# Embedding alignment methods in dynamic networks

Kamil Tagowski kamil.tagowski@pwr.edu.pl Department of Computational Intelligence, Wroclaw University of Science and Technology Wrocław, Poland Piotr Bielak piotr.bielak@pwr.edu.pl Department of Computational Intelligence, Wroclaw University of Science and Technology Wrocław, Poland Tomasz Kajdanowicz tomasz.kajdanowicz@pwr.edu.pl Department of Computational Intelligence, Wroclaw University of Science and Technology Wrocław, Poland

Dynamic graph embedding can be performed in two settings: continuous and discrete-time. The first one allows to handle a single

event that triggers updates of node embeddings. The latter setting

that is commonly utilized, involves the aggregation of graph data

into snapshots and computes embeddings for each one of them.

Such snapshot embeddings are further combined into a single node

embedding that captures the whole graph evolution. Unfortunately,

such decomposition of the embedding process suffers from the sto-

chastic nature of representation learning algorithms. Embeddings

of consecutive snapshots are algebraically incomparable due to

## ABSTRACT

In recent years, dynamic graph embedding has attracted a lot of attention due to its usefulness in real-world scenarios. In this paper, we consider discrete-time dynamic graph representation learning, where embeddings are computed for each time window, and then are aggregated to represent the dynamics of a graph. However, independently computed embeddings in consecutive windows suffer from the stochastic nature of representation learning algorithms and are algebraically incomparable. We underline the need for embedding alignment process and provide nine alignment techniques evaluated on real-world datasets in link prediction and graph reconstruction tasks. Our experiments show that alignment of Node2vec embeddings improves the performance of downstream tasks up to 11 pp compared to the not aligned scenario.

### **CCS CONCEPTS**

- Computing methodologies  $\rightarrow$  Learning latent representations.

### **KEYWORDS**

dynamic graphs, graph embedding, embedding alignment

#### ACM Reference Format:

Kamil Tagowski, Piotr Bielak, and Tomasz Kajdanowicz. 2021. Embedding alignment methods in dynamic networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

# **1 INTRODUCTION**

Node representation learning is pervasive across multiple applications, like social networks [12, 20], spatial networks [23, 24] or citation networks [8, 20]. The vast majority of node embedding methods are trained in an unsupervised manner, providing an automated way of discovering node representations for static networks. In many real-world scenarios, the network structure evolves and node embedding depends on such dynamics. However, the body of knowledge for dynamic graph node embedding methods is rather unaddressed [3].

Woodstock '18, June 03–05, 2018, Woodstock, NY

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00 https://doi.org/10.1145/1122445.1122456

the transformations (artifacts) induced by the embedding methods. Therefore, there exists an research gap of how to deal with these unwanted transformations. The expected outcome is to map embeddings from particular snapshots into a common space. This can be achieved by embedding alignment methods that mitigate transformations and provide the ability to compare embeddings along with consecutive snapshots. Performing downstream tasks on nonaligned node embedding vectors may provide inconclusive results In this paper, we focus on several node embedding alignment methods that allow finding unified representation for nodes in dynamic networks using static network embedding approaches (in our case: node2vec). Based on extensive experiments on several realworld datasets, we demonstrate that node embedding alignment is crucial and allows to increase performance up to 11 pp compared to not aligned embeddings (node2vec). We summarize our contributions as follows: (1) We propose nine embedding alignment methods for graph. (2) We provide a comprehensive evaluation showing that alignment is an indispensable operation in dynamic graph embedding based on a discrete approach, while dealing with node2vec embeddings. Additionally, in the Appendix B, (3) we formulate aligner performance measures (AMPs) for evaluating alignment algorithms, regardless of the downstream tasks.

# 2 RELATED WORKS

The literature on static node embedding methods is very rich [3]. We can distinguish approaches based on: random-walks (Node2vec [12], metapath2vec [8]), graph neural networks (GCN [13], GAT [22]) and matrix factorization (LLE [18], HOPE [16]). Despite being very powerful concepts, their applicability to dynamic graph embeddings is very limited. Embedding alignment is a tool that makes static embedding usable. Indeed, embedding alignment is crucial in many machine learning areas, e.g., in machine translation [11], cross-graph alignment [4, 5, 7], dynamic graph embedding [2, 19, 21]. Embedding alignment techniques are often based on solving Orthogonal Procrustes problem to obtain a linear transformation between pairs of embeddings [5]. We can also distinguish

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

approaches that utilize adversarial training [4, 7]. Dynamic graph embedding methods can update embeddings in an online manner (arrival of single events) [14, 15] or batched one (graph snapshot) [2, 10, 19, 21]. In the tNodeEmbed [19] and LCF [21] methods, alignment is achieved by solving the Orthogonal Procrustes problem using all common nodes. In FILDNE [2], the authors do not follow this scenario and they provide a mechanism for selecting only a subset of nodes used in the alignment process.

#### **3 GRAPH EMBEDDING ALIGNMENT**



Figure 1: Graph embedding alignment in the whole graph processing pipeline. For a dynamic graph in the form of snapshots, compute node embeddings, then (optionally) determine the reference nodes  $V_{\text{REF}}$  and align the newest embedding  $F_{t-1,t}$  to a given reference embedding (*previous* one  $F_{t-2,t-1}$ , or the *first* one overall  $F_{0,1}$ ). Aligned embeddings improve the performance of downstream tasks.

#### 3.1 Notation and problem statement

We denote a dynamic graph as  $G_{0,T} = (\mathcal{V}_{0,T}, \mathcal{E}_{0,T})$ , where  $\mathcal{V}_{0,T}$  and  $\mathcal{E}_{0,T}$  are the sets of all nodes (vertices) and edges observed between timestamp 0 and *T*, respectively. We model a dynamic graph as a series of snapshots  $G_{0,1}, G_{1,2}, \ldots, G_{T-1,T}$ . A node embedding function  $f: \mathcal{V} \to \mathbb{R}^{|\mathcal{V}| \times d}$  maps every node  $v \in \mathcal{V}$  into a low-dimensional vector representation of size  $d, d \ll |\mathcal{V}|$ , resulting in a node embedding matrix *F*, where each row represents an embedding of a single node. Embedding alignment  $g: \mathbb{R}^{|\mathcal{V}| \times d} \to \mathbb{R}^{|\mathcal{V}| \times d}$  transforms (aligns) a node embedding matrix *F* respective to another one, producing an aligned node embedding matrix  $F^*$ . The method is trained on the observed change of embeddings for a subset of nodes (reference nodes  $\mathcal{V}_{\text{REF}} \subseteq \mathcal{V}$ ).

#### 3.2 The importance of embedding alignment

Node embedding methods encode the structure of graphs in lowdimensional node representation vectors. The final form of the embedding space and the actual positions of node vectors highly depend on the particular method, e.g., in random walk-based methods (like node2vec), there are three sources: (1) the stochastic nature of random walk generation, (2) the random initialization of embedding vectors, and (3) the order of node-context pairs used for training the Skip-gram model. It results in the situation that calculating embeddings for the same graph twice, using the same method hyperparameters, we observe that the node embeddings in the second run end up in different positions. Deformations of embeddings may be caused by wide family of geometric transformations. For simplicity, we hypothesize that it is enough to consider a subset of linear transformations – translation, scaling, and rotations (see: Figure 2). These are examples of affine transformations, i.e., composed of linear transformations and translations of the embedding space. In such situations, two embeddings of the same node are incomparable.



Figure 2: Embedding the same graph twice.

The problem becomes fundamental in dynamic graph embedding, where representations are computed for every snapshot independently. In downstream tasks, these embeddings are combined to obtain a representation for the whole dynamic graph. To achieve a rational combination of snapshots' embeddings, we are forced to align them. All rotations, scaling, and translations must be eliminated (see: Figure 3). Proper alignment requires some transformation anchors (nodes). Sophisticated methods may automatically learn to perform the alignment against the common nodes between consecutive graphs. Notwithstanding, we developed a much simpler and computationally less complex methodology to select a subset of common nodes present in both graph snapshots. It significantly widens the applicability of the alignment to large scale networks. Our intuition is that nodes whose local structure has significantly changed, should not have been used to perform the alignment. The selection of appropriate reference nodes influences the performance of downstream tasks.



Figure 3: Embedding of dynamic graph snapshots with alignment method applied.

# 3.3 Dynamic graph embedding alignment methods

The embedding alignment problem might be addressed in two major ways: either (i) using post-hoc alignment of already computed embedding matrices, or (ii) adding an auxiliary loss to the embedding

Woodstock '18, June 03-05, 2018, Woodstock, NY

method that ensures alignment of node embeddings. The first one assumes the situation where all embeddings of consecutive static networks are already computed and the networks are not available at the time of dynamic embedding. On the other hand, the latter setting benefits from the availability of two network snapshots as it can address embedding deformation artifacts. In this work, we consider the post-hoc setting only leaving the second approach for the future work. To keep our proposed alignment methods computationally trackable, we focus in this work on the methods inspired by matrix alignment using the Orthogonal Procrustes problem [19]. Thus, we omit computationally expensive methods like [7].

Given two matrices  $A \in \mathbb{R}^{n \times d}$  and  $B \in \mathbb{R}^{n \times d}$  with matching rows, the Orthogonal Procrustes method find a transformation matrix Q such that:  $\operatorname{argmin}_{Q:Q^{\top}Q=I} ||BQ - A||_2^2$ . It can be solved as  $Q_{\text{opt}} = UW^{\top}$  with  $U\Sigma W^{\top}$  being the Singular Value Decomposition (SVD) of  $B^{\top}A$ . In the case of dynamic graph embedding alignment, the A and B matrices are two node embeddings.

As pointed out in Section 3.2, for dynamic graphs, we will select a subset of nodes, whose characteristics were the same (or at least similar) in both graph snapshots. Although, for completeness, we evaluate also the approach with all common nodes as reference nodes and call it **Procrustes Aligner (PA)**.

A simple yet quite restrictive approach, called **Procrustes Unchanged Aligner (PUA)**, is to select all nodes whose neighborhood does not change between snapshots, i.e., nodes having the same neighbors in both snapshots.

Another method for selecting only a subset of nodes as the reference nodes was already presented in FILDNE [2]. The authors proposed to use a node's activity (activity function), in the form of the *multi-degree* centrality, and then compare these activities in both snapshots using a scoring function to finally obtain a nodes' score s. We evaluated this approach in our experiments as FILDNE Aligner (FA).

We further explore this idea and propose to apply other activity functions. Due to the fact that the embedding is performed on dynamic graphs, we propose to utilize temporal (dynamic) node centrality measures for the activity functions (as in [25]): temporal betweenness **TB**, temporal closeness **TC**, temporal k-shell score **TK**, and temporal degree deviation **TDD**. We choose L1-norm as the scoring function:  $s(a_{t-1}^{(v)}, a_t^{(v)}) = |a_{t-1}^{(v)} - a_t^{(v)}|$ . We also further develop another perspective of the scoring function:

We also further develop another perspective of the scoring function that deeper exploits the structural graph properties, instead of node activity only. The **Edge Jaccard Aligner (EJA)** computes the Jaccard distance of neighbouring edges for every node existing in two snapshots:

$$s(E_{t-1}^{(v)}, E_t^{(v)}) = 1 - \frac{|E_{t-1}^{(v)} \cap E_t^{(v)}|}{|E_{t-1}^{(v)} \cup E_t^{(v)}|},$$
(1)

where  $E_t^{(v)}$  are edges in the snapshot  $G_{t-1,t}$  connected to node v.

The last proposed aligner is the **Embedding Neighbor Jaccard Aligner (ENJA)**, which utilizes the computed node embeddings. For all nodes existing in two snapshots, it extracts *n* percent of the closest neighbors in both embedding spaces, and then computes the Jaccard distance of neighbor sets:

$$s(F_{t-1}^{(v)}, F_t^{(v)}) = 1 - \frac{|CN_n(F_{t-1}^{(v)}) \cap CN_n(F_t^{(v)})|}{|CN_n(F_{t-1}^{(v)}) \cup CN_n(F_t^{(v)})|},$$
(2)

where  $CN_n(F_t^{(v)})$  is the set of the top *n* percent of closest neighbors of node *v* in the the embedding  $F_t$ .

We still need to select the actual reference nodes based on the computed nodes' scores. In our experiments, we consider the **top percent** scenario from FILDNE [2] as the other ones can be equivalently applied. We select the top **p** percent of the lowest scores: select(S, W) =  $V_{\text{REF}} \subseteq \text{sort}_S(W)$ , s.t.  $|V_{\text{REF}}| = \mathbf{p}|W|$ , where S are the node scores.

#### **4 EXPERIMENTS**

We evaluate all the proposed alignment methods on nine real-world datasets (see: Appendix A) on two downstream tasks: link prediction and graph reconstruction. The code, as well as the computational environment configuration (DVC pipeline), is made publicly available at https://gitlab.com/tgem/embedding-alignment to ensure reproducibility.

#### 4.1 Node embeddings

To compute node embeddings we utilize the Node2vec method implemented in the PyTorch Geometric library [9]. We embed each snapshot separately. Using the Hyperopt optimizer [1] (restricted to 200 iterations), we performed Node2vec hyperparameter search. We recomputed all embeddings 25 times to handle the stochastic nature of Node2vec (random initialization and random walks). The obtained embeddings were evaluated on a link prediction task on the same snapshot, i.e., embedding  $F_{t-1,t}$  was evaluated against graph  $G_{t-1,t}$  for all snapshots.

#### 4.2 Embedding aggregation

In our setting, all downstream tasks require a single embedding for a given node that captures its whole history in the dynamic graph. Hence, we need to combine nodes' embeddings from all snapshots. There are several approaches, like the simple averaging, convex combination with Bayesian inference [2], exponential decaying, linear combination [21], or deep neural networks [10, 19]. We decided to choose the most computationally efficient one – averaging node embedding vectors from all snapshots.

#### 4.3 Embedding alignment

We evaluate all the proposed aligners (Section 3.3) accompanied with not aligned embeddings N/AL as a baseline. As shown in Figure 1, one could either align a given snapshot embedding  $F_{t-1,t}$ to its *previous* one  $F_{t-2,t-1}^*$  or the **first embedding overall**  $F_{0,1}$ . We decided to use the latter setting, as it provides a common reference space for all the following snapshots. Moreover, we performed a grid search on the link prediction task using aggregated embeddings. The percent parameter was evaluated for a range  $p \in \{0.1, 0.2, ..., 0.9\}$  and (for ENJA aligner) the parameter *n* was evaluated for following values:  $n \in \{0.1, 0.2, ..., 1.0\}$ .

#### 4.4 Link prediction

*Setup.* Link prediction evaluation predicts the existence of edges in the last snapshot based on previous ones. We combine the snapshot embeddings  $F_{0,1}, \ldots, F_{T-2,T-1}$  using average operator. The link prediction dataset is generated from the last snapshot  $G_{T-1,T}$ , with edges in the graph as existing links (class 1). Additionally, we sample an equal number of non-existing edges (class 0). We split the dataset into train (75%) and test (25%). Using edge representations, obtained from the Hadamard product, we train a Logistic Regression classifier. To measure the performance, we report the mean and standard deviation of the AUC metric over 25 runs.

Results. The results can be found in Table 1 (upper half). Notice that for the vast majority of cases, the alignment of node embeddings improves the overall link prediction performance. We can gain up to 10 and 11pp (bitcoin-alpha with EJA, enron-employees with ENJA) over non-aligned embeddings. In the case of the Procrustes Unchanged aligner (PUA), we observe that five out of nine datasets could not be aligned. This occurs because this aligner relies on nodes, whose neighborhood was precisely the same between snapshots and for those datasets there were snapshot pairs with an empty set of reference nodes. Moreover, such a restrictive selection criterion affects the performance - see bitcoin-alpha and bitcoin-otc, where the aligned embeddings perform worse than not aligned ones (-0.95pp and -0.44pp, respectively). On the other hand, the biggest loss of -1.03pp was for hypertext using ENJA, but comparing the standard deviations of the results ENJA provides a more robust embedding. Among the best aligners we find Temporal Betweenness (fb-forum, fb-messages, ppi), Edge Jaccard (radoslaw-email, bitcoin-alpha) and Embedding Neighbor Jaccard (enron-employees, bitcoin-otc).

#### 4.5 Graph reconstruction

Setup. Graph reconstruction evaluation aims at reproducing the graph structure based on nodes' embedding. In our case, we expect to reconstruct the whole dynamic graph  $G_{0,T}$  from the dynamic embedding of all snapshots  $F_{0,T} = \operatorname{Avg}(F_{0,1}, \ldots, F_{T-1,T})$ . We compute the mean Average Precision score (mAP) for graphs [6]. This metric captures local graph properties, i.e. for any node and its embedding vector it checks how many of the nearest vectors in the embedding space (in the sense of euclidean norm) are actually first-order neighbors of this node (see FILDNE [2] for details). Similarly to link prediction results, we also report the mean and standard deviation of the mAP metric over 25 runs.

*Results.* The results can be found in Table 1 (lower half). For all but one case, we observe an improvement in the mAP metric values – up to about 8 pp (for enron-employees with PA and hypertext with TB). The only worse result occurs in hypertext with EJA (-0.53 pp). The best performing aligners are Procrustes (bitcoin-otc, fb-messages, enron-employees, ppi) and Temporal Betweenness (fb-forum, hypertext, radoslaw-email).

Overall, all the detailed results obtained in our study over embedding alignment methods allow claiming that (1) alignment is essential for embedding algorithms and (2) provide superior results in downstream tasks like link prediction and graph reconstruction. Ta wł da

statistically significant differences between alignment methods. The "*" symbol denotes methods that are significantly worse than the best method.
taset. We performed a Friedman test with Nemenyi post-hoc to confirm that all alignment methods are statistically different from N/AL case. There we
th gain (or loss) of aligned embeddings over not aligned ones (difference of mean values) in parenthesis. Bold values mark the best results for a sing
rere embeddings could not be aligned due to missing reference nodes. We present values as the mean and standard deviations over 25 embedding retrain
ble 1: Downstream task results (link prediction and graph reconstruction). N/AL denotes evaluation of not aligned embeddings and $ imes$ denotes a scenari

Graph Reconst. (mA	AP)	Link prediction (AUC	
PA PUA FA TB TC TK TDD	ALIGNER N/AL	PA PUA FA TB TC TK TDD EJA EJA	ALIGNER
$\begin{array}{l} 27.40 \pm 0.25 \left( +2.22 \right) \\ 25.21 \pm 0.28 \left( +0.03 \right) \ast \\ 26.44 \pm 0.24 \left( +1.26 \right) \ast \\ 27.34 \pm 0.28 \left( +2.16 \right) \\ 27.48 \pm 0.26 \left( +2.30 \right) \\ 26.70 \pm 0.31 \left( +1.52 \right) \ast \\ 26.70 \pm 0.31 \left( +1.52 \right) \ast \\ 26.70 \pm 0.31 \left( +1.52 \right) \ast \end{array}$	BITCOIN ALPHA 25.18 ± 0.20 *	$ \begin{array}{l} 5.80 \pm 13.95  (+3.08) \\ 46.77 \pm 14.17  (-0.95) \\ 50.91 \pm 13.75  (+3.19) \\ 53.42 \pm 13.35  (+5.70) \\ 53.42 \pm 13.35  (+5.70) \\ 55.21 \pm 12.61  (+7.49) \\ 55.91 \pm 12.77  (+9.19) \\ 56.91 \pm 12.77  (+9.19) \\ 56.91 \pm 12.77  (+9.19) \\ 56.60 \pm 12.06  (+8.88) \\ \end{array} $	BITCOIN ALPHA 47 72 + 14 44
$\begin{array}{c} \textbf{30.61} \pm \textbf{0.31}_{(+2.52)} \\ \textbf{28.23} \pm \textbf{0.39}_{(+0.14)} \ast \\ \textbf{30.44} \pm \textbf{0.35}_{(+2.35)} \ast \\ \textbf{30.53} \pm \textbf{0.34}_{(+2.44)} \\ \textbf{30.53} \pm \textbf{0.32}_{(+2.44)} \\ \textbf{30.47} \pm \textbf{0.32}_{(+2.48)} \\ \textbf{30.47} \pm \textbf{0.32}_{(+2.38)} \\ \textbf{30.47} \pm \textbf{0.48}_{(+2.38)} \\ \textbf{30.47} \pm \textbf{0.32}_{(+2.38)} \\ \textbf{30.47} \pm \textbf{0.34}_{(+2.38)} \\ \textbf{30.47} \pm \textbf{0.34}_{$	BITCOIN OTC 28.09 ± 0.36 *	$\begin{array}{c} 79.37\pm2.83\left(+9.18\right)\\ 69.75\pm5.41\left(-0.44\right)*\\ 77.48\pm4.76\left(+7.29\right)\\ 78.91\pm3.75\left(+8.72\right)\\ 78.82\pm3.75\left(+8.72\right)\\ 79.22\pm3.75\left(+7.83\right)\\ 79.28\pm2.82\left(+9.90\right)\\ 79.06\pm3.95\left(+8.87\right)\\ 79.67\pm2.87\left(+9.48\right)\\ \end{array}$	BITCOIN OTC 70 19 + 5.63 *
$\begin{array}{l} 14.08 \pm 0.32 (\pm 2.78) \\ 11.56 \pm 0.36 (\pm 0.26) \\ 14.15 \pm 0.31 (\pm 2.85) \\ 14.15 \pm 0.31 (\pm 2.85) \\ 14.35 \pm 0.38 (\pm 3.05) \\ 14.11 \pm 0.31 (\pm 2.81) \\ 13.97 \pm 0.32 (\pm 2.67) \\ 13.97 \pm 0.32 (\pm 0.57) \\ 13.97 \pm 0.57 \pm 0.57 \\ 13.97 \pm 0.57 \pm 0.57 \\ 13.97 \pm 0.57 \pm 0.57 \\ 14.97$	FB FORUM 11.30 ± 0.29 *	$\begin{array}{l} 90.36\pm1.36(+7.07)\\ 90.36\pm1.28(+7.07)\\ 83.72\pm2.33(+0.43)\\ 90.47\pm1.28(+7.18)\\ 91.07\pm1.37(+7.78)\\ 90.03\pm1.90(+6.74)\\ 90.71\pm1.04(+7.42)\\ 90.71\pm1.04(+7.42)\\ 90.71\pm1.04(+7.42)\\ 89.64\pm1.73(+6.35)\\ 89.05\pm1.49(+7.27)\\ \end{array}$	FB FORUM 83 29 + 2.80 *
$\begin{array}{c} \textbf{16.05} \pm \textbf{0.24}_{(+1.62)} \\ \times \\ \textbf{15.90} \pm \textbf{0.23}_{(+1.47)} \\ \textbf{15.98} \pm \textbf{0.25}_{(+1.55)} \\ \textbf{15.76} \pm \textbf{0.22}_{(+1.33)} \\ \textbf{15.90} \pm \textbf{0.24}_{(+1.47)} \\ \textbf{15.90} \pm \textbf{0.24}_{(+1.47)} \\ \textbf{15.90} \pm \textbf{0.24}_{(+1.47)} \\ \textbf{15.90} \pm \textbf{0.24}_{(+1.47)} \end{array}$	FB MESSAGES 14.43 ± 0.19 *	$\begin{array}{c} 64.6\pm 8.33(+4.96)\\ \times\\ 63.60\pm 6.61(+4.10)\\ 66.79\pm 6.74(+7.29)\\ 64.85\pm 7.26(+5.31)\\ 62.68\pm 6.31(+3.18)\\ 62.68\pm 6.31(+3.18)\\ 62.68\pm 6.31(+3.18)\\ 62.68\pm 6.31(+3.18)\\ 62.68\pm 6.31(+3.18)\\ 62.79\pm 7.99(+5.20)\\ 64.70\pm 7.99(+5.20)\\ \end{array}$	FB MESSAGES 59.50 + 6.80 *
$\begin{array}{l} {\rm 52.53}\pm0.90({}_{+8.00})\\ \times\\ {\rm 51.46}\pm1.09({}_{+6.93})*\\ {\rm 50.93}\pm1.06({}_{+6.40})*\\ {\rm 52.46}\pm1.12({}_{+7.93})\\ {\rm 52.16}\pm1.07({}_{+7.63})\\ {\rm 52.16}\pm1.07({}_{-7.63})\\ {\rm 52.16}({}_{-7.63},{}_{-7.63})\\ {\rm 52.16}({}_{-7.63},{}_{-$	ENRON EMPLOYEES 44.53 ± 1.85 *	$\begin{array}{c} 84.48 \pm 1.37_{(\pm10,19)} \\ \times \\ 83.09 \pm 1.39_{(\pm8.60)} \\ 79.83 \pm 2.04_{(\pm5.54)} \\ 83.10 \pm 2.08_{(\pm8.15)} \\ 83.10 \pm 2.08_{(\pm8.15)} \\ 83.50 \pm 0.97_{(\pm9.21)} \\ 83.50 \pm 0.97_{(\pm9.21)} \\ 83.50 \pm 0.97_{(\pm9.21)} \\ 84.46 \pm 1.25_{(\pm10,17)} \\ 84.29 \pm 1.29_{(\pm11,00)} \end{array}$	EMPLOYEES 74.29 + 3.95 *
$\begin{array}{l} 57.46\pm0.74_{(+4.98)}\\ \times\\ 59.04\pm0.93_{(+6.56)}\\ 60.42\pm0.87_{(+7.94)}\\ 55.76\pm0.88_{(+3.28)}\\ 54.60\pm0.70_{(+2.12)}\\ 54.60\pm0.70_{(+2.12)}\\ 54.60\pm0.70_{(+2.12)}\\ 54.60\pm0.70_{(+2.12)}\\ \end{array}$	HYPERTEXT 52.48 ± 1.15 *	$\begin{array}{c} 87.56\pm6.31(-0.43)\\ \times\\ 87.95\pm5.59(-0.04)\\ 88.95\pm4.68(+0.96)\\ 90.37\pm4.45(+2.43)\\ 90.42\pm2.02(+2.43)\\ 90.42\pm2.02(+2.43)\\ 90.42\pm5.00(+0.56)\\ 88.55\pm5.00(+0.56)\\ 86.99\pm4.80(-1.03)\\ \end{array}$	HYPERTEXT 87.99 + 12.83
$\begin{array}{l} 45.27\pm0.42_{(+6,42)}*\\ 42.94\pm0.59_{(+4,09)}*\\ 45.91\pm0.43_{(+7,06)}*\\ 46.67\pm0.41_{(+7,82)}\\ 46.62\pm0.52_{(+7,37)}\\ 44.64\pm0.58_{(+5,79)}*\\ 44.64\pm0.58_{(+5,7$	RADOSLAW EMAIL 38.85 ± 1.19 *	$\begin{array}{l} 92.89\pm0.38(_{+8.57})*\\ 86.55\pm1.05(_{+2.23})*\\ 92.96\pm0.38(_{+8.64})\\ 92.70\pm0.49(_{+8.38})*\\ 92.78\pm0.42(_{+8.48})*\\ 92.78\pm0.42(_{+8.46})*\\ 92.93\pm0.37(_{+8.61})*\\ 92.93\pm0.37(_{+8.61})*\\ 93.23\pm0.37(_{+8.91})\\ 93.02\pm0.43(_{+8.70}) \end{array}$	RADOSLAW EMAIL 84 32 + 1 34 *
$\begin{array}{c} 11.99 \pm 0.04  ({\scriptstyle +0.26}) \\ \times \\ 11.97 \pm 0.05  ({\scriptstyle +0.21}) \\ 11.88 \pm 0.04  ({\scriptstyle +0.15}) \ast \\ 11.88 \pm 0.04  ({\scriptstyle +0.12}) \ast \\ 11.88 \pm 0.05  ({\scriptstyle +0.25}) \\ 11.98 \pm 0.05  ({\scriptstyle +0.25}) \\ 1$	OGBL COLLAB 11.73 ± 0.05 *	$\begin{array}{l} 82.48 \pm 0.53_{(+1,24)} \\ \times \\ 82.40 \pm 0.67_{(+1,16)} \\ 82.19 \pm 0.53_{(+0,79)} \\ 82.19 \pm 0.53_{(+0,79)} \\ 82.41 \pm 0.55_{(+1,17)} \\ 82.41 \pm 0.55_{(+1,17)} \\ 82.41 \pm 0.55_{(+1,17)} \\ 82.42 \pm 0.65_{(+1,18)} \\ 82.38 \pm 0.46_{(+1,14)} \end{array}$	OGBL COLLAB 81.24 + 0.45 *
$\begin{array}{c} 9.43 \pm 0.05_{(+0.53)} \\ \times \\ 9.06 \pm 0.09_{(+0.16)} \\ 9.39 \pm 0.05_{(+0.49)} \end{array}$	ррI 8.90 ± 0.07 *	$\begin{array}{c} 60.22\pm0.78(+1.14)\\ \times\\ 60.26\pm0.74(+1.18)\\ 60.52\pm0.57(+1.45)\\ 59.80\pm0.68(+0.72)\\ 60.11\pm0.80(+1.03)\\ 60.11\pm0.80(+1.03)\\ 60.11\pm0.80(+1.03)\\ 60.14\pm0.77(+1.06)\\ 60.26\pm0.75(+1.18)\end{array}$	PPI

 $30.54 \pm 0.30$ 

14.10

51

(+6.57) \*

 $57.42 \pm 0.85$ 

11.97

 $\pm 0.04_{(+)}$ 

ŧ

# 4.6 Impact of the node fraction taken in the alignment process

In Table 1 we only reported the best scores for a particular aligner. We performed also grid search over parameter p, see Section 4.3, which describes percent of reference nodes taken from all common nodes. We compared each result with *PA* aligner (its mean and std result) that used all common nodes. It turned out that for bitcoinalpha, bitcoin-otc, fb-forum, fb-messages, ia-hypertext, ppi and ogbl-collab datasets it was sufficient to take only 10 percent of nodes, whereas for enron-employees 20 percent and for radoslaw-email 30 of common nodes to achieve comparative results. Such feature is especially crucial when dealing with large datasets, as it shortens computation time of Orthogonal Procrustes.

#### **5 CONCLUSIONS AND FUTURE WORK**

In this paper, we emphasize the importance of node embedding alignment in dynamic graph embedding. We formulate three aligner performance measures for the evaluation of alignment algorithms. We propose several embedding alignment methods for dynamic graphs. According to our experimental evaluation, alignment is an indispensable operation in dynamic graph embedding based on the discrete approach. Furthermore, embedding alignment improves the performance of downstream tasks up to 11 pp compared to the not aligned scenario. In terms of future work, we plan to exploit other approaches of embedding alignment and these that are directed to edges and subgraphs.

#### ACKNOWLEDGMENTS

The project was partially supported by The National Science Centre, Poland, the research project no. 2016/21/D/ST6/02948 and statutory funds of Department of Computational Intelligence.

#### REFERENCES

- James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*. 115–123.
- [2] Piotr Bielak, Kamil Tagowski, Maciej Falkiewicz, Tomasz Kajdanowicz, and Nitesh V. Chawla. 2020. FILDNE: A Framework for Incremental Learning of Dynamic Networks Embeddings. arXiv:1904.03423 [stat.ML]
- [3] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. 2020. Machine Learning on Graphs: A Model and Comprehensive Taxonomy. arXiv preprint arXiv:2005.03675 (2020).
- [4] Chaoqi Chen, Weiping Xie, Tingyang Xu, Yu Rong, Wen-bing Huang, Xinghao Ding, Yue Huang, and Junzhou Huang. 2019. Unsupervised Adversarial Graph Alignment with Graph Embedding. *CoRR* abs/1907.00544 (2019). arXiv:1907.00544 http://arxiv.org/abs/1907.00544
- [5] Xiyuan Chen, Mark Heimann, Fatemeh Vahedian, and Danai Koutra. 2020. CONE-Align: Consistent Network Alignment with Proximity-Preserving Node Embedding, Association for Computing Machinery, New York, NY, USA, 1985–1988. https: //doi.org/10.1145/3340531.3412136
- [6] Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. 2018. Representation tradeoffs for hyperbolic embeddings. *Proceedings of machine learning research* 80 (2018), 4460.
- [7] Tyler Derr, Hamid Karimi, Xiaorui Liu, Jiejun Xu, and Jiliang Tang. 2019. Deep Adversarial Network Alignment. *CoRR* abs/1902.10307 (2019). arXiv:1902.10307 http://arxiv.org/abs/1902.10307
- [8] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. 135–144.
- [9] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds.

- [10] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.
- [11] Edouard Grave, Armand Joulin, and Quentin Berthet. 2019. Unsupervised Alignment of Embeddings with Wasserstein Procrustes. In Proceedings of Machine Learning Research (Proceedings of Machine Learning Research, Vol. 89), Kamalika Chaudhuri and Masashi Sugiyama (Eds.). PMLR, 1880–1890. http: //proceedings.mlr.press/v89/grave19a.html
- [12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864.
- [13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).
- [14] John Boaz Lee, Giang Nguyen, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2019. Dynamic Node Embeddings from Edge Streams. arXiv:1904.06449 [cs.LG]
- [15] Yao Ma, Ziyi Guo, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming Graph Neural Networks. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 719–728. https://doi.org/10.1145/3397271.3401092
- [16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 1105– 1114.
- [17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 701–710.
- [18] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.
- [19] Uriel Singer, Ido Guy, and Kira Radinsky. 2019. Node embedding over temporal graphs. In Proceedings of the 28th International Joint Conference on Artificial Intelligence. AAAI Press, 4605–4612.
- [20] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web. 1067–1077.
- [21] Puja Trivedi, Alican Büyükçakır, Yin Lin, Yinlong Qian, Di Jin, and Danai Koutra. 2020. On Structural vs. Proximity-based Temporal Node Embeddings. (2020).
- [22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).
- [23] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019).
- [24] Dongwei Xu, Chenchen Wei, Peng Peng, Qi Xuan, and Haifeng Guo. 2020. GE-GAN: A novel deep learning framework for road traffic state estimation. *Transportation Research Part C: Emerging Technologies* 117 (2020), 102635.
- [25] En-Yu Yu, Yan Fu, Xiao Chen, Mei Xie, and Duan-Bing Chen. 2020. Identifying critical nodes in temporal networks by network embedding. *Scientific Reports* 10, 1 (2020), 1–8.

## A DATASETS

We perform experiments on nine real-world datasets (see Table 2). Each of them is split based on the timestamp frequency: daily (iahypertext), monthly (enron-employees, radoslaw-email, fb-forum, fb-messages) and yearly (bitcoin-alpha, bitcoin-otc, ppi, ogbl-collab). The total number of snapshots varies from 3 to 9. As dataset time characteristics are not ideal and some of the generated snapshots were too small, we performed the following operations: we merged the first snapshot into the second one (bitcoin-alpha, bitcoin-otc, fb-messages); we merged the last snapshot to the second last one, as validation on tiny snapshots would be biased (bitcoin-alpha, bitcoin-otc, employees, ia-radoslaw-email); we also ignored first four snapshots (ppi, ogbl-collab), as merging them would result in a much bigger time-span than other snapshots.

#### **B** ALIGNMENT PERFORMANCE MEASURES

We introduce a novel set of alignment performance measures (*APM*) that constitute the criteria an alignment algorithm should meet. We

Woodstock '18, June 03-05, 2018, Woodstock, NY

Table 2: Statistics of graph datasets.  $|\mathcal{V}|$  - number of nodes,  $|\mathcal{E}|$  - number of edges, Directed - whether the graph is directed or not

Dataset	$ \mathcal{V} $	3	Directed	Timespan	NUMBER OF SNAPSHOTS	Snapshot timespan
HYPERTEXT	113	20 818	×	2.5 days	3	1 day
ENRON-EMPLOYEES	151	50 572	×	37.9 months	6	6 MONTHS
RADOSLAW-EMAIL	167	82 927	√	9 months	9	1 month
FB-FORUM	899	33 720	×	5.5 months	5	1 month
FB-MESSAGES	1 899	61 734	×	7.2 months	7	1 month
BITCOIN-ALPHA	3 783	24 186	√	5.2 years	5	1 year
BITCOIN-OTC	5 881	35 592	V	5.2 years	5	1 year
PPI	16 386	141 836	×	24 years	5	5 years
OGBL-COLLAB	233 513	1 171 947	×	34 years	7	5 years

also propose measures to evaluate the fit of alignment methods to criteria.

Overall, graph representation learning methods derive low-dimensional vector embeddings for different entities in the graph, i.e., nodes, edges, or subgraphs. From now on, we will present a case of node embedding approach, but it can be easily transfigured to edge or subgraph related problems.

The aim of node embedding methods is, generally speaking, to encode structural information in vector representations by placing embeddings of similar nodes near in the embedding space and keeping dissimilar nodes at a further distance. The definitions of "distance" and "similarity" as well as "structural information" depend on the properties of the representation we want to achieve (e.g., Euclidean distance, cosine similarity, considering first or second-order neighborhoods).

The family of random walk-based embedding approaches (e.g., DeepWalk [17], Node2vec [12], metapath2vec [8]) explicitly preserve the distances between nodes in the graph. Thus, we postulate the first *APM* which is related to distances:

# APM 1. The pairwise distances of vectors in the embedding space should be preserved during alignment.

Changing the relative distances corrupts the information encoded by the embedding algorithm. To measure the magnitude of changes in relative distances, we propose **Pairwise Embedding Distance** (**PED**; see Equation 3). This performance measure quantifies how the embedding alignment method mitigates translation and rotation. High values of this metric indicate that the embedding structure is corrupted during the alignment process. Contrary, if the value is equal to zero ( $PED(F, F^*) = 0$ ), the embedding is perfectly preserved.

$$PED(F, F^*) = \frac{1}{|\mathcal{V}| * (|\mathcal{V}| - 1)} \sum_{\substack{(u,v) \in \mathcal{V} \times \mathcal{V} \\ u \neq v}} |D^{(F)}(u,v) - D^{(F^*)}(u,v)|,$$
(3)

where *F* is the initial node embedding matrix,  $F^*$  is the matrix after alignment,  $|\cdot|$  denotes the absolute value. For the distance measure  $D^{(F)}(u, v)$  between embeddings of nodes *u* and *v* in the embedding matrix *F*, we use the  $L_2$  distance, but one can employ other ones, like the cosine distance.

Besides translation and rotation, we also consider the scaling transformation, proposing the following APM.

APM 2. The scaling of distances between reference nodes after alignment should be the same for all other nodes.

We propose the Scaling Score Distance (SSD), see Equation 4.

$$SSD(F, F^*) = \left| \frac{1}{|\mathcal{V}_{\text{REF}}| * (|\mathcal{V}_{\text{REF}}| - 1)} \sum_{\substack{(u_R, v_R) \in \mathcal{V}_{\text{REF}} \times \mathcal{V}_{\text{REF}} \\ u_R \neq v_R}} \frac{D^{(F^*)}(u_R, v_R)}{D^{(F)}(u_R, v_R)} - \frac{1}{|\mathcal{V} \setminus \mathcal{V}_{\text{REF}}| * (|\mathcal{V} \setminus \mathcal{V}_{\text{REF}}| - 1)} \sum_{\substack{(u,v) \in (\mathcal{V} \setminus \mathcal{V}_{\text{REF}}) \times (\mathcal{V} \setminus \mathcal{V}_{\text{REF}}) \\ u \neq v}} \frac{D^{(F^*)}(u,v)}{D^{(F)}(u,v)} \right|$$

$$(4)$$

We also address the requirement of preserving the same positions of reference nodes in two snapshots of a dynamic graph. We assume that the alignment is performed according to those nodes. This leads us to our third APM:

APM 3. After the alignment of the embedding of the second snapshot, the vectors of reference nodes must be placed in the same positions as in the embedding of the first snapshot.



Figure 4: Visualization of RND alignment performance measure. We compare two scenarios: (1) RND computed using non-aligned embedding and (2) RND computed using aligned embeddings. We aggregate values across all datasets and show that in the RND for aligned embeddings are lower than for not aligned case. This indicates that the alignment is successful.

We propose the **Reference Nodes Distance** (**RND**; see Equation 5).

$$\mathbf{RND}(F_1, F_2^*) = \frac{1}{|\mathcal{V}_{\text{REF}}|} \sum_{u_R \in \mathcal{V}_{\text{REF}}} D^{(F_1, F_2^*)}(u_R),$$
(5)

where  $\mathcal{V}_{\text{REF}}$  is the set of reference nodes and  $D^{(F_1,F_2^*)}(u_R)$  is the  $L_2$  distance between  $u_R$ 's embedding vectors in  $F_1$  and  $F_2^*$ , respectively. The most desired case assumes a score equal to zero.

Assuming that the embedding algorithm introduces rotations, scaling, and translations, the alignment algorithm is performing perfectly if all the proposed measures are equal to zero.

#### C EXPERIMENTAL EVALUATION OF APMS

We compute the metrics mentioned in Section B for all aligned embeddings across all datasets and alignment algorithms. We observe that for the PED and SSD metrics we receive values close to Embedding alignment methods in dynamic networks

zero, which proves the alignment process preserves the information encoded in the node embeddings. In the case of the RND metric, we obtain values greater than zero, but after careful investigation, Woodstock '18, June 03-05, 2018, Woodstock, NY

we show that the distance between reference nodes decreases after alignment, respective to the targeted embedding (see: Figure 4).