-dependent [LQH15] and syntactic-aware [CPVDE17] word representations. Also, tensor factorization techniques have been applied to include the temporal dimension in recommender systems [XCH+10, WSD+19], knowledge graphs [LOU20,MTD19] and face-to-face contact networks [SPW+15,GPC14]. But in this chapter, we empirically show how to merge SGNS with tensor factorization, and then apply it to learn time-varying graph embeddings. HOSGNS differs from existing temporal network embeddings based on skip-gram [DWS+18, PLY+20], which are minor adaptations of standard SGNS to the dynamic setting. In fact, in Section 3.2.2 we show how the equations in the skip-gram framework can be completely rewritten to be naturally applied to inherently higher-order problems.

In the next sections, we first show the formal steps to the generalization of the skip-gram approach to higher-order data structures, and then we show how to apply HOSGNS on 3rd-order and 4th-order representations of time-varying graphs. In the same spirit that WORD2VEC refers to the word pairs (i, j) as (*word, context*), here we refer to the node pairs (i, j) as (*node, context*), and the time pairs (k, l) as (*time, context-time*).

3.2.2 SGNS for higher-order data structures

Here we address the problem of generalizing SGNS to learn embedding representations from higher-order co-occurrences. In Section 3.1.1 we described snapshot-based and multilayer-based representations of time-varying graphs, that can be seen as 3rd and 4th-order co-occurrence tensors; therefore in the remainder of this chapter we focus on 3rd and 4th-order structures. In this section, we describe in detail the generalization of SGNS to the 3rd-order case. In Appendix A, we discuss in more detail the derivation of the HOSGNS objective function to any Nth-order representation.

We consider a collection of training samples $\mathcal{D} = \{(i,j,k) :\ i,j \in \mathcal{V},\ k \in \mathcal{T}\}$ obtained by gathering co-occurrences among pairs of nodes and time-steps. While SGNS is limited to pairs of node-context (i,j), here $\mathcal{D}$ is constructed with three (or more) variables, e.g. random sampling walks over a higher-order data structure. We denote as $\#(i,j,k)$ the number of times the triple (i,j,k) appears in $\mathcal{D}$. Similarly we use $\#i = \sum_{j,k} \#(i,j,k),\ \#j = \sum_{i,k} \#(i,j,k)$ and $\#k = \sum_{i,j} \#(i,j,k)$ as the number of times each distinct element occurs in $\mathcal{D}$, with relative frequencies $P_{\mathcal{D}}(i,j,k) = \frac{\#(i,j,k)}{|\mathcal{D}|}$, $P_{\mathcal{D}}(i) = \frac{\#i}{|\mathcal{D}|}$, $P_{\mathcal{D}}(j) = \frac{\#j}{|\mathcal{D}|}$ and $P_{\mathcal{D}}(k) = \frac{\#k}{|\mathcal{D}|}$.

Optimization is performed as a binary classification task, where the objective is to discern occurrences actually coming from $\mathcal{D}$ from random occurrences. We define the likelihood for a single observation (i,j,k) by applying a sigmoid function to the higher-order inner product $[\![\cdot]\!]$ of corresponding D-dimensional representations:

$$\Pr[\ (i,j,k) \in \mathcal{D} \mid \mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k\ ] = \sigma\big(\ [\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k]\!]\ \big) \equiv \sigma\left(\ \sum\nolimits_{r=1}^{D} \mathbf{V}_{ir}\mathbf{C}_{jr}\mathbf{T}_{kr}\ \right) \tag{3.2}$$

where embedding vectors $\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k \in \mathbb{R}^D$ are respectively rows of $\mathbf{V}, \mathbf{C} \in \mathbb{R}^{|\mathcal{V}| \times D}$ and $\mathbf{T} \in \mathbb{R}^{|\mathcal{T}| \times D}$. In the 4th-order case we will also have a fourth embedding matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{T}| \times D}$. For negative sampling we fix an observed $(i,j,k) \in \mathcal{D}$ and independently sample j_n and k_n to generate κ negative examples (i, j_n, k_n). In this way, for a single occurrence $(i,j,k) \in \mathcal{D}$, the expected contribution to the loss is:

$$\log \sigma\big([\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k]\!]\big) + \kappa \cdot \mathop{\mathbb{E}}_{j_n, k_n \sim P_{\mathcal{N}}} \Big[\log \sigma\big(-[\![\mathbf{v}_i, \mathbf{c}_{j_n}, \mathbf{t}_{k_n}]\!]\big)\Big] \tag{3.3}$$

where the noise distribution is the product of independent unigram probabilities $P_{\mathcal{N}}(j,k) = P_{\mathcal{D}}(j) \cdot P_{\mathcal{D}}(k)$. Thus the global objective is the sum of all the quantities of Equation (3.3) weighted with the corresponding relative frequency $P_{\mathcal{D}}(i,j,k)$. The full loss function can be expressed as:

$$\mathcal{L} = -\sum_{i,j,k} \Big[P_{\mathcal{D}}(i,j,k) \log \sigma\big([\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k]\!]\big) + \kappa\ P_{\mathcal{N}}(i,j,k) \log \sigma\big(-[\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k]\!]\big)\Big] \tag{3.4}$$

In Appendix A we show the formal steps to obtain Equation (3.4) for the Nth-order case and that it can be optimized with respect to the embedding parameters, satisfying the low-rank tensor approximation of the multivariate shifted PMI tensor into factor matrices $\mathbf{V}, \mathbf{C}, \mathbf{T}$:

$$\sum_{r=1}^{D} \mathbf{V}_{ir}\mathbf{C}_{jr}\mathbf{T}_{kr} \approx \log\left(\frac{P_{\mathcal{D}}(i,j,k)}{P_{\mathcal{N}}(i,j,k)}\right) - \log\kappa \equiv \mathrm{SPMI}_{\kappa}(i,j,k) \tag{3.5}$$

Equation (3.5), like the analogous derived in Levy and Goldberg [LG14b] in Equation (2.11), assumes full rank embedding matrices with $D \approx R = \mathrm{rank}(\mathrm{SPMI}_{\kappa})$. For the case when $D \ll R$, a comprehensive theoretical analysis is missing, although recent works propose the feasibility of exact low-dimensional factorizations of real-world static networks [CMST20, CMST21]. Nevertheless, in Appendix A, we include an empirical analysis of the effectiveness of HOSGNS for low-rank factorization of time-varying graph representations.

3.2.3 Low-dimensional embedding of time-varying graphs

While a static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is uniquely represented by an adjacency matrix $\mathbf{A}(\mathcal{G}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, a time-varying graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}_{\mathcal{T}}, \mathcal{T})$ admits diverse possible higher-order adjacency relations (Section 3.1.1). Starting from these higher-order relations, we can either use them directly or use random walk realizations to build a dataset of higher-order co-occurrences. In the same spirit that random walk realizations lead to dyadic co-occurrences used to learn embeddings in SGNS, we use higher-order co-occurrences to learn embeddings via HOSGNS.

As discussed in Section 3.2.2, the statistics of higher-order relations can be summarized in multivariate PMI tensors, which derive from co-occurrence probabilities among elements. Once such PMI tensors are constructed, we can again factorize them via HOSGNS. To show the versatility of this approach, we choose probability tensors derived from two different types of higher-order relations:

- A 3rd-order tensor $\boldsymbol{\mathcal{P}}^{(stat)}(\mathcal{H}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times|\mathcal{T}|}$, which gathers relative frequencies of nodes occurrences in temporal edges:

$$(\boldsymbol{\mathcal{P}}^{(stat)})_{ijk} = \frac{\omega(i,j,k)}{\mathrm{vol}(\mathcal{H})} \tag{3.6}$$

  where $\mathrm{vol}(\mathcal{H}) = \sum_{i,j,k}\omega(i,j,k)$ is the total weight of interactions occurring in $\mathcal{H}$. These probabilities are associated to the snapshot sequence representation $\boldsymbol{\mathcal{A}}^{stat}(\mathcal{H}) = [\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(|\mathcal{T}|)}]$ and contain information about the topological structure of $\mathcal{H}$.

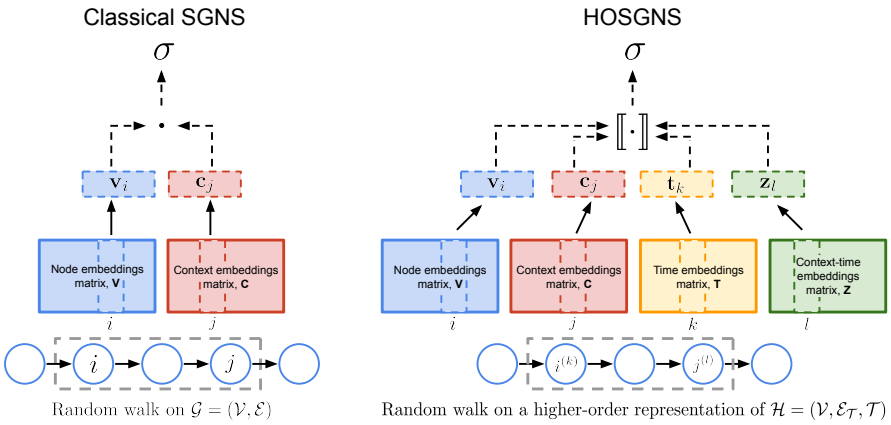


Figure 3.2: *Representation of SGNS and HOSGNS with embedding matrices and operations on embedding vectors. Starting from a random walk realization on a static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, SGNS takes as input nodes i and j within a context window of size T, and maximizes $\sigma(\mathbf{v}_i \cdot \mathbf{c}_j)$. HOSGNS starts from a random walk realization on a higher-order representation of time-varying graph $\mathcal{H} = (\mathcal{V}, \mathcal{E}_\mathcal{T}, \mathcal{T})$, takes as input nodes $i^{(k)}$ (node i at time k) and $j^{(l)}$ (node j at time l) within a context window of size T and maximizes $\sigma(\llbracket \mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, \mathbf{z}_l \rrbracket)$. In both cases, for each input sample, we fix i and draw κ combinations of j or j, k, l from a noise distribution, and we maximize $\sigma(-\mathbf{v}_i \cdot \mathbf{c}_j)$ (SGNS) or $\sigma(-\llbracket \mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, \mathbf{z}_l \rrbracket)$ (HOSGNS) with their corresponding embedding vectors (negative sampling).*

- A 4th-order tensor $\boldsymbol{\mathcal{P}}^{(dyn)}(\mathcal{H}) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|\times|\mathcal{T}|\times|\mathcal{T}|}$, which gathers occurrence probabilities of time-stamped nodes over random walks of the supra-adjacency graph $\mathcal{G}_\mathcal{H}$ (as used in DYANE). Using the numerator of Equation (2.15), with supra-adjacency indices $i^{(k)}$ and $j^{(l)}$ instead of usual indices i and j, tensor entries are given by:

$$(\boldsymbol{\mathcal{P}}^{(dyn)})_{ijkl} = \frac{1}{2T}\sum_{r=1}^{T}\left[\frac{d_{i^{(k)}}}{\mathrm{vol}(\mathcal{G}_\mathcal{H})}(\mathbf{P}^r)_{i^{(k)},j^{(l)}} + \frac{d_{j^{(l)}}}{\mathrm{vol}(\mathcal{G}_\mathcal{H})}(\mathbf{P}^r)_{j^{(l)},i^{(k)}}\right] \tag{3.7}$$

  These probabilities encode causal dependencies among temporal nodes and are correlated with the dynamical properties of spreading processes. Notice that the computation of $\boldsymbol{\mathcal{P}}^{(dyn)}(\mathcal{H})$ requires an undirected supra-adjacency graph, while in DYANE is directed.

We also combined the two representations in a single tensor that is the average of $\boldsymbol{\mathcal{P}}^{(stat)}$ and $\boldsymbol{\mathcal{P}}^{(dyn)}$:

$$(\boldsymbol{\mathcal{P}}^{(stat|dyn)})_{ijkl} = \frac{1}{2}\left[(\boldsymbol{\mathcal{P}}^{(stat)})_{ijk}\mathbb{1}[k=l] + (\boldsymbol{\mathcal{P}}^{(dyn)})_{ijkl}\right] \tag{3.8}$$

Figure 3.2 summarizes the differences between graph embedding via classical SGNS and time-varying graph embedding via HOSGNS. Here, indices (i, j, k, l) correspond to *(node, context, time, context-time)* in a 4th-order tensor representation of $\mathcal{H}$.

The above tensors gather empirical probabilities $P_{\mathcal{D}}(i, j, k, ..)$ corresponding to positive examples of observable higher-order relations. The probabilities of negative examples $P_{\mathcal{N}}(i, j, k, ..)$ can be obtained as the product of marginal distributions $P_{\mathcal{D}}(i) \cdot P_{\mathcal{D}}(j) \ldots$

Objective functions like Equation (3.4) can be computed with a sampling strategy: picking positive tuples according to the data distribution $P_{\mathcal{D}}$ and negative ones according to independent sampling $P_{\mathcal{N}}$, HOSGNS objective can be optimized through the following weighted cross-entropy loss:

$$\mathcal{L}^{(\text{bce})} = -\frac{1}{B}\Big[\sum_{(ijk\ldots)\sim P_{\mathcal{D}}}^{B} \log \sigma([\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, ..]\!]) + \kappa \sum_{(ijk\ldots)\sim P_{\mathcal{N}}}^{B} \log \sigma(-[\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, ..]\!])\Big] \tag{3.9}$$

where B is the number of samples drawn in a training step and κ is the negative sampling constant. We additionally apply the *warm-up* steps explained in the Appendix A to speed up the main training stage.

3.3 Experiments

For the experiments, we use time-varying graphs collected by the SocioPatterns collaboration (http://www.sociopatterns.org) using wearable proximity sensors that sense the face-to-face proximity relations of individuals wearing them. After training the proposed models (HOSGNS applied to $\boldsymbol{\mathcal{P}}^{(stat)}$, $\boldsymbol{\mathcal{P}}^{(dyn)}$ or $\boldsymbol{\mathcal{P}}^{(stat|dyn)}$) on each dataset, embedding matrices $\mathbf{V}, \mathbf{C}, \mathbf{T}$ (and $\mathbf{Z}$ except for $\boldsymbol{\mathcal{P}}^{(stat)}$) are mapped to embedding vectors $\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k$ (and $\mathbf{z}_l$) where $i, j \in \mathcal{V}$ and $k, l \in \mathcal{T}$. In the next paragraphs, we illustrate how datasets are processed, how we train HOSGNS, and how we use learned representations to solve different downstream tasks: *node classification*, *temporal event reconstruction*, and *missing event prediction*.

3.3.1 Datasets

We performed experiments with both empirical and synthetic datasets describing face-to-face proximity of individuals. We used publicly available empirical contact data collected by the SocioPatterns collaboration [CVdBB$^+$10], with a temporal resolution of 20 seconds, in a variety of contexts: in a school ("LyonSchool"), a conference ("SFHH"), a hospital ("LH10"), a high school ("Thiers13"), and in offices ("InVS15") [GB18]. This is currently the largest collection of open datasets sensing proximity in the same range and temporal resolution used by modern contact tracing systems. In addition, we used social interaction data generated by

Table 3.1: *Summary statistics of empirical and synthetic time-varying graph datasets. In order: number of single nodes $|\mathcal{V}|$, number of time-steps $|\mathcal{T}|$, number of events $|\mathcal{E}_\mathcal{T}|$, number of temporally-active nodes $|\mathcal{V}^{(\mathcal{T})}|$, average weight of events $\frac{1}{|\mathcal{E}_\mathcal{T}|}\sum_{e\in\mathcal{E}_\mathcal{T}}\omega(e)$, nodes density $\frac{|\mathcal{V}^{(\mathcal{T})}|}{|\mathcal{V}||\mathcal{T}|}$ and links density $\frac{2|\mathcal{E}_\mathcal{T}|}{|\mathcal{V}|(|\mathcal{V}|-1)|\mathcal{T}|}$.*

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{V}^{(\mathcal{T})}\rvert$	Average weight	Nodes density	Links density
LyonSchool	242	104	44,820	17,174	2.81	0.682	0.0148
SFHH	403	127	17,223	10,815	4.08	0.211	0.0017
LH10	76	321	7,435	4,880	4.45	0.200	0.0081
Thiers13	327	246	35,862	32,546	5.26	0.405	0.0027
InVS15	217	691	18,791	22,451	4.16	0.150	0.0012
OpenABM-2k-100	2,000	100	1,243,551	198,537	1.0	0.993	0.0062
OpenABM-5k-20	5,000	20	632,523	99,966	1.0	0.999	0.0025

the agent-based model OpenABM-Covid19 [HPN+21] to simulate an outbreak of COVID-19 in an urban setting.

We built a time-varying graph from each dataset, and for the empirical data we performed aggregation on 600 seconds time windows, neglecting those snapshots without registered interactions at that time-step. The choice of the time scale may have a substantial impact on the system's properties under study [KKB+12, RPB13], and in Appendix A we report a sensitivity analysis with the change of performance as the window length is altered. In a practical setting, the optimal aggregation width can be chosen based on domain knowledge, cross-validation, or –possibly– directly learned from data [SBWG10, LCF15, FC17]

The weight of the link (i, j, k) is the number of events recorded between nodes (i, j) in a certain aggregated window k. For synthetic data, we maintained the original temporal resolution and we set links weights to 1. Table 3.1 shows statistics for each dataset.

3.3.2 Parameter settings and baseline methods

HOSGNS variants are optimized with Adam [KB15] fixing the negative samples parameter $\kappa = 5$, linearly decaying the learning rate from a starting value of 0.05 for 4000 iterations. In the case of empirical datasets, the implementation of event sampling is made through the realization of a tensor with the probability distributions $\boldsymbol{\mathcal{P}}^{(stat)}$ and $\boldsymbol{\mathcal{P}}^{(dyn)}$ defined in Equations (3.6) and (3.7). For $\boldsymbol{\mathcal{A}}^{(dyn)}$ we set the random walks context window $T = 10$. Before training we apply 100 warm-up steps, fixing the sample size $B = 50000$.

In the case of synthetic datasets, due to the huge size and low sparsity of the

probabilities tensor, HOSGNS was implemented by sampling positive and negative events from a corpus of random walks. For HOSGNS$^{(stat)}$ random walks are sampled from the set of temporal snapshots $\{\mathcal{G}^{(k)}\}_{k\in\mathcal{T}}$ with window size $T = 1$, and for HOSGNS$^{(dyn)}$ random walks are sampled from the supra-adjacency graph $\mathcal{G}_{\mathcal{H}}$ with window size $T = 10$. With these sampling strategies, positive examples are drawn from the same probability distributions as in $\boldsymbol{\mathcal{P}}^{(stat)}$ and $\boldsymbol{\mathcal{P}}^{(dyn)}$. Embedding parameters are initialized with 1000 warm-up steps, fixing the batch size of positive examples to 20000.

We compare our approach with several baseline methods from the literature of time-varying graph embeddings, which learn time-stamped node representations:

- DYANE [SOBC21], learns temporal node embeddings training DEEPWALK on the supra-adjacency graph. As in the original paper, we optimized NODE2VEC[1] with default hyperparameters ($p = q = 1$, $\kappa = 5$, $walk\_length = 80$, $num\_walks = 10$ and the same context window size $T = 10$ that we chose for HOSGNS). The number of SGD epochs is 1 since we did not observe any improvement in downstream tasks by increasing the number of epochs.

- DYNGEM [GKHL17], a deep autoencoder architecture that dynamically reconstructs each graph snapshot initializing model weights with parameters learned in previous time frames. With the code made available online by the authors[2], we trained the model with SGD with momentum (learning rate 10^{-3} and momentum coefficient 0.99) for 100 iterations in the first time-step and 30 for the others. We set the internal layer sizes of the autoencoder to $[400, 250, D]$.

- DYNAMICTRIAD [ZYR$^+$18], which captures structural information and temporal patterns of nodes, modeling the *triadic closure* process. The model is trained using Adagrad (learning rate 10^{-1}) with 100 epochs and negative/positive samples ratio set to 5. Coefficients β_0 and β_1 related to social homophily and temporal smoothness are set to 0.1. We used the reference implementation available in the official repository[3].

- DYSAT [SWG$^+$20], a deep neural model that computes node embeddings by a joint self-attention mechanism applied to structural neighborhood and temporal dynamics. The algorithm is trained using the standard implementation[4] with Adam optimizer (initial learning rate 10^{-3}) for 100 epochs with

[1] https://github.com/snap-stanford/snap/tree/master/examples/node2vec

[2] http://www-scf.usc.edu/~nkamra/

[3] https://github.com/luckiezhou/DynamicTriad

[4] https://github.com/aravindsankar28/DySAT

window size for temporal attention set to 10, spatial and temporal drop-out probabilities equal to 0.1 and 0.5 respectively.

- ISGNS [PLY+20], an incremental skip-gram embedding model based on DEEPWALK. ISGNS is trained using the reference code[5] with standard NODE2VEC parameters (the same as DYANE).

3.3.3 Downstream tasks

Node classification

The aim of this task is to classify nodes in epidemic states according to a SIR process with infection rate β and recovery rate μ. We simulated 30 realizations of the SIR process on top of each empirical graph with different combinations of parameters (β, μ). We used similar combinations of epidemic parameters and the same dynamical process to produce SIR states as described in [SOBC21]. Then we set a logistic regression to classify epidemic states S-I-R assigned to each active node $i^{(k)}$ during the unfolding of the spreading process. We combine the embedding vectors of HOSGNS using the Hadamard (element-wise) product $\mathbf{v}_i * \mathbf{t}_k$, compared with dynamic node embeddings learned from baselines.

Temporal event reconstruction

In this task, we aim to determine if a generic event (i, j, k) (occurred or not) is in $\mathcal{H} = (\mathcal{V}, \mathcal{E}_\mathcal{T}, \mathcal{T})$, i.e. if there is an edge between nodes i and j at time k. We create a random time-varying graph $\bar{\mathcal{H}} = (\mathcal{V}, \bar{\mathcal{E}_\mathcal{T}}, \mathcal{T})$ with same active nodes $\mathcal{V}^{(\mathcal{T})}$ and a number of $|\mathcal{E}_\mathcal{T}|$ events that are not part of $\mathcal{E}_\mathcal{T}$ (i.e., $|\mathcal{E}_\mathcal{T}| = |\bar{\mathcal{E}_\mathcal{T}}|$ and $|\mathcal{E}_\mathcal{T} \cap \bar{\mathcal{E}_\mathcal{T}}| = 0$). In other words, $\bar{\mathcal{E}_\mathcal{T}}$ contains random events that may occur only between the nodes that are active in each snapshot, disregarding other possible edges that involve inactive nodes. Embedding representations learned from $\mathcal{H}$ are used as features to train a logistic regression to predict if a given event (i, j, k) is in $\mathcal{E}_\mathcal{T}$ or in $\bar{\mathcal{E}_\mathcal{T}}$. We combine the embedding vectors of HOSGNS as follows: for HOSGNS$^{(stat)}$, we use the Hadamard product $\mathbf{v}_i * \mathbf{c}_j * \mathbf{t}_k$; for HOSGNS$^{(dyn)}$ and HOSGNS$^{(stat|dyn)}$, we use $\mathbf{v}_i * \mathbf{c}_j * \mathbf{t}_k * \mathbf{z}_k$. For baseline methods, we aggregate vector embeddings to obtain link-level representations with binary operators (*Average*, *Hadamard*, *Weighted-L1*, *Weighted-L2* and *Concat*) as already used in previous works [GL16, TMKM18].

Missing event prediction

In this task, we aim to predict the occurrence of an event (i, j, k) *previously removed* from $\mathcal{H} = (\mathcal{V}, \mathcal{E}_\mathcal{T}, \mathcal{T})$. We create a pruned time-varying graph $\mathcal{H}^\dagger = (\mathcal{V}, \mathcal{E}^\dagger_\mathcal{T}, \mathcal{T})$

[5] https://github.com/RingBDStack/dynamic_network_embedding

with the same active nodes $\mathcal{V}^{(\mathcal{T})}$ and a number of events $|\mathcal{E}_{\mathcal{T}}^{\dagger}| = 70\%\ |\mathcal{E}_{\mathcal{T}}|$ sampled from $\mathcal{H}$. Embedding representations learned from $\mathcal{H}^{\dagger}$ are used as features to train a logistic regression to predict missing occurred events $(i, j, k) \in \mathcal{E}_{\mathcal{T}} \setminus \mathcal{E}_{\mathcal{T}}^{\dagger}$ against the events $\bar{\mathcal{E}_{\mathcal{T}}}$ of a random time-varying graph $\bar{\mathcal{H}} = (\mathcal{V}, \bar{\mathcal{E}_{\mathcal{T}}}, \mathcal{T})$ (see the construction above). We combine the embedding vectors of HOSGNS for the classification task as explained in the event reconstruction task.

3.4 Results

In this section, we first show downstream task performance results for the empirical and synthetic datasets, then we compare the different approaches in terms of training complexity (Section 3.4.1), by measuring the number of trainable parameters and the training time with a fixed number of training steps. Finally, we show the visualization of the two-dimensional projections of the embeddings for one of the chosen empirical datasets (Section 3.4.2).

3.4.1 Task performances and training complexity

Tasks were evaluated using train-test splits. To avoid information leakage from training to test, we randomly split $\mathcal{V}$ and $\mathcal{T}$ in train and test sets $(\mathcal{V}_{tr}, \mathcal{V}_{ts})$ and $(\mathcal{T}_{tr}, \mathcal{T}_{ts})$, with proportion $70\% - 30\%$. For node classification, only nodes in $\mathcal{V}_{tr}$ at times in $\mathcal{T}_{tr}$ were included in the train set, and only nodes in $\mathcal{V}_{ts}$ at times in $\mathcal{T}_{ts}$ were included in the test set. For event reconstruction and prediction, only events with $i, j \in \mathcal{V}_{tr}$ and $k \in \mathcal{T}_{tr}$ were included in the train set, and only events with $i, j \in \mathcal{V}_{ts}$ and $k \in \mathcal{T}_{ts}$ were included in the test set.

All approaches were evaluated for downstream tasks in terms of Macro-F1 scores in all datasets. For a fair comparison, all models produce time-stamped node representations with dimension $D = 128$ and event representations with dimension $D = 192$, as input to the logistic regression. Five different runs of the embedding model are evaluated on 30 different train-test splits in every downstream task. We report the average score with standard error over all splits. In node classification, every SIR realization is assigned to a single embedding run to compute prediction scores. In event reconstruction and prediction tasks, a different random time-varying graph realization $\bar{\mathcal{H}}$ to produce samples of non-occurring events is assigned to each train-test subset.

Empirical datasets

Results for the classification of nodes in epidemic states are shown in Table 3.2. DYNGEM and DYNAMICTRIAD have low scores, since they are not devised to learn from graph dynamics. Also DYSAT has a bad performance in this task, since this method uses a context prediction objective that preserves the local structure

Table 3.2: *Macro-F1 scores on node classification of epidemic states according to different SIR processes over empirical datasets. For each (β, μ) we highlight the two highest scores and underline the best one.*

(β, μ)	Model	Dataset LYONSCHOOL	SFHH	LH10	THIERS13	INVS15
(0.25,0.002)	DYANE	**78.1 ± 0.5**	67.0 ± 1.2	52.5 ± 1.7	**71.9 ± 0.6**	**64.3 ± 0.8**
	DYNGEM	58.7 ± 2.8	35.9 ± 1.1	34.5 ± 0.7	35.5 ± 1.2	58.8 ± 1.1
	DYNAMICTRIAD	31.0 ± 0.4	28.8 ± 0.4	29.9 ± 0.3	30.3 ± 0.2	30.4 ± 0.2
	DYSAT	27.3 ± 0.2	27.4 ± 0.3	29.7 ± 0.2	30.2 ± 0.2	30.5 ± 0.2
	ISGNS	63.5 ± 0.6	60.7 ± 0.8	54.1 ± 1.1	56.4 ± 0.6	52.3 ± 0.6
	HOSGNS$^{(stat)}$	55.5 ± 0.8	57.3 ± 1.1	45.9 ± 0.9	46.9 ± 0.7	44.5 ± 0.7
	HOSGNS$^{(dyn)}$	**79.2 ± 0.5**	**69.1 ± 1.1**	**59.6 ± 1.5**	71.8 ± 1.2	**64.6 ± 0.7**
	HOSGNS$^{(stat\|dyn)}$	77.4 ± 0.6	**67.4 ± 1.2**	**59.7 ± 1.2**	**72.5 ± 0.7**	64.2 ± 1.0
(0.0625,0.002)	DYANE	72.2 ± 0.6	64.9 ± 1.7	**59.0 ± 1.2**	68.0 ± 0.5	**60.2 ± 0.5**
	DYNGEM	56.4 ± 2.7	35.9 ± 4.1	35.8 ± 1.2	32.9 ± 1.2	55.0 ± 0.6
	DYNAMICTRIAD	29.5 ± 0.5	33.1 ± 2.5	29.6 ± 0.4	27.4 ± 0.3	28.4 ± 0.2
	DYSAT	26.4 ± 0.2	29.5 ± 1.3	29.5 ± 0.3	26.5 ± 0.2	28.5 ± 0.2
	ISGNS	59.2 ± 0.3	57.1 ± 1.6	55.9 ± 1.0	49.0 ± 0.3	47.2 ± 0.3
	HOSGNS$^{(stat)}$	55.5 ± 0.7	57.6 ± 2.2	49.4 ± 0.8	45.5 ± 0.4	43.6 ± 0.5
	HOSGNS$^{(dyn)}$	**73.5 ± 0.5**	**65.7 ± 1.6**	**61.1 ± 1.2**	**69.5 ± 0.3**	**59.6 ± 0.5**
	HOSGNS$^{(stat\|dyn)}$	**72.9 ± 0.6**	**66.3 ± 1.9**	58.2 ± 1.1	**68.5 ± 0.4**	59.0 ± 0.7
(0.1875,0.001)	DYANE	**74.7 ± 0.7**	67.7 ± 1.2	**63.4 ± 1.8**	72.7 ± 0.4	**68.6 ± 0.4**
	DYNGEM	57.4 ± 2.8	36.2 ± 2.6	41.4 ± 1.3	34.8 ± 1.3	61.2 ± 0.9
	DYNAMICTRIAD	32.3 ± 0.5	31.5 ± 0.8	30.5 ± 0.4	27.9 ± 0.3	30.0 ± 0.2
	DYSAT	26.4 ± 0.2	29.4 ± 0.8	30.0 ± 0.3	27.7 ± 0.3	29.9 ± 0.2
	ISGNS	65.1 ± 0.5	63.0 ± 1.4	60.2 ± 1.7	56.0 ± 0.5	52.5 ± 0.5
	HOSGNS$^{(stat)}$	56.9 ± 0.8	59.4 ± 1.7	48.5 ± 1.1	49.0 ± 0.6	46.2 ± 0.8
	HOSGNS$^{(dyn)}$	**76.5 ± 0.4**	**68.6 ± 1.1**	62.4 ± 1.7	**74.8 ± 0.5**	**67.9 ± 0.7**
	HOSGNS$^{(stat\|dyn)}$	74.5 ± 0.4	**69.4 ± 1.4**	**62.5 ± 2.0**	**73.6 ± 0.6**	67.3 ± 0.5
(0.125, 0.002)	DYANE	**77.1 ± 0.4**	**68.4 ± 0.9**	54.8 ± 1.5	71.6 ± 0.4	62.4 ± 0.5
	DYNGEM	57.1 ± 2.7	32.8 ± 1.3	35.0 ± 0.8	34.4 ± 1.0	57.1 ± 0.7
	DYNAMICTRIAD	30.4 ± 0.4	29.3 ± 0.4	30.1 ± 0.3	29.0 ± 0.3	29.4 ± 0.2
	DYSAT	27.0 ± 0.1	27.1 ± 0.3	30.3 ± 0.3	28.3 ± 0.3	29.4 ± 0.2
	ISGNS	61.8 ± 0.4	58.2 ± 0.7	54.6 ± 1.2	51.4 ± 0.3	49.1 ± 0.4
	HOSGNS$^{(stat)}$	55.4 ± 0.9	55.9 ± 0.8	44.9 ± 1.0	46.3 ± 0.4	44.8 ± 0.6
	HOSGNS$^{(dyn)}$	**77.5 ± 0.5**	**68.8 ± 0.8**	**58.7 ± 1.1**	**72.6 ± 0.5**	**63.3 ± 0.6**
	HOSGNS$^{(stat\|dyn)}$	75.2 ± 0.6	68.1 ± 0.8	**59.7 ± 1.1**	**72.0 ± 0.5**	**63.4 ± 0.6**
(0.125, 0.001)	DYANE	**75.3 ± 0.4**	**71.6 ± 1.9**	59.0 ± 1.8	72.4 ± 0.3	65.8 ± 0.6
	DYNGEM	58.9 ± 2.9	37.0 ± 4.1	41.0 ± 1.4	32.5 ± 1.2	59.0 ± 1.2
	DYNAMICTRIAD	31.2 ± 0.5	35.0 ± 3.3	30.5 ± 0.7	27.4 ± 0.3	29.5 ± 0.2
	DYSAT	25.9 ± 0.2	30.4 ± 1.2	30.3 ± 0.7	26.9 ± 0.2	29.3 ± 0.2
	ISGNS	65.5 ± 0.5	59.4 ± 0.8	57.9 ± 1.3	54.0 ± 0.4	50.6 ± 0.4
	HOSGNS$^{(stat)}$	56.8 ± 0.9	61.8 ± 2.4	49.1 ± 1.9	47.3 ± 0.6	45.9 ± 0.7
	HOSGNS$^{(dyn)}$	**76.0 ± 0.4**	**71.5 ± 2.0**	**59.6 ± 2.0**	**74.2 ± 0.4**	**65.9 ± 0.6**
	HOSGNS$^{(stat\|dyn)}$	74.6 ± 0.4	70.2 ± 1.9	**59.9 ± 2.3**	**74.8 ± 0.4**	**66.0 ± 0.6**

without properly encoding dynamical patterns. HOSGNS$^{(stat)}$ is not able to capture the graph dynamics due to the static nature of $\boldsymbol{\mathcal{P}}^{(stat)}$. ISGNS, due to the incremental training, performs only marginally better than HOSGNS$^{(stat)}$. DYANE,

Table 3.3: *Macro-F1 scores on temporal event reconstruction over empirical datasets. We highlight in bold the two best scores for each dataset. For baseline models, we underline their highest score.*

Model	Operator	Dataset LYONSCHOOL	SFHH	LH10	THIERS13	INVS15
DYANE	Average	56.4 ± 0.4	52.9 ± 0.5	52.3 ± 0.6	51.0 ± 0.4	52.7 ± 0.4
	Hadamard	89.7 ± 0.3	$\underline{86.5} \pm 0.3$	$\underline{74.6} \pm 0.6$	94.7 ± 0.1	94.1 ± 0.1
	Weighted-L1	90.2 ± 0.2	83.3 ± 0.5	73.3 ± 0.7	94.7 ± 0.1	94.4 ± 0.2
	Weighted-L2	$\underline{90.6} \pm 0.2$	84.5 ± 0.5	72.0 ± 0.5	$\underline{95.0} \pm 0.1$	$\underline{94.8} \pm 0.2$
	Concat	65.7 ± 0.4	53.8 ± 0.4	56.2 ± 0.6	57.0 ± 0.4	50.9 ± 0.4
DYNGEM	Average	57.7 ± 0.5	56.8 ± 0.7	$\underline{54.8} \pm 1.5$	40.4 ± 1.5	42.8 ± 0.9
	Hadamard	$\underline{62.2} \pm 0.4$	55.1 ± 1.0	52.5 ± 1.6	40.8 ± 1.5	43.7 ± 1.0
	Weighted-L1	58.4 ± 0.6	52.3 ± 0.7	50.9 ± 1.2	$\underline{41.3} \pm 1.6$	44.8 ± 0.9
	Weighted-L2	53.7 ± 0.6	47.0 ± 0.8	47.0 ± 1.3	39.2 ± 1.2	43.6 ± 0.6
	Concat	60.4 ± 0.4	$\underline{57.8} \pm 0.3$	48.9 ± 1.7	36.9 ± 1.3	$\underline{45.7} \pm 1.0$
DYNAMICTRIAD	Average	51.7 ± 0.2	56.9 ± 0.4	60.2 ± 0.6	58.1 ± 0.2	56.1 ± 0.3
	Hadamard	60.3 ± 0.3	58.9 ± 0.4	59.5 ± 0.5	62.2 ± 0.3	64.7 ± 0.3
	Weighted-L1	$\underline{79.1} \pm 0.4$	72.3 ± 0.4	75.5 ± 0.6	70.8 ± 0.3	78.1 ± 0.2
	Weighted-L2	77.4 ± 0.4	$\underline{73.4} \pm 0.4$	$\underline{77.4} \pm 0.5$	$\underline{72.4} \pm 0.2$	$\underline{78.9} \pm 0.3$
	Concat	52.2 ± 0.2	53.4 ± 0.3	55.9 ± 0.7	55.1 ± 0.2	53.2 ± 0.3
DYSAT	Average	51.1 ± 0.3	49.6 ± 0.4	51.6 ± 0.5	50.4 ± 0.2	50.1 ± 0.3
	Hadamard	$\underline{75.1} \pm 0.5$	$\underline{52.9} \pm 0.3$	54.8 ± 0.6	$\underline{71.1} \pm 0.4$	$\underline{66.8} \pm 0.5$
	Weighted-L1	72.4 ± 0.5	51.5 ± 0.3	56.1 ± 0.6	66.4 ± 0.4	64.8 ± 0.3
	Weighted-L2	72.4 ± 0.5	51.7 ± 0.3	$\underline{56.8} \pm 0.7$	66.5 ± 0.4	63.7 ± 0.4
	Concat	50.0 ± 0.3	50.1 ± 0.4	52.3 ± 0.5	49.8 ± 0.2	50.9 ± 0.3
ISGNS	Average	53.4 ± 0.4	50.3 ± 0.5	48.1 ± 0.6	49.4 ± 0.4	45.9 ± 0.5
	Hadamard	$\underline{90.1} \pm 0.3$	87.2 ± 0.4	80.8 ± 0.7	96.7 ± 0.2	96.7 ± 0.2
	Weighted-L1	89.9 ± 0.3	87.7 ± 0.4	81.6 ± 0.4	96.8 ± 0.2	96.4 ± 0.2
	Weighted-L2	89.7 ± 0.3	$\mathbf{\underline{88.2} \pm 0.4}$	$\mathbf{\underline{81.7} \pm 0.5}$	$\mathbf{\underline{96.9} \pm 0.1}$	$\mathbf{\underline{96.8} \pm 0.2}$
	Concat	57.1 ± 0.5	50.2 ± 0.4	48.8 ± 0.7	52.7 ± 0.4	43.8 ± 0.4
HOSGNS$^{(stat)}$	Hadamard	$\mathbf{98.5 \pm 0.1}$	$\mathbf{98.8 \pm 0.1}$	$\mathbf{99.8 \pm 0.1}$	$\mathbf{99.6 \pm 0.1}$	$\mathbf{99.1 \pm 0.1}$
HOSGNS$^{(dyn)}$	Hadamard	90.3 ± 0.2	80.9 ± 0.4	68.1 ± 0.7	93.5 ± 0.2	87.2 ± 0.2
HOSGNS$^{(stat\|dyn)}$	Hadamard	$\mathbf{91.8 \pm 0.2}$	86.7 ± 0.4	73.6 ± 0.6	94.3 ± 0.1	89.0 ± 0.2

HOSGNS$^{(stat|dyn)}$ and HOSGNS$^{(dyn)}$ show good performance, with these two HOSGNS variants outperforming DYANE in most of the combinations of datasets and SIR parameters.

Results for the temporal event reconstruction task are reported in Table 3.3. Temporal event reconstruction is not performed well by DYNGEM. DYNAMICTRIAD has better performance with Weighted-L1 and Weighted-L2 operators, while DYANE, DYSAT and ISGNS have better performance using Hadamard and Weighted-L2. ISGNS has the second best performance in most of the datasets. Since Hadamard product is explicitly used in Equation (3.2) to optimize HOSGNS, all HOSGNS variants show the best scores with this operator. HOSGNS$^{(stat)}$ outperforms all approaches, setting new state-of-the-art results in this task. The $\boldsymbol{\mathcal{P}}^{(dyn)}$ representation used as input to HOSGNS$^{(dyn)}$ does not focus on events but on dynamics, so the performance for event reconstruction is slightly below DYANE, while

Table 3.4: *Macro-F1 scores on missing event prediction over empirical datasets. We highlight in bold the two best scores for each dataset. For baseline models, we underline their highest score.*

Model	Operator	Dataset LyonSchool	SFHH	LH10	Thiers13	InVS15
DyANE	Average	56.8 ± 0.6	50.6 ± 0.8	51.3 ± 1.0	49.1 ± 0.6	49.3 ± 0.8
	Hadamard	87.3 ± 0.3	73.5 ± 0.6	$\underline{67.0} \pm 1.0$	$\underline{87.2} \pm 0.3$	$\underline{80.1} \pm 0.8$
	Weighted-L1	87.8 ± 0.3	73.3 ± 0.6	65.9 ± 1.0	84.0 ± 0.4	78.4 ± 0.6
	Weighted-L2	$\underline{88.5} \pm 0.2$	$\underline{73.7} \pm 0.5$	66.1 ± 1.0	84.4 ± 0.4	78.9 ± 0.6
	Concat	64.4 ± 0.5	52.4 ± 0.8	51.9 ± 1.0	57.0 ± 0.6	51.4 ± 0.7
DynGEM	Average	56.2 ± 0.5	$\underline{51.8} \pm 0.8$	$\underline{52.0} \pm 1.1$	49.7 ± 0.5	50.9 ± 0.7
	Hadamard	54.8 ± 0.6	51.3 ± 0.7	51.7 ± 1.2	44.7 ± 0.7	$\underline{50.9} \pm 0.6$
	Weighted-L1	55.5 ± 0.4	48.5 ± 0.8	50.2 ± 1.0	$\underline{52.2} \pm 0.4$	49.8 ± 0.7
	Weighted-L2	53.2 ± 0.7	47.8 ± 0.9	48.0 ± 1.1	48.9 ± 0.6	45.3 ± 0.6
	Concat	$\underline{58.2} \pm 0.5$	50.4 ± 0.8	46.4 ± 1.4	48.8 ± 0.5	49.9 ± 0.6
DynamicTriad	Average	51.4 ± 0.4	52.6 ± 0.6	53.0 ± 0.8	52.0 ± 0.4	49.9 ± 0.7
	Hadamard	53.1 ± 0.4	49.5 ± 0.6	52.0 ± 0.8	51.7 ± 0.5	49.8 ± 0.6
	Weighted-L1	64.3 ± 0.4	56.6 ± 0.7	54.2 ± 0.9	53.6 ± 0.4	47.2 ± 0.6
	Weighted-L2	$\underline{64.5} \pm 0.4$	$\underline{57.3} \pm 0.7$	$\underline{54.9} \pm 0.9$	$\underline{54.5} \pm 0.5$	47.0 ± 0.6
	Concat	52.6 ± 0.3	51.8 ± 0.5	52.7 ± 0.9	51.5 ± 0.3	$\underline{49.9} \pm 0.6$
DySAT	Average	51.3 ± 0.4	51.6 ± 0.6	52.5 ± 0.8	50.0 ± 0.4	50.3 ± 0.6
	Hadamard	$\underline{73.8} \pm 0.6$	$\underline{52.5} \pm 0.7$	56.6 ± 0.7	$\underline{68.5} \pm 0.5$	61.5 ± 0.8
	Weighted-L1	71.3 ± 0.5	52.0 ± 0.6	$\underline{57.6} \pm 0.8$	63.2 ± 0.6	$\underline{64.4} \pm 0.5$
	Weighted-L2	70.7 ± 0.5	51.5 ± 0.7	56.5 ± 0.8	63.1 ± 0.5	63.4 ± 0.5
	Concat	49.2 ± 0.4	48.8 ± 0.8	52.4 ± 0.9	49.8 ± 0.5	50.4 ± 0.6
ISGNS	Average	52.4 ± 0.6	49.5 ± 0.8	44.9 ± 0.9	48.0 ± 0.4	42.7 ± 0.8
	Hadamard	79.8 ± 0.4	59.3 ± 0.7	61.1 ± 1.2	59.3 ± 0.6	$\underline{51.7} \pm 0.7$
	Weighted-L1	80.8 ± 0.3	59.8 ± 0.7	61.7 ± 1.0	59.0 ± 0.6	49.8 ± 0.7
	Weighted-L2	$\underline{81.5} \pm 0.3$	$\underline{60.2} \pm 0.7$	$\underline{62.5} \pm 0.9$	$\underline{59.9} \pm 0.6$	51.5 ± 0.7
	Concat	55.8 ± 0.7	50.8 ± 0.6	46.8 ± 0.8	52.2 ± 0.5	48.5 ± 0.6
HOSGNS$^{(stat)}$	Hadamard	52.1 ± 0.4	43.8 ± 0.6	34.2 ± 0.2	55.9 ± 0.6	43.0 ± 0.5
HOSGNS$^{(dyn)}$	Hadamard	$\mathbf{89.2 \pm 0.2}$	$\mathbf{74.9 \pm 0.6}$	$\mathbf{67.1 \pm 0.8}$	$\mathbf{90.7 \pm 0.3}$	$\mathbf{81.4 \pm 0.5}$
HOSGNS$^{(stat\|dyn)}$	Hadamard	$\mathbf{89.2 \pm 0.3}$	$\mathbf{76.3 \pm 0.7}$	$\mathbf{68.5 \pm 1.0}$	$\mathbf{89.9 \pm 0.3}$	$\mathbf{80.8 \pm 0.6}$

HOSGNS$^{(stat|dyn)}$ is comparable to DyANE.

Table 3.4 outlines the results for the missing event prediction task. In this case HOSGNS$^{(stat)}$ has lower performance but is comparable with DynGEM and DynamicTriad. DySAT and ISGNS works slightly better with Hadamard or Weighted-L1/L2 operator, but they are outperformed by DyANE that has an excellent performance with Hadamard or Weighted-L2. However, HOSGNS$^{(dyn)}$ and HOSGNS$^{(stat|dyn)}$ have the best scores, which emphasize the importance of leveraging dynamics to learn and predict missing information.

We observe an overall good performance of HOSGNS$^{(stat|dyn)}$ in all downstream tasks, being in almost all cases among the two highest scores, compared to the other two HOSGNS variants which excel in certain tasks but have lower performance in the others.

Synthetic datasets

Here we report the performance of downstream tasks with the two synthetic datasets only for HOSGNS$^{(stat)}$ and HOSGNS$^{(dyn)}$, given the similar performance of HOSGNS$^{(dyn)}$ and HOSGNS$^{(stat|dyn)}$ in previous experiments. We also chose DYANE as the only baseline, given its better performance compared to other baselines in empirical datasets.

Results for the node classification task are reported in Table 3.5, reflecting previous results on empirical datasets, with HOSGNS$^{(dyn)}$ performance comparable or superior to DYANE. Results for the event reconstruction and prediction tasks are reported in Table 3.6. DYANE performs well with Hadamard operation, but nevertheless the scores are below HOSGNS$^{(dyn)}$ and HOSGNS$^{(stat)}$ scores. Especially with HOSGNS$^{(stat)}$, the performance of event reconstruction is not much larger than even prediction, contrary to empirical datasets. This difference might be due to the different topological features of synthetic networks with respect to empirical ones.

Table 3.5: *Macro-F1 scores on node classification of epidemic states according to different SIR processes over synthetic datasets. For each (β, μ) we highlight the best score.*

(β, μ)	Model	Dataset	
		OPENABM-2k-100	OPENABM-5k-20
(0.25,0.002)	DYANE	$\mathbf{57.9 \pm 1.8}$	59.6 ± 1.7
	HOSGNS$^{(stat)}$	31.2 ± 0.1	27.8 ± 0.6
	HOSGNS$^{(dyn)}$	57.5 ± 1.8	$\mathbf{61.0 \pm 1.1}$
(0.0625,0.002)	DYANE	$\mathbf{61.8 \pm 0.4}$	53.8 ± 1.3
	HOSGNS$^{(stat)}$	29.8 ± 0.2	29.4 ± 1.4
	HOSGNS$^{(dyn)}$	59.5 ± 0.9	$\mathbf{54.5 \pm 1.4}$
(0.1875,0.001)	DYANE	60.3 ± 1.4	59.6 ± 1.5
	HOSGNS$^{(stat)}$	31.9 ± 0.2	27.4 ± 0.7
	HOSGNS$^{(dyn)}$	$\mathbf{60.5 \pm 1.1}$	$\mathbf{60.9 \pm 1.0}$
$(0.125, 0.002)$	DYANE	$\mathbf{60.7 \pm 1.1}$	$\mathbf{61.3 \pm 0.6}$
	HOSGNS$^{(stat)}$	30.8 ± 0.1	27.4 ± 1.2
	HOSGNS$^{(dyn)}$	58.9 ± 1.4	60.7 ± 0.6
$(0.1875, 0.001)$	DYANE	60.3 ± 1.4	59.6 ± 1.5
	HOSGNS$^{(stat)}$	31.9 ± 0.2	27.4 ± 0.7
	HOSGNS$^{(dyn)}$	$\mathbf{60.5 \pm 1.1}$	$\mathbf{60.9 \pm 1.0}$

Training complexity

We report in Table 3.7 the number of trainable parameters and training time duration for each considered algorithm, when applied to an empirical graph (LYONSCHOOL) and to the synthetic ones. The proposed HOSGNS model requires a number of

Table 3.6: *Macro-F1 scores on temporal event reconstruction and missing event prediction over synthetic datasets. We highlight in bold the best two scores for each dataset. For baseline models, we underline their highest score.*

Model	Operator	Dataset			
		OPENABM-2k-100		OPENABM-5k-20	
		Reconstruction	Prediction	Reconstruction	Prediction
DYANE	Average	52.2 ± 0.1	51.7 ± 0.1	51.9 ± 0.1	51.9 ± 0.1
	Hadamard	$\underline{76.4} \pm 0.1$	$\underline{72.4} \pm 0.2$	$\mathbf{\underline{90.5}} \pm \mathbf{0.3}$	$\underline{77.8} \pm 0.2$
	Weighted-L1	70.3 ± 0.1	67.4 ± 0.2	78.2 ± 0.7	70.5 ± 0.3
	Weighted-L2	70.3 ± 0.1	67.7 ± 0.1	78.8 ± 0.5	70.9 ± 0.3
	Concat	53.8 ± 0.1	54.6 ± 0.1	52.5 ± 0.1	52.5 ± 0.2
HOSGNS$^{(stat)}$	Hadamard	$\mathbf{91.1 \pm 0.1}$	$\mathbf{87.0 \pm 0.1}$	$\mathbf{98.7 \pm 0.1}$	$\mathbf{86.0 \pm 0.1}$
HOSGNS$^{(dyn)}$	Hadamard	$\mathbf{78.7 \pm 0.1}$	$\mathbf{79.8 \pm 0.2}$	82.8 ± 0.3	$\mathbf{82.4 \pm 0.2}$

trainable parameters that are orders of magnitude smaller than other approaches, with a training time considerably shorter as the number of nodes increases, given a fixed number of training iterations. ISGNS has a comparable number of parameters because it incrementally updates $\mathcal{O}(|\mathcal{V}|)$ parameters moving across the $|\mathcal{T}|$ snapshots. DYSAT training time is considerably higher due to the computational overhead of the self-attention mechanism.

Table 3.7: *Number of trainable parameters and training time of time-varying graph representation learning models trained on LYONSCHOOL and synthetic datasets. The embedding dimension is fixed to 128.*

Model	Dataset					
	LYONSCHOOL $\lvert\mathcal{V}\rvert = 242, \lvert\mathcal{T}\rvert = 104$		OPENABM-2k-100 $\lvert\mathcal{V}\rvert = 2000, \lvert\mathcal{T}\rvert = 100$		OPENABM-5k-20 $\lvert\mathcal{V}\rvert = 5000, \lvert\mathcal{T}\rvert = 20$	
	Tr. parameters	Tr. time	Tr. parameters	Tr. time	Tr. parameters	Tr. time
DYANE	4,396,544	62s	50,825,472	1,014s	25,591,296	448s
DYNGEM	459,270	516s	1,867,428	10,765s	4,270,428	23,307s
DYNAMICTRIAD	3,221,632	1,131s	25,600,128	17,191s	12,800,128	12,625s
DYSAT	98,336	18,323s	323,232	152,976s	707,232	8,958s
ISGNS	61,952	381s	512,000	5,895s	1,280,000	3,062s
HOSGNS$^{(stat)}$	75,264	316s	524,800	548s	1,282,560	724s
HOSGNS$^{(dyn)}$	88,576	303s	537,600	565s	1,285,120	734s

3.4.2 Embedding space visualization

One of the main advantages of HOSGNS is that it is able to disentangle the role of nodes and time by learning representations of nodes and time intervals separately. In this section, we include plots with two-dimensional projections of these embeddings, made with UMAP [MHM18] for manifold learning and non-linear dimensionality reduction. With these plots, we show that the embedding matrices learned by

HOSGNS$^{(stat)}$ and HOSGNS$^{(dyn)}$ successfully capture both the structure and the dynamics of the time-varying graph.

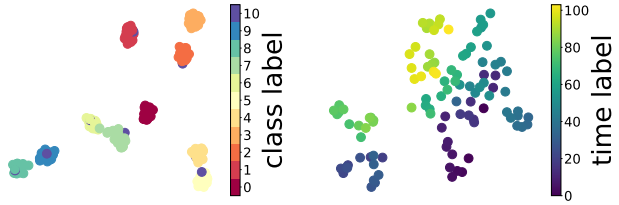
Dynamical information can be represented by associating each embedding vector to its corresponding time interval $k \in \mathcal{T}$, and graph structure can be represented by associating each embedding vector to a community. While community membership can be estimated by different community detection methods, we choose to use a dataset with ground-truth data containing node membership information. We consider the LYONSCHOOL dataset as a case study, widely investigated in literature with respect to structural and spreading properties [SVB$^+$11, BCC$^+$13, SBBPS12, PGBC13, SBCG18, GBB$^+$18]. This dataset spans two days and includes metadata (Table 3.8) concerning the class of each participant of the school (10 different labels for children and 1 label for teachers), and we identify the community membership of each individual according to these labels (*class* labels). Moreover, we also assign *time* labels according to the activation of individual nodes in temporal snapshots.

Table 3.8: *Number of class components for each labeled class in LYONSCHOOL.*

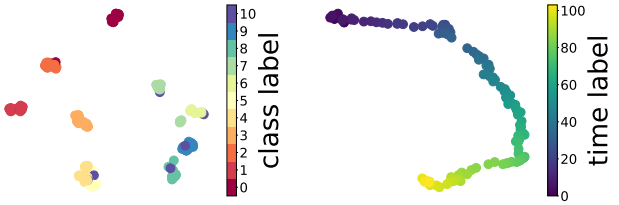
Class name	Class label	Number of children or teachers
CP-A	0	23
CP-B	1	25
CE1-A	2	23
CE1-B	3	26
CE2-A	4	23
CE2-B	5	22
CM1-A	6	21
CM1-B	7	23
CM2-A	8	22
CM2-B	9	24
Teachers	10	10

To show how disentangled representations capture different aspects of the evolving graph, in Figure 3.3 we plot individual representations of nodes $i \in \mathcal{V}$ and time slices $k \in \mathcal{T}$ labeled according to the class membership and the time snapshot respectively. Both HOSGNS$^{(stat)}$ and HOSGNS$^{(dyn)}$ capture the community structure (left of each panel) with node embeddings clustered into the ground-truth classes, but dynamical information expressed by time embeddings (right of each panel) is different for the two methods. Due to the time-respecting topology of the supra-adjacency graph, HOSGNS$^{(dyn)}$ captures the causality of node co-occurrences encoding temporal slices into a time-ordered one-dimensional manifold. HOSGNS$^{(stat)}$ is built on the snapshot representation, invariant over time permutation, and thus the temporal encoding is constrained to the local connectivity structure of graph slices.

In Figure 3.4 we visualize representations of temporal nodes $i^{(k)} \in \mathcal{V}^{(\mathcal{T})}$,

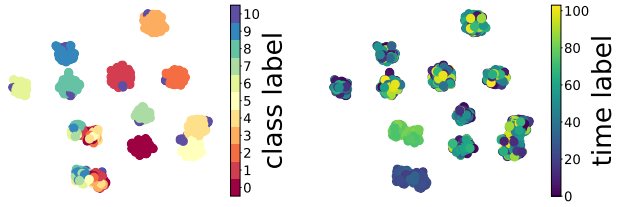


(a) *HOSGNS*$^{(stat)}$

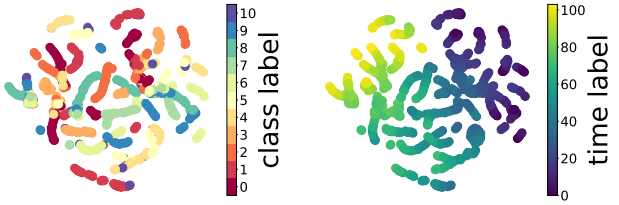


(b) *HOSGNS*$^{(dyn)}$

Figure 3.3: *Two-dimensional projections of 128-dim HOSGNS node embeddings* $\{\mathbf{v}_i\}_{i\in\mathcal{V}}$ *and time embeddings* $\{\mathbf{t}_k\}_{k\in\mathcal{T}}$*, trained on* LYONSCHOOL *dataset. On the left of each panel, node embeddings are labeled according to class communities; on the right of each panel, time embeddings are labeled according to time-steps.*

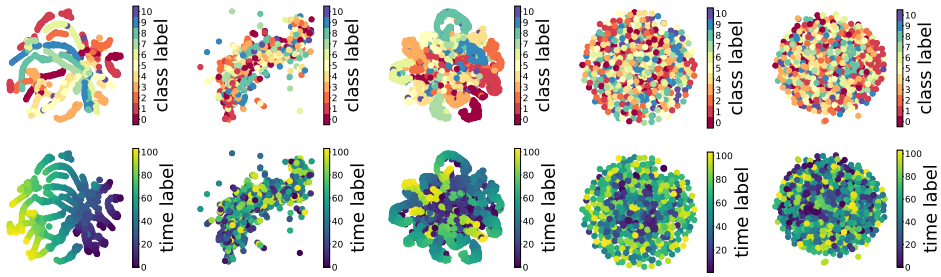


(a) *HOSGNS*$^{(stat)}$



(b) *HOSGNS*$^{(dyn)}$

Figure 3.4: *Two-dimensional projections of 128-dim HOSGNS time-resolved node embeddings obtained with Hadamard products* $\{\mathbf{v}_i * \mathbf{t}_k\}_{(i,k)\in\mathcal{V}^{(\mathcal{T})}}$*, trained on* LYONSCHOOL *dataset. On the left of each panel, embeddings are labeled according to temporal participation in class communities; on the right of each panel, embeddings are labeled according to the time-step of activation.*



(*a*) *DYANE* (*b*) *DYNGEM* (*c*) *DYNTRIAD* (*d*) *DYSAT* (*e*) *ISGNS*

Figure 3.5: *Two-dimensional projections of 128-dim time-resolved node embeddings trained on LYONSCHOOL dataset with baseline methods. On the top of each panel, embeddings are labeled according to temporal participation in class communities; on the bottom of each panel, embeddings are labeled according to the time-step of activation.*

computed as Hadamard products of nodes and time embeddings. HOSGNS$^{(stat)}$ projections show clusters of nodes active at multiple times representing different social situations: interactions during lectures present uniform class labels and heterogeneous time labels, whereas interactions occurred in social spaces with mixed classes present uniform time labels and heterogeneous class labels. This is in line with previous studies [GPC14], where different patterns of interactions are found during school activities, and gatherings in social spaces (such as the canteen and playground) are more concentrated during lunchtime. HOSGNS$^{(dyn)}$ projected embeddings, due to the causality information encoded in time representations, display trajectories of social interactions that span over time in the embedding space, with communities interacting and mixing at different points of the day.

In Figure 3.5 we see dynamic node embeddings computed with baseline methods without dissociating structure and time. The embedding space in DYANE encodes properly the time-aware topology, since the model is based on the supra-adjacency graph like HOSGNS$^{(dyn)}$. Also DYNAMICTRIAD captures significant temporal structures, but it is less effective to express the overall dynamics since it is limited in modeling the triadic closure process. Other relevant interaction patterns are instead accounted with supra-adjacency random walks. DYNGEM, DYSAT and ISGNS embedding spaces do not encode any structural or temporal information.

3.5 Summary

In this chapter, we introduce Higher-Order Skip-Gram with Negative Sampling (HOSGNS) for time-varying graph representation learning. We generalize the skip-gram embedding approach that implicitly performs a factorization of the shifted

PMI matrix to perform implicit factorization of a shifted PMI tensor. We show how to optimize HOSGNS for the generic Nth-order case, and how to apply 3rd-order and 4th-order SGNS on different higher-order representations of time-varying graphs. The embedding representations learned by HOSGNS outperform other methods in the literature and set new state-of-the-art results for solving downstream tasks. By learning embeddings on empirical time-resolved face-to-face proximity data, such representations can be effectively used to predict the outcomes of a SIR spreading process over the time-varying graph. They also can be effectively used for network reconstruction and link prediction.

HOSGNS is able to learn more compact representations of time-varying graphs due to the reduced number of parameters, with computational complexity that is comparable to or lower than other state-of-the-art methods. By learning disentangled representations of nodes and time intervals, HOSGNS uses a number of parameters in the order of $\mathcal{O}(|\mathcal{V}| + |\mathcal{T}|)$, while models that learn node-time representations need a number of parameters that is at least $\mathcal{O}(|\mathcal{V}| \cdot |\mathcal{T}|)$. While other methods such as DYANE assume that the whole temporal network has to be known, here we relax this assumption and show that the learned representations can be used also for predicting events that are not seen during the representation learning phase.

CHAPTER 4

Representation Learning on Simplicial Complexes for Effective Higher-Order Link Prediction and Reconstruction

NETWORK science provides the dominant paradigm for the study of structure and dynamics of complex systems, thanks to its focus on their underlying relational properties. In data mining applications, topological node embeddings of networks are standard representation learning methods that help solve downstream tasks, such as network reconstruction, link prediction, and node classification [Ham20]. Complex interacting systems have been usually represented as graphs. This representation however suffers from the obvious limitation that it can only capture pairwise relations among nodes, while many systems are characterized by group interactions [BCI+20]. Indeed, simplicial complexes are generalized graphs that encode group-wise edges as sets of nodes, or *simplices*, with the additional requirement that any subset of nodes forming a simplex must also itself form a simplex belonging to the complex. Unlike alternative high-order representations, e.g. hypergraphs, which also overcome the dyadic limitation of the graph formalism [TBBER21], the simplicial *downward closure* constraint works particularly well when studying systems with subset dependencies, such as brain

networks and social networks (e.g., people interacting as a group also engage in pairwise interactions).

Due to the increased interest in studying complex systems as generalized graph structures, topological representation learning techniques on simplicial complexes are also emerging as tools to solve learning tasks on systems with polyadic relations. In particular, here we focus on tasks based on the reconstruction and prediction of higher-order edges. While for standard graphs these problems have been extensively studied with traditional machine learning approaches [LZ11, CMS21] and representation learning [CMST21, MLDB20], the literature for their higher-order counterparts is more limited. In fact, reconstruction and prediction of higher-order interactions have been investigated mainly starting from pairwise data [YPP21, BAS+18] or time series [WMC+22, SBPA23], without particular attention to representation learning methods.

In this chapter and in [PPP22] we study low-dimensional embeddings of simplicial complexes for link prediction and reconstruction in higher-order networks. Our main contributions are:

- We introduce an embedding framework to compute low-rank representations of simplicial complexes.
- We formalize network reconstruction and link prediction tasks for polyadic graph structures.
- We show that simplicial similarities computed from embedding representations outperform classical network-based reconstruction and link prediction methods.

Since the problems of link prediction and network reconstruction are not yet well-defined in the literature for the higher-order case, none of the available state-of-the-art methods were previously evaluated in terms of both these tasks. In this chapter we properly delineate the formal steps to perform higher-order link prediction and reconstruction, and we make a comprehensive evaluation of different methods adding many variations such as the use of multi-node proximities and simplicial weighted random walks.

4.1 Preliminaries and Related Work

4.1.1 Simplicial complexes and their mathematical representations

Simplicial complexes can be considered as generalized graphs that include higher-order interactions. Given a set of nodes $\mathcal{V}$, a simplicial complex $\mathcal{K}$ is a collection of subsets of $\mathcal{V}$, called *simplices*, satisfying *downward inclusion*: for any simplex

$\sigma \in \mathcal{K}$, any other simplex τ which is a subset of σ belongs to the simplicial complex $\mathcal{K}$ (for any $\sigma \in \mathcal{K}$ and $\tau \subset \sigma$, we also have $\tau \in \mathcal{K}$). This constraint makes simplicial complexes different from *hypergraphs*, for which there is no prescribed relation between hyperedges. Choosing for one or the other higher-order representation frequently appears to be driven by technical convenience and, in many cases, different choices can result in dissimilar outcomes when studying group interactions [ZLB23]. Moreover, the validity of the closure assumption strongly depends on the system under study [TBBER21]. For instance, if three authors in a collaboration network have written a paper together, it is not necessarily true that each pair of scientists have co-authored a paper as well. Instead, with face-to-face social relations, a group-wise interaction usually implies pairwise links.

A simplex σ is called a k-simplex if $|\sigma| = k + 1$, where k is its *dimension* (or order). A simplex σ is a *coface* of τ (or equivalently, τ is a *face* of σ) if $\tau \subset \sigma$. We denote with $\dim(\sigma)$ the order of simplex σ, and with N_k the number of k-simplices in $\mathcal{K}$. Each simplicial complex can be unfolded in its canonical graph of inclusions, called Hasse Diagram (HD): formally, the Hasse diagram of complex $\mathcal{K}$ is the multipartite graph $\mathcal{G}_{\mathcal{K}} = (\mathcal{V}_{\mathcal{K}}, \mathcal{E}_{\mathcal{K}})$, such that each simplex $\sigma \in \mathcal{K}$ corresponds to a node of $\mathcal{V}_{\mathcal{K}}$, and two simplices $\sigma, \tau \in \mathcal{K}$ are connected by the undirected edge $(\sigma, \tau) \in \mathcal{E}_{\mathcal{K}}$ iff σ is a coface of τ and $\dim(\tau) = \dim(\sigma) - 1$. In other words, each simplicial order corresponds to a graph layer in $\mathcal{G}_{\mathcal{K}}$, and two simplices in different layers are linked if they are (upper/lower) adjacent in the original simplicial complex.

4.1.2 Representation learning on higher-order structures

Representation learning on usual graphs [Ham20] allows obtaining low-dimensional vector representations of nodes that convey information useful for solving machine learning tasks. Shallow embedding methods generate node representations as a result of an unsupervised task (e.g., matrix factorization [QDM$^+$18]), while GNN methods obtain node vectors from iterative message passing operations, e.g. graph convolutions and graph attention networks [VCC$^+$18]. In hypergraph settings, node embedding methods typically leverage hyperedge relations similarly to what is done for standard graph edges: for example, spectral decomposition [ZHS06], random walk sampling [HCY$^+$19, HLS19], autoencoders [TCW$^+$18]. Recently, Maleki et al. [MSWP22] proposed a hierarchical approach for scalable node embedding in hypergraphs. In simplicial complexes, random walks over simplices are exploited to compute embeddings of interacting groups with uniform or mixed sizes [Hac20, BHL$^+$19], expanding hypergraph methods that compute only node representations. Extensions of GNNs have been proposed to generalize convolution and attention mechanisms to hypergraphs [YNY$^+$19, FYZ$^+$19, ZZM20, BZT21] and simplicial complexes [EDS20, BFW$^+$21, GBL22].

4.1.3 Link prediction and network reconstruction with higher-order interactions

The *link prediction* [LZ11] task infers the presence of unobserved links in a graph by estimating their occurrence likelihood, while *network reconstruction* consists in the inference of a graph structure based on indirect data [Pei19], missing or noisy observations [New18]. In this chapter, we use latent embedding variables to assess the reconstruction and prediction of a given edge, relying on similarity indices. In higher-order systems, link prediction has been investigated primarily for hypergraphs, in particular with methods based on matrix factorization [ZCJC18, SPM20], resource allocation metric [KDPR20], loop structure [PSL$^+$21], and representation learning [YNN$^+$20, CP20]. The higher-order link prediction problem was introduced in a temporal setting by Benson et al. [BAS$^+$18] (reformulating the term *simplicial closure* [PPV17]), while Liu et al. [LML22] studied the prediction of several higher-order patterns with neural networks. Yoon et al. [YSSY20] investigated the use of opportune k-order projected graphs to represent group interactions, and Patil et al. [PSM20] analyzed the problem of finding relevant candidate hyperlinks as negative examples. Recently Choo et al. [CS22] investigated a related problem on the predictability of persistence in higher-order interactions. Despite these early results, reconstruction of higher-order interactions is an ongoing challenge: for example, Young et al. [YPP21] proposed a Bayesian inference method to distinguish between hyperedges and combinations of low-order edges in pairwise data, while Musciotto et al. [MBM21] developed a filtering approach to detect statistically significant hyperlinks in hypergraph data. In addition, some works studied approaches for the inference of higher-order structures from time series data [WMC$^+$22, SBPA23].

4.2 Methods and Tasks Description

4.2.1 Low-dimensional embedding of simplicial complexes

Given a simplicial complex $\mathcal{K}$, we want to learn a mapping function $g : \tau \in \mathcal{K} \mapsto \mathbf{v}_\tau \in \mathbb{R}^D$ from elements of $\mathcal{K}$ to a D-dimensional low-rank feature space ($D \ll |\mathcal{K}|$). The mapping g must preserve topological information incorporated in the simplicial complex, in such a way that adjacency relations are preserved into geometric distances between vectors of the embedding space. Here we propose that representations of simplices can be obtained by random-walking over the inclusions hierarchy of $\mathcal{K}$ and learning the embeddings according to the simplex proximity observed through such walks, preserving high-order information about the topological structure of the complex itself. Here we discuss the sampling strategies to obtain a corpus of simplicial random walks and the optimization framework to compute simplex embeddings.

Random walk sampling

The navigation of the downward inclusion chain can be performed with usual graph random walk sampling, unfolding the simplicial complex in its canonical graph of inclusions, i.e. the Hasse diagram. In the following Experiments, we consider several weighting schemes [BHL$^+$19] to bias the random walks between the vertices of the HD:

Unweighted The jump to a given τ is made by a uniform sampling among the set of neighbors $\Gamma_\sigma = \Gamma_\sigma^\downarrow \cup \Gamma_\sigma^\uparrow$ of the node σ in the HD (corresponding to faces $\Gamma_\sigma^\downarrow$ and cofaces $\Gamma_\sigma^\uparrow$ of the simplex σ in the simplicial complex).

Counts To every node τ of the HD is attached an empirical weight $\omega(\tau)$, counting the number of times that τ appears in the data. The probability to jump from σ to τ is given by $P(\sigma \to \tau) = \frac{\omega(\tau)}{\sum_{r\in\Gamma_\sigma}\omega(r)}$.

LObias With the definition of transition probability as before, the weight $\omega(\tau)$ is defined to introduce a bias for the random walker towards low-order simplices: as explained in [BHL$^+$19], every time a n-simplex σ appears in the data its weight is increased by 1, and the weight of any subface of dimension $n-k$ is increased by $\frac{(n+1)!}{(n-k+1)!}$. There is an equivalent scheme for biasing towards high-order simplices, but we empirically observed that the performance of the first one is better.

EQbias Starting from the weights' set $\{\omega(\tau)\}_{\tau\in\mathcal{V}_\mathcal{K}}$ computed with empirical counts, we attach additional weights $\{\omega(\sigma,\tau)\}_{(\sigma,\tau)\in\mathcal{E}_\mathcal{K}}$ to the Hasse diagram's edges in order to have equal probability of choosing neighbors from $\Gamma_\sigma^\downarrow$ or $\Gamma_\sigma^\uparrow$. Transition weights for the downward (upward) step (σ,τ) are defined by normalizing $\omega(\tau)$ with respect to all the downward (upward) weights $\omega(\sigma,\tau) \propto \frac{\omega(\tau)}{\sum_{r\in\Gamma_\sigma^{\downarrow(\uparrow)}}\omega(r)}$, with the probability of the step given by $P(\sigma\to\tau) = \frac{\omega(\sigma,\tau)}{\sum_{r\in\Gamma_\sigma^\downarrow\cup\Gamma_\sigma^\uparrow}\omega(r)}$.

Optimization framework

Inspired by language models such as WORD2VEC [MCCD13], we start from a corpus $\mathcal{W} = \{\sigma_1, \sigma_2, \dots\}$ of simplicial random walks, and we aim to maximize the log-likelihood of a target simplex σ_l given the multi-set $\mathcal{C}_T(\sigma_l) = \{\sigma_{l-T}\dots\sigma_{l-1}, \sigma_{l+1}\dots\sigma_{l+T}\}$ of context simplices within a distance T, determined as the number of steps between the target and the context simplex. The objective function is as follows:

$$\sum_{\sigma_l\in\mathcal{W}} \log \Pr[\, \sigma_l \mid \{\mathbf{v}_\tau : \tau \in \mathcal{C}_T(\sigma_l)\}\,] \tag{4.1}$$

where the probability $\Pr[\ \sigma_l \mid \{\mathbf{v}_\tau, \dots\}\ ] \propto \exp\left[\sum_{\tau \in \mathcal{C}_T(\sigma_l)} \mathbf{v}_\tau \cdot \mathbf{v}_{\sigma_l}\right]$ is a soft-max function which represents the likelihood of observing simplex σ_l given context simplices in $\mathcal{C}_T(\sigma_l)$. The soft-max is normalized via the standard partition function $Z_{\sigma_l} = \sum_{u \in \mathcal{K}} \exp\left[\sum_{\tau \in \mathcal{C}_T(\sigma_l)} \mathbf{v}_\tau \cdot \mathbf{v}_u\right]$. This leads to the maximization of the loss:

$$\sum_{\sigma_l \in \mathcal{W}} \big[-\log Z_{\sigma_l} + \sum_{\tau \in \mathcal{C}_T(\sigma_l)} \mathbf{v}_\tau \cdot \mathbf{v}_{\sigma_l} \big] \tag{4.2}$$

Our method of choice –SIMPLEX2VEC [BHL+19]– is implemented by sampling random walks from $\mathcal{G}_\mathcal{K}$ and learning simplicial embeddings with Continuous-Bag-Of-Words model [MCCD13]. To overcome the expensive computation of Z_{σ_l}, we train CBOW with Negative Sampling. While SIMPLEX2VEC is conceptually similar to k-SIMPLEX2VEC [Hac20], there are important differences: (i) by fixing k as simplex dimension, k-SIMPLEX2VEC uses exclusively upper connections through (k+1)-cofaces and lower connections through (k-1)-faces to compute random walk transitions; (ii) random walks focus on a fixed dimension, allowing the embedding computation only for k-simplices. SIMPLEX2VEC instead computes embedding representations for *all* simplex orders simultaneously because the random walks are sampled from the entire Hasse diagram.

4.2.2 Reconstruction and prediction of higher-order interactions

Network reconstruction [CMST21] and link prediction [MLDB20] are common tasks performed to assess the quality of node embeddings. In standard graphs, most popular methods are based on similarity metrics, used to rank pairs of nodes (candidate links), and can be obtained from local (e.g., number of common neighbors) or global indices (e.g., number of connecting paths). Here we formalize these tasks as binary classification problems for simplicial complexes.

Given a simplicial complex $\mathcal{K}$, by *reconstruction* of higher-order interactions we mean the task of correctly classifying whether a group of $k+1$ nodes $s = (i_0, i_1, \dots, i_k)$ is a k-simplex of $\mathcal{K}$ or not. This task is intended to study if the information encoded in the embedding space can preserve the original training structure. More specifically, we consider $\mathcal{S} = \{s \in \mathcal{K} : |s| > 1\}$ as the set of interactions (simplices with order greater than 0) that belongs to the simplicial complex $\mathcal{K}$. Given any group $s = (i_0, i_1, \dots, i_k)$, with the reconstruction task we aim to discern if the elements in s interact within the same simplex, and so $s \in \mathcal{S}$, or s is a group of lower-order simplices, and so $s \notin \mathcal{S}$ (but subsets of s may be existing simplices). When group s interacts within a simplex, we say that s is $closed$, conversely it is $open$.

By higher-order link *prediction* we mean instead the task of predicting whether a non-interacting group $\bar{s} \notin \mathcal{S}$ at the present time will appear in the future as

Table 4.1: *Summary statistics of empirical higher-order datasets, referred to the largest connected component of the projected graph. In order: total number of time-stamped simplices; number of unique simplices; number of training nodes $|\mathcal{V}|$ and edges $|\mathcal{E}|$ in the first 80% of $\mathcal{D}$; number of triangles in the first 80% $|\Delta|$ / new triangles in the last 20% $|\Delta^*|$; number of training tetrahedra in the first 80% $|\Theta|$ / new tetrahedra in the last 20% $|\Theta^*|$.*

Dataset	Time-stamped simplices	Unique simplices	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\Delta\rvert/\lvert\Delta^*\rvert$	$\lvert\Theta\rvert/\lvert\Theta^*\rvert$
HIGH-SCHOOL	172,035	7,818	327	5,225	2,050 / 320	218 / 20
PRIMARY-SCHOOL	106,879	12,704	242	7,575	4,259 / 880	310 / 71
EMAIL-EU	234,559	25,008	952	26,582	143,280 / 17,325	631,590 / 82,945
EMAIL-ENRON	10,883	1,512	140	1,607	5,517 / 1,061	14,902 / 3,547
TAGS-MATH	819,546	150,346	893	60,258	167,306 / 34,801	101,649 / 26,344
CONGRESS-BILLS	103,758	18,626	97	3,207	32,692 / 371	90,316 / 3,309
COAUTH-HISTORY	114,447	11,072	4,034	9,255	4,714 / 1,297	3,966 / 1,008
COAUTH-GEOLOGY	275,565	29,414	3,835	27,950	17,946 / 3,852	12,072 / 3,168

a closed interaction. This task is intended to study the generalization ability of embedding methods to predict unseen interactions that are likely to occur. To describe this task, we refer to $\mathcal{S}^*$ as the set of new incoming simplices. Given any open configuration $\bar{s} \in \mathcal{U}_{\mathcal{S}}$ coming from the set of unobserved interactions $\mathcal{U}_{\mathcal{S}} = \left\{s \in 2^{\mathcal{V}} : |s| > 1, s \notin \mathcal{S}\right\}$, namely the complement of $\mathcal{S}$ with respect to the power set of the vertices $2^{\mathcal{V}}$, the prediction task consists in classifying which groups will give rise to a simplicial closure in the future ($\bar{s} \in \mathcal{S}^*$) versus those that will remain open ($\bar{s} \in \mathcal{U}_{\mathcal{S}} \setminus \mathcal{S}^*$).

4.3 Experiments

Here we describe the experimental setup used to quantify the accuracy of SIMPLEX2VEC in reconstructing and predicting higher-order interactions. In the next paragraphs, we illustrate which datasets we use, how we sample unobserved hyperlinks, and how we use them in downstream tasks.

4.3.1 Datasets

We consider data in the form of collections of time-stamped interactions $\mathcal{D} = \{(s,t), s \in \mathcal{S}, t \in \mathcal{T}\}$, where each $s = (i_0, i_1, \ldots, i_k)$ is a k-simplex of the node set $\mathcal{V}$ and $\mathcal{T}$ is the set of time-stamps at which interactions occur. We split $\mathcal{D}$ in two subsets, $\mathcal{D}^{train}$ and $\mathcal{D}^{test}$, corresponding to the 80th percentile $t^{(80)}$ of time-stamps, namely $\mathcal{D}^{train} = \{(s,t) \in \mathcal{D}, t^{(0)} \leq t \leq t^{(80)}\}$ and $\mathcal{D}^{test} = \{(s,t) \in \mathcal{D}, t^{(80)} < t \leq t^{(100)}\}$, where $t^{(0)}$ and $t^{(100)}$ are the 0th and the 100th percentiles of the set $\mathcal{T}$.

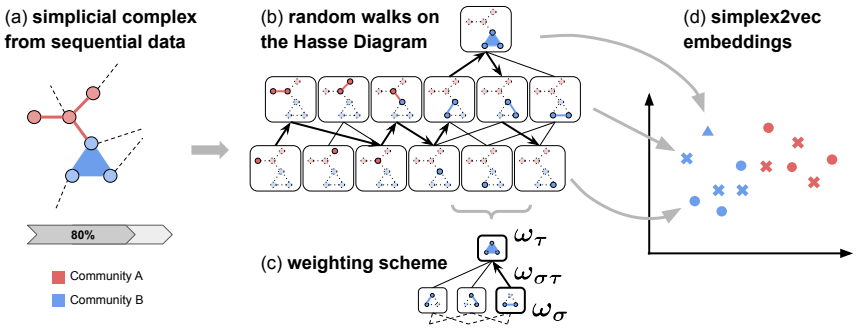


Figure 4.1: *Schematic view of SIMPLEX2VEC. Starting from simplicial sequential data (a), we construct a simplicial complex on whose Hasse diagram we sample random walks (b) with different weighting (c), from which we construct the embedding space (d).*

We use real-world time-stamped data, indicated above with the collection $\mathcal{D}$, from different domains [BAS+18]: face-to-face proximity ("HIGH-SCHOOL" and "PRIMARY-SCHOOL"), email exchange ("EMAIL-EU" and "EMAIL-ENRON"), online tags ("TAGS-MATH"), US congress bills ("CONGRESS-BILLS"), coauthorships ("COAUTH-HISTORY" and "COAUTH-GEOLOGY"). When the datasets came in pairwise format, we associated simplices to cliques obtained by integrating edge information over short time intervals [BAS+18].

We considered, for all datasets, only nodes in the largest connected component of the projected graph (two nodes of the projected graph are connected if they appear in at least one simplex of $\mathcal{D}$). In addition, to lighten the embedding computations, for CONGRESS, TAGS and COAUTH datasets we apply a filtering approach in order to reduce their sizes: similarly to [LNK07] with the `Core` set, here we selected the nodes incident in at least 5 cliques in every temporal quartile (except in COAUTH-HISTORY where we applied a threshold of 1 clique per temporal quartile). In Table 4.1, we report statistics for every considered dataset after these pre-processing steps (i.e., extraction of the largest projected component and filtering of unfrequent nodes).

4.3.2 Training process and baseline methods

We build from $\mathcal{D}^{train}$, disregarding time-stamps, a simplicial complex $\mathcal{K}^{train}$ from which we sample random walk realizations for learning low-dimensional embeddings. In all the experiments we train WORD2VEC[1] on the Hasse diagram $\mathcal{G}_{\mathcal{K}^{train}}$, running the CBOW model with window $T = 10$ and 5 epochs, and

[1] `https://radimrehurek.com/gensim/`

sampling 10 random walks of length 80 for any simplex $\sigma \in \mathcal{K}^{train}$, to obtain D-dimensional feature representations $\mathbf{v}_\sigma \in \mathbb{R}^D$.

Due to the combinatorial explosion of the number of simplicial vertices in the HD, we constrain the maximum order of the interactions to $M \in \{1, 2, 3\}$ in a reduced Hasse diagram $\mathcal{G}_{\mathcal{K}^{train},M}$ (referred simply as $\mathcal{K}_M$ from here). Consequently, every simplex with a dimension larger than M is represented in $\mathcal{K}_M$ by node combinations of size up to $M + 1$. This reduction approach introduces a distortion on the topology of the HD that might affect the final results, but on the other hand it allows the application of the embedding algorithm to middle-sized systems. However, scalability remains a concern when dealing with larger simplicial complexes. To address this challenge, future work should focus on devising methods to properly compute reduced complexes while keeping the topology unchanged [VDM13]. Alternatively, efforts can be directed towards efficiently limiting the complexity of the embedding encoder [ZDW+19]. In Figure 4.1, we show the feature learning process explained before.

To assess the reconstruction and prediction performances of the embedding model, we compare our approach with several baseline methods:

- *Projected metrics*. Local and global node-level features computed from the projected graph. The projected graph is defined as $\mathcal{G}^{train} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of 0-simplices of $\mathcal{K}^{train}$ and $\mathcal{E} = \left\{s \in \mathcal{K}^{train} : |s| = 2\right\}$ is the set of links between training nodes that interacted in at least one simplex of $\mathcal{D}^{train}$. Moreover, edges (i, j) can be weighted with the number of simplices of $\mathcal{D}^{train}$ containing both i and j. For triangles-related tasks we considered several 3-way metrics computed with the code[2] released by [BAS+18] (we show the best performant: *Harmonic mean*, *Geometric mean*, *Katz*, *PPR*, *Logistic Regression*). We exploited also the pairwise random walk measure PPMI_T [CM20], for tetrahedra-related tasks where 4-way implementations of the above listed scores are not available. PPMI is widely used as similarity function for node embeddings, and variations of the window size T allow us to take into account both local and global information.

- *Spectral embedding*. Eigenvector features from the spectral decomposition of combinatorial k-Laplacians [Gol02]. Given the set of *boundary* matrices[3] $\mathbf{B}_k \in \{0, \pm 1\}^{N_{k-1} \times N_k}$, the unweighted k-Laplacian $\mathbf{L}_k = \mathbf{B}_k^{\mathrm{T}}\mathbf{B}_k + \mathbf{B}_{k+1}\mathbf{B}_{k+1}^{\mathrm{T}} \equiv \mathbf{L}_k^{\downarrow} + \mathbf{L}_k^{\uparrow}$ encodes the adjacency information of k-simplices with their lower-adjacent $(k-1)$-faces ($\mathbf{L}_k^{\downarrow}$) and upper-adjacent $(k+1)$-cofaces ($\mathbf{L}_k^{\uparrow}$). The weighted k-Laplacian [CM21] is calculated with the

[2] `https://github.com/arbenson/ScHoLP-Tutorial`

[3] Boundary matrices require the definition of oriented simplices, see [BCI+20] for additional details.

substitutions $\mathbf{B}_k \rightarrow \mathbf{W}_{k-1}^{-1/2}\mathbf{B}_k\mathbf{W}_k^{1/2}$, where every $\mathbf{W}_k$ is a diagonal matrix containing empirical counts for k-simplices[4]. Following the same procedure used in graph spectral embeddings [BN03], we compute the eigenvectors matrix $\mathbf{Q}_k \in \mathbb{R}^{N_k \times D}$ corresponding to the first D smallest non-zero eigenvalues of $\mathbf{L}_k$ and we use the rows of $\mathbf{Q}_k$ as D-dimensional spectral embeddings for k-simplices.

- k-SIMPLEX2VEC. Features learned with an extension of NODE2VEC [Hac20][5] that samples random walks from higher-order transition probabilities (e.g., edge-to-edge occurrences) in a single simplicial dimension. This model is based on sampling from a uniform structure without taking into account simplicial weights. We sample the same number of random walks per simplex, with the same length, as the ones used for SIMPLEX2VEC.

4.3.3 Downstream Tasks

Classification of higher-order links

Similarly to the standard graph case, non-existing links are usually the majority class in classification tasks and this imbalance is even more pronounced in the higher-order case [ZCJC18] (in graphs we have $\mathcal{O}(|\mathcal{V}|^2)$ potential links, but the number of potential hyperlinks/simplices is $\mathcal{O}(2^{|\mathcal{V}|})$ in higher-order structures). In our analyses we focus on 3-node and 4-node groups, reducing the number of potential hyperedges to $\mathcal{O}(|\mathcal{V}|^3)$ and $\mathcal{O}(|\mathcal{V}|^4)$ respectively. Hence, we restrict the set of possible interactions $\mathcal{S}$ to be exclusively closed triangles Δ, or closed tetrahedra Θ, and the corresponding complementary sets $\mathcal{U}_{\Delta/\Theta}$ of unobserved groups:

$$\Delta = \left\{ s \in \mathcal{K}^{train} : |s| = 3 \right\}, \quad \mathcal{U}_\Delta = \binom{\mathcal{V}}{3} \setminus \Delta \tag{4.3}$$

$$\Theta = \left\{ s \in \mathcal{K}^{train} : |s| = 4 \right\}, \quad \mathcal{U}_\Theta = \binom{\mathcal{V}}{4} \setminus \Theta \tag{4.4}$$

where we used $\binom{\mathcal{V}}{k}$ as the set of fixed-size node combinations. For a concise presentation, in the next lines we describe mainly the 3-way case. In particular, with the reconstruction task we aim to discern those triplets δ interacting as a group in the training window $[0, t^{(80)}]$, and so $\delta \in \Delta$, from those that are groups of lower-order simplices, meaning $\delta \in \mathcal{U}_\Delta$. Moreover, defining Δ^* as the set of new closed interactions in the interval $(t^{(80)}, t^{(100)}]$, with the prediction task we aim to classify those open groups $\bar{\delta} \in \mathcal{U}_\Delta$ that will give rise to a simplicial closure on the

[4]Weights matrices satisfy the consistency relations $\mathbf{W}_k = |\mathbf{B}_{k+1}|\mathbf{W}_{k+1}$, see [CM21] for further details.

[5]https://github.com/celiahacker/k-simplex2vec

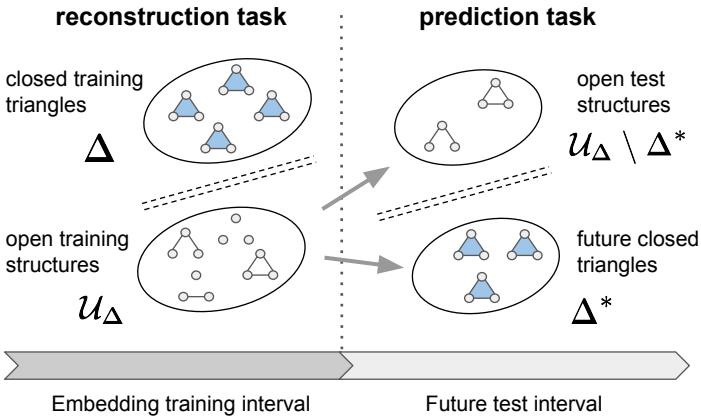


Figure 4.2: *Schematic description of classification tasks (reconstruction and prediction) in the case of 3-node group interactions.*

test time-span ($\bar{\delta} \in \Delta^*$) respect to those ones that remain open ($\bar{\delta} \in \mathcal{U}_\Delta \setminus \Delta^*$) on the same interval. In Figure 4.2, we sketch the task's formulation based on 2-simplices (3-node configurations).

Using the learned simplicial embeddings we assign to each higher-order link candidate δ a k-order similarity score based on the average pairwise inner product among 0-simplex embeddings of nodes $\{\mathbf{v}_i, i \in \delta\}$ or any high-order k-simplices $\{\mathbf{v}_\sigma, \sigma \subset \delta\}$:

$$s_k(\delta) = \frac{1}{\left|\binom{\delta}{k+1}\right|} \sum_{(\sigma,\tau)\in\binom{\delta}{k+1}} \mathbf{v}_\sigma \cdot \mathbf{v}_\tau \tag{4.5}$$

Likelihood scores of candidate higher-order links are assigned for the baseline embedding models with the same metric of Equation (4.5) used for SIMPLEX2VEC.

Open configurations sampling

Positive examples, in reconstruction and prediction tasks, are trivial to find from empirical data. For negative examples instead, enumerating all the unseen configurations (respectively $\mathcal{U}_\Delta$ and $\mathcal{U}_\Theta$ for triangles and tetrahedra) is typically unfeasible. Thus, we perform sampling of fixed-size groups of nodes to collect unseen instances for the classification tasks. Sampling non-existing hyperedges may result in different outcomes: an independent sampling of vertices most likely leads to weakly tied groups of nodes that are unlikely to occur (thus easy to distinguish from real higher-order links); instead, non-existing hyperedges obtained by slightly perturbing observed interactions leads to strongly tied configurations that are more similar to actual hyperlinks [YSSY20]. Studying the embedding performance in

Table 4.2: *Number of unobserved configurations obtained with the sampling approach in different datasets.*

Dataset	Unseen configurations sampled from $\mathcal{U}_\Delta$ $n_\mathcal{E}(\times 10^3)$			
	0	1	2	3
HIGH-SCHOOL	3,476	1,150	107	25
EMAIL-EU	8,096	1,392	1,654	186
TAGS-MATH	6,229	2,473	5,467	1,725
COAUTH-HISTORY	9,958	30	60	2

Dataset	Unseen configurations sampled from $\mathcal{U}_\Theta$ $n_\Delta(\times 10^3)$				
	0	1	2	3	4
PRIMARY-SCHOOL	17,683	396	19	2	< 1
EMAIL-ENRON	7,048	400	28	2	< 1
CONGRESS-BILLS	1,462	1,264	325	149	80
COAUTH-GEOLOGY	15,473	593	30	3	< 1

downstream tasks, we consider the variety of open configurations as the structure of internal connections changes. In practice, we sample *stars*, *cliques* and other network *motifs* [PSM20] from the projected graph to collect group configurations with distinct densities of lower-order interactions. We independently sample nodes to obtain (more likely) groups with unconnected units. For each sampled 3-node group δ we count the number of involved training edges $n_\mathcal{E}(\delta)$, and we analyze tasks performances for open configurations characterized by fixing $n_\mathcal{E}(\delta) \in \{0, 1, 2, 3\}$. For 4-node configurations, instead of $n_\mathcal{E}(\delta)$, we consider the number of training triangles $n_\Delta(\delta) \in \{0, 1, 2, 3, 4\}$ to differentiate open groups. In Table 4.2 we report the number of open configurations randomly selected from $\mathcal{U}_\Delta$ and $\mathcal{U}_\Theta$. We extracted with replacement 10^7 samples of candidate open configurations for each pattern (*stars*, *cliques*, *motifs*, and *independent* node groups). Reported values refer to the exact number of negative examples available for reconstruction tasks. For prediction tasks instead, they may contain also future closed interactions (that are anyway unobserved in the training interval) that must be subtracted to obtain negative examples.

We claim that quantities $n_\mathcal{E}(\delta)$ and $n_\Delta(\delta)$ are related to the concept of $hardness$ of non-hyperlinks [PSM20], i.e. the propensity of open groups to be misclassified as closed interaction, and they influence the difficulty of downstream classification tasks. In fact, increasing the number of lower-order faces -$n_\mathcal{E}$ or n_Δ- engaged into a fake hyperlink, the latter becomes more and more structurally similar to true hyperlinks, making the classification task more difficult.

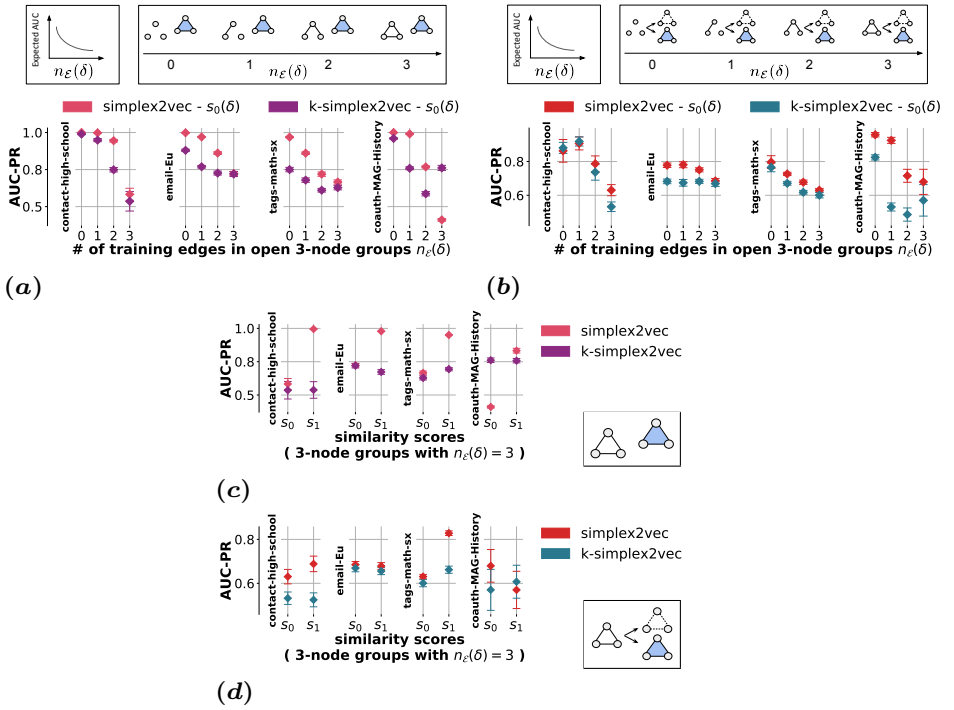


Figure 4.3: *Calibrated AUC-PR scores on 3-way link reconstruction (a)(c) and prediction (b)(d) for SIMPLEX2VEC and k-SIMPLEX2VEC with: (a)(b) similarity metric s_0 varying the parameter $n_{\mathcal{E}}$; (c)(d) similarity metric s_k (with k in $\{0, 1\}$) on highly edge-dense open configurations ($n_{\mathcal{E}} = 3$). Metrics are computed in unweighted representations, with SIMPLEX2VEC trained on $\mathcal{K}_{k+1}$ when showing results for metric s_k. The label unbalancing in each sample is uniformly drawn between 1:1 and 1:5000. A schematic view of positive and negative examples is reported for each classification task.*

4.4 Results

Here we show experimental results with 3-node configurations on datasets HIGH-SCHOOL, EMAIL-EU, TAGS-MATH, COAUTH-HISTORY (Section 4.4.1) and with 4-node configurations on the remaining ones (Section 4.4.2). We highlight models performance when using different embedding similarities $s_k(\delta)$ on open configurations with different $n_{\mathcal{E}}(\delta)$ or $n_{\Delta}(\delta)$. For each case, the classification of triangles and tetrahedra respectively, we examine: (i) the comparison with k-SIMPLEX2VEC embeddings in the *unweighted* scenario, to study how different embedding models learn statistical patterns from the simplicial structure; (ii) the comparison with classical metrics in the *weighted* scenario, to study how the addition of empir-

ical weights influences the embedding performance with respect to traditional weighted approaches. We also include in Appendix B supplemental experiments with hypergraph-based embeddings not shown in the main text.

Results are presented in terms of average binary classification scores, where test sets are generated by randomly chosen open and closed groups. Contrarily to previous work [BAS+18, CP20], we evaluate models without a fixed class imbalance because we cannot access the entire negative classes (e.g., $\mathcal{U}_\Delta$ and $\mathcal{U}_\Delta \setminus \Delta^*$ respectively in 3-way reconstruction and prediction). Instead, in every test set we uniformly sample the cardinality of the two classes to be between 1 and the number of available samples according to the task. We report calibrated AUC-PR scores [SFHG+20] to account for the difference in class imbalance as a consequence of our sampling choice[6].

In Figures 4.3 and 4.4, for a fair comparison with the other projected and embedding metrics, we report the similarity s_k training SIMPLEX2VEC on $\mathcal{K}_{k+1}$. For instance, when comparing node embedding performance (k=0), we use the Hasse diagram $\mathcal{K}_1$ to neglect triadic and higher-order information not explicitly incorporated with node-to-node proximities in k-SIMPLEX2VEC and spectral node embeddings. Best average scores are chosen for embedding models with a search on vector sizes D in the set $\{8, 16, 32, 64, 128, 256, 512, 1024\}$.

4.4.1 Reconstruction and prediction of triadic interactions

Comparison of pairwise node proximities with uniform weights

In Figure 4.3(a)(b), we show evaluation metrics on higher-order link classification (reconstruction and prediction) for 3-way interactions, computed with *unweighted* node-level information from different models, varying the quantity $n_{\mathcal{E}}(\delta)$ referred to the open configurations. We recall that in this case k-SIMPLEX2VEC is equivalent to the standard embedding of the projected graph. Hasse diagram $\mathcal{K}_1$ scores $s_0(\delta)$ computed with SIMPLEX2VEC perform overall better than proximities of the projected graph (i.e., k-SIMPLEX2VEC scores) in almost all cases, meaning that the information given by the pairwise structures is enriched by considering multiple layers of interactions, even without leveraging interaction weights (both in $\mathcal{G}^{train}$ and $\mathcal{K}^{train}$).

Generally, we observe an expected decrease in performance for every model with respect to parameter $n_{\mathcal{E}}$. For example, a few datasets show less sensitivity in the performance of prediction tasks to variations of $n_{\mathcal{E}}(\delta)$ (e.g., EMAIL-EU). We ascribe this difference to domain-specific effects and peculiarities of those datasets. Embedding similarity $s_0(\delta)$ from $\mathcal{K}_1$ diagram outperforms k-SIMPLEX2VEC prox-

[6]For this purpose we fix the reference class ratio $\pi_0 = 0.5$. See [SFHG+20] for additional details. We also tested the AUC-ROC metric with similar findings.

Table 4.3: *Calibrated AUC-PR scores on 3-way link reconstruction (top) and prediction (bottom), with the hardest class of negative configurations ($n_{\mathcal{E}} = 3$). The best scores for different methods are reported in boldface letters; among these ones, the best overall score is blue-shaded and the second best score is grey-shaded.*

	Features Type		Dataset							
			HIGH-SCHOOL		EMAIL-EU		TAGS-MATH		COAUTH-HISTORY	
			$s_0(\delta)$	$s_1(\delta)$	$s_0(\delta)$	$s_1(\delta)$	$s_0(\delta)$	$s_1(\delta)$	$s_0(\delta)$	$s_1(\delta)$
Neural Embedding	Hasse diagram $\mathcal{K}_1$	*Unweighted*	57.5±1.9	51.4±1.2	72.0±0.3	64.0±0.2	66.7±0.2	57.1±0.1	41.1±0.9	75.5±1.1
		Counts	79.5±1.0	84.4±0.9	**76.3±0.4**	73.3±0.2	80.5±0.1	**87.8±0.1**	41.6±1.0	**76.0±1.1**
		LObias	81.6±2.4	**89.5±0.8**	76.1±0.3	71.2±0.2	76.9±0.1	83.7±0.1	41.7±0.7	57.7±1.2
	Hasse diagram $\mathcal{K}_2$	*Unweighted*	55.5±3.0	**99.5±0.1**	61.0±0.4	**97.9±0.0**	66.7±0.1	**95.1±0.0**	40.0±0.5	83.1±1.3
		Counts	57.0±1.3	91.2±0.9	54.5±0.2	92.6±0.1	66.2±0.1	89.4±0.1	35.3±0.4	82.1±1.3
		LObias	84.7±2.2	91.9±0.8	80.6±0.3	81.6±0.2	77.9±0.1	84.3±0.1	57.3±1.0	70.4±1.4
		EQbias	72.7±1.1	89.2±0.7	71.8±0.3	75.0±0.2	78.2±0.2	88.0±0.1	39.3±0.7	**87.3±1.1**
Spectral Embedding	Combinatorial Laplacians	*Unweighted*	52.4±3.7	**77.0±1.3**	67.3±0.3	65.3±0.2	58.4±0.2	50.7±0.1	72.1±1.1	63.5±1.4
		Weighted	70.4±1.6	75.3±1.6	**79.4±0.2**	76.4±0.1	**79.9±0.1**	50.4±0.1	**82.3±1.0**	68.4±1.2
Projected Metrics	Harm. mean	*Weighted*	85.5±1.5		**74.0±0.2**		83.1±0.1		53.3±1.1	
	Geom. mean		**85.8±1.1**		72.5±0.2		**86.8±0.1**		52.9±1.3	
	Katz		78.6±1.1		65.6±0.2		81.8±0.1		49.2±1.5	
	PPR		76.9±1.4		70.7±0.2		81.8±0.1		**74.8±1.3**	

	Features Type		Dataset							
			HIGH-SCHOOL		EMAIL-EU		TAGS-MATH		COAUTH-HISTORY	
			$s_0(\delta)$	$s_1(\delta)$	$s_0(\delta)$	$s_1(\delta)$	$s_0(\delta)$	$s_1(\delta)$	$s_0(\delta)$	$s_1(\delta)$
Neural Embedding	Hasse diagram $\mathcal{K}_1$	*Unweighted*	62.9±5.2	50.6±4.7	68.5±0.7	57.6±0.5	63.2±0.3	54.0±0.5	**69.5±8.2**	63.2±6.6
		Counts	**74.2±3.0**	73.0±3.4	**74.3±0.8**	67.3±0.7	74.3±0.4	**84.0±0.3**	68.7±8.4	66.6±8.6
		LObias	70.6±2.8	65.6±5.3	70.5±0.6	64.5±0.8	71.3±0.5	79.1±0.5	68.8±8.7	66.5±8.7
	Hasse diagram $\mathcal{K}_2$	*Unweighted*	62.5±6.3	69.5±4.9	66.2±0.7	67.8±0.6	62.5±0.2	**83.1±0.2**	65.9±8.5	55.6±8.0
		Counts	64.3±3.6	72.8±3.6	61.8±0.7	69.1±0.6	62.9±0.3	82.3±0.3	67.3±8.2	61.0±9.6
		LObias	69.7±3.5	65.4±5.1	69.0±0.6	60.3±0.6	71.2±0.7	79.2±0.4	67.3±7.9	64.2±9.6
		EQbias	72.4±3.6	**73.5±3.5**	**71.3±0.6**	66.1±0.6	71.2±0.4	82.3±0.3	**67.8±8.6**	65.7±9.3
Spectral Embedding	Combinatorial Laplacians	*Unweighted*	56.4±3.6	56.7±6.8	63.8±0.6	53.5±0.7	55.1±0.2	50.4±0.2	57.8±6.0	56.4±5.7
		Weighted	**66.5±5.3**	56.1±6.5	**65.2±0.8**	55.6±0.7	**72.8±0.4**	50.3±0.3	**70.1±8.3**	53.5±6.8
Projected Metrics	Harm. mean	*Weighted*	71.4±4.3		64.5±0.8		79.0±0.2		61.6±8.2	
	Geom. mean		**73.1±3.8**		66.7±0.8		**83.3±0.2**		62.4±7.7	
	Katz		69.3±3.7		63.2±0.6		77.8±0.3		62.4±7.0	
	PPR		69.8±3.9		**68.8±0.5**		75.7±0.4		57.7±4.6	
	Logistic Regression	*Unweighted*	68.7±3.1		68.1±0.7		81.2±0.2		**65.4±6.9**	

imities in almost every reconstruction task, except for COAUTH-HISTORY on open configurations with $n_{\mathcal{E}} = 3$. This fact seems connected with some specific graph features of collaborations (even possibly related to the filtering approach utilized). Moreover, coauthorship relations usually are not characterized by subset dependencies (writing a paper as a group does not imply pairwise collaborations) that are encoded with simplicial complexes. In prediction tasks, we observe the same advantage of SIMPLEX2VEC respect to k-SIMPLEX2VEC, except in HIGH-SCHOOL where the models perform similarly on $n_{\mathcal{E}} < 2$.

Comparison of higher-order edge proximities with uniform weights

In the previous sections, the metric $s_0(\delta)$ was computed from feature representations of 0-simplices. Here we analyze instead how performances change when we use embedding representations of 1-simplices (edge representations) to compute $s_1(\delta)$. Intuitively, group representations like 1-simplex embeddings should convey higher-order information useful to improve classification with respect to node-level features.

In Figure 4.3(c)(d), we show evaluation metrics on higher-order link classification for 3-way interactions, comparing *unweighted* node-level and edge-level information from different models, fixing the quantity $n_{\mathcal{E}}(\delta) = 3$ referred to the open configurations. We consider fully connected triangle configurations because, besides being the harder configurations to be classified, they consist of the set of links necessary to compute $s_1(\delta)$.

Generally, we notice an increase in classification scores when using $s_1(\delta)$ similarity rather $s_0(\delta)$ with SIMPLEX2VEC embeddings, instead k-SIMPLEX2VEC exhibits reduced gains in most datasets. The SIMPLEX2VEC performance gain is quite large (between 30% and 100%) in all reconstruction tasks, and for prediction tasks it is noticeable on HIGH-SCHOOL and TAGS-MATH while it is even negative on COAUTH-HISTORY. Regarding the latter dataset, the use of edge-level similarity balances the node-level reconstruction loss noticed in Figure 4.3(a).

Role of simplicial weights

Previously we showed that feature representations learned through the hierarchical organization of the HD enhance the classification accuracy of closed triangles when considering unweighted complexes. We now integrate these results by studying the effect of introducing weights. In particular, we analyze the importance of weighted interactions in our framework, focusing on the case where fully connected open triangles are the negative examples for downstream tasks.

On the top of Table 4.3, we show higher-order link reconstruction results. Simplicial similarity $s_1(\delta)$ on the unweighted HD $\mathcal{K}_2$ outperforms all other methods, in particular weighted metrics based on Laplacian similarity and projected graph

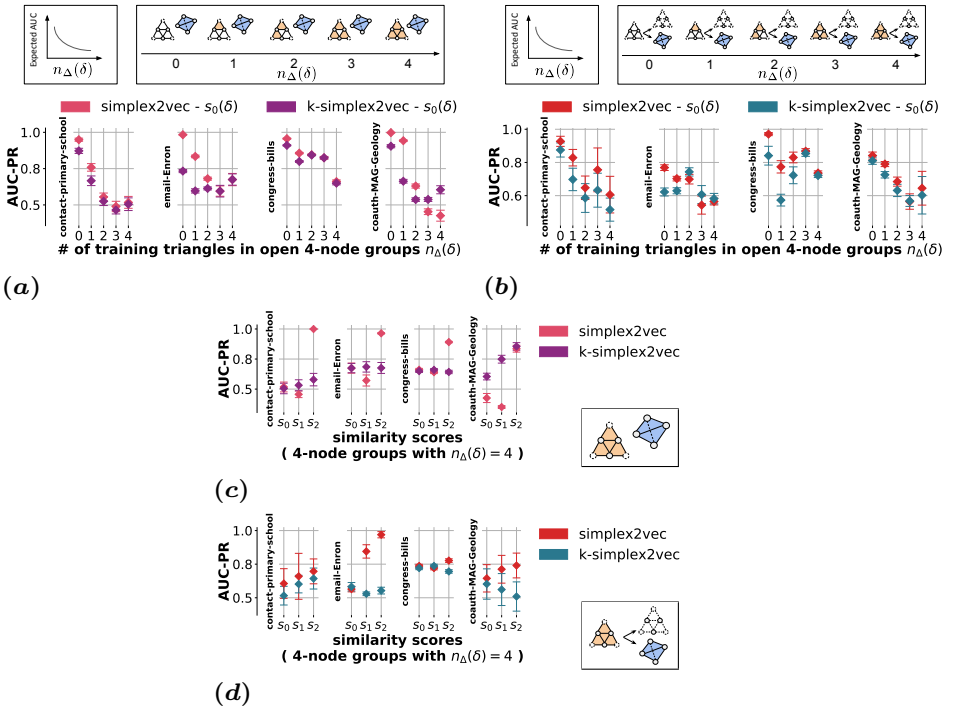


Figure 4.4: *Calibrated AUC-PR scores on 4-way link reconstruction (a)(c) and prediction (b)(d) for SIMPLEX2VEC and k-SIMPLEX2VEC with: (a)(b) similarity metric s_0 varying the parameter n_Δ; (c)(d) similarity metric s_k (with k in $\{0, 1, 2\}$) on highly triangle-dense open configurations ($n_\Delta = 4$). Metrics are computed in unweighted representations, with SIMPLEX2VEC trained on $\mathcal{K}_{k+1}$ when showing results for metric s_k. Label unbalancing in each sample is uniformly drawn between 1:1 and 1:5000. A schematic view of positive and negative examples is reported for each classification task.*

geometric mean, allowing almost perfect reconstruction in 3 out of 4 datasets. Compared with projected graph metrics, this was expected since 3-way information is incorporated in $\mathcal{K}_2$, and the optimal scores reflect the goodness of fit of the embedding algorithm. Weighting schemes *Counts* and *EQbias* also obtain excellent scores with $s_1(\delta)$ metric, while metric $s_0(\delta)$ benefits from the use of *LObias* weights. Differently, even simplicial similarity $s_1(\delta)$ on Hasse diagram $\mathcal{K}_1$ outperforms baseline scores in half of the datasets (with weighting schemes *Counts* and *LObias*), showing the feasibility of reconstructing 2-order interactions from weighted lower-order simplices (vertices in $\mathcal{K}_1$ are simplices of dimension 0 and 1) similarly to previous work on hypergraph reconstruction [YPP21].

On the bottom of Table 4.3, we show higher-order link prediction results. SIM-

PLEX2VEC embeddings trained on $\mathcal{K}_1$ with *Counts* and *EQbias* weights give better results: in HIGH-SCHOOL and EMAIL-EU with $s_0(\delta)$ metric, in TAGS-MATH with $s_1(\delta)$ metric. In dataset COAUTH-HISTORY the unweighted $s_0(\delta)$ score is outperformed uniquely by the weighted $\mathbf{L}_0$ embedding, with weighted simplicial counterparts resulting in similar performances. In the space of projected graph scores, good results are obtained with *geometric mean* and *logistic regression*, which were among the best metrics in one of the seminal works on higher-order link prediction [BAS+18].

Overall, we observe that weighting schemes for neural simplicial embeddings overall positively contribute to classification tasks both for reconstruction and prediction.

4.4.2 Reconstruction and prediction of tetradic interactions

Unweighted case

In Figure 4.4(a), we show node-level evaluation metrics for 4-way higher-order reconstruction. Metric $s_0(\delta)$ of SIMPLEX2VEC computed on $\mathcal{K}_1$ shows overall slightly better performances respect to k-SIMPLEX2VEC similarities, especially when the density of triangles is low ($n_\Delta < 3$). In COAUTH-GEOLOGY we observe also a remarkable increment of k-SIMPLEX2VEC reconstruction scores for negative examples with increasing $n_\Delta(\delta)$, and this is also observable in EMAIL-ENRON.

In Figure 4.4(b), we report node-level evaluation metrics for 4-way higher-order prediction. Node-level SIMPLEX2VEC embedding performs better than k-SIMPLEX2VEC, on PRIMARY-SCHOOL and, to a lesser extent, on COAUTH-GEOLOGY. In EMAIL-ENRON and CONGRESS-BILLS SIMPLEX2VEC performance increases when the density of triangles is low ($n_\Delta < 3$).

Higher-order similarity measures from k-SIMPLEX2VEC in Figure 4.4(c)(d), are outperformed by the SIMPLEX2VEC ones in many cases, especially $s_2(\delta)$ metric for PRIMARY-SCHOOL, EMAIL-ENRON and CONGRESS-BILLS in reconstruction tasks. In prediction tasks over EMAIL-ENRON and COAUTH-GEOLOGY, SIMPLEX2VEC obtains mainly good results overcoming the simplicial baseline. These results generally confirm our previous findings on 3-way tasks, which displayed an increasing classification capability when using higher-order proximities s_k ($k > 0$) for SIMPLEX2VEC.

Weighted case

On the top of Table 4.4, we show reconstruction scores of tetrahedra, when simplicial embeddings are trained on Hasse diagram $\mathcal{K}_2$ and negative examples are given by open 4-way configurations with four triangular faces. Due to $\mathcal{K}_2$ characteristics, features learned from the simplicial complex are not aware of tetrahedral structures and this task results on reconstructing 4-node groups from training data with most

Table 4.4: *Calibrated AUC-PR scores on 4-way link reconstruction (top) and prediction (bottom), with the hardest class of negative configurations ($n_\Delta = 4$). The best scores for different methods are reported in boldface letters; among these ones, the best overall score is blue-shaded and the second best score is grey-shaded.*

Dataset		Neural Embedding (Hasse diagram $\mathcal{K}_2$)			Spectral Embedding (Combinatorial Laplacians)				Projected Graph PPMI Metric		
		$s_0(\delta)$	$s_1(\delta)$	$s_2(\delta)$		$s_0(\delta)$	$s_1(\delta)$	$s_2(\delta)$	$T=1$	$T=10$	$T=\infty$
PRIMARY-SCHOOL	*Unweighted*	52.9±3.3	45.2±2.7	64.5±2.8	*Unweighted*	52.1±3.8	**58.2±2.0**	53.4±3.0	**51.5±3.1**	50.2±3.0	50.2±3.0
	Counts	48.4±3.0	46.2±2.8	59.1±3.3							
	LObias	50.6±3.2	61.6±3.3	**70.7±3.9**	*Weighted*	54.0±2.8	55.9±2.8	53.4±2.1	47.9±3.1	47.0±2.7	48.5±2.5
	EQbias	45.2±3.6	47.0±3.0	58.5±3.3							
EMAIL-ENRON	*Unweighted*	**69.0±0.4**	56.0±0.4	58.2±0.3	*Unweighted*	69.0±0.5	68.0±0.4	55.5±0.3	**68.5±0.4**	66.7±0.5	66.9±0.4
	Counts	60.6±0.5	61.3±0.5	54.0±0.4							
	LObias	68.0±0.5	46.5±0.5	57.4±0.5	*Weighted*	71.1±0.4	**79.0±0.3**	76.9±0.2	58.3±0.4	57.9±0.5	62.0±0.5
	EQbias	62.1±0.7	44.4±0.3	53.1±0.4							
CONGRESS-BILLS	*Unweighted*	63.1±0.2	64.4±0.1	51.8±0.2	*Unweighted*	56.1±0.2	58.4±0.1	49.8±0.1	65.9±0.1	**66.0±0.1**	65.9±0.1
	Counts	43.1±0.1	70.4±0.1	72.5±0.1							
	LObias	49.0±0.1	**74.2±0.1**	60.6±0.2	*Weighted*	55.0±0.1	**62.8±0.2**	55.3±0.2	49.1±0.1	47.8±0.1	47.3±0.1
	EQbias	65.7±0.2	69.0±0.1	**74.2±0.1**							
COAUTH-GEOLOGY	*Unweighted*	71.6±0.5	34.6±0.3	**84.2±0.7**	*Unweighted*	62.6±0.6	61.7±0.9	49.3±0.9	**86.0±0.4**	77.8±0.4	75.5±0.5
	Counts	40.5±0.3	36.2±0.4	74.1±0.3							
	LObias	64.1±0.5	34.4±0.3	73.3±0.5	*Weighted*	**85.8±0.7**	65.7±0.5	44.9±0.7	76.3±0.6	71.9±0.5	70.6±0.6
	EQbias	36.7±0.3	37.5±0.2	79.2±0.4							

Dataset		Neural Embedding (Hasse diagram $\mathcal{K}_3$)			Spectral Embedding (Combinatorial Laplacians)				Projected Graph PPMI Metric		
		$s_0(\delta)$	$s_1(\delta)$	$s_2(\delta)$		$s_0(\delta)$	$s_1(\delta)$	$s_2(\delta)$	$T=1$	$T=10$	$T=\infty$
PRIMARY-SCHOOL	*Unweighted*	56.4±1.8	58.6±2.3	66.8±2.4	*Unweighted*	82.1±4.0	85.4±1.7	**85.9±3.1**	49.3±2.2	45.8±1.6	45.7±1.7
	Counts	63.0±2.7	67.8±0.7	**72.2±1.6**							
	LObias	60.4±1.6	61.2±2.2	62.4±2.6	*Weighted*	57.8±2.4	81.3±4.4	70.6±1.5	**61.1±2.3**	47.4±1.6	48.6±1.6
	EQbias	62.7±2.0	65.6±1.2	68.3±2.2							
EMAIL-ENRON	*Unweighted*	88.3±6.6	**98.0±2.1**	96.9±2.3	*Unweighted*	92.7±2.9	67.6±5.7	**97.1±1.8**	50.3±0.2	50.9±0.5	50.8±0.5
	Counts	77.0±5.6	88.7±4.0	83.5±4.5							
	LObias	60.5±3.1	73.7±5.4	88.4±4.0	*Weighted*	84.8±5.6	88.7±3.7	95.8±2.4	**55.8±2.2**	53.3±1.3	54.7±1.5
	EQbias	57.9±2.5	84.9±3.6	80.4±5.6							
CONGRESS-BILLS	*Unweighted*	47.9±0.1	34.0±0.0	**77.7±0.3**	*Unweighted*	60.8±0.2	**64.3±0.3**	48.8±0.2	**74.7±0.2**	**74.7±0.2**	**74.7±0.2**
	Counts	49.9±0.2	37.4±0.1	74.6±0.3							
	LObias	40.2±0.2	76.9±0.3	74.0±0.3	*Weighted*	40.2±0.1	53.1±0.3	50.8±0.2	40.2±0.1	40.8±0.1	40.2±0.1
	EQbias	64.2±0.2	58.4±0.3	71.4±0.2							
COAUTH-GEOLOGY	*Unweighted*	55.1±7.7	60.1±7.2	74.8±4.8	*Unweighted*	57.0±6.9	48.1±7.8	52.1±7.3	50.7±3.5	54.6±6.3	55.3±7.4
	Counts	54.0±5.9	74.1±3.6	78.6±4.4							
	LObias	75.9±5.0	**84.2±2.9**	73.9±4.3	*Weighted*	**88.5±3.2**	52.0±7.7	52.7±7.3	54.9±4.5	**56.1±5.9**	55.3±4.8
	EQbias	51.3±4.7	76.1±4.3	72.8±6.1							

triadic structures. Previous work analyzed the problem of higher-order edge reconstruction from pairwise data [YPP21], but here we focus on a not previously studied task based on triadic data. From the comparison with spectral embeddings and PPMI proximities, we notice that SIMPLEX2VEC weighted $s_2(\delta)$ similarity (*LObias* and *EQbias*) is the best on half of the datasets in classifying closed tetrahedra respect to triangle-rich open groups. In EMAIL-ENRON weighted $\mathbf{L}_1$ embedding outperforms the unweighted (and weighted ones) $s_0(\delta)$ simplicial metric, while in COAUTH-GEOLOGY the best score is given by the unweighted PPMI_1 (which is also the best projected metric in the other 3 datasets).

On the bottom of Table 4.4, we report classification scores for the prediction of simplicial closures on tetrahedra, when neural embeddings are trained on Hasse diagram $\mathcal{K}_3$ (we empirically observed better results with respect to $\mathcal{K}_2$). We compare these results with spectral embeddings and PPMI projected metrics in predicting which mostly triangle-dense configurations will close in a tetrahedron in the last 20% of data. Unusually, best scores obtained with SIMPLEX2VEC come from the unweighted setting in EMAIL-ENRON and CONGRESS-BILLS with respectively $s_1(\delta)$ and $s_2(\delta)$ metrics. There is not a unique best metric, which was also observed in the 3-way prediction reports at the bottom of Table 4.3. Spectral embedding outperforms neural methods for PRIMARY-SCHOOL (unweighted s_2) and COAUTH-GEOLOGY (weighted s_0).

4.5 Summary

In this chapter, we introduce SIMPLEX2VEC for representation learning on simplicial complexes. In particular, we focus on formalizing network reconstruction and link prediction tasks for higher-order structures, and we test the proposed model on solving such downstream tasks. We show that SIMPLEX2VEC-based representations are more effective in classification than traditional approaches and previous higher-order embedding methods. In our experiments, we prove the feasibility of using simplicial embedding of Hasse diagrams in reconstructing the system's polyadic interactions from lower-order edges, in addition to adequately predicting future simplicial closures.

In conclusion, the employment of SIMPLEX2VEC for representation learning on simplicial complexes opens up new possibilities for understanding and modeling higher-order complex systems. The ability to reconstruct and predict links in these structures can provide valuable insights into the underlying dynamics of such systems. SIMPLEX2VEC enables the investigation of the impact of different topological features, and we showed that weighted and unweighted models have different predictive power. Overall, tools for learning representations of higher-order systems are promising for understanding and predicting complex behavior.

CHAPTER 5

Interpretability of Dimensions in Node Representations for Graphs

The main goal of graph representation learning is to convert the vertices of a graph into a latent vector space through the computation of node representations [HYL17b]. Unsupervised node embeddings are commonly used to achieve this by optimizing a structural prediction task and encoding graph proximity relationships into the embedding space. These techniques are widely used in web and social network analysis for tasks such as link prediction and community detection [Ham20]. However, there is a lack of comprehension of how to interpret the dimensions of these embeddings in a way that is easily understandable by humans [WYT+19, LHLH18, KMA21]. This chapter aims to address this gap by providing an important missing aspect for the interpretability of node representations.

The individual dimensions of node embeddings are often challenging to interpret [LHLH18, DG18, GBH19]. They are not easily understandable to humans because there is no emphasis on making the embedding dimensions interpretable during the learning process. As a result, it is unclear how the resulting embedding entries relate to understandable properties of the input graph. Previous research on the interpretability of node embeddings has mainly focused on two aspects. Firstly,

some studies have attempted to map dimensions to ground-truth node labels, when they are available [GBH19, DNA19]. Secondly, there have been efforts to check if specific node properties, such as node centrality or clustering coefficient, are encoded in embeddings for a given prediction task [DG18, BKB+19]. Our proposed methods aim to retrofit existing node embeddings to make latent dimensions more interpretable, an important aspect that has not been explored in prior works for graph representation learning.

The main objective of this chapter is to understand the meaning of the dimensions in node embeddings and associate them with understandable structural units of the input graph. Even though this is a *post-hoc* approach, we cannot use the existing tools for interpreting prediction models as they are mainly designed for supervised tasks [RSG16, LL17]. We will analyze unsupervised node embeddings to find evidence for interpretable latent units associated with semantic concepts of the input data. Given this as the principal objective, we summarize the major contributions of this chapter as follows:

- We propose a novel per-dimension measure of interpretability that is grounded in game-theoretic notions.
- We propose a new, modular, and simple approach that aims to enhance the interpretability of the embedding dimensions by reconfiguring existing node representations.
- We experimentally show clear gains in the interpretability-performance trade-offs using our retrofitting approach in comparison with competitive available methods.

We propose to use communities as interpretable semantic concepts of the input data, as many real-world graphs have an underlying community structure [GN02]. To interpret any latent dimension d of node embeddings, we firstly establish an importance measure -$\mu_d(\mathbf{u}, \mathbf{v})$- based on score contributions when predicting the occurrence of edge (u, v). By constructing saliency maps for dimensions (refer to Figure 5.1), we can recognize groups of edges (salient subgraphs induced by our importance measure). We also define metrics to quantify the interpretability of latent dimensions by assessing the degree of association with individual graph communities, in addition to the sparsity level of these associations. As an example, in Figure 5.1(a), most of the DEEPWALK dimensions are not immediately interpretable since they do not clearly match with a single clique of the synthetic graph. To quantify this "degree of matching", we introduce a score I_d and use it as a dimension-wise community-aware interpretability score.

Secondly, we propose a novel modular approach to improve the interpretability of existing node representations. The approach, based on autoencoder architectures

[Bal12], maps the input embeddings into a new sparse, interpretable, and low-entropy vector space. Figure 5.1(b) shows the result of this post-processing step on DEEPWALK vectors. Moreover, the autoencoder's representational power also preserves the topological graph information to be used in usual downstream tasks with minimal performance loss. We conduct extensive experimental evaluations comparing our approach to other embedding methods in terms of interpretability, link prediction performance, and scalability over multiple synthetic and real-world graph datasets. Empirical results show that our retrofitting approach consistently outperforms existing baselines in terms of per-dimension interpretability, with minimal or no performance losses under most experimental conditions.

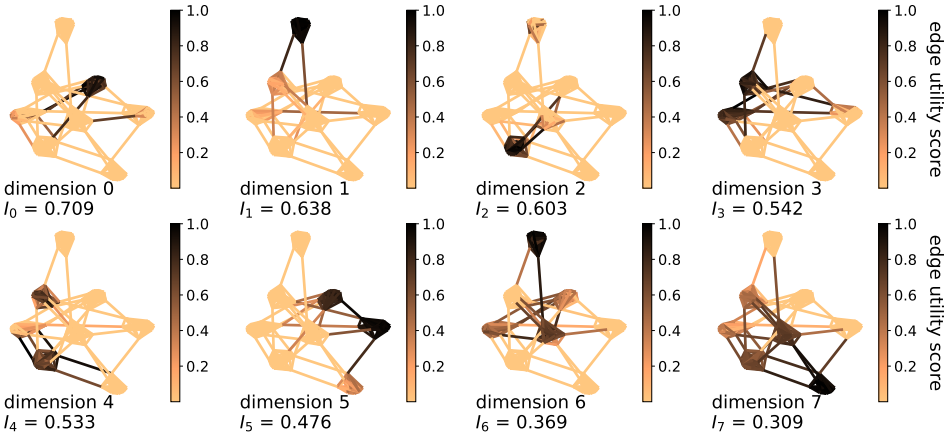
5.1 Related Work

5.1.1 Interpretability for node embeddings

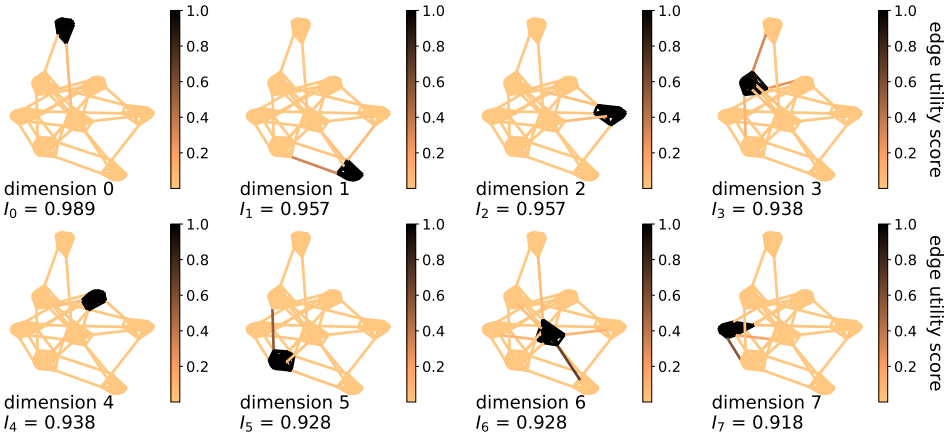
From the node embeddings perspective, interpretability is a multi-faceted concept that has been studied from different angles. In [DG18, BKB+19] authors investigate, by means of prediction tasks, whether specific topological graph features (e.g., degree centrality, clustering coefficient, etc.) are encoded into node representations. These works significantly differ from our approach, where the aim is to find the interpretable meaning of embedding dimensions, associating single dimensions with interpretable graph structures (e.g., communities). Other methods that focus on explaining individual dimensions [GBH19, KMA21] do not optimize graph reconstruction and edge prediction, as the method presented here does. In [LHLH18] global interpretations are given in the form of a hierarchy of graph partitions, but they do not focus on interpreting single dimensions. Instead, [WYT+19] study the impact on learned node embeddings when removing edges from the input graph. Another line of research focuses on producing interpretable-by-design representations based on graph clustering [DNA19], which are conceptually analogous to community-preserving node embeddings [WCW+17, RDSS19].

5.1.2 Interpretability for link prediction

Our approach focuses on the interpretation of embedding dimensions according to graph structural reconstruction task, and it is conceptually similar to methods for the interpretability of embedding-based link prediction. For instance, ExplaiNE [KLDB22] quantify the variation in the probability of a link when adding or removing neighboring edges. In the same way, many methods have been proposed for explaining link prediction in knowledge graphs [RFMT22, ZPZ+19], but these works differ from our approach since we aim to quantify the most influential representation dimensions in scoring links, whereas those methods focus on identifying other relevant links that enable a given prediction. Moreover, the task of



(*a*) DEEPWALK



(*b*) *RF*-DEEPWALK

Figure 5.1: *Saliency plots with per-dimension edge utilities (normalized between 0 and 1) for 8-dim embeddings trained on a graph generated via Stochastic Block Model [Abb17]. The figures show edge utility scores and interpretability metrics when using:* DEEPWALK *embeddings (a), and retrofitted RF-*DEEPWALK *embeddings (b). Graphs have 80 nodes divided into 8 fully connected cliques.*

interpreting link prediction has been used to generate explanations in graph-based recommendations [GBSRW20, PKFD20].

5.1.3 Interpretability for word embeddings

Since part of the literature on graph representation learning is built upon word embeddings, methods for interpreting such representations are also relevant for the graph domain. Previous works in this area focus on interpreting word embedding dimensions based on semantic information [PDCM22, ŞUY+18], or analyzing geometric properties of the embedding space [SMF18, PBO17]. Several studies also show different approaches to learning interpretable representations by design, where the main goal is achieving *sparsity* [SGL+16, CZ17], or by leveraging on external lexicon-based resources [BAMK16, ŞUŞ+20]. Another category of techniques consists of retrofitting pre-trained models (e.g., WORD2VEC or GLOVE [PSM14]) to improve their interpretability. These post-processing methods can be supervised approaches that utilize semantic information [FDJ+15, JDH15], or unsupervised transformations that convert dense word vectors into sparse, overcomplete representations [SPJ+18, FTY+15].

5.2 Methods Description

In this part, we first present a novel methodology to quantify the per-dimension interpretability of node embeddings (Sections 5.2.1 and 5.2.2). Then, in Section 5.2.3 we introduce a retrofitting approach to reconfigure existing node embeddings, making them more interpretable according to previously defined metrics. We will refer to $\mathbf{u}$ as the embedding vectors of node $u \in \mathcal{V}$ mapped through an encoder enc, and with u_d as entries of these vectors corresponding to dimension d. Embedding vectors are collected into the entries of the embedding matrix $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}| \times D}$, i.e. $\mathbf{V}_{ud} = \mathrm{u}_d$. Later we will refer to D as the cardinality of the set $\Omega = \{1, \ldots, D\}$ containing the enumerated dimensions.

5.2.1 Importance-based utility of latent dimensions with edge scoring

The score returned by the decoder function $\mathrm{dec}(\mathbf{u}, \mathbf{v})$ can be used to perform edge reconstruction, i.e. assessing the occurrence of edges (u, v): the higher the score, the higher the likelihood of observing the link on the input graph. Here we consider popular methods such as DEEPWALK and NODE2VEC, where the decoder $\mathrm{dec}(\mathbf{u}, \mathbf{v}) \equiv \sigma(\mathbf{u} \cdot \mathbf{v})$, i.e. the sigmoid of the inner product of the embedding vectors. Let us define a decoder for any subset of dimensions $\mathcal{B} \subseteq \Omega$ replacing σ

with a linear scoring function $\xi_{\mathcal{B}} : \mathbb{R}^{|\mathcal{B}|} \times \mathbb{R}^{|\mathcal{B}|} \to \mathbb{R}$, defined as:

$$\xi_{\mathcal{B}}(\mathbf{u}, \mathbf{v}) = \frac{1}{|\mathcal{B}|} \sum_{d \in \mathcal{B}} \mathrm{u}_d \mathrm{v}_d \tag{5.1}$$

To study the interpretability of node embeddings, we calculate the contribution score of a single dimension d in the reconstruction of edge (u, v) as:

$$\mu_d(\mathbf{u}, \mathbf{v}) = \xi_{\Omega}(\mathbf{u}, \mathbf{v}) - \xi_{\Omega \setminus \{d\}}(\mathbf{u}, \mathbf{v}) \tag{5.2}$$

This measure quantifies the variation of the likelihood function ξ_Ω when removing dimension d from the whole set of embedding directions. Given a well-known result from cooperative game theory [Sha16], a theoretically principled contribution score is given by the Shapley value, which takes into account the marginal importance of a feature across all possible subsets of dimensions:

$$\phi_d(\mathbf{u}, \mathbf{v}) = \sum_{\mathcal{B} \subseteq \Omega \setminus \{d\}} \frac{|\mathcal{B}|!\,(|\Omega| - |\mathcal{B}| - 1)!}{|\Omega|!} \left[\xi_{\mathcal{B} \cup \{d\}}(\mathbf{u}, \mathbf{v}) - \xi_{\mathcal{B}}(\mathbf{u}, \mathbf{v})\right] \tag{5.3}$$

where the difference $\left[\xi_{\mathcal{B} \cup \{d\}}(\mathbf{u}, \mathbf{v}) - \xi_{\mathcal{B}}(\mathbf{u}, \mathbf{v})\right]$ corresponds to the *marginal utility* of adding d to the *coalitional set* of dimensions $\mathcal{B} \subset \Omega$. Since the exhaustive computation of all coalitional contributions requires exponential time complexity, in place of ϕ_d we rely on the utility score μ_d in Equation (5.2), which is based only on maximal coalitions (with $\mathcal{B} = \Omega \setminus \{d\}$). This simple yet effective approach has a tractable computational complexity and attributes feature importance based on the effect of isolated dimensions.

Although previous work studied how to explain embedding dimensions by looking at feature values [GBH19, ŞUY+18, DNA19], we argue that studying the influence of dimensions on edge scores is more appropriate because it relies on the same functional dependency that is optimized during model training, i.e. the decoder. For this reason, from here we will rely on the above utility scores to define the interpretability of embeddings. Since we have not made any other assumption about the embeddings, apart from the inner product decoder, this procedure is appropriate for a broad range of representation learning models. In particular, both dense and sparse vector embeddings can be evaluated with this framework.

5.2.2 Interpretability metrics for latent dimensions

We define metrics to quantify the interpretability of dimensions based on their marginal utilities $\mu_d(\mathbf{u}, \mathbf{v})$. To evaluate the interpretability of a dimension d, we look at the set of edges with *positive* marginal utility. By considering only values greater than 0, we look at multiple induced subgraphs $\{\mathcal{G}_d\}_{d \in \Omega}$ with edge sets $\mathcal{E}_d = \{(u, v) \in \mathcal{E} : \mu_d(\mathbf{u}, \mathbf{v}) > 0\}$ that highlight relevant structures in different

dimensions. The focus is on positive payoffs because our main interest is to find those dimensions that are more effective in predicting a given edge, and we will leave the analysis of negative effects for future work.

Our goal is to relate "utility-induced" subgraphs to factors of the input graph that are easy for humans to understand. Since we are working with unattributed and homogeneous graphs, these factors can be seen as important structural components, which we identify with the *communities* of the graph. In reality, communities are one of the fundamental organizing principles in real-world graphs [GN02], where sets of nodes group into densely connected clusters due to specific affinities, and they appear as a reasonable choice to identify the meaning of representation dimensions.

In the following paragraphs, we introduce two interpretability metrics in terms of: (i) level of matching between per-dimension subgraphs $\{\mathcal{G}_d\}_{d\in\Omega}$ and the community structure; (ii) level of sparsity of edge subgraphs $\{\mathcal{E}_d\}_{d\in\Omega}$.

Community-aware metric

We evaluate *community-aware interpretability* measuring the relevance of edges $\mathcal{E}_d$ with retrieval scores. This procedure is conceptually similar to evaluation methods used in node-based community detection [YL13], but here adopts an edge-based approach. Starting from extracted link partitions $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots\}$ and a membership function $\pi : \mathcal{E} \to \mathcal{P}$, we define precision and recall metrics:

$$\text{precision}(\mathcal{E}_d, \mathcal{P}_i) = \frac{|\{(u,v) \in \mathcal{E}_d : \pi(u,v) = \mathcal{P}_i\}|}{|\mathcal{E}_d|} \tag{5.4}$$

$$\text{recall}(\mathcal{E}_d, \mathcal{P}_i) = \frac{|\{(u,v) \in \mathcal{E}_d : \pi(u,v) = \mathcal{P}_i\}|}{|\mathcal{P}_i|} \tag{5.5}$$

In this way we can determine the partition $\hat{\mathcal{P}}_d$ which better matches with dimension d, plus the corresponding interpretability score I_d:

$$\hat{\mathcal{P}}_d = \underset{\mathcal{P}_i \in \mathcal{P}}{\text{argmax}}\ \text{F1}(\mathcal{E}_d, \mathcal{P}_i); \qquad I_d^{(com)} = \max_{\mathcal{P}_i \in \mathcal{P}} \text{F1}(\mathcal{E}_d, \mathcal{P}_i) \tag{5.6}$$

where F1-score is the harmonic mean between $\text{precision}(\mathcal{E}_d, \mathcal{P}_i)$ and $\text{recall}(\mathcal{E}_d, \mathcal{P}_i)$. Higher values of $I_d^{(com)}$ indicate the dimension d is strongly associated with a single community. Global community-aware interpretability can be quantified with the average $I^{(com)} = \frac{1}{|\Omega|} \sum_{d\in\Omega} I_d^{(com)}$.

Sparsity-aware metric

We evaluate *sparsity-aware interpretability* by using the cardinality of edge sets $\{\mathcal{E}_d\}_{d\in\Omega}$. In fact, highly interpretable directions of the embedding space should

highlight subgraphs without exceeding in size. In other words, even without community structure information, we can anyhow quantify whether dimensions are associated with few structure-relevant edges. We formulate this idea inspired by entropy-based sparsity for explanation masks in graph neural networks [FKRA22]:

$$I_d^{(sp)} = -\frac{1}{\log|\mathcal{E}|} \sum_{(u,v)\in\mathcal{E}} \left(\frac{\mathbb{1}[(u,v)\in\mathcal{E}_d]}{Z_d}\right) \log\left(\frac{\mathbb{1}[(u,v)\in\mathcal{E}_d]}{Z_d}\right) \tag{5.7}$$

where $Z_d = \sum_{(u,v)\in\mathcal{E}} \mathbb{1}[(u,v)\in\mathcal{E}_d]$ is a normalization for the correct computation of the Shannon entropy. Lower values indicate that embedding dimensions are associated with small-sized subgraphs. Global sparsity-aware interpretability can be quantified with the average $I^{(sp)} = \frac{1}{|\Omega|}\sum_{d\in\Omega} I_d^{(sp)}$.

5.2.3 Node embeddings with interpretable dimensions

We have previously introduced the per-dimension utility score $\mu_d(\mathbf{u},\mathbf{v})$, associated with latent embedding direction d, and how it can be used to evaluate the interpretability of product-based decoder function. As previously shown in Figure 5.1, standard embedding methods (e.g., DEEPWALK) have poor interpretability mainly because they are trained with the sole goal of optimizing performance in neighborhood reconstruction. Our objective is to learn a new embedding function $h^* : \mathcal{V} \to \mathbb{R}^K$ which preserves both task performance and interpretable decoder scores over the edges.

Specifically for the textual domain, the interpretability of word embeddings can be promoted by facilitating the alignment of dimensions with concepts from external semantic sources [BAMK16, ŞUŞ+20], or by adding sparsity constraints during the training phase [PSB19, SGL+16]. However, due to the high popularity of some word embedding approaches [PSM14, MSC+13], post-processing techniques that are built upon these methods have been often preferred rather than interpretable-by-design methods [SPJ+18, FTY+15, FDJ+15]. In the field of graph learning, the restricted availability of taxonomical resources associated with real-world networks poses several limits in using knowledge bases to explain node embeddings [IKA20]. Consequently, learning interpretable embeddings for graph-structured data is an even more challenging task, as it cannot fully rely on anything other than topological information. For this reason, the few available methods perform clustering-aware regularization to learn explainable node representations [DNA19, RDSS19].

Here, we propose a fully unsupervised approach that can operate without external information or node attributes. Moreover, instead of building a new graph representation algorithm from scratch, we present a retrofitting technique that enables the processing of any pre-trained embedding with proven downstream performance. More formally, we present a retrofitting approach for node embeddings,

learning an opportune mapping $h : \mathbb{R}^D \rightarrow \mathbb{R}^K$ such that the function composition $h \circ \mathrm{enc}$ is exactly the interpretable embedding function $h^* : \mathcal{V} \rightarrow \mathbb{R}^K$ stated before. While similar algorithms for word embeddings enforce learning sparse representations [SPJ+18], we instead encourage the sparsity of the utility-induced subgraphs that directly affect the interpretability metrics defined in Section 5.2.2. Intuitively, we aim to have sparse explanations rather than sparse embeddings themselves, because it is more straightforward to associate them with human-understandable graph units. In the next, we will use both $\mathbf{u}^*$ and $h(\mathbf{u})$ to indicate the embedding vectors of node $u \in \mathcal{V}$ mapped with the encoder function h^*. We will also refer to K as the cardinality of the set containing the enumerated dimensions of the new space $\Omega^* = \{1, \ldots, K\}$.

Retrofitting node embeddings

Here we describe the retrofitting optimization task, whereas in the next section we further discuss the beneficial effects on the interpretability of latent dimensions. Since we aim to work in an unsupervised framework, we do not use external attributes to improve interpretability, but we rely only on structural graph information.

Inspired by disentangled learning [LTL+19], we enforce *orthogonality* in edge reconstruction patterns of representations h. Specifically, we optimise the element-wise products $\mathrm{h}_d(\mathbf{u}) \cdot \mathrm{h}_d(\mathbf{v}) \equiv m_d(u, v)$, collected as the entries of K continuous-valued graph masks $\{\mathbf{M}_d \in [0, 1]^{|\mathcal{V}|\times|\mathcal{V}|}\}_{d\in\Omega^*}$, which hide the structure-irrelevant edges for any dimension in $\mathbb{R}^K$. We implemented h as the latent projection of a single-layer autoencoder, namely $h(\mathbf{u}) = \sigma(\Psi^{(0)}\mathbf{u} + \mathbf{b}^{(0)})$, which returns $\tilde{\mathbf{u}} = \Psi^{(1)}h(\mathbf{u}) + \mathbf{b}^{(1)}$ as output. We define with $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times K}$ as the hidden layer matrix of the autoencoder, whose entries $\mathbf{H}_{ud} = \mathrm{h}_d(\mathbf{u}) = \mathrm{u}^*_d$ are the components of the post-processed embedding vectors.

We jointly optimise the hidden layer matrix $\mathbf{H}$, which depends on autoencoder parameters $\{\Psi^{(0)} \in \mathbb{R}^{K\times D}, \Psi^{(1)} \in \mathbb{R}^{D\times K}, \mathbf{b}^{(0)} \in \mathbb{R}^K, \mathbf{b}^{(1)} \in \mathbb{R}^D\}$ and masks matrices $\{\mathbf{M}_d \in [0, 1]^{|\mathcal{V}|\times|\mathcal{V}|}\}_{d\in\Omega^*}$ computed from $\mathbf{H}$, with the following losses:

Autoencoder Loss In order to preserve graph structural information incorporated in input node representations, we minimize $\mathcal{L}_{ac} = ||\mathbf{V} - \tilde{\mathbf{V}}||_F$, where $\mathbf{V}, \tilde{\mathbf{V}} \in \mathbb{R}^{|\mathcal{V}|\times D}$ are the input and reconstructed embedding matrices of the autoencoder.

Orthogonality Loss Inspired by graph clustering techniques [BGA20], we squeeze embedding mask matrices into one partition matrix $\mathbf{\Pi} \in \mathbb{R}^{K\times|\mathcal{V}|}$, with entries $\mathbf{\Pi}_{dv} = \sum_{u\in\mathcal{V}} m_d(u, v)$, induced by aggregating edge reconstruction scores with the same target node . In order to encourage structure-relevant subgraphs to be incorporated into different embedding axes, we optimize $\mathcal{L}_{orth} = ||\frac{\mathbf{\Pi}\mathbf{\Pi}^{\mathrm{T}}}{||\mathbf{\Pi}\mathbf{\Pi}^{\mathrm{T}}||_F} - \frac{\mathbf{I}_K}{||\mathbf{I}_K||_F}||_F$.

For scalability reasons, partition matrix entries $\mathbf{\Pi}_{dv} = \sum_u m_d(u,v) = h_d(\mathbf{v})\big[\sum_u h_d(\mathbf{u})\big]$ can be computed avoiding the explicit calculation and caching of $\mathcal{O}(K \times |\mathcal{V}| \times |\mathcal{V}|)$ parameter for graph masks, reducing the complexity to $\mathcal{O}(K \times |\mathcal{V}|)$.

Size Loss In order to avoid degenerate solutions, e.g. all relevant subgraphs reconstructed in the same dimension, we enforce the size of every mask $s_d = \sum_{u,v} m_d(u,v)$ to be non-zero. This constraint is accomplished by maximizing the entropy of the size variables $\{s_d\}_{d\in\Omega^*}$, opportunely normalized, $\mathcal{L}_{size} = \log|\Omega^*| + \sum_{d\in\Omega^*}\left(\frac{s_d}{\sum_{q\in\Omega^*} s_q}\right)\log\left(\frac{s_d}{\sum_{q\in\Omega^*} s_q}\right)$.

The full objective loss is given by:

$$\mathcal{L} = \mathcal{L}_{ac} + \mathcal{L}_{orth} + \mathcal{L}_{size} \tag{5.8}$$

Essentially, optimizing the loss $\mathcal{L}_{ac}$ ensures to preserve structural graph information, instead minimization of the losses $\mathcal{L}_{orth} + \mathcal{L}_{size}$ transform the original embedding space associating relevant graph clusters with different embedding dimensions.

Interpretable characteristics of retrofitted dimensions

Here we list remarkable properties obtained through our post-processing approach that directly affect the interpretability of retrofitted node embeddings. Although we do not go into theoretical detail, we empirically show the effectiveness of such an approach in the Experiments section. In particular, we reach the following interpretability features:

- *Nonnegativity*. Constraining weights to be nonnegative [CZ14] is usually an important requirement for interpretable learning models. The sigmoid function ensures the nonnegativity of representations $h(\mathbf{u})$ for any node $u \in \mathcal{V}$.

- *Decomposability*. It is often desirable for a graph model to produce edge likelihoods in an interpretable fashion [YL13]. The optimization of graph masks facilitates straightforward interpretations by decomposing the likelihood scoring function $\sum_{d\in\Omega^*} \mathrm{h}_d(\mathbf{u}) \cdot \mathrm{h}_d(\mathbf{v})$ into different bounded terms $\mathrm{h}_d(\mathbf{u}) \cdot \mathrm{h}_d(\mathbf{v}) \in [0,1]$ that represent the magnitude with which the edge (u,v) participates in the associated subgraph induced by $\mathcal{E}_d$.

- *Sparsity*. Encouraging models to use fewer features for prediction is a widely adopted constraint to promote interpretability [SPJ+18]. Due to regularization constraints employed to achieve non-degenerate orthogonality of graph masks $m_d(u,v)$, the decoded edge score $\sum_{d\in\Omega^*} \mathrm{h}_d(\mathbf{u}) \cdot \mathrm{h}_d(\mathbf{v})$ receives non-zero contributions only by few relevant dimensions.

Moreover, as we will empirically show in the Experiments, learning optimal graph masks directly affects the utility measures $\mu_d\big(h(\mathbf{u}), h(\mathbf{v})\big)$ used to evaluate interpretability. The following theorem states the per-dimension relations between utility scores and graph masks entries:

Theorem. *Let $h^* : \mathcal{V} \rightarrow \mathbb{R}^K$ be a retrofitted encoder function on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The per-dimension marginal utility score for edge $(u, v) \in \mathcal{E}$ can be expressed as:*

$$\mu_d\Big(h(\mathbf{u}), h(\mathbf{v})\Big) = \big[m_d(u,v) - \frac{1}{K}\sum_{d\in\Omega^*} m_d(u,v)\big]\left(\frac{1}{K} + \frac{1}{K^2} + \dots\right)$$

The proof is given in Appendix C, and it is valid for any graph encoder (in point of fact, entries of graph masks are equal to element-wise vector products).

From Decomposability and Sparsity properties described above for the function h, it follows that $\sum_{d\in\Omega^*} m_d(u,v) = \alpha_{uv}K$, where typically $\alpha_{uv} << 1$ is the fraction of non-zero elements in the inner product. With sufficient high dimensionality K, we can neglect infinitesimal terms in parenthesis obtaining $K \cdot \mu_d\Big(h(\mathbf{u}), h(\mathbf{v})\Big) + \alpha_{uv} \approx m_d(u,v)$ which expresses a simplified relation between dimensional graph masks and utility measure.

5.3 Experiments

In this section, we describe the experimental analyses oriented to study the interpretability of embedding methods from different perspectives. In particular, we aim to discover if the proposed retrofitting approach is more understandable than other models (in particular those ones that incorporate alternative interpretability constraints, e.g. sparsity of vector entries, self-clustering, etc.), and if it is capable of effectively achieving link prediction despite maintaining good scalability performance. The next sections specify which data and models are used for the comparison, as well as the description of the tasks performed in order to answer our research questions.

5.3.1 Datasets and baseline methods

We present our results on a variety of real-world benchmark datasets used in prior work [YSX+20]: three citation networks ("CORA", "CITESEER" and "PUBMED"), two social networks ("BLOGCATALOG" and "FLICKR"), and a web pages network ("WIKI"). Despite their original format, we restrict our analyses to the larger connected component of any graph, considered unweighted and undirected. We ran Louvain detection method [BGLL08] to extract the node-level communities $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots\}$. Moreover, we use two small-scale synthetic graphs

Table 5.1: *Summary statistics of synthetic and empirical graph datasets. In order: number of nodes $|\mathcal{V}|$, number of edges $|\mathcal{E}|$, number of extracted communities $|\mathcal{C}|$, average node degree and average clustering coefficient.*

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$	Average degree	Average clustering coeff.
RINGCLIQUES	160	736	16	9.20	0.960
SBMODEL	160	846	16	10.6	0.750
CORA	2,485	5,069	28	4.08	0.238
CITESEER	2,110	3,668	35	3.48	0.171
PUBMED	19,717	44,324	38	4.50	0.060
BLOGCATALOG	5,196	171,743	10	66.1	0.122
FLICKR	7,575	239,738	9	63.3	0.330
WIKI	2,357	11,592	17	9.84	0.383

consisting of sixteen 10-vertex cliques linked with different connectivity patterns: in "RINGCLIQUES", cliques are connected through single links to form a ring shape; in "SBMODEL", cliques are fully connected communities generated by Stochastic Block Model [Abb17] ($intra\_cluster\_density = 1$ and $inter\_cluster\_density = 0.01$). We report datasets statistics in Table 5.1.

For experimental comparisons with our approach, we use the following baseline methods:

- DEEPWALK [PARS14], skip-gram based model that computes node embeddings from random walks co-occurrence statistics. We train NODE2VEC[1] for 5 epochs with the following hyperparameters: $p = q = 1$, $T = 5$, $walk\_length = 10$, $num\_walks = 20$.
- GAE [TW16], neural network model with a GCN encoder trained on adjacency matrix reconstruction. The model[2] is trained for 200 iterations using Adam optimizer and learning rate 0.01 as described in the main paper. The GCN hidden layer size is taken double the output size.
- GEMSEC [RDSS19], a variation of DEEPWALK which jointly learns node embeddings and node clusters. We train the model[3] with the same configuration as DEEPWALK, plus the number of clusters fixed equal to the number of dimensions.
- SPINE [SPJ+18], a post-processing technique based on k-sparse denoising autoencoder to generate overcomplete sparse embeddings. We train the

[1] `https://github.com/eliorc/node2vec`
[2] `https://github.com/zfjsail/gae-pytorch`
[3] `https://github.com/benedekrozemberczki/karateclub`

original model[4] using DEEPWALK vectors as input, for 2000 iterations with sparsity coefficient 0.15 and learning rate 0.1.

We chose to apply the proposed retrofitting method on DEEPWALK and GAE embeddings, calling them *RF*-DEEPWALK and *RF*-GAE, to analyze the outcome when handling different inductive biases. In doing this, we use the same number of iterations and learning rate as with SPINE.

5.3.2 Evaluation tasks

Interpretation of dimensions

We compare all the methods in terms of per-dimension interpretability metrics defined in Section 5.2.2. About indexes $I_d^{(com)}$, we define the edge-level partition labels $\pi(u, v)$ using node-level Louvain communities with the assignment $\pi(u, v) = \{c(u), c(v)\}$, where $c : \mathcal{V} \to \mathcal{C}$ is the per-node community membership function. Despite the considered empirical graphs having ground-truth labels attached to single nodes, the use of detected communities is preferable because the usage of node metadata as structural ground-truth has been criticized by previous work [PLC17].

To compare models together, instead of computing global scores with the average over all the dimensions, we decided to focus on a subset of effective dimensions Ω_{eff} that encode the majority of edge information, removing the less impacting features. In particular, after sorting the scores $\{I_d^{(com)}\}_{d\in\Omega}$ and $\{I_d^{(sp)}\}_{d\in\Omega}$, we take from the top-ranked dimensions those ones that, cumulatively, have a positive utility in the reconstruction of at least 90% of graph edges, i.e. $|\bigcup_{d\in\Omega_{eff}} \mathcal{E}_d| = 90\%|\mathcal{E}|$. Thus we compute global scores as $I_{eff}^{(com|sp)} = \frac{1}{|\Omega_{eff}|} \sum_{d\in\Omega_{eff}} I_d^{(com|sp)}$.

Link prediction

We compare all the methods in terms of link prediction performance. To do so, before training every method, we randomly remove 10% of the edges that are used as positive examples for the link prediction task, whereas we sample the same number of node pairs from the set of non-existing links as negative examples. The task consists in ranking the collected node pairs with the scoring function ξ_Ω and evaluating the classification performance with the ROC-AUC score.

[4] https://github.com/harsh19/SPINE

5.4 Results

In this section, we first show the outcomes of interpretation experiments for the synthetic and empirical datasets (Section 5.4.1), then we evaluate the different approaches in terms of link prediction performance (Section 5.4.2). Finally, we compare the scalability of the embedding methods by measuring the training execution times (Section 5.4.3).

All the methods, except SPINE, are trained on empirical datasets to return embedding vectors with dimensions in the list $[2, 4, 8, 16, 32, 64, 128]$, which we call *output* dimensions. In synthetic-generated data, output dimensions are in the list $[2, 4, 8, 12, 16, 24, 32]$. Instead, we refer to *input* dimensions as the dimensionality of DEEPWALK and GAE vectors used for the proposed retrofitting (*RF*-DEEPWALK and *RF*-GAE) and for SPINE post-processing (*SPINE*-DEEPWALK), which are taken from the list $[8, 16, 32, 64, 128, 256, 512]$. For SPINE, due to the overcomplete layer, we chose output dimensions to be multiples of the input dimensions, according to integer multiplicative factors between $\times 1$ and $\times 8$.

In our analyses, we compare dense and sparse baseline methods separately. When comparing with sparse methods, we show the best performance of *SPINE*-DEEPWALK and *RF*-DEEPWALK across different input DEEPWALK dimensions. When comparing with dense methods, we show the best performance of *RF*-DEEPWALK and *RF*-GAE against other baseline methods across different output dimensions. Results for every method are reported with the average evaluation score and standard deviation over 5 training runs.

5.4.1 Interpretability

Synthetic datasets

Plot with interpretation scores $I_{eff}^{(com)}$ and $I_{eff}^{(sp)}$ for synthetic datasets are reported in Figures 5.2 and 5.3. In Figure 5.2, we observe that the model *RF*-DEEPWALK performs well in both interpretations based on communities and sparsity when compared with algorithms that return dense representations. In particular, near-perfect interpretability performance is achieved when the number of output embedding dimensions approaches 16, i.e. the number of synthetic cliques. In Figure 5.3, *RF*-DEEPWALK is even the best model when compared with the natively sparse method *SPINE*-DEEPWALK. In the same plot, DEEPWALK has increasing performance (higher community-aware scores and lower sparsity-aware scores) when augmenting the number of input embedding dimensions, and it reaches sparsity results comparable to *RF*-DEEPWALK with the maximum size 512. The latter observation suggests that DEEPWALK needs more embedding dimensions to reach the interpretability performance of *RF*-DEEPWALK.

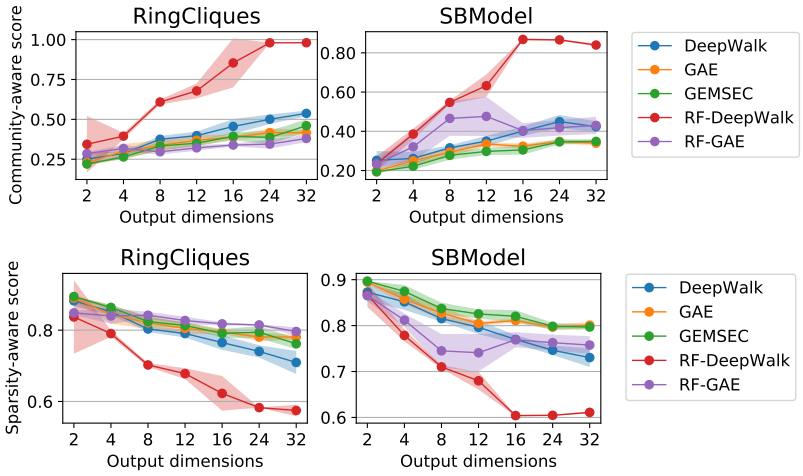


Figure 5.2: *Interpretation scores compared among RF-*DEEPWALK*, RF-GAE and different dense embedding methods trained on synthetic datasets, when varying the number of output dimensions and choosing the best score among models with a different number of input dimensions. On the top, we compare the community-aware scores (higher is better); on the bottom, we compare the sparsity-aware scores (lower is better).*

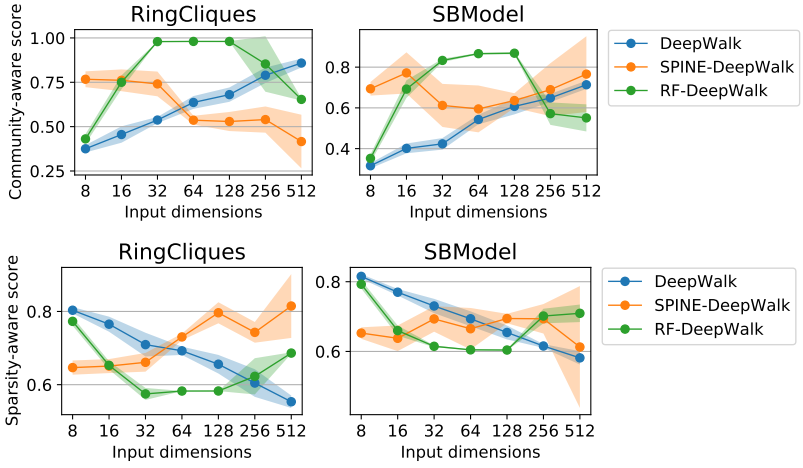


Figure 5.3: *Interpretation scores compared among* DEEPWALK*, SPINE-*DEEPWALK *and RF-*DEEPWALK *methods trained on synthetic datasets, when varying the number of input dimensions and choosing the best score among models with a different number of output dimensions. On the top, we compare the community-aware scores (higher is better); on the bottom, we compare the sparsity-aware scores (lower is better).*

Empirical datasets

Best interpretation scores $I_{eff}^{(com)}$ and $I_{eff}^{(sp)}$ for empirical datasets are reported in Tables 5.2 and 5.3. We notice that the combination *RF*-DEEPWALK performs well with respect to dense embeddings in almost every dataset, with both interpretations based on communities and sparsity. The model *RF*-GAE is less interpretable than *RF*-DEEPWALK, but still more interpretable than the other dense baselines. With respect to sparsity-based models, *RF*-DEEPWALK and *SPINE*-DEEPWALK are characterized by good interpretability, obtaining the best scores in half of the datasets each and outperforming the DEEPWALK embeddings. In Appendix C, we report extensive results for the entire set of dimension sizes, noticing that both interpretability metrics improve when increasing latent dimensions, confirming results found on synthetic data and giving an important insight that help to choose the embedding size in real-world applications.

5.4.2 Link prediction

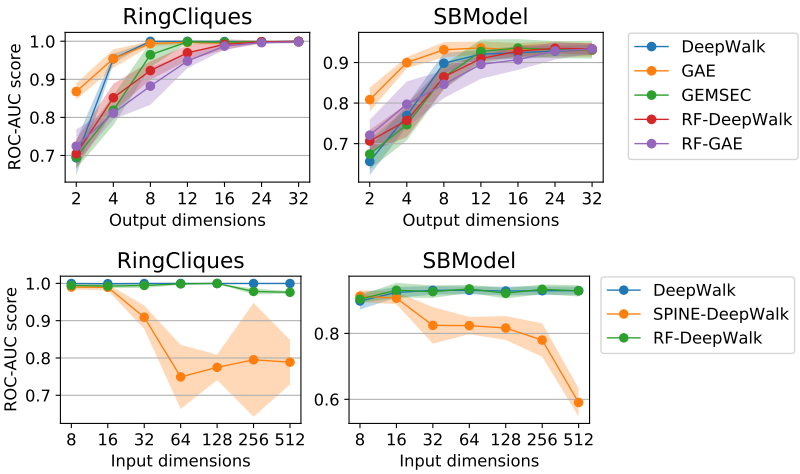


Figure 5.4: *ROC-AUC scores on link prediction for different embedding methods trained on synthetic datasets. On the top, we compare RF-*DEEPWALK*, RF-GAE and different dense embedding methods when varying the number of output dimensions and choosing the best score among models with a different number of input dimensions; on the bottom, we compare* DEEPWALK*, SPINE-*DEEPWALK *and RF-*DEEPWALK *when varying the number of input dimensions and choosing the best score among models with a different number of output dimensions.*

Table 5.2: *Community-aware scores for evaluating the interpretations of different embedding methods. For each dataset, we highlight the best score.*

	CORA	CITESEER	PUBMED	BLOGCATALOG	FLICKR	WIKI
DEEPWALK	44.8±0.8	43.3±1.4	42.2±1.1	43.2±0.7	49.9±10.5	44.5±2.5
GAE	37.2±0.6	42.0±1.0	48.7±1.9	49.6±0.6	62.5±2.1	46.0±1.4
GEMSEC	40.5±1.3	44.6±1.4	43.1±0.9	39.2±3.0	42.1±2.2	45.4±0.2
RF-DEEPWALK	**62.3±1.9**	**64.1±2.7**	**60.5±2.7**	59.0±2.3	**65.7±1.4**	**65.2±1.0**
RF-GAE	55.0±1.5	52.6±0.8	59.3±1.1	**60.0±0.8**	59.5±1.3	63.1±1.7
DEEPWALK	49.5±2.3	49.1±1.6	45.7±0.6	48.3±0.9	46.9±1.8	49.6±0.9
DEEPWALK+SPINE	59.0±5.1	54.3±6.0	**60.8±3.7**	**63.2±8.2**	**70.3±0.1**	61.1±1.5
RF-DEEPWALK	**62.3±1.9**	**64.1±2.7**	60.5±2.7	59.0±2.3	65.7±1.4	**65.2±1.0**

Table 5.3: *Sparsity-aware scores for evaluating the interpretations of different embedding methods. For each dataset, we highlight the best score.*

	CORA	CITESEER	PUBMED	BLOGCATALOG	FLICKR	WIKI
DEEPWALK	79.2±0.6	77.8±0.6	84.9±0.1	86.5±0.2	87.4±0.4	82.0±0.3
GAE	83.7±0.5	82.5±0.4	85.1±0.5	87.1±0.9	78.5±0.1	84.4±0.2
GEMSEC	82.7±0.3	81.7±0.3	86.5±0.2	91.1±0.3	91.6±0.2	84.6±0.4
RF-DEEPWALK	**66.0±1.2**	**63.0±1.5**	**68.4±24.2**	**74.9±0.6**	**37.2±6.9**	**68.8±0.3**
RF-GAE	72.8±0.1	72.8±0.9	80.5±0.3	82.0±1.2	71.1±2.1	75.6±0.3
DEEPWALK	76.7±0.5	75.5±0.3	83.6±0.1	84.0±0.1	87.4±0.4	79.6±0.4
DEEPWALK+SPINE	69.7±2.0	70.3±2.5	**74.1±3.3**	**61.0±8.9**	72.9±12.5	**52.9±9.7**
RF-DEEPWALK	**66.0±1.2**	**63.0±1.5**	74.8±0.8	74.9±0.6	**68.1±6.0**	68.8±0.3

Table 5.4: *ROC-AUC scores for evaluating the link prediction performance of different embedding methods. For each dataset, we show in boldface letters the best score, and we higlight with a gray background other methods that reach at least* 95% *of the best model score.*

	CORA	CITESEER	PUBMED	BLOGCATALOG	FLICKR	WIKI
DEEPWALK	92.3±0.7	94.5±0.7	**96.4±0.2**	75.6±0.1	65.6±0.1	84.8±0.3
GAE	92.6±0.4	**95.2±0.6**	95.8±0.2	**86.9±0.3**	89.4±0.4	**94.2±0.2**
GEMSEC	**92.8±0.6**	94.2±0.7	95.1±0.3	79.4±0.2	67.0±0.3	90.5±0.4
RF-DEEPWALK	92.2±0.7	94.7±1.0	93.3±1.2	74.7±0.6	75.9±0.9	90.5±0.4
RF-GAE	88.6±1.2	92.0±0.9	96.1±0.1	82.0±1.0	**91.1±0.3**	**94.2±0.4**
DEEPWALK	**92.5±0.6**	**94.9±0.8**	**96.5±0.1**	**75.8±0.1**	66.7±0.2	84.8±0.2
DEEPWALK+SPINE	87.2±1.6	89.7±2.1	90.2±0.6	67.3±1.4	63.2±3.5	81.6±1.4
RF-DEEPWALK	92.2±0.7	94.7±1.0	93.3±1.2	74.7±0.6	**75.9±0.9**	**90.5±0.4**

Synthetic datasets

Plots with ROC-AUC scores in link prediction are shown in Figure 5.4 for synthetic datasets. In the top panel, we observe that any model converges to optimal performance increasing the output embedding size. GAE is the best model because it achieves almost-best scores with only 4 dimensions, and in general any model reaches the optimum at roughly 16 dimensions. With respect to the bottom panel of Figure 5.4, we observe that *SPINE*-DEEPWALK performance degrades with more input dimensions, contrary to the other methods which maintain optimal scores with any input size.

Empirical datasets

Best ROC-AUC scores in link prediction are reported in Table 5.4 for empirical datasets. In citation networks, we notice that all dense models perform close to the optimal one, and in other datasets best scores are obtained by GAE and *RF*-GAE. For sparse embeddings, *SPINE*-DEEPWALK is consistently outperformed by *RF*-DEEPWALK, with the latter that obtains a comparable performance (or even superior in FLICKR and WIKI) with DEEPWALK.

5.4.3 Scalability

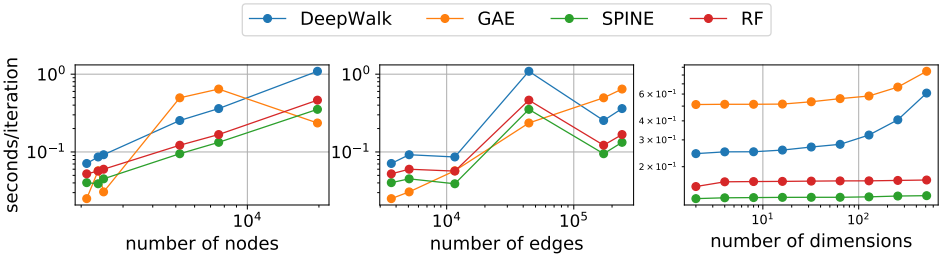


Figure 5.5: *Normalized execution times for different embedding methods when trained on multiple datasets. On the left, we compare running times in function of the number of nodes with 128-dim output embeddings; in the center, we compare running times in function of the number of edges with 128-dim output embeddings; on the right, we compare running times in function of the number of embedding dimensions in FLICKR dataset.*

Training times for different methods are reported in Figure 5.5, where we normalize the intervals with respect to the number of iterations/epochs. For DEEPWALK we also divide respect to the num_walks parameter to cut out the dependency from the number of walks sampled per node. From left and center panels, we

observe that training times for the GAE method scale with the number of edges, instead in all the other cases the running times scale with the number of nodes. Moreover, *RF* has slightly higher training times than SPINE, and both are faster than DEEPWALK. On the right panel, we recognize that in GAE and DEEPWALK the training times have a notable dependency with respect to the number of embedding dimensions, while this is less significant in the other ones.

5.5 Summary

In this chapter, we introduce a comprehensive framework for interpreting node embedding techniques. Relying on feature importance methods derived from Shapley values, we propose a novel, model-agnostic utility measure for determining the per-dimension contributions to edge reconstruction scores. Additionally, we establish appropriate metrics for evaluating the interpretability of embeddings based on community structure and sparsity. Finally, we propose an auto-encoder framework to further enhance the interpretability of existing node embedding methods.

Experimental results demonstrate that our approach improves human understandability of embedding dimensions compared to standard methods, while maintaining comparable performance on link prediction tasks. Furthermore, our model is superior to the sparse method SPINE, achieving better results in link prediction tasks. Additionally, our approach is scalable and suitable for use in graphs with diverse edge densities, and it demonstrates acceptable performance in terms of different input graph sizes.

Overall, our proposed framework for interpreting node embedding methods provides a valuable tool for understanding the underlying structure of complex networks and offers a means for enhancing the interpretability of existing embedding methods. This framework has the potential to facilitate further research in the field of network analysis and improve the interpretability of node embedding methods for real-world applications.

CHAPTER 6

Conclusions

THE widespread use of technology and the internet has led to an unprecedented amount of behavioral information being generated and collected. This has created new challenges for representing, analyzing and understanding the structure and dynamics of techno-social systems.

Graphs are a natural way to represent pairwise interconnected relations that characterize a wide range of complex systems, and traditional network-based methods of analysis have been widely used for mining knowledge from relational data. However, analyzing and understanding empirical graph data structures is becoming more and more a challenging task, due to the increasing complexity and heterogeneity of such data. Network representation learning and low-dimensional embeddings of graphs are examples of tools that have been developed to model interconnected structures with a data-driven perspective. By leveraging similarities and hidden geometry of graph-shaped data, these techniques learn latent representations that help to understand the original data, revealing the important structures, and enabling further analyses. Additionally, these latent encodings can be used to perform various downstream tasks, including link prediction, multi-label node classification, and graph clustering.

Real-world networks are inherently dynamic, with connections that change over time, and accounting for their time-evolving nature poses important challenges in

terms of modeling and analysis. While existing methods have been largely focused on static network data, the relevance of applying graph representation learning to time-varying networks has become more prominent, especially with the growing amount of high-resolution time series data on social, biological, and technical systems.

Moreover, the network science and graph learning communities have recently highlighted the significance of non-dyadic, higher-order interactions that cannot be decomposed into multiple dyadic relations, which are typically captured by standard network models. Failing to account for higher-order structures, which often play an important role in our understanding of real-world interacting systems, presents a current limitation of traditional graph formalism. Despite the growing interest in extending graphs toward higher-order models, we still lack a solid understanding of representation learning for such complex interactions. It specifically limits our ability to apply machine learning on socio-technical systems data, where non-dyadic interactions are common.

Finally, while the majority of previous research has focused on the development of more performant learning architectures, explaining resulting decisions is a crucial challenge for the responsible application of representation learning and feature extraction in real-world prediction tasks. In recent years, there has been a growing emphasis on addressing the limitations of explainability in representation learning, particularly in the domain of graph machine learning. This challenge is closely linked with the interpretability of the learned feature dimensions, and modifying representation learning techniques to produce interpretable features is thus a crucial research topic in machine learning for graphs.

This thesis studies novel representation learning techniques for graph-structured complex systems, presenting empirical evidence of their predictive power and interpretability. This section summarizes our main research achievements and presents a perspective on potential future investigations.

6.1 Main Contributions

Contributions of this thesis can be described as follows:

- Firstly, in Chapter 3 and in [PP22] we introduced Higher-Order Skip-Gram with Negative Sampling (HOSGNS), an embedding method for time-resolved graphs that uses higher-order co-occurrences over time-aware random walks to learn low-dimensional encodings of node and temporal slices. We showed that the model effectively performs implicit tensor factorization of different higher-order representations of time-varying graphs (e.g., the supra-adjacency representation [SOBC21]) and achieves state-of-the-art performance in several downstream tasks for time-evolving networks of face-to-face proximity

contacts, i.e. temporal classification of epidemic states, event prediction and reconstruction.

- Secondly, in Chapter 4 and in [PPP22] we studied low-dimensional embeddings of simplicial complexes for link prediction and reconstruction on higher-order structures. We showed how to carefully design network reconstruction [TCW+18] and link prediction [KSSB20] in the case of polyadic edges, and we conducted experiments to analyze the performance of simplicial complex embeddings [BHL+19] in contrast with traditional and spectral approaches. Our findings highlight the effectiveness of performing classification tasks on hyperedges, even without explicit records of higher-order links in training data [BAB+21].

- Lastly, in Chapter 5 we discussed the interpretability [DLH19] of latent dimensions for node embeddings in graphs. We proposed an attribution method to compute per-dimension importance with product-based edge scoring, applied to latent directions of pre-trained embeddings. By means of this method, we are able to evaluate whether encoded node dimensions are associated with relevant subgraphs (e.g, communities [GN02]) and quantify the sparsity level of this association. Moreover, we presented a novel retrofitting approach capable to enhance the interpretability of existing embedding methods, maintaining optimal performance in link prediction.

Overall, the research presented in this dissertation has made significant contributions to the field of representation learning and its applications to temporal and higher-order structure, in addition to shedding light on the interpretability of such models when applied to graph-structured data. We present some potential research questions for future work in the paragraphs below.

6.2 Future Work

Representation learning is a broad field of research that encompasses several aspects and has been extensively studied from various perspectives in the last decade. In this thesis, we have directed our attention towards studying the advantages and constraints of representation learning for socio-technical systems structured as graphs, with an emphasis on empirical understanding. The research findings we have presented in the previous chapters provide a solid foundation for further investigation and exploration in the future. Specifically, our work has identified several promising research directions at the intersection between complex systems and machine learning.

The ability to estimate the outcome of dynamical processes [BBV08] is a key challenge in network science, and nowadays there is a growing attraction for

integrating classical approaches based on mechanistic modeling with machine learning [RPC+19, NTLL19, MLA21]. In the context of infectious disease forecasting, the COVID-19 pandemic has underscored the importance of effective tools to predict and control the spread of contagions [CSL+21, CRL+22]. From a machine learning perspective, novel techniques based on graph neural networks have been proposed to tackle epidemic forecasting tasks [KBL+20, GSQ+21, FDR22, TRDL+22], and the approach presented in Chapter 3 is situated in this framework of research. However, several advancements are necessary to improve the effectiveness and reliability of these methodologies for epidemic prevention and mitigation [BBB+21]. In fact, these methods often require complete knowledge of the social network topology [TRDL+22], which makes them unsuitable to operate in a realistic scenario for practical interventions. Moreover, they are not robust to the effects of noise and errors in tracing data [SOBC21].

Predicting the future evolution of real-world complex systems is another ongoing research topic [VZMZ15, SYS16]. Even in the binary case, the analysis of link predictability is a current challenge [Zho21], and the integration of established theoretical frameworks [ZX15, SFX+20] with machine learning techniques for link prediction [DM20, GHG+20] can represent a crucial step towards dealing with the predictability of empirical time-evolving networks [TDS+20]. Contributions of Chapter 4 suggest that forthcoming goals should entail gaining a more comprehensive understanding of the key topological and temporal features that impact the emergence of new interactions [LPZ+15, GPAGS20]. Future research should also explore the significance of higher-order evolving patterns [CS22, KKS20] in modeling and predicting social behavior accurately. This is especially important, as these patterns have been shown to play a critical role in shaping the social dynamics [IPBL19, CKC+21, IPBB22].

The increasing interest in the field of link prediction is leading to further investigations of explainable models for achieving the task [BBM14, EBT16]. The extraction of explanations in the context of edge prediction involves identifying the most influential features of the input graph, such as subgraphs or node attributes, that contribute to the prediction of a new link [RFMT22, ZZS+23]. In Chapter 5, a similar problem has been addressed in the interpretation of latent dimensions for shallow embedding models. However, there is a significant lack of exploration into the use of explanation models for GNN-based link prediction [ZC18, ZZXT21], despite their widespread usage in node and graph classification [YYW+21, LCX+20, YBY+19]. Further research in this area would provide insight into how structural network features positively or negatively impact edge prediction, possibly enabling causal explanations of link predictability [ZLW+22].

Future work includes addressing scalability limitations that affect the application of representation learning to large-scale graphs [HFZ+20], even exasperated by the inclusion of temporal [LYZ+23] and higher-order interactions [MSWP22]. In

particular, the embedding models utilized in this thesis can be significantly improved in terms of space and time complexity. One potential approach to efficiently handle temporal data, especially in streaming-oriented applications, is the integration of incremental learning techniques [GSG+22] into existing algorithms. Additionally, to mitigate the challenges posed by growing combinatorial complexities (e.g., increasing orders of simplices), solutions based on decentralized algorithms [GWS+20], or hierarchical coarse graining approaches [LGP21, BMD+20] may be implemented to reduce computational costs.

Appendices

Appendix A

Low-rank Tensor Decomposition

Low-rank tensor decomposition [KB09] aims to factorize a generic tensor into a sum of rank-one tensors. For example, given a 3rd-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, the rank-$R$ decomposition of $\boldsymbol{\mathcal{X}}$ takes the form of a ternary product between three factor matrices:

$$(\boldsymbol{\mathcal{X}})_{ijk} \approx [\![\mathbf{a}_i, \mathbf{b}_j, \mathbf{c}_k]\!] \equiv \sum_{r=1}^{R} \mathbf{A}_{ir}\mathbf{B}_{jr}\mathbf{C}_{kr} \tag{A.1}$$

where $\mathbf{a}_i$, $\mathbf{b}_j$, $\mathbf{c}_k \in \mathbb{R}^R$ are rows of the factor matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$ and $\mathbf{C} \in \mathbb{R}^{K \times R}$. If Equation (A.1) holds with the equality, the above operation is called Canonical Polyadic (CP) decomposition. For 2nd-order tensors (matrices) the operation is equivalent to the low-rank matrix decomposition ($\mathbf{X} \approx \mathbf{A}\mathbf{B}^{\mathrm{T}}$).

For a generic Nth-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, low-rank decomposition is expressed as:

$$(\boldsymbol{\mathcal{X}})_{i_1 i_2 \ldots i_N} \approx [\![\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{i_2}, \ldots, \mathbf{a}^{(N)}_{i_N}]\!] \equiv \sum_{r=1}^{R} \mathbf{A}^{(1)}_{i_1 r}\, \mathbf{A}^{(2)}_{i_2 r} \ldots \mathbf{A}^{(N)}_{i_N r} \tag{A.2}$$

where $\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{i_2}, \ldots, \mathbf{a}^{(N)}_{i_N} \in \mathbb{R}^R$ ($i_n \in \{1, \ldots, I_n\},\ n \in \{1, \ldots, N\}$) are rows of factor matrices $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times R}$, $\ldots$, $\mathbf{A}^{(N)} \in \mathbb{R}^{I_N \times R}$.

Derivation of the HOSGNS Loss Function

We consider a set of training samples $\mathcal{D} = \{(i_1, i_2, \dots, i_N) : \ i_1 \in \mathcal{V}_1, \ i_2 \in \mathcal{V}_2, \ \dots, \ i_N \in \mathcal{V}_N\}$ obtained by collecting co-occurrences among elements from N sets $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_N$. We denote as $\#(i_1, i_2, \dots, i_N)$ the number of times the tuple $(i_1, i_2, \dots, i_N)$ appears in $\mathcal{D}$. Similarly we use $\#i_n$ as the number of times each distinct element occurs in $\mathcal{D}$, with relative frequencies $P_{\mathcal{D}}(i_1, \dots, i_N) = \frac{\#(i_1, \dots, i_N)}{|\mathcal{D}|}$ and $P_{\mathcal{D}}(i_n) = \frac{\#(i_n)}{|\mathcal{D}|}$.

Optimization is performed as a binary classification task, where the objective is to discern occurrences actually coming from $\mathcal{D}$ from random occurrences. We define the likelihood for a single observation $(i_1, \dots, i_N) \in \mathcal{D}$ by applying a sigmoid to the higher-order inner product $[\![\cdot]\!]$ of corresponding D-dimensional representations:

$$P[\ (i_1, \dots, i_N) \in \mathcal{D} \mid \mathbf{a}^{(1)}_{i_1}, \dots, \mathbf{a}^{(N)}_{i_N}\ ] = \sigma\big(\ [\![\mathbf{a}^{(1)}_{i_1}, \dots, \mathbf{a}^{(N)}_{i_N}]\!]\ \big) \tag{A.3}$$

where we have N trainable embedding matrices $\mathbf{A}^{(1)} \in \mathbb{R}^{|\mathcal{V}_1| \times D}, \dots, \mathbf{A}^{(N)} \in \mathbb{R}^{|\mathcal{V}_N| \times D}$ and each embedding vector $\mathbf{a}^{(n)}_{i_n}$ is the i_n-th row of the matrix $\mathbf{A}^{(n)}$. We define the loss with negative sampling fixing i_1 and picking negative tuples $(\nu_2, \dots, \nu_N)$ according to the noise distribution $P_{\mathcal{N}}(\nu_2, \dots, \nu_N) = \prod_{n=2}^{N} \frac{\#\nu_n}{|\mathcal{D}|} \equiv \prod_{n=2}^{N} P_{\mathcal{D}}(\nu_n)$:

$$\log \sigma\big([\![\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{i_2}, \dots, \mathbf{a}^{(N)}_{i_N}]\!]\big) + \kappa \cdot \mathop{\mathbb{E}}_{(\nu_2 \dots \nu_N) \sim P_{\mathcal{N}}} \Big[\log \sigma\big(-[\![\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{\nu_2}, \dots, \mathbf{a}^{(N)}_{\nu_N}]\!]\big) \Big] \tag{A.4}$$

The expectation term can be explicited:

$$\mathop{\mathbb{E}}_{(\nu_2 .. \nu_N) \sim P_{\mathcal{N}}} \Big[\log \sigma\big(-[\![\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{\nu_2}, .., \mathbf{a}^{(N)}_{\nu_N}]\!]\big) \Big] = \sum_{j_2 .. j_N} P_{\mathcal{N}}(j_2, .., j_N) \log \sigma\big(-[\![\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{j_2}, .., \mathbf{a}^{(N)}_{j_N}]\!]\big)$$

Weighting the loss error for each tuple $(i_1, i_2, \dots, i_N)$ with their empirical probability $P_{\mathcal{D}}(i_1, i_2, \dots, i_N)$, and defining $[\![\mathbf{a}^{(1)}_{i_1}, \mathbf{a}^{(2)}_{i_2}, \dots, \mathbf{a}^{(N)}_{i_N}]\!] \equiv m_{i_1 i_2 \dots i_N}$, we obtain the global objective with the sum over all combinations of vocabulary elements:

$$\begin{aligned}
\mathcal{L} &= - \sum_{i_1 i_2 \dots i_N} P_{\mathcal{D}}(i_1, i_2, .., i_N) \Big[\log \sigma(m_{i_1 i_2 \dots i_N}) + \kappa \sum_{j_2 \dots j_N} P_{\mathcal{N}}(j_2, .., j_N) \log \sigma(-m_{i_1 j_2 \dots j_N}) \Big] \\
&= - \sum_{i_1 i_2 \dots i_N} P_{\mathcal{D}}(i_1, i_2, .., i_N) \log \sigma(m_{i_1 i_2 \dots i_N}) + \\
&\qquad - \kappa \sum_{i_1 i_2 \dots i_N} P_{\mathcal{D}}(i_1, i_2, .., i_N) \sum_{j_2 \dots j_N} P_{\mathcal{N}}(j_2, .., j_N) \log \sigma(-m_{i_1 j_2 .. j_N})
\end{aligned}$$

In the second term, we can notice that only $P_{\mathcal{D}}(i_1, i_2, \dots, i_N)$ depends on the $N - 1$ indices $(i_2, \dots, i_N)$, so performing the sum over that subset of indices

we obtain the marginal distribution $\sum_{i_2 \dots i_N} P_{\mathcal{D}}(i_1, i_2, \dots, i_N) = P_{\mathcal{D}}(i_1)$. Finally renaming indices $\{j_h\} \to \{i_h\}$ and observing that $P_{\mathcal{D}}(i_1) P_{\mathcal{N}}(i_2, \dots, i_N) \equiv P_{\mathcal{N}}(i_1, i_2, \dots, i_N)$, we obtain the final loss:

$$\mathcal{L}^{(hosgns)} = -\sum_{i_1 \dots i_N} \Big[P_{\mathcal{D}}(i_1, .., i_N) \log \sigma(m_{i_1 \dots i_N}) + \kappa \, P_{\mathcal{N}}(i_1, .., i_N) \log \sigma(-m_{i_1 \dots i_N}) \Big] \tag{A.5}$$

Implicit Tensor Factorization Theorem

Theorem. *Let* $\mathcal{D} = \{(i_1, i_2, \dots, i_N),\ i_1 \in \mathcal{V}_1,\ i_2 \in \mathcal{V}_2,\ \dots,\ i_N \in \mathcal{V}_N\}$ *be a training set of higher-order co-occurrences and* $\mathrm{PMI}(i_1, .., i_N) = \log \left(\frac{P_{\mathcal{D}}(i_1, .., i_N)}{P_{\mathcal{N}}(i_1, .., i_N)} \right)$ *the entries of the pointwise mutual information tensor computed from* $\mathcal{D}$*. Let* $\mathbf{A}^{(1)} \in \mathbb{R}^{|\mathcal{V}_1| \times D}, \dots, \mathbf{A}^{(N)} \in \mathbb{R}^{|\mathcal{V}_N| \times D}$ *be embedding matrices of* HOSGNS*. For* D *sufficiently large,* HOSGNS *has the same global optimum as the canonical polyadic decomposition of* SPMI_κ*, the PMI tensor shifted by* $\log \kappa$*.*

Proof. We consider each relation $[\![\mathbf{a}^{(1)}_{i_1}, \dots, \mathbf{a}^{(N)}_{i_N}]\!] \equiv m_{i_1 \dots i_N}$ as the entry of a tensor $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{|\mathcal{V}_1| \times \dots \times |\mathcal{V}_N|}$. The global loss $\mathcal{L}^{(hosgns)} = \sum_{i_1 \dots i_N} \ell(i_1, \dots, i_N)$ in Equation (A.5) is the sum of local losses computed from elements of $\boldsymbol{\mathcal{M}}$:

$$\ell(i_1, .., i_N) = -\big[P_{\mathcal{D}}(i_1, .., i_N) \log \sigma(m_{i_1 \dots i_N}) + \kappa \, P_{\mathcal{N}}(i_1, .., i_N) \log \sigma(-m_{i_1 \dots i_N}) \big]$$

For sufficiently large D (i.e., allowing for a perfect reconstruction of SPMI_κ), each $m_{i_1 \dots i_N}$ can assume a value independently to the others, and we can treat the loss function $\mathcal{L}$ as a sum of independent addends, restricting the optimization problem to looking at the local objective and its derivative respect to $m_{i_1 \dots i_N}$:

$$\begin{aligned} \frac{\partial \ell(i_1, .., i_N)}{\partial m_{i_1 \dots i_N}} &= \kappa \, P_{\mathcal{N}}(i_1, .., i_N) \sigma(m_{i_1 \dots i_N}) - P_{\mathcal{D}}(i_1, .., i_N) \big[1 - \sigma(m_{i_1 \dots i_N}) \big] \\ &= \big[P_{\mathcal{D}}(i_1, .., i_N) + \kappa \, P_{\mathcal{N}}(i_1, .., i_N) \big] \sigma(m_{i_1 \dots i_N}) - P_{\mathcal{D}}(i_1, .., i_N) \end{aligned}$$

where we have used $\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x))$. To compare the derivative with zero, we use the identities $P_{\mathcal{D}} = (P_{\mathcal{D}} + \kappa \, P_{\mathcal{N}})(1 + \frac{\kappa \, P_{\mathcal{N}}}{P_{\mathcal{D}}})^{-1}$ and $(1 + x)^{-1} = \sigma(\log x^{-1})$:

$$\frac{\partial \ell(i_1, .., i_N)}{\partial m_{i_1 \dots i_N}} = [P_{\mathcal{D}}(i_1, .., i_N) + \kappa \, P_{\mathcal{N}}(i_1, .., i_N)] \left[\sigma(m_{i_1 \dots i_N}) - \sigma\left(\log \frac{P_{\mathcal{D}}(i_1, .., i_N)}{\kappa \, P_{\mathcal{N}}(i_1, .., i_N)} \right) \right]$$

from which it follows that the derivative is 0 when elements $m_{i_1 \dots i_N}$ are equal to the shifted PMI tensor entries:

$$\frac{\partial \ell(i_1, .., i_N)}{\partial m_{i_1 \dots i_N}} = 0 \ \Leftrightarrow \ \sum_{r=1}^{D} \mathbf{A}^{(1)}_{i_1 r} \dots \mathbf{A}^{(N)}_{i_N r} = \log \left(\frac{P_{\mathcal{D}}(i_1, .., i_N)}{\kappa \, P_{\mathcal{N}}(i_1, .., i_N)} \right) = \mathrm{SPMI}_\kappa(i_1, .., i_N) \tag{A.6}$$

Since we have assumed that D is large enough to ensure an exact reconstruction of SPMI_κ, and this is true if $D \approx R = \mathrm{rank}(\mathrm{SPMI}_\kappa)$, Equation (A.6) is consistent with the canonical polyadic decomposition of the shifted PMI tensor. □

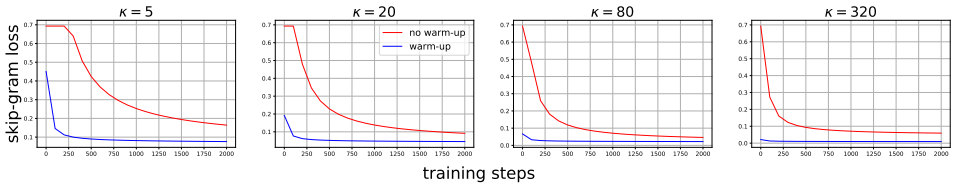


Figure A.1: *Impact of additional warm-up steps in the decrease of* $\mathrm{HOSGNS}^{(dyn)}$ *loss function* $\mathcal{L}^{(\mathrm{bce})}$ *with respect to the number of training iterations, for* LYONSCHOOL *dataset and with different negative sampling sizes* κ*. The loss function is normalized with a factor* $(\kappa + 1)$ *for accounting the different contributions of the negative sampling parameter in* $\mathcal{L}^{(\mathrm{bce})}$*.*

Description of the Warm-up Procedure

Here we propose a warm-up strategy with the aim of finding an advantageous configuration of model parameters to initialize trainable weights. In particular, we show that we can preliminarily optimize embedding vectors in order to ensure that all higher-order products $m_{ijk\ldots} = [\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, ..]\!]$ return the same quantity m, regardless of the indices combination $(i, j, k, ..)$. The value m can be chosen in order to make the cross-entropy error as minimum as possible before passing empirical data samples to the model. We start with a random initialization where embedding weights are realizations of random variables i.i.d. according to a normal distribution:

$$\mathbf{V}_{ir}, \mathbf{C}_{jr}, \mathbf{T}_{kr}, .. \sim \mathcal{N}(0, D^{-2}) \ , \ r = 1 \ldots D$$

Once chosen m we can fix Hadamard products by optimizing a squared error loss function:

$$\mathcal{L}^{(\mathrm{warm-up})} = \sum_{ijk\ldots} \Big([\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, ..]\!] - m \Big)^2 \tag{A.7}$$

The optimal value of m is stated by the following theorem:

Theorem. *Let* $m_{ijk\ldots} = [\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, ..]\!]$ *be the higher-order inner product among multiple embedding entities in HOSGNS, and* $\mathcal{Q} = \{m_{ijk\ldots},\ i, j \in \mathcal{V},\ k \in \mathcal{T}, \ldots\}$ *the set of embedding products over all the combinations of indices* $(i, j, k, ..)$*. Assuming the same value for each element* $m_{ijk\ldots} \equiv m$ *in the set* $\mathcal{Q}$*, the cross-entropy error of the model is minimum when* $m = -\log \kappa$*.*

Proof. Given the hypothesis, the objective function in Equation (3.9) takes the

form:

$$\mathcal{L}^{(\mathrm{bce})} = -\frac{1}{B}\Big[\sum_{(ijk\dots)\sim P_{\mathcal{D}}}^{B} \log\sigma(m) + \kappa \sum_{(ijk\dots)\sim P_{\mathcal{N}}}^{B} \log\sigma(-m)\Big]$$
$$= -[\log\sigma(m) + \kappa\log\sigma(-m)] \equiv \ell(m)$$

where m is the returned value for each $m_{ijk\dots}$. Solving the equation $\frac{dl}{dm} = 0$ we get:

$$\frac{1}{\sigma(m)}\frac{d\sigma}{dx}\Big|_{x=m} - \kappa\frac{1}{\sigma(-m)}\frac{d\sigma}{dx}\Big|_{x=-m} = 0 \Rightarrow \kappa = \frac{\sigma(-m)}{\sigma(m)} = e^{-m} \Rightarrow m = -\log\kappa$$

□

In Figure A.1 is shown the effectiveness of the addition of extra warm-up steps in loss optimization.

Additional Results of Embedding Representations in Downstream Tasks

In Figures A.2, A.3 and A.4 we report a sensitivity analysis with the effect of the embedding size D, the negative sampling constant κ and the number of training steps E on prediction performances in node classification and event-related tasks. In every experiment, HOSGNS embeddings have been combined using Hadamard product.

In Figure A.5 we show the prediction performance of embedding models DYANE and HOSGNS$^{(dyn)}$ in the node classification task, changing the width of the aggregation window in SocioPatterns data. In HOSGNS$^{(dyn)}$ we notice a modest descent of classification performance for larger aggregation windows, while results for DYANE are more unstable, resulting in a less robust performance with respect to the proposed HOSGNS model.

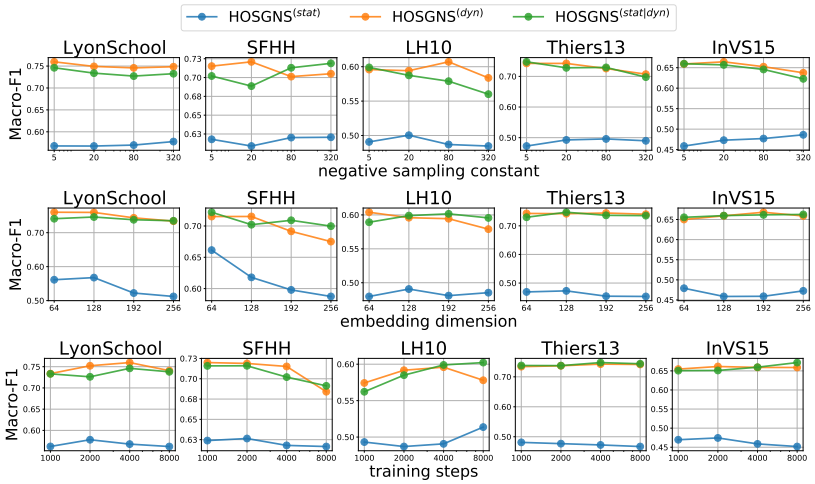


Figure A.2: *Macro-F1 scores related to the node classification of SIR states with epidemic parameters* $(\beta, \mu) = (0.125, 0.001)$*, computed with HOSGNS embeddings. On the top, classification scores are reported varying the negative sampling parameter* κ*; in the center, varying the embedding dimension* D*; on the bottom, varying the number of training iterations* E*. In each panel remaining parameters are fixed to* $D = 128$*,* $\kappa = 5$ *and* $E = 4000$*.*

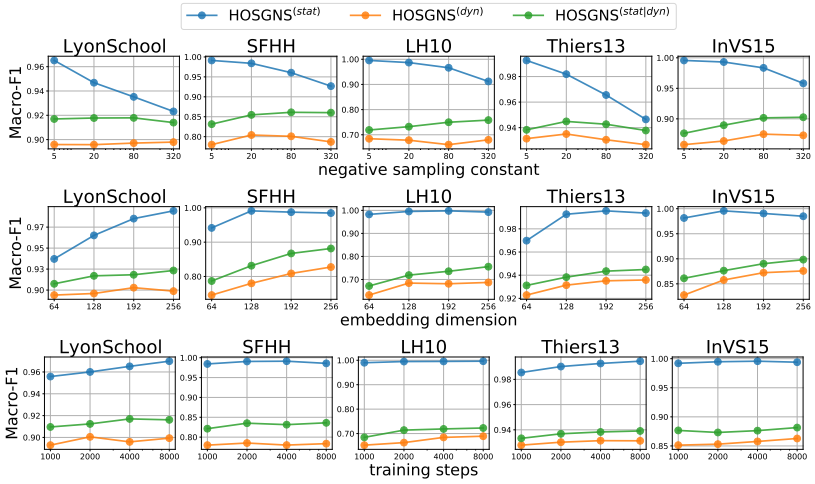


Figure A.3: *Macro-F1 scores related to temporal event reconstruction, computed with HOSGNS embeddings. On the top, classification scores are reported varying the negative sampling parameter* κ*; in the center, varying the embedding dimension* D*; on the bottom, varying the number of training iterations* E*. In each panel remaining parameters are fixed to* $D = 128$*,* $\kappa = 5$ *and* $E = 4000$*.*

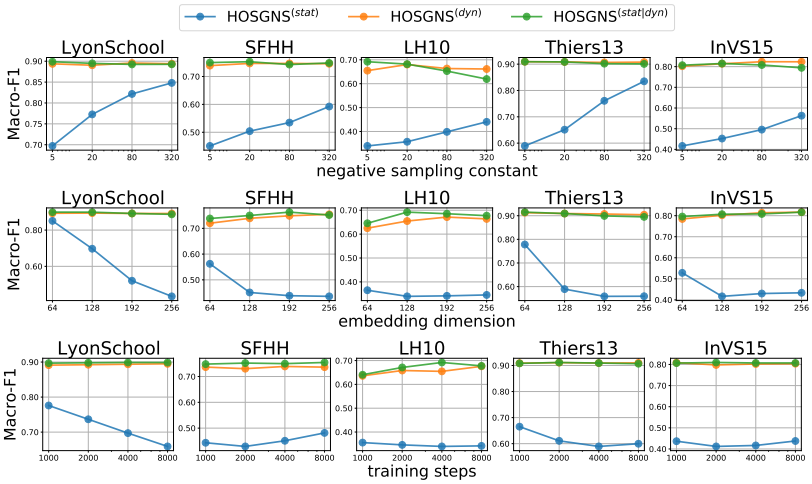


Figure A.4: *Macro-F1 scores related to missing event prediction, computed with HOSGNS embeddings. On the top, classification scores are reported varying the negative sampling parameter κ; in the center, varying the embedding dimension D; on the bottom, varying the number of training iterations E. In each panel remaining parameters are fixed to $D = 128$, $\kappa = 5$ and $E = 4000$.*

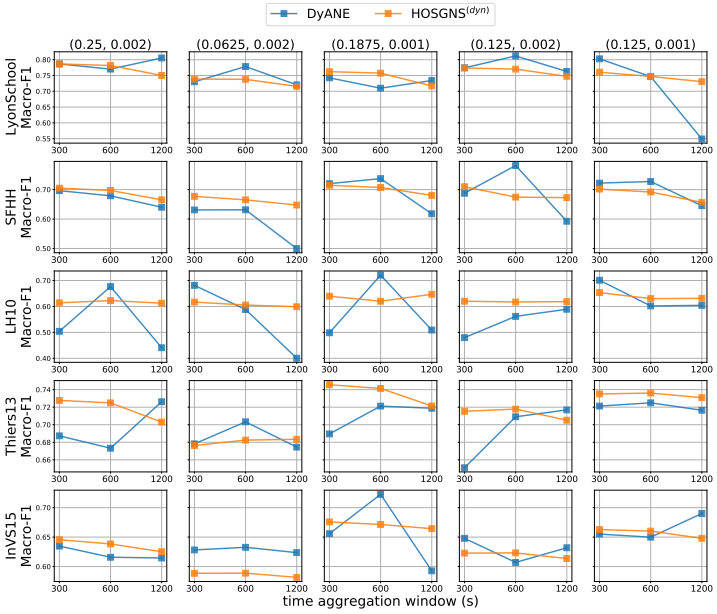


Figure A.5: *Macro-F1 scores related to node classification of epidemic states for different SIR processes, varying the time window used to aggregate empirical datasets, with embedding dimension $D = 128$.*

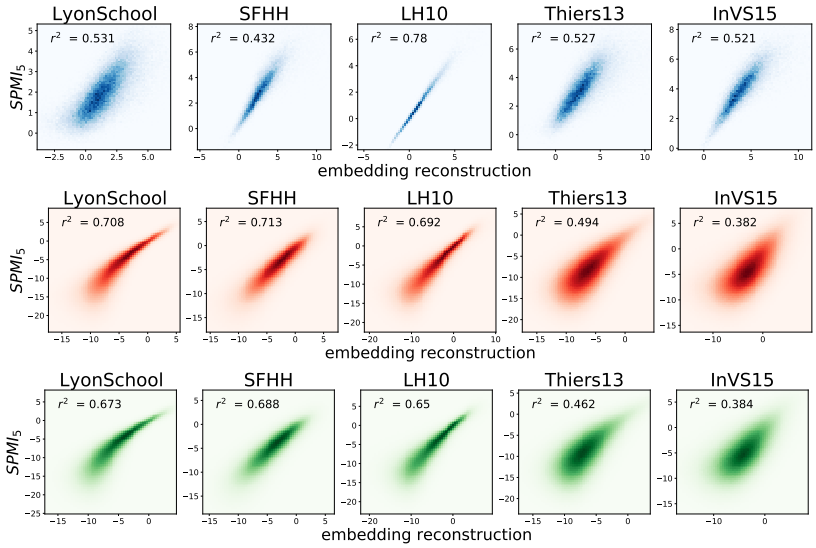


Figure A.6: *Histograms of shifted PMI values* $\mathrm{SPMI}_5(i, j, k \dots)$ *(whereas are greater than* $-\infty$*) versus embedding reconstructed values from higher-order inner products* $[\![\mathbf{v}_i, \mathbf{c}_j, \mathbf{t}_k, ..]\!]$*. From the top to the bottom, HOSGNS model has been trained respectively on* $\mathcal{P}^{(stat)}$*,* $\mathcal{P}^{(dyn)}$ *and* $\mathcal{P}^{(stat|dyn)}$*. The histograms were built by uniformly sampling* 10^7 *entries from the* SPMI_5 *tensors.*

Tensor Decomposition Evaluation

In Figure A.6 we probe the capability of the HOSGNS models to reconstruct the shifted PMI tensor entries in empirical datasets by computing the higher-order product of embedding vectors, operation optimized during the training phase to classify non-zero elements of the tensor itself. We verify the goodness of approximation estimating the square of the Pearson coefficient between the distribution of actual PMI values and the estimated ones, having fixed the model $\kappa = 5$ during training.

Appendix B

Additional Comparison with Walk-based Hypergraph Embeddings

In Figures B.1 and B.2 we compare classification scores respectively for reconstruction and prediction of higher-order links, among SIMPLEX2VEC and Skip-Gram node embeddings generated with 1st-order random walks [ZHS06] on the unweighted hypergraph structure of the input data (we use the same setup for WORD2VEC : $T = 10$, 5 epochs, $walk\_length = 80$, $num\_walks = 10$). Even SIMPLEX2VEC is trained with unweighted walk transitions, leading to a similar 1st-order random walk strategy (but, on a different topological structure). The hypergraph contains hyperedges (formed by at least 2 nodes) that are simplices of $\mathcal{K}_k$, where $k = 2, 3$ is the order of simplices involved in the classification task. Even comparing node-level similarity indices, we notice that SIMPLEX2VEC outperforms hypergraph-based node embeddings in the majority of the datasets, except in the reconstruction of densely connected configurations for co-authorship data.

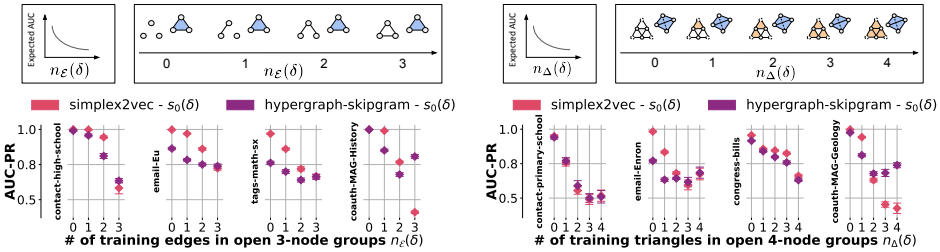


Figure B.1: *Calibrated AUC-PR scores on higher-order link reconstruction for* SIMPLEX2VEC *(trained on $\mathcal{K}_1$) compared with walk-based hypergraph embeddings, with similarity metric s_0. On the left, classification scores are reported varying the parameter $n_{\mathcal{E}}$ for 3-node interactions; on the right, varying the parameter n_{Δ} for 4-node interactions. Metrics are computed in unweighted representations. Label unbalancing in each sample is uniformly drawn between 1:1 and 1:5000.*

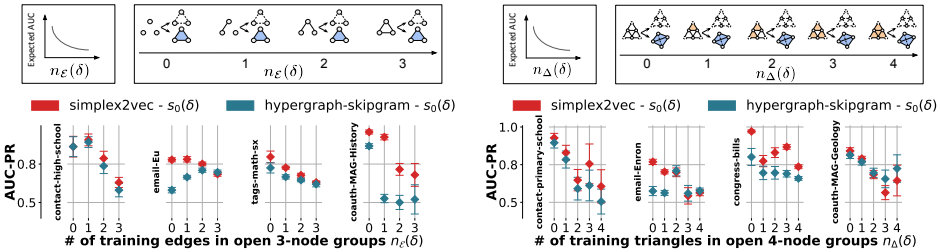


Figure B.2: *Calibrated AUC-PR scores on higher-order link prediction for* SIMPLEX2VEC *(trained on $\mathcal{K}_1$) compared walk-based hypergraph embeddings, with similarity metric s_0. On the left, classification scores are reported varying the parameter $n_{\mathcal{E}}$ for 3-node interactions; on the right, varying the parameter n_{Δ} for 4-node interactions. Metrics are computed in unweighted representations. Label unbalancing in each sample is uniformly drawn between 1:1 and 1:5000.*

Additional Comparison with Hypergraph Neural Networks

In Figures B.3 and B.4 we compare classification scores respectively for reconstruction and prediction of higher-order links, among SIMPLEX2VEC and Hyper-SAGNN [ZZM20] node embeddings on the unweighted hypergraph structure of the input data. Due to the model architecture, we compute hyperedge likelihood scores for Hyper-SAGNN combining embeddings with the same euclidean functional form optimized during model training, as $e_0(\delta) = \frac{1}{|\delta|}\sum_{i\in\delta}|\mathbf{d}_i - \mathbf{s}_i|^2$, where the pair $(\mathbf{s}_i, \mathbf{d}_i)$ corresponds to the (*static*, *dynamic*) embeddings of node i as explained in the paper. In this setup, we notice that SIMPLEX2VEC outperforms Hyper-SAGNN embeddings in the larger part of experiments.

One of the main drawbacks of existing hypergraph-based methods (e.g., [HLS19, ZZM20, BZT21, MSWP22]) is that they are limited to compute 0-simplex representations (node embeddings), making impossible the use of higher-order proximities (computed with interaction embeddings, like edges and triangles) similarly to the ones showed in Figures 4.3 and 4.4 (c)(d).

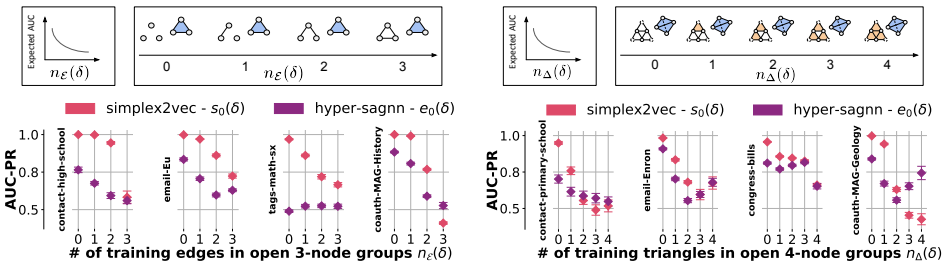


Figure B.3: *Calibrated AUC-PR scores on higher-order link reconstruction for* SIMPLEX2VEC *(trained on $\mathcal{K}_1$) compared with Hyper-SAGNN node embeddings, with similarity metric s_0. On the left, classification scores are reported varying the parameter $n_{\mathcal{E}}$ for 3-node interactions; on the right, varying the parameter n_{Δ} for 4-node interactions. Metrics are computed in unweighted representations. Label unbalancing in each sample is uniformly drawn between 1:1 and 1:5000.*

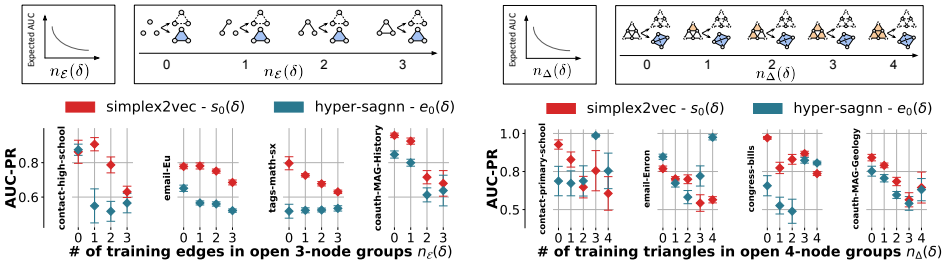


Figure B.4: *Calibrated AUC-PR scores on higher-order link prediction for* SIMPLEX2VEC *(trained on $\mathcal{K}_1$) compared with Hyper-SAGNN node embeddings, with similarity metric s_0. On the left, classification scores are reported varying the parameter $n_{\mathcal{E}}$ for 3-node interactions; on the right, varying the parameter n_{Δ} for 4-node interactions. Metrics are computed in unweighted representations. Label unbalancing in each sample is uniformly drawn between 1:1 and 1:5000.*

Appendix C

Per-dimension Utility Function Theorem

Theorem. *Let* $\mathrm{enc} : \mathcal{V} \to \mathbb{R}^D$ *be any embedding encoder function which returns a feature vector* $\mathbf{u} = \mathrm{enc}(u)$ *for any node* $u \in \mathcal{V}$ *of the graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*. The per-dimension marginal utility score for edge* $(u, v) \in \mathcal{E}$ *can be expressed as:*

$$\mu_d(\mathbf{u}, \mathbf{v}) = \left[\mathrm{u}_d \mathrm{v}_d - \frac{1}{D}\mathbf{u} \cdot \mathbf{v}\right] \left(\frac{1}{D} + \frac{1}{D^2} + \dots\right)$$

Proof. Referring to $\Omega = \{1, \dots, D\}$ as the set containing the enumerated dimensions, from Equation (5.2) we start writing the explicit formula for the definition of $\mu_d(\mathbf{u}, \mathbf{v})$:

$$\mu_d(\mathbf{u}, \mathbf{v}) = \frac{1}{D} \sum_{q \in \Omega} \mathrm{u}_q \mathrm{v}_q - \frac{1}{D-1} \sum_{q \in \Omega \setminus \{d\}} \mathrm{u}_q \mathrm{v}_q$$

Using the expression $\sum_{q \in \Omega} \mathrm{u}_q \mathrm{v}_q = \mathbf{u} \cdot \mathbf{v} = \mathrm{u}_d \mathrm{v}_d + \sum_{q \in \Omega \setminus \{d\}} \mathrm{u}_q \mathrm{v}_q$, we find that:

$$\begin{aligned}
\mu_d(\mathbf{u}, \mathbf{v}) &= \frac{1}{D} \mathrm{u}_d \mathrm{v}_d - \left(\frac{1}{D-1} - \frac{1}{D}\right) \sum_{q \in \Omega \setminus \{d\}} \mathrm{u}_q \mathrm{v}_q \\
&= \frac{1}{D} \mathrm{u}_d \mathrm{v}_d - \frac{1}{D} \left(\frac{1}{1 - \frac{1}{D}} - 1\right) (\mathbf{u} \cdot \mathbf{v} - \mathrm{u}_d \mathrm{v}_d)
\end{aligned}$$

Ignoring the case of 1-dimensional embeddings, $D > 1$ and $0 < \frac{1}{D} < 1$, then the expression within the parenthesis can be rewritten using the geometric series:

$$\frac{1}{1 - \frac{1}{D}} - 1 = \sum_{k=0}^{\infty} \frac{1}{D^k} - 1 = \frac{1}{D} + \frac{1}{D^2} + \dots \tag{C.1}$$

Replacing the formula in parenthesis with Equation (C.1), we prove the theorem:

$$
\begin{aligned}
D \cdot \mu_d(\mathbf{u}, \mathbf{v}) &= \mathrm{u}_d \mathrm{v}_d - \left(\frac{1}{D} + \frac{1}{D^2} + \dots \right) (\mathbf{u} \cdot \mathbf{v} - \mathrm{u}_d \mathrm{v}_d) \\
&= \mathrm{u}_d \mathrm{v}_d \left(1 + \frac{1}{D} + \frac{1}{D^2} + \dots \right) - \mathbf{u} \cdot \mathbf{v} \left(\frac{1}{D} + \frac{1}{D^2} + \dots \right) \\
&= \left[\mathrm{u}_d \mathrm{v}_d - \frac{1}{D} \mathbf{u} \cdot \mathbf{v} \right] \left(1 + \frac{1}{D} + \frac{1}{D^2} + \dots \right)
\end{aligned}
$$

□

Additional Experimental Results on Empirical Datasets

In Figures C.1, C.2 and C.3 we report the complete set of experimental results for interpretation metrics and link prediction performance on multiple real-world datasets, when varying the number of embedding dimensions.

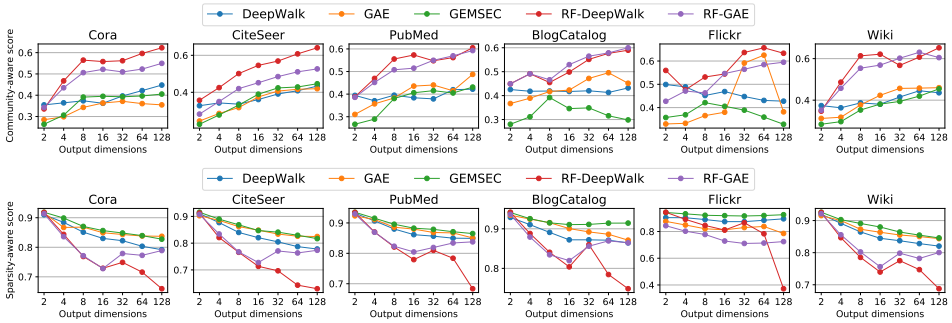


Figure C.1: *Interpretation scores compared among RF-*DEEPWALK*, RF-GAE and different dense embedding methods trained on real-world datasets, when varying the number of output dimensions and choosing the best score among models with a different number of input dimensions. On the top, we compare the community-aware scores (higher is better); on the bottom, we compare the sparsity-aware scores (lower is better).*

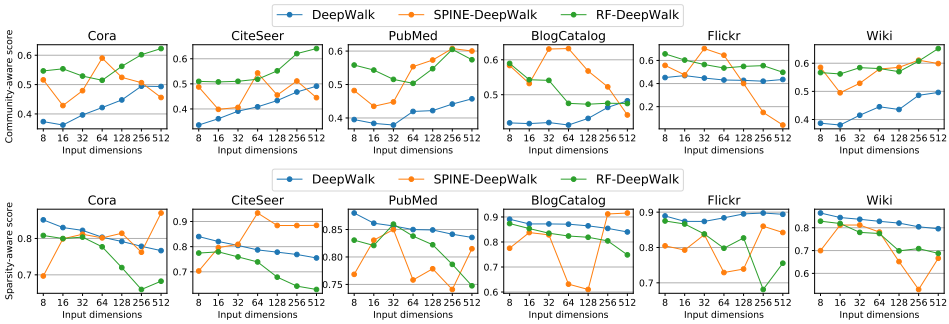


Figure C.2: *Interpretation scores compared among* DEEPWALK, *SPINE-*DEEPWALK *and RF-*DEEPWALK *methods trained on real-world datasets, when varying the number of input dimensions and choosing the best score among models with a different number of output dimensions. On the top, we compare the community-aware scores (higher is better); on the bottom, we compare the sparsity-aware scores (lower is better).*

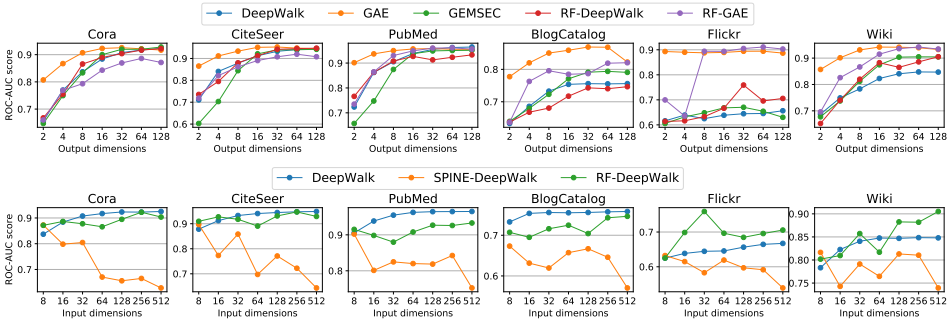


Figure C.3: *ROC-AUC scores on link prediction for different embedding methods trained on real-world datasets. On the top, we compare RF-*DEEPWALK, *RF-GAE and different dense embedding methods when varying the number of output dimensions and choosing the best score among models with a different number of input dimensions; on the bottom, we compare* DEEPWALK, *SPINE-*DEEPWALK *and RF-*DEEPWALK *methods when varying the number of input dimensions and choosing the best score among models with a different number of output dimensions.*

Bibliography

[AA03] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[Abb17] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

[ABH19] Carl Allen, Ivana Balazevic, and Timothy Hospedales. What the vec? towards probabilistically grounded embeddings. In *Advances in Neural Information Processing Systems*, pages 7465–7475, 2019.

[ALL+16] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

[ASK+21] Sarwan Ali, Muhammad Haroon Shakeel, Imdadullah Khan, Safiullah Faizullah, and Muhammad Asad Khan. Predicting attributes of nodes using network structure. *ACM Transactions on Intelligent Systems and Technology*, 12(2):1–23, 2021.

[BAB+21] Federico Battiston, Enrico Amico, Alain Barrat, Ginestra Bianconi, Guilherme Ferraz de Arruda, Benedetta Franceschiello, Iacopo Iacopini, Sonia Kéfi, Vito Latora, Yamir Moreno, et al. The physics of higher-order interactions in complex systems. *Nature Physics*, 17(10):1093–1098, 2021.

[Bal12] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[BAMK16] Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Kenichi Kawarabayashi. Joint word representation learning using a corpus and a semantic lexicon. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[Bar12] Albert-László Barabási. The network takeover. *Nature Physics*, 8(1):14–16, 2012.

[BAS+18] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

[BBB+21] Antoine Baker, Indaco Biazzo, Alfredo Braunstein, Giovanni Catania, Luca Dall'Asta, Alessandro Ingrosso, Florent Krzakala, Fabio Mazza, Marc Mézard, Anna Paola Muntoni, et al. Epidemic mitigation by statistical inference from contact tracing data. *Proceedings of the National Academy of Sciences*, 118(32):e2106548118, 2021.

[BBM14] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. Who to follow and why: link prediction with explanations. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1266–1275, 2014.

[BBV08] Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes on complex networks*. Cambridge university press, 2008.

[BCC+13] Alain Barrat, Ciro Cattuto, Vittoria Colizza, Francesco Gesualdo, Lorenzo Isella, Elisabetta Pandolfi, J F Pinton, Lucilla Ravà, Caterina Rizzo, Mariateresa Romano, et al. Empirical temporal networks of face-to-face human interactions. *The European Physical Journal Special Topics*, 222:1295–1309, 2013.

[BCI+20] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and dynamics. *Physics Reports*, 874:1–92, 2020.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[BFW+21] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pages 1026–1037. PMLR, 2021.

[BGA20] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pages 874–883. PMLR, 2020.

[BGL16] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[BHL+19] Jacob Charles Wright Billings, Mirko Hu, Giulia Lerda, Alexey N Medvedev, Francesco Mottes, Adrian Onicas, Andrea Santoro, and Giovanni Petri. Simplex2vec embeddings for community detection in simplicial complexes. *arXiv preprint arXiv:1906.09068*, 2019.

[BJN+02] Albert-Laszlo Barabâsi, Hawoong Jeong, Zoltan Néda, Erzsebet Ravasz, Andras Schubert, and Tamas Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3-4):590–614, 2002.

[BKB+19] Stephen Bonner, Ibad Kureshi, John Brennan, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. Exploring the semantic content of unsupervised graph embeddings: An empirical study. *Data Science and Engineering*, 4:269–289, 2019.

[BKPB19] Ferenc Béres, Domokos M Kelen, Róbert Pálovics, and András A Benczúr. Node embeddings in dynamic graphs. *Applied Network Science*, 4(1):64, 2019.

[BKTK19] Caleb Belth, Fahad Kamran, Donna Tjandra, and Danai Koutra. When to remember where you came from: Node representation learning in higher-order networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 222–225, 2019.

[BMD+20] Ayan Kumar Bhowmick, Koushik Meneni, Maximilien Danisch, Jean-Loup Guillaume, and Bivas Mitra. Louvainne: Hierarchical louvain method for high quality and scalable network embedding. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 43–51, 2020.

[BMVZ21] Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani. A survey on embedding dynamic graphs. *ACM Computing Surveys*, 55(1):1–37, 2021.

[BN03] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[BZT21] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[CFQS12] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[CH90] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[CKC+21] Sandeep Chowdhary, Aanjaneya Kumar, Giulia Cencetti, Iacopo Iacopini, and Federico Battiston. Simplicial contagion in temporal higher-order networks. *Journal of Physics: Complexity*, 2(3):035019, 2021.

[CLX15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 891–900, 2015.

[CLX19] Ren-Meng Cao, Si-Yuan Liu, and Xiao-Ke Xu. Network embedding for link prediction: The pitfall and improvement. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(10):103102, 2019.

[CM20] Sudhanshu Chanpuriya and Cameron Musco. Infinitewalk: Deep network embeddings as laplacian embeddings with a nonlinearity. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1325–1333, 2020.

[CM21] Yu-Chia Chen and Marina Meila. The decomposition of the higher-order homology embedding constructed from the k-laplacian. *Advances in Neural Information Processing Systems*, 34, 2021.

[CMS21] Giulio Cimini, Rossana Mastrandrea, and Tiziano Squartini. *Reconstructing networks*. Cambridge University Press, 2021.

[CMST20] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. Node embeddings and exact low-rank representations of complex networks. *Advances in Neural Information Processing Systems*, 33:13185–13198, 2020.

[CMST21] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos Tsourakakis. Deepwalking backwards: From embeddings back to graphs. In *International Conference on Machine Learning*, pages 1473–1483. PMLR, 2021.

[CP20] Neeraj Chavan and Katerina Potika. Higher-order link prediction using triangle embeddings. In *IEEE International Conference on Big Data*, pages 4535–4544. IEEE, 2020.

[CPVDE17] Ryan Cotterell, Adam Poliak, Benjamin Van Durme, and Jason Eisner. Explaining and generalizing skip-gram through exponential family principal component analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 175–181, 2017.

[CRL+22] Estee Y Cramer, Evan L Ray, Velma K Lopez, Johannes Bracher, Andrea Brennen, Alvaro J Castro Rivadeneira, Aaron Gerding, Tilmann Gneiting, Katie H House, Yuxin Huang, et al. Evaluation of individual and ensemble probabilistic forecasts of covid-19 mortality in the united states. *Proceedings of the National Academy of Sciences*, 119(15):e2113561119, 2022.

[CS22] Hyunjin Choo and Kijung Shin. On the persistence of higher-order interactions in real-world hypergraphs. In *Proceedings of the SIAM International Conference on Data Mining*, pages 163–171. SIAM, 2022.

[CSL+21] Giulia Cencetti, Gabriele Santin, Antonio Longa, Emanuele Pigani, Alain Barrat, Ciro Cattuto, Sune Lehmann, Marcel Salathe, and Bruno Lepri. Digital proximity tracing on empirical contact networks for pandemic control. *Nature Communications*, 12(1):1655, 2021.

[CVdBB+10] Ciro Cattuto, Wouter Van den Broeck, Alain Barrat, Vittoria Colizza, Jean-François Pinton, and Alessandro Vespignani. Dynamics of person-to-person interactions from distributed rfid sensor networks. *PloS One*, 5(7), 2010.

[CZ14] Jan Chorowski and Jacek M Zurada. Learning understandable neural networks with nonnegative weight constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 26(1):62–69, 2014.

[CZ17] Yu Chen and Mohammed J Zaki. Kate: K-competitive autoencoder for text. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 85–94, 2017.

[CZC18] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[DDSRC+13] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[DG18] Ayushi Dalmia and Manish Gupta. Towards interpretation of node embeddings. In *Companion Proceedings of the The World Wide Web Conference*, pages 945–952, 2018.

[DGE15] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[DKA11] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data*, 5(2):1–27, 2011.

[DLH19] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[DM20] Aswathy Divakaran and Anuraj Mohan. Temporal link prediction: A survey. *New Generation Computing*, 38(1):213–258, 2020.

[DNA19] Chi Thang Duong, Quoc Viet Hung Nguyen, and Karl Aberer. Interpretable node embeddings with mincut loss. In *Learning and Reasoning with Graph-Structured Representations Workshop-ICML*, 2019.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[DWS+18] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. Dynamic network embedding: An extended approach for skip-gram based network embedding. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2086–2092, 2018.

[EBT16] Jesper E van Engelen, Hanjo D Boekhout, and Frank W Takes. Explainable and efficient link prediction in real-world network data. In *International Symposium on Intelligent Data Analysis*, pages 295–307. Springer, 2016.

[EDS20] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial neural networks. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

[EMS04] Jean-Pierre Eckmann, Elisha Moses, and Danilo Sergi. Entropy of dialogues creates coherent structures in e-mail traffic. *Proceedings of the National Academy of Sciences*, 101(40):14333–14337, 2004.

[FC17] Benjamin Fish and Rajmonda S Caceres. A task-driven approach to time scale detection in dynamic networks. In *Proceedings of the 13th international workshop on mining and learning with graphs (MLG)*, 2017.

[FDJ+15] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, 2015.

[FDR22] Cornelius Fritz, Emilio Dorigatti, and David Rügamer. Combining graph neural networks and spatio-temporal disease models to improve the prediction of weekly covid-19 cases in germany. *Scientific Reports*, 12(1):3930, 2022.

[FKRA22] Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. Z orro: Valid, sparse, and stable explanations in graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[FTY+15] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, 2015.

[FYZ+19] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019.

[GB18] Mathieu Génois and Alain Barrat. Can co-location be used as a proxy for face-to-face contacts? *EPJ Data Science*, 7(1):11, 2018.

[GBB+18] Edoardo Galimberti, Alain Barrat, Francesco Bonchi, Ciro Cattuto, and Francesco Gullo. Mining (maximal) span-cores from temporal networks. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 107–116, 2018.

[GBH19] Antonia Gogoglou, C. Bayan Bruss, and Keegan E. Hines. On the Interpretability and Evaluation of Graph Representation Learning. *NeurIPS Workshop on Graph Representation Learning*, 2019.

[GBL22] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio. Simplicial attention networks. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[GBSRW20] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. Prince: Provider-side interpretability with counterfactual explanations in recommender systems. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 196–204, 2020.

[GCC20] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

[GF18] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[GHG+20] Amir Ghasemian, Homa Hosseinmardi, Aram Galstyan, Edoardo M Airoldi, and Aaron Clauset. Stacking models for nearly optimal link prediction in complex networks. *Proceedings of the National Academy of Sciences*, 117(38):23393–23400, 2020.

[GKHL17] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. In *IJCAI Workshop on Representation Learning for Graphs (ReLiG)*, 2017.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 855–864, 2016.

[GMR+18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42, 2018.

[GN02] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[Gol02] Timothy E Goldberg. Combinatorial laplacians of simplicial complexes. *Senior Thesis, Bard College*, 2002.

[GPAGS20] Guillermo García-Pérez, Roya Aliakbarisani, Abdorasoul Ghasemi, and M Ángeles Serrano. Precision as a measure of predictability of missing links in real networks. *Physical Review E*, 101(5):052318, 2020.

[GPC14] Laetitia Gauvin, André Panisson, and Ciro Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS One*, 9(1):e86028, 2014.

[GSG+22] Deniz Gurevin, Mohsin Shan, Tong Geng, Weiwen Jiang, Caiwen Ding, and Omer Khan. Towards real-time temporal graph learning. In *IEEE International Conference on Computer Design*, pages 263–271. IEEE, 2022.

[GSP09] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[GSQ+21] Junyi Gao, Rakshith Sharma, Cheng Qian, Lucas M Glass, Jeffrey Spaeder, Justin Romberg, Jimeng Sun, and Cao Xiao. Stan: spatio-temporal attention network for pandemic prediction using real-world evidence. *Journal of the American Medical Informatics Association*, 28(4):733–743, 2021.

[GTP+20] Laetitia Gauvin, Michele Tizzoni, Simone Piaggesi, Andrew Young, Natalia Adler, Stefaan Verhulst, Leo Ferres, and Ciro Cattuto. Gender gaps in urban mobility. *Humanities and Social Sciences Communications*, 7(1):1–13, 2020.

[GVF+15] Mathieu Génois, Christian L Vestergaard, Julie Fournet, André Panisson, Isabelle Bonmarin, and Alain Barrat. Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. *Network Science*, 3(3):326–347, 2015.

[GWS+20] Lingbing Guo, Weiqing Wang, Zequn Sun, Chenghao Liu, and Wei Hu. Decentralized knowledge graph representation learning. *arXiv preprint arXiv:2010.08114*, 2020.

[Hac20] Celia Hacker. k-simplex2vec: a simplicial extension of node2vec. In *NeurIPS Workshop on Topological Data Analysis and Beyond*, 2020.

[Ham20] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[Har54] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[HCY+19] Jie Huang, Chuan Chen, Fanghua Ye, Jiajing Wu, Zibin Zheng, and Guohui Ling. Hyper2vec: Biased random walk for hyper-network embedding. In *Database Systems for Advanced Applications: DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22–25, 2019, Proceedings 24*, pages 273–277. Springer, 2019.

[HFZ+20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

[HGC20] Patrick Hall, Navdeep Gill, and Benjamin Cox. *Responsible Machine Learning*. O'Reilly Media, Incorporated, 2020.

[HLS19] Jie Huang, Xin Liu, and Yangqiu Song. Hyper-path-based representation learning for hyper-networks. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 449–458, 2019.

[HPN+21] Robert Hinch, William JM Probert, Anel Nurtay, Michelle Kendall, Chris Wymant, Matthew Hall, Katrina Lythgoe, Ana Bulas Cruz, Lele Zhao, Andrea Stewart, et al. Openabm-covid19—an agent-based model for non-pharmaceutical interventions against covid-19 including contact tracing. *PLoS computational biology*, 17(7):e1009146, 2021.

[HS12] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[HYL17a] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

[HYL17b] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.

[IKA20] Maximilian Idahl, Megha Khosla, and Avishek Anand. Finding interpretable concept spaces in node embeddings using knowledge bases. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 229–240. Springer, 2020.

[IPBB22] Iacopo Iacopini, Giovanni Petri, Andrea Baronchelli, and Alain Barrat. Group interactions modulate critical mass dynamics in social convention. *Communications Physics*, 5(1):64, 2022.

[IPBL19] Iacopo Iacopini, Giovanni Petri, Alain Barrat, and Vito Latora. Simplicial models of social contagion. *Nature Communications*, 10(1):2485, 2019.

[ISB+11] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. What's in a crowd? analysis of face-to-face behavioral networks. *Journal of theoretical biology*, 271(1):166–180, 2011.

[JDH15] Sujay Kumar Jauhar, Chris Dyer, and Eduard H Hovy. Ontologically grounded multi-sense representation learning for semantic vector space models. In *HLT-NAACL*, pages 683–693, 2015.

[JPC+21] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.

[Kat53] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[KB15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[KBL+20] Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. Examining covid-19 forecasting using spatio-temporal gnns. In *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*, 2020.

[KDPR20] Tarun Kumar, K Darwin, Srinivasan Parthasarathy, and Balaraman Ravindran. Hpra: Hyperedge prediction using resource allocation. In *ACM Conference on Web Science*, pages 135–143, 2020.

[KGJ+20] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[Kit14] Rob Kitchin. Big data, new epistemologies and paradigm shifts. *Big data & society*, 1(1):2053951714528481, 2014.

[KKB+12] Gautier Krings, Márton Karsai, Sebastian Bernhardsson, Vincent D Blondel, and Jari Saramäki. Effects of time window size and placement on the structure of an aggregated communication network. *EPJ Data Science*, 1(1):1–16, 2012.

[KKS20] Yunbum Kook, Jihoon Ko, and Kijung Shin. Evolution of real-world hypergraphs: Patterns and models without oracles. In *IEEE International Conference on Data Mining*, pages 272–281. IEEE, 2020.

[KLDB22] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Explanations for network embedding-based link predictions. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021, Virtual Event, September 13-17, 2021, Proceedings, Part I*, pages 473–488. Springer, 2022.

[KMA21] Shima Khoshraftar, Sedigheh Mahdavi, and Aijun An. Centrality-based interpretability measures for graph embeddings. In *IEEE International Conference on Data Science and Advanced Analytics*, pages 1–10. IEEE, 2021.

[KR08] Matt J. Keeling and Pejman Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, 2008.

[KSSB20] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

[KZL19] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019.

[LCF15] Yannick Léo, Christophe Crespelle, and Eric Fleury. Non-altering time scales for aggregation of dynamic networks into series of graphs. In *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies*, pages 1–7, 2015.

[LCX+20] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 33:19620–19631, 2020.

[LG14a] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 171–180, 2014.

[LG14b] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[LGP21] Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. Mile: A multi-level framework for scalable graph embedding. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 15, pages 361–372, 2021.

[LH08] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the World Wide Web Conference*, pages 915–924, 2008.

[LHLH18] Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. On interpretation of network embedding via taxonomy induction. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1812–1820, 2018.

[LJZ09] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Physical Review E*, 80(4):046122, 2009.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[LML22] Yunyu Liu, Jianzhu Ma, and Pan Li. Neural predicting higher-order patterns in temporal networks. In *Proceedings of the World Wide Web Conference*, pages 1340–1351, 2022.

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[LOU20] Timothée Lacroix, Guillaume Obozinski, and Nicolas Usunier. Tensor decompositions for temporal knowledge base completion. In *International Conference on Learning Representations*, 2020.

[LPA+09] David Lazer, Alex Pentland, Lada Adamic, Sinan Aral, Albert-László Barabási, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. Computational social science. *Science*, 323(5915):721–723, 2009.

[LPZ+15] Linyuan Lü, Liming Pan, Tao Zhou, Yi-Cheng Zhang, and H Eugene Stanley. Toward link predictability of complex networks. *Proceedings of the National Academy of Sciences*, 112(8):2325–2330, 2015.

[LQH15] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015.

[LRS19] Renaud Lambiotte, Martin Rosvall, and Ingo Scholtes. From networks to optimal higher-order models of complex systems. *Nature Physics*, 15(4):313–320, 2019.

[LTL+19] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. Is a single vector enough? exploring node polysemy for network embedding. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 932–940, 2019.

[LYZ+23] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. Scaling up dynamic graph representation learning via spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[LZ11] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[LZP+18] Taisong Li, Jiawei Zhang, S Yu Philip, Yan Zhang, and Yonghong Yan. Deep dynamic network embedding for link prediction. *IEEE Access*, 6:29219–29230, 2018.

[MBM21] Federico Musciotto, Federico Battiston, and Rosario N Mantegna. Detecting informative higher-order interactions in statistically validated hypergraphs. *Communications Physics*, 4(1):1–9, 2021.

[MCCD13] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations, Workshop Track Proceedings*, 2013.

[MG17] Oren Melamud and Jacob Goldberger. Information-theory interpretation of the skip-gram negative-sampling objective function. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 167–171, 2017.

[MHM18] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[MK13] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273, 2013.

[MKA18] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *IEEE International Conference on Big Data*, pages 3762–3765. IEEE, 2018.

[MLA21] Charles Murphy, Edward Laurence, and Antoine Allard. Deep learning of contagion dynamics on complex networks. *Nature Communications*, 12(1):4720, 2021.

[MLB+22] Abdelrahman Mohamed, Hung-yi Lee, Lasse Borgholt, Jakob D Havtorn, Joakim Edin, Christian Igel, Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaløe, et al. Self-supervised speech representation learning: A review. *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[MLDB20] Alexandru Cristian Mara, Jefrey Lijffijt, and Tijl De Bie. Benchmarking network embedding models for link prediction: are we making progress? In *IEEE International Conference on Data Science and Advanced Analytics*, pages 138–147. IEEE, 2020.

[MM20] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[MMS+21] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, 2021.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[MSWP22] Sepideh Maleki, Donya Saless, Dennis P Wall, and Keshav Pingali. Hypernetvec: Fast and scalable hierarchical embedding for hypergraphs. In *International Conference on Network Science*, pages 169–183. Springer, 2022.

[MTD19] Yunpu Ma, Volker Tresp, and Erik A Daxberger. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, 59:100490, 2019.

[MUH+21] Osman Asif Malik, Shashanka Ubaru, Lior Horesh, Misha E Kilmer, and Haim Avron. Dynamic graph convolutional networks using the tensor m-product. In *Proceedings of the SIAM International Conference on Data Mining*, pages 729–737. SIAM, 2021.

[New01] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2):025102, 2001.

[New03] Mark EJ Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[New18] Mark EJ Newman. Network structure from rich but noisy data. *Nature Physics*, 14(6):542–545, 2018.

[NLR+18] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The World Wide Web Conference*, pages 969–976, 2018.

[NMCC16] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *Companion Proceedings of the World Wide Web Conference*, pages 83–84, 2016.

[NTLL19] Qi Ni, Ming Tang, Ying Liu, and Ying-Cheng Lai. Machine learning dynamical phase transitions in complex networks. *Physical Review E*, 100(5):052312, 2019.

[OCP+16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1105–1114, 2016.

[OSH+07] Jukka-Pekka Onnela, Jari Saramäki, Jörkki Hyvönen, Gábor Szabó, M Argollo De Menezes, Kimmo Kaski, Albert-László Barabási, and János Kertész. Analysis of a large-scale weighted network of one-to-one human communication. *New journal of physics*, 9(6):179, 2007.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 701–710. ACM, 2014.

[PBO17] Sungjoon Park, JinYeong Bak, and Alice Oh. Rotated Word Vector Representations and their Interpretability. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 401–411, Copenhagen, Denmark, 2017. Association for Computational Linguistics.

[PDCM22] Thibault Prouteau, Nicolas Dugué, Nathalie Camelin, and Sylvain Meignier. Are embedding spaces interpretable? results of an intrusion detection evaluation on a large french corpus. In *LREC 2022*, 2022.

[Pei19] Tiago P Peixoto. Network reconstruction and community detection from dynamics. *Physical Review Letters*, 123(12):128301, 2019.

[PGBC13] André Panisson, Laetitia Gauvin, Alain Barrat, and Ciro Cattuto. Fingerprinting temporal networks of close-range human proximity. In *IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 261–266. IEEE, 2013.

[PKCS17] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 258–265, 2017.

[PKFD20] Namyong Park, Andrey Kan, Christos Faloutsos, and Xin Luna Dong. J-recs: Principled and scalable recommendation justification. In *IEEE International Conference on Data Mining*, pages 1208–1213. IEEE, 2020.

[PLC17] Leto Peel, Daniel B Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science advances*, 3(5):e1602548, 2017.

[PLY+20] Hao Peng, Jianxin Li, Hao Yan, Qiran Gong, Senzhang Wang, Lin Liu, Lihong Wang, and Xiang Ren. Dynamic network embedding via incremental skip-gram with negative sampling. *Science China Information Sciences*, 63(10):1–19, 2020.

[PMH18] Nikolaos Pappas, Lesly Miculicich, and James Henderson. Beyond weight tying: Learning joint input-output embeddings for neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, 2018.

[PP22] Simone Piaggesi and André Panisson. Time-varying graph representation learning via higher-order skip-gram with negative sampling. *EPJ Data Science*, 11(1):33, 2022.

[PPP22] Simone Piaggesi, André Panisson, and Giovanni Petri. Effective higher-order link prediction and reconstruction from simplicial complex embeddings. In *Learning on Graphs Conference*, pages 55–1. PMLR, 2022.

[PPV17] Alice Patania, Giovanni Petri, and Francesco Vaccarino. The shape of collaborations. *EPJ Data Science*, 6:1–16, 2017.

[PSB19] Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. Word2sense: Sparse interpretable word embeddings. In *Proceedings of the 57th annual meeting of the Association for Computational Linguistics*, pages 5692–5705, 2019.

[PSCVMV15] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925, 2015.

[PSG+13] René Pfitzner, Ingo Scholtes, Antonios Garas, Claudio J Tessone, and Frank Schweitzer. Betweenness preference: Quantifying correlations in the topological dynamics of temporal networks. *Physical Review Letters*, 110(19):198701, 2013.

[PSHM23] Mathilde Papillon, Sophia Sanborn, Mustafa Hajij, and Nina Miolane. Architectures of topological deep learning: A survey on topological neural networks. *arXiv preprint arXiv:2304.10031*, 2023.

[PSL+21] Liming Pan, Hui-Juan Shang, Peiyan Li, Haixing Dai, Wei Wang, and Lixin Tian. Predicting hyperlinks via hypernetwork loop structure. *Europhysics Letters*, 135(4):48005, 2021.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[PSM20] Prasanna Patil, Govind Sharma, and M. Narasimha Murty. Negative Sampling for Hyperlink Prediction in Networks. In Hady W. Lauw, Raymond Chi-Wing Wong, Alexandros Ntoulas, Ee-Peng Lim, See-Kiong Ng, and Sinno Jialin Pan, editors, *Advances in Knowledge Discovery and Data Mining*, pages 607–619. Springer International Publishing, 2020.

[PSV02] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical Review E*, 65(3):036104, 2002.

[QDM+18] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 459–467. ACM, 2018.

[RB18] Maja Rudolph and David Blei. Dynamic embeddings for language evolution. In *Proceedings of the World Wide Web Conference*, pages 1003–1011, 2018.

[RDSS19] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 65–72, 2019.

[RFMT22] Andrea Rossi, Donatella Firmani, Paolo Merialdo, and Tommaso Teofili. Explaining link prediction systems based on knowledge graph embeddings. In *Proceedings of the 2022 International Conference on Management of Data*, pages 2062–2075, 2022.

[RPB13] Bruno Ribeiro, Nicola Perra, and Andrea Baronchelli. Quantifying the effect of temporal resolution on time-varying networks. *Scientific Reports*, 3(1):1–5, 2013.

[RPC+19] Francisco A Rodrigues, Thomas Peron, Colm Connaughton, Jurgen Kurths, and Yamir Moreno. A machine learning approach to predicting dynamical observables from network structure. *arXiv preprint arXiv:1910.00544*, 2019.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1135–1144, 2016.

[RZHW22] Omar F Robledo, Xiu-Xiu Zhan, Alan Hanjalic, and Huijuan Wang. Influence of clustering coefficient on network embedding in link prediction. *Applied Network Science*, 7(1):1–20, 2022.

[SBB+12] Marcel Salathe, Linus Bengtsson, Todd J Bodnar, Devon D Brewer, John S Brownstein, Caroline Buckee, Ellsworth M Campbell, Ciro Cattuto, Shashank Khandelwal, Patricia L Mabry, et al. Digital epidemiology. *PLoS Computational Biology*, 8(7), 2012.

[SBBPS12] Michele Starnini, Andrea Baronchelli, Alain Barrat, and Romualdo Pastor-Satorras. Random walks on temporal networks. *Physical Review E*, 85(5):056115, 2012.

[SBCG18] Anna Sapienza, Alain Barrat, Ciro Cattuto, and Laetitia Gauvin. Estimating the outcome of spreading processes on networks with incomplete information: A dimensionality reduction approach. *Physical Review E*, 98(1):012317, 2018.

[SBPA23] Andrea Santoro, Federico Battiston, Giovanni Petri, and Enrico Amico. Higher-order organization of multivariate time series. *Nature Physics*, pages 1–9, 2023.

[SBWG10] Rajmonda Sulo, Tanya Berger-Wolf, and Robert Grossman. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 127–136, 2010.

[SCKC20] Mandana Saebi, Giovanni Luca Ciampaglia, Lance M Kaplan, and Nitesh V Chawla. Honem: learning embedding for higher order networks. *Big Data*, 8(4):255–269, 2020.

[SCL18] Vsevolod Salnikov, Daniele Cassese, and Renaud Lambiotte. Simplicial complexes and complex systems. *European Journal of Physics*, 40(1):014001, 2018.

[SFHG+20] Wissam Siblini, Jordan Fréry, Liyun He-Guelton, Frédéric Oblé, and Yi-Qing Wang. Master your metrics with calibration. In *International Symposium on Intelligent Data Analysis*, pages 457–469. Springer, 2020.

[SFX+20] Jiachen Sun, Ling Feng, Jiarong Xie, Xiao Ma, Dashun Wang, and Yanqing Hu. Revealing the predictability of intrinsic structure in complex networks. *Nature Communications*, 11(1):1–10, 2020.

[SGL+16] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. Sparse word embeddings using l1 regularized online learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2915–2921, 2016.

[SGT+08] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[Sha16] Lloyd S Shapley. 17. a value for n-person games. In *Contributions to the Theory of Games (AM-28), Volume II*, pages 307–318. Princeton University Press, 2016.

[Sha20] Govind Sharma. *Hypergraph Network Models: Learning, Prediction, and Representation in the Presence of Higher-Order Relations*. PhD thesis, Indian Institute of Science Bangalore, 2020.

[SKL+10] Marcel Salathé, Maria Kazandjieva, Jung Woo Lee, Philip Levis, Marcus W Feldman, and James H Jones. A high-resolution human contact network for infectious disease transmission. *Proceedings of the National Academy of Sciences*, 107(51):22020–22025, 2010.

[SM19] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 5–22. Springer, 2019.

[SMF18] Jamin Shin, Andrea Madotto, and Pascale Fung. Interpreting word embeddings with eigenvector analysis. In *NeurIPS Workshop on Interpretability and Robustness in Audio, Speech, and Language*, 2018.

[SOBC21] Koya Sato, Mizuki Oka, Alain Barrat, and Ciro Cattuto. Predicting partially observed processes on temporal networks by dynamics-aware node embeddings (dyane). *EPJ Data Science*, 10(1):22, 2021.

[SPJ+18] Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard Hovy. Spine: Sparse interpretable neural embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[SPM20] Govind Sharma, Prasanna Patil, and M. Narasimha Murty. C3MM: Clique-Closure based Hyperlink Prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3364–3370, 2020.

[SPW+15] Anna Sapienza, André Panisson, Joseph Wu, Laetitia Gauvin, and Ciro Cattuto. Detecting anomalies in time-varying networks using tensor decomposition. In *IEEE International Conference on Data Mining Workshop*, pages 516–523. IEEE, 2015.

[ŞUŞ+20] Lütfi Kerem Şenel, Ihsan Utlu, Furkan Şahinuç, Haldun M. Ozaktas, and Aykut Koç. Imparting interpretability to word embeddings while preserving semantic structure. *Natural Language Engineering*, page 1–26, 2020.

[ŞUY+18] Lütfi Kerem Şenel, Ihsan Utlu, Veysel Yücesoy, Aykut Koc, and Tolga Cukur. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1769–1779, 2018.

[SVB+11] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS One*, 6(8), 2011.

[SWG+20] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 519–527, 2020.

[SYS16] Ke-ke Shang, Wei-sheng Yan, and Michael Small. Evolving networks—using past structure to predict the future. *Physica A: Statistical Mechanics and its Applications*, 455:120–135, 2016.

[TBBER21] Leo Torres, Ann S Blevins, Danielle Bassett, and Tina Eliassi-Rad. The why, how, and when of representations for complex systems. *SIAM Review*, 63(3):435–485, 2021.

[TCW+18] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. Structural deep embedding for hyper-networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[TDS+20] Disheng Tang, Wenbo Du, Louis Shekhtman, Yijie Wang, Shlomo Havlin, Xianbin Cao, and Gang Yan. Predictability of real temporal networks. *National Science Review*, 7(5):929–937, 2020.

[TKG20] Maddalena Torricelli, Márton Karsai, and Laetitia Gauvin. weg2vec: Event embedding for temporal networks. *Scientific Reports*, 10(1):1–11, 2020.

[TMKM18] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the World Wide Web Conference*, pages 539–548, 2018.

[TPM19] Dane Taylor, Mason A. Porter, and Peter J. Mucha. *Supracentrality Analysis of Temporal Networks with Directed Interlayer Coupling*, pages 325–344. Springer International Publishing, 2019.

[TQW+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the World Wide Web Conference*, pages 1067–1077, 2015.

[TRDL+22] Abhishek Tomy, Matteo Razzanelli, Francesco Di Lauro, Daniela Rus, and Cosimo Della Santina. Estimating the state of epidemics spreading with graph neural networks. *Nonlinear Dynamics*, 109(1):249–263, 2022.

[TW16] N Kipf Thomas and Max Welling. Variational graph auto-encoders.(2016). In *Neural Information Processing Systems Workshop on Bayesian Deep Learning*, 2016.

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[VDM13] Anais Vergne, Laurent Decreusefond, and Philippe Martins. Reduction algorithm for simplicial complexes. In *Proceedings IEEE INFOCOM*, pages 475–479. IEEE, 2013.

[Ves09] Alessandro Vespignani. Predicting the behavior of techno-social systems. *Science*, 325(5939):425–428, 2009.

[Ves12] Alessandro Vespignani. Modelling dynamical processes in complex socio-technical systems. *Nature Physics*, 8(1):32–39, 2012.

[VFPC15] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2):021005, 2015.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[VZMZ15] Alexandre Vidmer, An Zeng, Matúš Medo, and Yi-Cheng Zhang. Prediction in complex systems: The case of the international trade network. *Physica A: Statistical Mechanics and its Applications*, 436:188–199, 2015.

[W+01] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[WCW+17] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[WK17] Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[WMC+22] Huan Wang, Chuang Ma, Han-Shuang Chen, Ying-Cheng Lai, and Hai-Feng Zhang. Full reconstruction of simplicial complexes from binary contagion and ising data. *Nature Communications*, 13(1):1–10, 2022.

[WPC+20] 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.

[WSD+19] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh V Chawla. Neural tensor factorization for temporal interaction learning. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 537–545, 2019.

[WY21] Xinru Wang and Ming Yin. Are explanations helpful? a comparative study of the effects of explanations in ai-assisted decision-making. In *26th International Conference on Intelligent User Interfaces*, pages 318–328, 2021.

[WYT+19] Yaojing Wang, Yuan Yao, Hanghang Tong, Feng Xu, and Jian Lu. Discerning edge influence for network embedding. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 429–438, 2019.

[XCH+10] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM International Conference on Data Mining*, pages 211–222. SIAM, 2010.

[XRK+20] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020.

[XSY+21] Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.

[Xu21] Mengjia Xu. Understanding graph embedding methods and their applications. *SIAM Review*, 63(4):825–853, 2021.

[YBY+19] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[YCA+18] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 2672–2681, 2018.

[YDZ+20] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *Proceedings of the International Conference on Knowledge Discovery & Data Mining*, pages 1666–1676, 2020.

[YL13] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 587–596, 2013.

[YNN+20] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 1705–1714, 2020.

[YNY+19] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in Neural Information Processing Systems*, 32, 2019.

[YPP21] Jean-Gabriel Young, Giovanni Petri, and Tiago P Peixoto. Hypergraph reconstruction from network data. *Communications Physics*, 4(1):1–11, 2021.

[YSSY20] Se-eun Yoon, Hyungseok Song, Kijung Shin, and Yung Yi. How much and when do we need higher-order information in hypergraphs? a case study on hyperedge prediction. In *Proceedings of the World Wide Web Conference*, pages 2627–2633, 2020.

[YSX+20] Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, Juncheng Liu, and Sourav S. Bhowmick. Scaling attributed network embedding to massive graphs. *Proceedings of the VLDB Endowment*, 14(1), 2020.

[YYW+21] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021.

[ZC18] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[ZCJC18] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. Beyond link prediction: Predicting hyperlinks in adjacency space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[ZCP+18] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. Timers: Error-bounded svd restart on dynamic networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[ZDW+19] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. Prone: Fast and scalable network representation learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[ZGY+16] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable temporal latent space inference for link prediction in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2765–2777, 2016.

[Zho21] Tao Zhou. Progresses and challenges in link prediction. *Iscience*, 24(11):103217, 2021.

[ZHS06] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in Neural Information Processing Systems*, 19, 2006.

[ZLB23] Yuanzhao Zhang, Maxime Lucas, and Federico Battiston. Higher-order interactions shape collective dynamics differently in hypergraphs and simplicial complexes. *Nature Communications*, 14(1):1605, 2023.

[ZLM+20] Xiu-Xiu Zhan, Ziyu Li, Naoki Masuda, Petter Holme, and Huijuan Wang. Susceptible-infected-spreading-based network embedding in static and temporal networks. *EPJ Data Science*, 9(1):30, 2020.

[ZLW+22] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. Learning from counterfactual links for link prediction. In *International Conference on Machine Learning*, pages 26911–26926. PMLR, 2022.

[ZPZ+19] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. Interaction embeddings for prediction and explanation in knowledge graphs. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 96–104, 2019.

[ZSBB11] Kun Zhao, Juliette Stehlé, Ginestra Bianconi, and Alain Barrat. Social network dynamics of face-to-face interactions. *Physical Review E*, 83(5):056109, 2011.

[ZX15] Boyao Zhu and Yongxiang Xia. An information-theoretic model for link prediction in complex networks. *Scientific Reports*, 5(1):1–11, 2015.

[ZYHD20] Chao Zhang, Zichao Yang, Xiaodong He, and Li Deng. Multimodal intelligence: Representation learning, information fusion, and applications. *IEEE Journal of Selected Topics in Signal Processing*, 14(3):478–493, 2020.

[ZYR+18] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[ZYR21] Yi-Jiao Zhang, Kai-Cheng Yang, and Filippo Radicchi. Systematic comparison of graph embedding methods in practical tasks. *Physical Review E*, 104(4):044315, 2021.

[ZYZZ18] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network representation learning: A survey. *IEEE Transactions on Big Data*, 6(1):3–28, 2018.

[ZZM20] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*, 2020.

[ZZS+23] Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, and Yizhou Sun. Page-link: Path-based graph neural network explanation for heterogeneous link prediction. *arXiv preprint arXiv:2302.12465*, 2023.

[ZZXT21] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34:29476–29490, 2021.