Embedding alignment methods in dynamic networks

Kamil Tagowski[1 [0000−0003−4809−3587]], Piotr Bielak[1 [0000−0002−1487−2569]], and Tomasz Kajdanowicz[1 [0000−0002−8417−1012]]

Department of Computational Intelligence, Wroclaw University of Science and Technology, Poland
{kamil.tagowski, piotr.bielak, tomasz.kajdanowicz}@pwr.edu.pl

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.

Keywords: dynamic graphs · graph embedding · embedding alignment

1 Introduction

Node representation learning is pervasive across multiple applications, like social networks [13, 21], spatial networks [24, 25] or citation networks [9, 21]. The vast majority of node embedding methods are trained in an unsupervised manner, providing an automated way of discovering node representations for static networks. However, the body of knowledge for dynamic graph node embedding methods is rather unaddressed [4]. There are not many approaches to deal with real-world scenarios, where the structure of the network evolves and node embedding depends on such dynamics.

The embedding of dynamic graphs can be performed according to two scenarios: continuous and discrete-time approaches. The continuous approach 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 stochastic nature of representation learning algorithms. Embeddings of consecutive snapshots are algebraically incomparable due to the transformations

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

(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 linear 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 real-world datasets for link prediction and graph reconstruction tasks, we demonstrate that node embedding alignment is crucial and allows to increase performance up to 11 pp compared to not aligned embeddings.

We summarize our contributions as follows. (1) We formulate aligner performance measures (AMPs) for evaluating alignment algorithms, regardless of the downstream tasks. (2) We propose nine embedding alignment methods for graph. (3) 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.

This paper is structured as follows: in **Section 2** we discuss other work related to our topic. Then, we discuss applications of graph embedding alignment and emphasize its importance (**Section 3.2**). We also formulate aligner performance measures in **Section 3.3**. Next, we propose several methods for dynamic graph embedding alignment (**Section 3.4**) and evaluate them in downstream tasks as well as by means of introduced measures (**Section 4**). We conclude our work and point out future directions in **Section 5**.

2 Related works

The literature on static node embedding methods is very rich [4]. We can distinguish many approaches that are based on random-walks: DeepWalk [18], Node2vec [13], metapath2vec [9]; graph neural networks: GCN [14], GAT [23]; and matrix factorization: LLE [19], Laplacian Eigenmaps [1], HOPE [17]. Even though all of them are 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 [12], cross-graph alignment [5, 6, 8], dynamic graph embedding [3, 20, 22]. Embedding alignment techniques are often based on solving Orthogonal Procrustes problem to obtain a linear transformation between pairs of embeddings [6]. We can also distinguish approaches that utilize adversarial training [5, 8]. Dynamic Graph Embedding methods provide an embedding update mechanism for changes in the graph structure (appearing or disappearing nodes and edges). Embedding update may be performed in an online manner with the arrival of single events [15, 16] or with arrival of a new batch (graph snapshot) [3, 11, 20, 22]. In the tNodeEmbed [20] and LCF [22]

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

methods, alignment is achieved by solving the Orthogonal Procrustes problem using all common nodes to obtain the transformation matrix. In FILDNE [3], 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

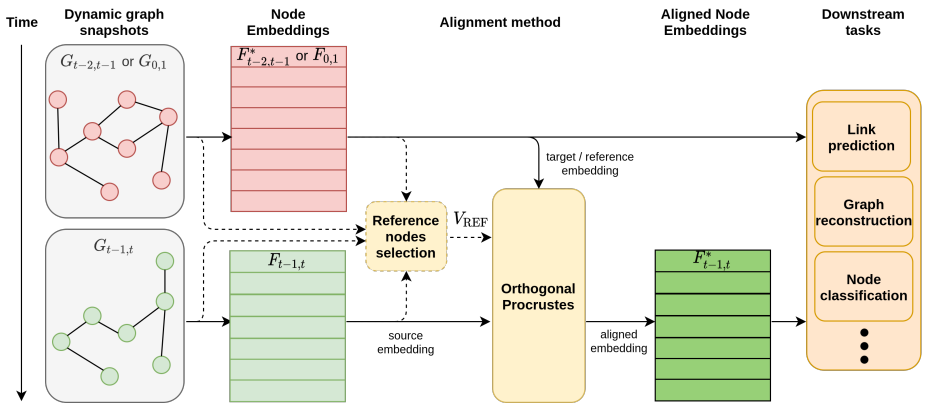

Fig. 1: Graph embedding alignment in the whole graph processing pipeline. For a dynamic graph in the form of snapshots, compute node embeddings, then (optionally) determine the reference nodes $\mathcal{V}_{\text{REF}}$ and align the newest embedding $F_{t-1,t}$ to a given target / reference embedding (*previous* one $F_{t-2,t-1}$, or the *first* one overall $F_{0,1}$). Such aligned embeddings can be used in downstream tasks, improving the performance compared to non-aligned embeddings.

3.1 Notation and problem statement

We denote a dynamic graph $G_{0,T}$ as a tuple $(\mathcal{V}_{0,T}, \mathcal{E}_{0,T})$, where $\mathcal{V}_{0,T}$ is the set of all nodes (vertices) observed between timestamp 0 and T, and $\mathcal{E}_{0,T}$ is the set of edges in the same timestamp range. We model such 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 $d, d << |\mathcal{V}|$, resulting in a node embedding matrix F, where each row represents an embedding of a single node.

An embedding alignment is a function $g : \mathbb{R}^{|\mathcal{V}| \times d} \to \mathbb{R}^{|\mathcal{V}| \times d}$ that transforms (aligns) a given node embedding matrix F respective to another one, producing an aligned node embedding matrix F^*. The method is trained on the observed change of embedding for a subset of nodes – called reference nodes – $\mathcal{V}_{\text{REF}} \subseteq \mathcal{V}$.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

3.2 The importance of embedding alignment

Node embedding methods capture the structure of graphs and encode it in low-dimensional representation vectors for every node. The final form of the embedding space and the actual positions of node vectors highly depend on the optimized cost function as well as the optimization procedure itself. For instance, 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 vector values, and (3) the order of node-context pairs used for training the Skip-gram model influence the final node vectors. It results in the situation that calculating the embedding for the same graph twice, with the same parameters of the embedding method, we observe that the node embeddings in the second run end up in different positions. Deformations of embeddings may be caused by a 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). Such transformations are examples of affine transformations, i.e., are composed of linear transformations and translations of the embedding space. In such situations, two embeddings of the same node are incomparable.

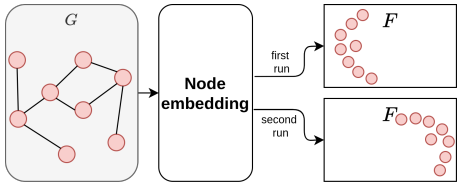

Fig. 2: Two runs of embedding calculation of the same graph.

The problem becomes fundamental in the case of dynamic graph representation learning, where embeddings are computed for every snapshot independently. In downstream tasks, these embeddings are often combined to obtain a representation for the whole dynamic graph, e.g., as in itebielak2020fildne. To obtain a rational combination of snapshots' embeddings, we are forced to align them. All rotations, scaling, and translations must be eliminated (see: Figure 3). Note that 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.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

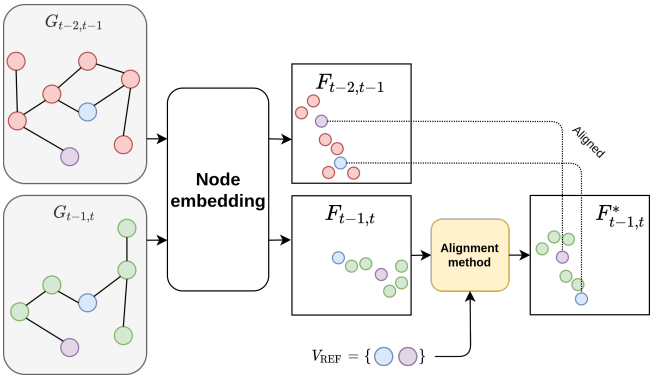

Fig. 3: Embedding of dynamic graph snapshots with alignment method applied.

3.3 Alignment performance measures

In this subsection, we introduce a novel set of alignment performance measures (APM) that constitute the criteria an alignment algorithm should meet. We 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 [18], Node2vec [13], metapath2vec [9]) 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**):

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

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

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

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.

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.

Besides translation and rotation, we also consider the scaling transformation, proposing the following APM. We propose the **Scaling Score Distance (SSD**; see Equation 2**)**.

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

$$\mathbf{SSD}(F, F^*) = |\frac{1}{|\mathcal{V}_{\text{REF}}| * (|\mathcal{V}_{\text{REF}}| - 1)} \sum_{\substack{(u_R, v_R) \in \mathcal{V}_{\text{REF}} \times \mathcal{V}_{\text{REF}} \\ u_R \neq v_R}} \frac{D^{(F^*)}(u_R, v_R)}{D^{(F)}(u_R, v_R)} - \frac{1}{|\mathcal{V} \backslash \mathcal{V}_{\text{REF}}| * (|\mathcal{V} \backslash \mathcal{V}_{\text{REF}}| - 1)} \sum_{\substack{(u,v) \in (\mathcal{V} \backslash \mathcal{V}_{\text{REF}}) \times (\mathcal{V} \backslash \mathcal{V}_{\text{REF}}) \\ u \neq v}} \frac{D^{(F^*)}(u, v)}{D^{(F)}(u, v)}|. \tag{2}$$

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 and we propose the **Reference Nodes Distance** (**RND**; see Equation 3).

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.*

$$\mathbf{RND}(F_1, F_2^*) = \frac{1}{|\mathcal{V}_{\text{REF}}|} \sum_{u_R \in \mathcal{V}_{\text{REF}}} D^{(F_1, F_2^*)}(u_R), \tag{3}$$

$\mathcal{V}_{\text{REF}}$ 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.

3.4 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)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

adding an auxiliary loss to the embedding 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 [20]. Thus, we omit computationally expensive methods based on neural networks, like [8]. In this section, we will present how the embedding alignment techniques work and how they select reference nodes.

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:

$$\underset{Q:Q^\intercal Q=I}{\operatorname{argmin}} ||BQ - A||_2^2 \tag{4}$$

It can be solved as $Q_{\text{opt}} = UW^\intercal$ with $U\Sigma W^\intercal$ being the Singular Value Decomposition (SVD) of $B^\intercal A$. In the case of dynamic graph embedding alignment, the A and B matrices are two node embeddings.

As we already 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 [3]. 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 the function presented in Equation 5 to finally obtain a nodes' score s (**scoring function**).

$$s(a_{t-1}^{(v)}, a_t^{(v)}) = |a_{t-1}^{(v)} - a_t^{(v)}| \left(\frac{\pi}{2} - \arctan(\max\{a_{t-1}^{(v)}, a_t^{(v)}\}) \right), \tag{5}$$

where $a_t^{(v)}$ is the v node's activity in snapshot $G_{t-1,t}$. This method is evaluated 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 from a body of knowledge related to complex network analysis, e.g. [26]. We postulate to measure the activity using: temporal betweenness **TB**, temporal closeness **TC**, temporal k-shell score **TK**, and temporal degree deviation **TDD**. See details of the measures in [26]. We choose L1-norm as the scoring function:

$$s(a_{t-1}^{(v)}, a_t^{(v)}) = |a_{t-1}^{(v)} - a_t^{(v)}| \tag{6}$$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

Both for the FILDNE and all temporal node-activity-based aligners, 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 [3] as the other ones can be equivalently applied. We select the top **p** percent of the lowest scores: $\text{select}(S, \mathcal{V}) = \mathcal{V}_{\text{REF}} \subseteq \text{sort}_S(\mathcal{V}),$ s.t. $|\mathcal{V}_{\text{REF}}| = \mathbf{p}|\mathcal{V}|$, where S are the node scores.

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

$$s(E_{t-1}^{(v)}, E_t^{(v)}) = 1 - \frac{|E_{t-1}^{(v)} \cap E_t^{(v)}|}{|E_{t-1}^{(v)} \cup E_t^{(v)}|}, \tag{7}$$

where $E_t^{(v)}$ is the set of edges in 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:

$$s(F_{t-1}^{(v)}, F_t^{(v)}) = 1 - \frac{|\mathcal{CN}_n(F_{t-1}^{(v)}) \cap \mathcal{CN}_n(F_t^{(v)})|}{|\mathcal{CN}_n(F_{t-1}^{(v)}) \cup \mathcal{CN}_n(F_t^{(v)})|}, \tag{8}$$

where $\mathcal{CN}_n(F_t^{(v)})$ is the set of the top n percent of closest neighbors of node v in the the embedding F_t.

4 Experiments

We evaluate all the proposed alignment methods on real-world datasets on two downstream tasks: link prediction and graph reconstruction. Moreover, we provide the analysis of the introduced Alignment Performance Measures. 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 Datasets

We perform experiments on nine real-world datasets (see Table 1). Each of them is split based on the timestamp frequency: daily (ia-hypertext), 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

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

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.

Table 1: Statistics of graph datasets. $|\mathcal{V}|$ - number of nodes, $|\mathcal{E}|$ - number of edges, **Directed** - whether the graph is directed or not

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	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	√	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

4.2 Node embeddings

To compute node embeddings we utilize the Node2vec method implemented in the PyTorch Geometric library [10]. We embed each snapshot separately. Using the Hyperopt optimizer [2] (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.

4.3 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, [3], exponential decaying, linear combination [22], or deep neural networks [11, 20]. We decided to choose the most computationally efficient one – averaging node embedding vectors from all snapshots.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

4.4 Embedding alignment

We evaluate all the proposed aligners (Section 3.4) 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 $p \in \{0.1, 0.2, \ldots, 0.9\}$ and (for ENJA aligner) the parameter n was evaluated for following values: $n \in \{0.1, 0.2, \ldots 1.0\}$.

4.5 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 2 (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 TB (fb-forum, fb-messages, ppi), EJA (radoslaw-email, bitcoin-alpha) and ENJA (enron-employees, bitcoin-otc).

4.6 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 $F_{0,T} = \mathrm{Avg}(F_{0,1}, \ldots, F_{T-1,T})$. We compute the mean Average Precision score (mAP) for graphs [7]. This metric captures local graph properties, i.e. for any node and its

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

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 [3] 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 2 (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 PA (bitcoin-otc, fb-messages, enron-employees, ppi) and TB (fb-forum, hypertext, radoslaw-email).

4.7 Impact of the node fraction taken in the alignment process

In Table 2 we only reported the best scores for a particular aligner. We performed also grid search over parameter p, see Section 4.4, 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 bitcoin-alpha, 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.

4.8 Embedding alignment performance metrics

We compute the metrics mentioned in Section 3.3 for all aligned embeddings across all datasets and alignment algorithms. We observe that for the PED and SSD metrics we receive values close to 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, we show that the distance between reference nodes decreases after alignment, respective to the targeted embedding (see: Figure 4).

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

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. Furthermore, embedding alignment improves the performance of downstream tasks up to 11 pp compared to the not aligned scenario. We plan to exploit other approaches of embedding alignment that are directed to edges and subgraphs.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

Table 2: Downstream task results (link prediction and graph reconstruction). **N/AL** denotes evaluation of not aligned embeddings and × denotes a scenario where embeddings could not be aligned due to missing reference nodes. We present values as the mean and standard deviations over 25 embedding retrains with gain (or loss) of aligned embeddings over not aligned ones (difference of mean values) in parenthesis. **Bold** values mark the best results for a single dataset. We performed a Friedman test with Nemenyi post-hoc to confirm that all alignment methods are statistically different from N/AL case. There were no statistically significant differences between alignment methods. The "∗" symbol denotes methods that are significantly worse than the best method.

	Aligner	bitcoin alpha	bitcoin otc	fb forum	fb messages	enron employees	hypertext	radoslaw email	ogbl collab	ppi
Link prediction (AUC)	N/AL	47.72 ± 14.44	70.19 ± 5.63 ∗	83.29 ± 2.80 ∗	59.50 ± 6.80 ∗	74.29 ± 3.95 ∗	87.99 ± 12.83	84.32 ± 1.34 ∗	81.24 ± 0.45 ∗	59.08 ± 0.80 ∗
	PA	$50.80 \pm 13.95_{(+3.08)}$	$79.37 \pm 2.83_{(+9.18)}$	$90.36 \pm 1.36_{(+7.07)}$	$64.46 \pm 8.33_{(+4.96)}$	$84.48 \pm 1.37_{(+10.19)}$	$87.56 \pm 6.31_{(-0.43)}$	$92.89 \pm 0.38_{(+8.57)}$ ∗	$\mathbf{82.48 \pm 0.53}_{(+1.24)}$	$60.22 \pm 0.78_{(+1.14)}$
	PUA	$46.77 \pm 14.17_{(-0.95)}$	$69.75 \pm 5.41_{(-0.44)}$ ∗	$83.72 \pm 2.53_{(+0.43)}$ ∗	×	×	×	$86.55 \pm 1.05_{(+2.23)}$ ∗	×	×
	FA	$50.91 \pm 14.78_{(+3.19)}$	$77.48 \pm 4.76_{(+7.29)}$	$90.47 \pm 1.28_{(+7.18)}$	$63.60 \pm 6.61_{(+4.10)}$	$83.09 \pm 1.39_{(+8.80)}$ ∗	$87.95 \pm 5.59_{(-0.04)}$	$92.96 \pm 0.38_{(+8.64)}$	$82.40 \pm 0.67_{(+1.16)}$	$60.26 \pm 0.74_{(+1.18)}$
	TB	$53.42 \pm 13.35_{(+5.70)}$	$78.91 \pm 3.75_{(+8.72)}$	$\mathbf{91.07 \pm 1.37}_{(+7.78)}$	$\mathbf{66.79 \pm 6.74}_{(+7.29)}$	$79.83 \pm 2.04_{(+5.54)}$ ∗	$88.95 \pm 4.68_{(+0.96)}$	$92.70 \pm 0.40_{(+8.38)}$ ∗	$82.19 \pm 0.53_{(+0.95)}$	$\mathbf{60.53 \pm 0.57}_{(+1.45)}$
	TC	$55.21 \pm 12.61_{(+7.49)}$	$78.02 \pm 3.75_{(+7.83)}$	$90.03 \pm 1.90_{(+6.74)}$	$64.85 \pm 7.26_{(+5.35)}$	$83.10 \pm 2.08_{(+8.81)}$ ∗	$90.37 \pm 4.45_{(+2.38)}$	$92.78 \pm 0.42_{(+8.46)}$ ∗	$82.03 \pm 0.51_{(+0.79)}$ ∗	$59.80 \pm 0.68_{(+0.72)}$ ∗
	TK	$56.91 \pm 12.77_{(+9.19)}$	$79.28 \pm 2.82_{(+9.09)}$	$90.71 \pm 1.64_{(+7.42)}$	$62.68 \pm 6.31_{(+3.18)}$	$83.50 \pm 0.97_{(+9.21)}$ ∗	$90.42 \pm 2.02_{(+2.43)}$	$92.93 \pm 0.37_{(+8.61)}$ ∗	$82.41 \pm 0.55_{(+1.17)}$	$60.11 \pm 0.80_{(+1.03)}$
	TDD	$56.91 \pm 12.77_{(+9.19)}$	$79.28 \pm 2.82_{(+9.09)}$	$90.71 \pm 1.64_{(+7.42)}$	$62.68 \pm 6.31_{(+3.18)}$	$83.50 \pm 0.97_{(+9.21)}$ ∗	$\mathbf{90.42 \pm 2.02}_{(+2.43)}$	$92.93 \pm 0.37_{(+8.61)}$ ∗	$82.41 \pm 0.55_{(+1.17)}$	$60.11 \pm 0.80_{(+1.03)}$
	EJA	$\mathbf{57.75 \pm 12.10}_{(+10.03)}$	$79.06 \pm 3.95_{(+8.87)}$	$89.64 \pm 1.73_{(+6.35)}$ ∗	$63.17 \pm 7.69_{(+3.67)}$	$84.46 \pm 1.25_{(+10.17)}$	$88.55 \pm 5.00_{(+0.56)}$	$\mathbf{93.23 \pm 0.31}_{(+8.91)}$	$82.42 \pm 0.65_{(+1.18)}$	$60.14 \pm 0.77_{(+1.06)}$
	ENJA	$56.60 \pm 12.06_{(+8.88)}$	$\mathbf{79.67 \pm 2.87}_{(+9.48)}$	$90.56 \pm 1.49_{(+7.27)}$	$64.70 \pm 7.93_{(+5.20)}$	$\mathbf{85.29 \pm 1.29}_{(+11.00)}$	$86.96 \pm 4.80_{(-1.03)}$	$93.02 \pm 0.43_{(+8.70)}$	$82.38 \pm 0.46_{(+1.14)}$	$60.26 \pm 0.75_{(+1.18)}$

	Aligner	bitcoin alpha	bitcoin otc	fb forum	fb messages	enron employees	hypertext	radoslaw email	ogbl collab	ppi
Graph Reconst. (mAP)	N/AL	25.18 ± 0.20 ∗	28.09 ± 0.36 ∗	11.30 ± 0.29 ∗	14.43 ± 0.19 ∗	44.53 ± 1.85 ∗	52.48 ± 1.15 ∗	38.85 ± 1.19 ∗	11.73 ± 0.05 ∗	8.90 ± 0.07 ∗
	PA	$27.40 \pm 0.25_{(+2.22)}$	$\mathbf{30.61 \pm 0.31}_{(+2.52)}$	$14.08 \pm 0.32_{(+2.78)}$ ∗	$\mathbf{16.05 \pm 0.24}_{(+1.62)}$	$\mathbf{52.53 \pm 0.90}_{(+8.00)}$	$57.46 \pm 0.74_{(+4.98)}$	$45.27 \pm 0.42_{(+6.42)}$ ∗	$\mathbf{11.99 \pm 0.04}_{(+0.26)}$	$\mathbf{9.43 \pm 0.05}_{(+0.53)}$
	PUA	$25.21 \pm 0.28_{(+0.03)}$ ∗	$28.23 \pm 0.39_{(+0.14)}$ ∗	$11.56 \pm 0.36_{(+0.26)}$ ∗	×	×	×	$42.94 \pm 0.59_{(+4.09)}$ ∗	×	×
	FA	$26.44 \pm 0.24_{(+1.26)}$ ∗	$30.44 \pm 0.35_{(+2.35)}$ ∗	$14.15 \pm 0.31_{(+2.85)}$	$15.90 \pm 0.23_{(+1.47)}$	$51.46 \pm 1.09_{(+6.93)}$ ∗	$59.04 \pm 0.93_{(+6.56)}$	$45.91 \pm 0.43_{(+7.06)}$ ∗	$11.97 \pm 0.05_{(+0.24)}$	$9.06 \pm 0.09_{(+0.16)}$ ∗
	TB	$27.34 \pm 0.28_{(+2.16)}$	$30.53 \pm 0.34_{(+2.44)}$	$\mathbf{14.35 \pm 0.38}_{(+3.05)}$	$15.98 \pm 0.25_{(+1.55)}$	$50.93 \pm 1.06_{(+6.40)}$ ∗	$\mathbf{60.42 \pm 0.87}_{(+7.94)}$	$\mathbf{46.67 \pm 0.41}_{(+7.82)}$	$11.88 \pm 0.04_{(+0.15)}$ ∗	$9.39 \pm 0.05_{(+0.49)}$
	TC	$\mathbf{27.48 \pm 0.26}_{(+2.30)}$	$30.53 \pm 0.32_{(+2.44)}$	$14.11 \pm 0.31_{(+2.81)}$	$15.76 \pm 0.23_{(+1.33)}$ ∗	$52.46 \pm 1.12_{(+7.93)}$	$55.76 \pm 0.88_{(+3.28)}$ ∗	$46.22 \pm 0.52_{(+7.37)}$	$11.85 \pm 0.04_{(+0.12)}$ ∗	$9.19 \pm 0.08_{(+0.29)}$ ∗
	TK	$26.70 \pm 0.31_{(+1.52)}$ ∗	$30.47 \pm 0.32_{(+2.38)}$	$13.97 \pm 0.32_{(+2.67)}$ ∗	$15.90 \pm 0.24_{(+1.47)}$ ∗	$52.16 \pm 1.07_{(+7.63)}$	$54.60 \pm 0.70_{(+2.12)}$ ∗	$44.64 \pm 0.58_{(+5.79)}$ ∗	$11.98 \pm 0.05_{(+0.25)}$	$9.39 \pm 0.05_{(+0.49)}$
	TDD	$26.70 \pm 0.31_{(+1.52)}$ ∗	$30.47 \pm 0.32_{(+2.38)}$ ∗	$13.97 \pm 0.32_{(+2.67)}$ ∗	$15.90 \pm 0.24_{(+1.47)}$ ∗	$52.16 \pm 1.07_{(+7.63)}$	$54.60 \pm 0.70_{(+2.12)}$ ∗	$44.64 \pm 0.58_{(+5.79)}$ ∗	$11.98 \pm 0.05_{(+0.25)}$	$9.39 \pm 0.05_{(+0.49)}$
	EJA	$26.70 \pm 0.30_{(+1.52)}$ ∗	$30.38 \pm 0.34_{(+2.29)}$ ∗	$13.84 \pm 0.30_{(+2.54)}$ ∗	$15.96 \pm 0.22_{(+1.53)}$	$51.10 \pm 1.08_{(+6.57)}$ ∗	$51.95 \pm 0.51_{(-0.53)}$ ∗	$46.04 \pm 0.42_{(+7.19)}$	$\mathbf{11.99 \pm 0.04}_{(+0.26)}$	$9.39 \pm 0.05_{(+0.49)}$
	ENJA	$26.70 \pm 0.25_{(+1.52)}$ ∗	$30.54 \pm 0.30_{(+2.45)}$	$14.10 \pm 0.29_{(+2.80)}$	$15.92 \pm 0.26_{(+1.49)}$ ∗	$51.70 \pm 0.87_{(+7.17)}$	$57.42 \pm 0.85_{(+4.94)}$ ∗	$46.16 \pm 0.57_{(+7.31)}$	$11.97 \pm 0.04_{(+0.24)}$	$9.42 \pm 0.05_{(+0.52)}$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

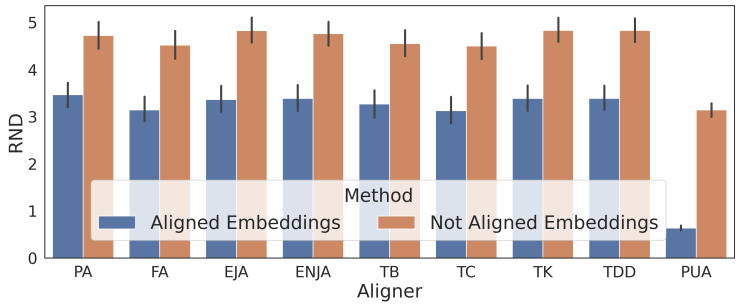

Fig. 4: Visualization of RND measure. We compare RND computed using: (1) non-aligned embedding and (2) aligned embeddings. We aggregate values across all datasets and show that in the RND for aligned embeddings are lower than for not aligned case. This indicates that the alignment is successful.

Acknowledgment

The project was partially supported by The National Science Centre, Poland, the research project no. 2016/21/D/ST6/02948, 2016/23/B/ST6/01735 and statutory funds of Department of Computational Intelligence.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6), 1373–1396 (2003)
2. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: International conference on machine learning. pp. 115–123 (2013)
3. Bielak, P., Tagowski, K., Falkiewicz, M., Kajdanowicz, T., Chawla, N.V.: Fildne: A framework for incremental learning of dynamic networks embeddings (2020)
4. Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., Murphy, K.: Machine learning on graphs: A model and comprehensive taxonomy. arXiv preprint arXiv:2005.03675 (2020)
5. Chen, C., Xie, W., Xu, T., Rong, Y., Huang, W., Ding, X., Huang, Y., Huang, J.: Unsupervised adversarial graph alignment with graph embedding. CoRR **abs/1907.00544** (2019), `http://arxiv.org/abs/1907.00544`
6. Chen, X., Heimann, M., Vahedian, F., Koutra, D.: CONE-Align: Consistent Network Alignment with Proximity-Preserving Node Embedding, p. 1985–1988. Association for Computing Machinery, New York, NY, USA (2020), `https://doi.org/10.1145/3340531.3412136`
7. De Sa, C., Gu, A., Ré, C., Sala, F.: Representation tradeoffs for hyperbolic embeddings. Proceedings of machine learning research **80**, 4460 (2018)
8. Derr, T., Karimi, H., Liu, X., Xu, J., Tang, J.: Deep adversarial network alignment. CoRR **abs/1902.10307** (2019), `http://arxiv.org/abs/1902.10307`
9. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 135–144 (2017)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48

10. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
11. Goyal, P., Chhetri, S.R., Canedo, A.: dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems **187**, 104816 (2020)
12. Grave, E., Joulin, A., Berthet, Q.: Unsupervised alignment of embeddings with wasserstein procrustes. In: Chaudhuri, K., Sugiyama, M. (eds.) Proceedings of Machine Learning Research. Proceedings of Machine Learning Research, vol. 89, pp. 1880–1890. PMLR (16–18 Apr 2019), http://proceedings.mlr.press/v89/grave19a.html
13. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
15. Lee, J.B., Nguyen, G., Rossi, R.A., Ahmed, N.K., Koh, E., Kim, S.: Dynamic node embeddings from edge streams (2019)
16. Ma, Y., Guo, Z., Ren, Z., Tang, J., Yin, D.: Streaming graph neural networks. In: Huang, J., Chang, Y., Cheng, X., Kamps, J., Murdock, V., Wen, J., Liu, Y. (eds.) Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020. pp. 719–728. ACM (2020). https://doi.org/10.1145/3397271.3401092
17. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1105–1114 (2016)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
19. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. science **290**(5500), 2323–2326 (2000)
20. Singer, U., Guy, I., Radinsky, K.: Node embedding over temporal graphs. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. pp. 4605–4612. AAAI Press (2019)
21. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. pp. 1067–1077 (2015)
22. Trivedi, P., Büyükçakır, A., Lin, Y., Qian, Y., Jin, D., Koutra, D.: On structural vs. proximity-based temporal node embeddings (2020)
23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
24. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019)
25. Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H.: Ge-gan: A novel deep learning framework for road traffic state estimation. Transportation Research Part C: Emerging Technologies **117**, 102635 (2020)
26. Yu, E.Y., Fu, Y., Chen, X., Xie, M., Chen, D.B.: Identifying critical nodes in temporal networks by network embedding. Scientific Reports **10**(1), 1–8 (2020)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_48