Continual representation learning for node classification in power-law graphs

# Continual Representation Learning for Node Classification in Power-Law Graphs

Gianfranco Lombardo[a], Agostino Poggi[a], Michele Tomaiuolo[a]

[a]*Department of Engineering and Architecture, University of Parma, Italy*

**Abstract**

The recent advent of node embedding techniques enabled a more efficient application of machine learning techniques on graphs. These techniques allow each node of a network to be encoded into an arbitrary low-dimensional vector representation, which can be exploited by statistical learning models. However, the main limitation of these approaches is that the embedding task is solved as an optimization problem on a static snapshot of the graph. In a real scenario, temporal dynamics should be considered with some consequences: new nodes might join the network and get a representation of only these new ones. As a consequence, a new training step over the entire graph is required. Even more, training models with static approaches can have resource-intensive requirements, especially when dealing with large networks. In light of this, a continual feature learning that builds on top of previously already learned knowledge (previous partial embedding of the network) and well-known properties can be a solution to address both limitations efficiently in real scenarios. Our approach is suitable for graphs whose degree distribution is described by a power-law function that is a common property of real systems. This research work presents three main scientific contributions: (a) a continual feature learning meta-algorithm for node embedding, which exploits properties of power-law distribution and spaces alignment techniques; It is suitable with any traditional node embedding techniques that relies on embedding spaces (b) we demonstrate empirically, by performing node labeling tasks, that a lightweight solution to encode new nodes, based on limited knowledge of the embedding of the network hub-nodes, can provide comparable or better performances, with respect to static approaches. (c) Finally, we experimented our algorithm in the temporal graphs domain and we achieved better results in node classification compared with other state of the art techniques.

*Preprint submitted to Future Generation Computer Systems* *October 13, 2021*

*Keywords:* Embedding, Node Embedding, Incremental learning, Continual Learning, Dynamic Networks, Representation learning

## 1. Introduction

Graphs (or networks) representations are fundamental in modeling many systems, particularly for expressing relational knowledge about interacting entities. Thus, they are ubiquitous in a wide range of disciplines, such as Computer Science, Sociology, Finance, Biology, and others. Moreover, in addition to being useful as knowledge repositories, these data structures also play a key role in Knowledge Discovery within the application of Machine Learning techniques. Several examples are available in the scientific literature of different research fields: in [1] the authors modeled the interaction between proteins as a network, to predict a correct label for each node, for describing its functionalities; in [2] the authors use a temporal network describing the US stock market to cluster correlated stocks' behaviors, in order to predict financial crises; in [3] an online community of patients is exploited to extract latent information about the emotional effects of their pathology. The main issue when applying machine learning on graphs is finding a way to encode structural information in the feature space required by learning models.

Representation learning is the branch of machine learning that aims to solve this problem by generalizing low-dimensional representations learning for data structures or sequences (e.g. text, graph). State of the art is learning these embeddings by solving an optimization problem that seeks to preserve local and global properties [4]. The first attempt in this field is the Neural language model [5] and its most famous breakthrough implementation for word embedding: Word2Vec [6].In this model, a shallow neural network is trained to predict the missing word given the context or the contrary of predicting the context given the word. The training task is an auxiliary task to get feature vectors associated with each word, named representation learning. In particular, Word2vec exploits the Skip-gram model to optimize the objective function using Stochastic Gradient Descent (SGD) as an iterative learning function and hierarchical Softmax. Both are used in the form of back-propagation on a single hidden layer feed-forward neural network with a linear activation function. The Skip-gram model has been adapted in the last years for node embedding in networks, by using streams of random walks instead of text corpora.

Although these approaches raised state-of-art benchmarks in both fields, they present a limitation when dealing with dynamical contexts. They are designed to work on a static context, for example, a snapshot of a graph at a certain time. In a real scenario, a new node might join the network or a word in a text might be unseen at training time. It is necessary to run the entire representation learning process again, to get a feature vector for these new elements. Addressing this issue in a dynamic context is still challenging and of increasing interest in Machine Learning, as demonstrated by the high number of papers published in the last two years.

In this research work, we propose a meta-algorithm that aims to learn features of new nodes incrementally joining a graph, by using a continual learning approach, partial embedding alignment and exploiting properties of power-law graphs. This approach is suitable for graphs whose degree distribution follows a power-law function. However, several works [7] have primarily demonstrated that real systems tend to present this distribution in their node degree. We demonstrate that it is possible to embed new data by learning features incrementally, building on top of previously learned knowledge and well-known properties, with comparable or even outperforming results with respect to the static methods, without training the whole model again. Moreover, we experimented our algorithm for Node Classification on temporal graphs achieving better results than other state of the art methods.

The paper is organized as follow: in Section 2, we present the current state of the art in this domain, in Section 3 we introduce some properties related to power-law graphs that are involved as an assumption in our algorithm. In Section 4, we describe the proposed solution, named WalkHub2vec and finally in Section 5 the experimental part, and the results are presented.

## 2. State of the art

The increasing interest in Representation Learning for graphs in the Machine Learning community is due to the idea that exploiting links among data in the form of a graph can increase learning performances for several tasks. In particular, the witnessed rapid growth in knowledge graph (KG) construction and the application made this task crucial in several domains. However, finding a way to incorporate information about the structure of a network into a statistical model is still challenging because of the heterogeneity of this information. For example, in the case of link prediction, one might want to focus more on structural properties to encode pairwise properties between nodes (e.g. common friends in a social network). On the contrary, in the case of node classification, one might want to preserve more global properties in the node description (e.g. nodes sharing many labels need to be similar). Moreover, in knowledge graphs, each edge is represented as a triple of the form (head entity, relation, tail entity), also called a fact, indicating that a specific relation connects two entities. Thus, the underlying symbolic nature of such triples usually makes this kind of graph hard to manipulate for machine learning tasks [8].

The idea behind the representation Learning is to learn a mapping function that embeds nodes as points in a low-dimensional vector space by solving a downstream task. Several approaches have been presented in the scientific literature, with multiple and separate lines of research [9] [10]. In knowledge graphs, the key idea is to embed components of a KG including entities and relations into this kind of continuous vector spaces, to learn features that preserve the inherent structure of the graph. Those embeddings can further be used to benefit all kinds of tasks, such as KG completion [11],[12], entity resolution [13], node classification [14], [15] and link prediction [16].

The State of the Art is represented by several methods that learn static node representations combining neural networks, random walks in graphs and the neural language model Skip-gram [17]. A classic approach to node embedding is DeepWalk [18], which captures second-order proximity. In this algorithm, random walks are modeled as sentences and later are fed into the Skip-gram model. Values from the hidden layers are the resulting node embedding vectors.

The LINE algorithm [19] addresses the efficiency issues, that previous approaches have when applied on large networks, by improving scalability with the use of negative sampling and asynchronous stochastic gradient descent to solve the optimization problem.

Finally, another relevant algorithm is Node2vec [20], which is an important modification of DeepWalk with significant performance gains. It introduces a parametrization of random walks generation by combining DFS and BFS-like behavior and also negative sampling. A common formulation of the optimization problem addressed by these algorithms can be summarized by Equation 1. where $f$ is the function that maps each node $u \in V$ in the vector space and $N_s(u)$ is the neighborhood of node $u$, sampled in different ways by the algorithms.

$$\max_f \sum_{u \in V} \log Pr(N_s(u)|f(u)) \tag{1}$$

However, although much progress has been reached in terms of performances, scalability over large networks is still problematic. It can become a limit even with just thousands of nodes when the network is dense. Additionally, the static nature of these methods can hardly apply to the analysis of real systems that are intrinsically dynamic.

In the last two years, several works have tried to address the issue of node embedding in a dynamic context, and this increasing production of different solutions highlights the importance of the topic both in academia and industry. The most used approach in the literature is based on modeling dynamics as a sequence of static snapshots.

The first algorithm, based on network snapshots, is the Continuous-time Dynamic Network Embedding (**CTDNE**) [21], proposed for the link prediction task. It relies on the concept of temporal random walks, using timestamps to preserve the time order of edges and thus their temporal dependencies. It can preserve temporal structural properties for link prediction, but it requires retraining at every change in the network.

Other successful snapshots-based works, relying on a different approach, are **DynGEM** [22], the one proposed in [23], **Dynnode2vec** [24] and **Dyngraph2vec** [25]. Dyngem and Dyngraph2vec estimate node representation with a deep autoencoder and an LSTM, respectively, by initializing network weights with the previously computed embedding at time $t-1$. Both algorithms address the problem of graph visualization and link prediction without node classification. Their main limitation is their use of their own learning model (PropSize), so they only work with this embedding method. The algorithm proposed in [23] is an extension of the Skip-gram model and, in particular of the LINE algorithm. Its interesting contribution is that when a new node joins the network, changes in the representations of other nodes are limited. In light of this, it computes a new representation with new walks only for a selected set of vertices. It offers impressive performance in several tasks, including a simple task of node classification, independently to the static algorithm used for random walk generation.

Finally, Dynnode2vec can be seen as a combination that takes the advantages of CTDNE and the algorithm in [23]. It uses the concept of evolving walk generation and computes new representation only for new nodes that join the graph in a particular timeframe by initializing the hidden weights of the neural network with the ones used to embed the snapshot at time $t-1$.

More recently, in [26], the authors proposed an algorithm, named Credit probing network embedding, that aims to compute node embedding for evolving networks with a partial monitoring of the network. The basic idea is to update each node vector dealing with it as a timestamps vector and to exploit a probing system to decide which node and when it should be updated. However, in this case, the evaluation task is represented by the link prediction task, and the source code is not currently available.

Finally, in [27], the authors propose **tNodeEmbed**, an algorithm based on Node2Vec that learns new nodes representation by aligning snapshots using Orthogonal Procustes [28] and then training an LSTM for a given task to optimize feature learning. It is not easy to assert which one is the best, because they are experimented on different datasets and with some gaps for what concerns other more difficult tasks like Node classification. The only tNodeEmbed takes into consideration the task of multi-class node classification, i.e., the benchmark task allowing a comparison with static approaches. This last algorithm reaches very interesting performance, but a complete comparison with the static case remains difficult. The authors experimented with tNodeEmbed on different datasets and the used parameters for embedding are unclear.

From direct experiments and the analysis of recent literature, we can assert that the idea of dynamic network embedding based on snapshot does not solve scalability issues and retraining is time-consuming, although less than the static case under certain conditions. In particular, these are important limitations when dealing with real networks that can often be large and dense. We have to be able to maintain previous embeddings and sometimes the set of previous walks whenever there is a change. It is possible to conclude that several issues and open problems are present in dynamic network embedding. However, considering the recent growth of this research line, all of the works reviewed in this section propose interesting approaches with important progress that trace a way to more generalized solutions.

In this work, we take into account specifically the multi-class node labeling task in light of *(i)* the reported lacks in state of the art, and *(ii)* the different issues which this case presents, with respect to the more studied case of link prediction.

## 3. Properties of power-law graphs

WalkHub2Vec founds its theoretical justifications in several properties of power-law graphs (Scale-free networks). The importance of this particular category of networks has been widely discussed in several research fields (E.g. [29],[30],[31]). The main reason is related to the fact that it fits the relationships among entities in many real systems, in various domains. The difference between a random network and a power law one lies in the probability distribution, which better describes a node's choice with a particular degree. In random networks, the randomness of links generation brings to a Poisson distribution and a lack of a component composed of nodes with a higher degree than other nodes, i.e., the hub nodes [7]. On the other hand, real systems are often better described by a power-law because only a few nodes present a high degree, while most of the remaining nodes have a small degree. When a system presents a power-law degree distribution, it means that the probability of randomly selecting a node with degree K is well approximated by Equation 2

$$P_k \sim k^{-\gamma} \tag{2}$$

This difference has an important consequence on how networks tend to grow. In power-law graphs, the evolution is well described by the Barabási-Albert Model [32]. This model assumes that in power-law graphs (or Scale-free graphs) the growth is described by a property called **Preferential Attachment**. This property claims that the probability $\Pi(k)$ of a new node to be connected with node $i$ depends on the degree $k_i$, as in Equation 3. That means that the growth in this kind of graph follows a probabilistic mechanism where a new node is free to connect to any node in the network, whether it is a hub or has a single link.

However, if a new node has a choice between a degree-two and a degree-four node, it is twice as likely that it connects to the degree-four node [7]. This behavior related to the power-law distribution is well known also as the Pareto principle, or 80/20 rule [33]. This rule can be summarized, saying that roughly 80% of the effects come from 20% of the causes. For example, as Pareto noticed in the 19th century, 20% of the population earned most of the money, while most of the population (80%) earned rather small amounts. Thus, it is likely that new incomes are destined to that 20% of people. Similarly, if a new node joins a network, it will likely be connected with the graph's 20% of hub nodes. This property relies on the long-tail of the power law distribution: in this kind of graphs, it is possible to observe that most nodes have a small degree while few nodes have a big degree (the so called long-tail). For this reason, as illustrated in [7] in power-law graphs is present a giant component that is composed by hub nodes (nodes with a high degree) that connect most of the normal nodes with a small degree.

$$\prod(k_i) = \frac{k_i}{\sum_j k_j} \tag{3}$$

## 4. WalkHub2Vec

In this section we describe the incremental feature learning algorithm for node embedding. In light of the properties summarized in Section 3, we take into account power-law graphs. First of all because WalkHub2Vec aims to be a solution to embed real systems graphs, thus this choice does not represent a limitation. Secondarily, to the best of our knowledge, these properties have not been taken into account by previous works. Finally, as we demonstrate in the result section, these properties represent a useful tool to address the dynamical case of a new node joining the network and in need for a vector representation. Also, we believe that an important part of the information required by the node classification can be directly found in the hubs component of each graph. The algorithm that we present is based on DeepWalk for simplicity, but it can be used with other embedding solution (e.g. Node2Vec or LINE) and also combining them in different moment. Additionally, being it based on a static embedding of a network, it is possible to retrieve past trained embeddings of a network to obtain the representation of new nodes without retraining. Every time a new node has to be embedded with respect to the previous static embedding, we seek to optimize a variant of the problem in Equation 1 specifically for incremental feature learning. This optimization maximizes the probability of observing a second-order neighborhood among the hubs component for the new node $i$, conditioned on an aligned feature representation between the target original space and a lightweight embedding of the hubs, or in other words the embedding of node $i$ with respect to the original embedding space:

$$\max Pr(N_H(i)|R \cdot f_H(i)) \tag{4}$$

Where $R$ is the rotational matrix necessary to align the drifted embedding space and is later discussed in detail. Being $A$ defined as the static algorithm chosen for the unique snapshot of the graph, WalkHub2Vec has the following steps:

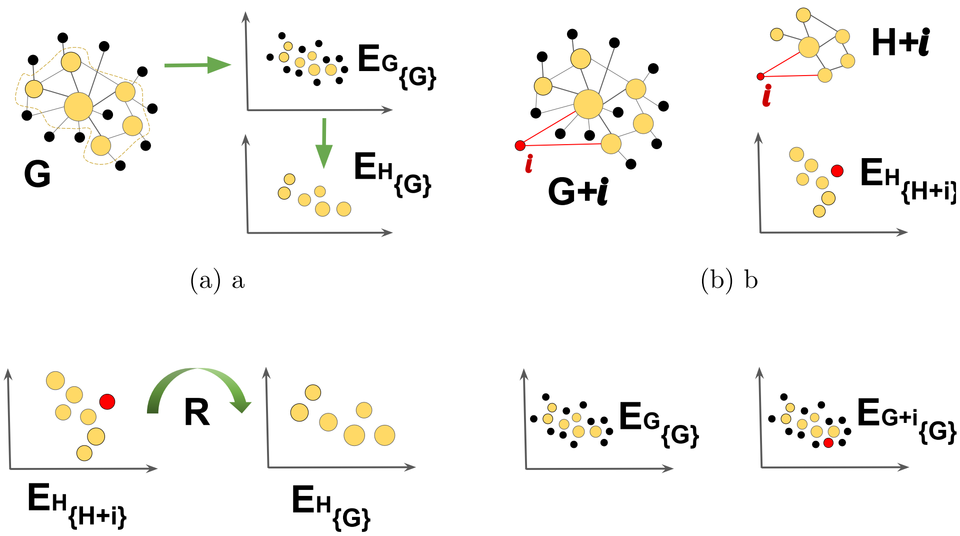
1. It computes the static embedding of the network $G$ with $A$ following the specific parameters of the considered algorithm and solving the general optimization problem in Equation 1. Given a graph $G$ we define also $E_{G_{\{G\}}}$ as the embedding of $G$ made with respect to $G$ itself.
2. The algorithm extracts the hub component of the network by selecting the sub-graph $H$ induced by the percentage $\lambda$ of nodes with the highest degree. The parameter $\lambda$ is set as default equal to 20 in light of the Pareto principle. For construction $E_{H_{\{G\}}} \subset E_{G_{\{G\}}}$ without any other computation (Figure 1a).
3. When a new node $i$ joins the network, it is considered a graph composed by the sub-graph $H$ plus the node $i$: $H + i$. In the rare case when the new node in the power-law graph has not links with the hubs, a single random walk from $i$ to one of the hubs is considered in the construction of $H + i$. Given the smaller network $H + i$ a lightweight embedding $E_{H_{\{H+i\}}}$ is computed with $A$ or an arbitrary algorithm (Figure 1b).
4. In order to get the embedding of node $i$ with respect to the original embedding space, the algorithm proceed to align $E_{H_{\{H+i\}}}$ with $E_{H_{\{G\}}}$ by learning the optimal rotation matrix. The alignment involves only nodes' representations who belong to the second-order neighborhood of $i$ in the hub component (Figure 1c). Alignment details are discussed later in 4.1.
5. Once the rotational matrix $R$ is computed, the embedding of $i$ with respect of the original embedding space is the dot product between $R$ and the embedding of $i$ with respect to $H + i$ space (Figure 1d).

### *4.1. The orthogonal Procrustes problem*

The orthogonal Procustes problem [28] is a matrix approximation technique that aims to learn the orthogonal rotational matrix $R$ which closely maps a matrix $A$ to a matrix $B$ (Equation 5). It is based on the Frobenius norm and a closed-form solution is provided with the Single Value Decomposition (SVD).

$$R = argmin_{\Omega} ||\Omega A - B||_F \;\; subject\, to \;\; \Omega\Omega^T \tag{5}$$

In WalkHubs2Vec, matrix $A$ is represented by the embedding of the second-order neighborhood of node $i$ in $E_{H_{\{H+i\}}}$ and respectively the correspondents




(a) a

(b) b

(c) c

(d) d

Figure 1: (a) First and second steps of the algorithm; The embedding of the hub component is extracted as a subset of the entire embedding $E_G$(b) Third step: A new node $i$ joins the network and a new embedding of only the $H$ sub-graph is computed; (c) The two embedding spaces have all nodes in common expect for the new node $i$. They have to be aligned to get $i$ coordinates with respect to the original space. Embedding alignments by learning the optimal rotation matrix; (d) Node $i$ with its resulting embedding in the original space

in $E_{H_{\{G\}}}$. In order to increase the quality of the alignment the two matrix are previously pre-processed following the full Procustes superimposition [34]. It requires that matrices, that can be seen as geometric shapes to be aligned, are previously uniformly scaled and translated to have the average value in the origin.

## 5. Results

In this section we provide an overview of the datasets and methods which we will use in our experiments. Code and data to reproduce our results are available on GitHub (see footnote 1). In order to evaluate WalkHubs2Vec and the related methodology, we performed three different kind of node labelling tasks: multi-label classification, multi-class classification and node classification with a dynamic graph. The difference between the first and the second is that in the first one each node can have more than one label at the same time and each one has to be correctly predicted. For both tasks, with the aim of a fair comparison between our method and the static techniques, we used the exact experimental procedure as in [35, 36], DeepWalk [18] and Node2Vec [20] and their common datasets. Finally, we evaluated WalkHubs2Vec in a dynamic context with temporal edges and nodes, making a comparison with CTDNE[21], tNodeEmbed [27] and DySAT [37]. For each task and experiment, we randomly selected a percentage of the labeled nodes between 10 and 90%, and use them as training data. The rest of the nodes are used as test set Specifically, we repeated 10 times, and we report the average performance in terms of both Macro-F1 and Micro-F1. The machine learning model used for all the tasks is a One-Vs-Rest logistic regression. [1]

### *5.1. Datasets*

For the multi-label classification we considered three networks: PPI and Wikipedia. We selected these networks as a benchmark because have also been used to evaluate Node2Vec and DeepWalk in their respective papers.

- **PPI**: Protein-Protein interactions for Homo Sapiens, is a network with 3,890 nodes, 76,584 edges and 50 different labels, where labels represent the biological states.

[1] https://github.com/gfl-datascience/walkHub2vec

- **Wikipedia**: This network is composed by 4,777 nodes with 40 different labels and 184,812 edges. It is a co-occurrence network of word appearing in the first million bytes of the Wikipedia dump. Labels are Part-of-Speech (POS) tags associated to each word.

- **Blogcatalog**: It is a social network of relationships among blogger authors on the BlogCatalog website. The network has 10,312 nodes, 333,983 edges and 39 different labels that represent blogger interest inferred through metadata.

For the multi-class classification task we used Cora, that is one of the most famous in literature for this task (E.g. [38],[39], [27]) with its various versions. It is a citation network composed by papers about different research areas of Computer Science and their citations. We used the largest version available with all of the nodes and the labels [40]. The network has 23,567 nodes, 93965 edges and 10 classes, which represent the topic of the articles. We exploited Cora also for the node classification task in dynamic graphs. We used temporal directed edges that represent citations from one paper to another, with timestamps of the citing paper's publication date as in [27]. We trained a starting embedding with papers between 1900 and 1990 and then we used papers until 1999 as new nodes and edges coming-up in the network. We used the same methodology also to evaluate CTDNE, tNodeEmbed and DySAT.

### *5.2. Experimental setup*

In light of the reasons explained in Section 2 about a necessary fair comparison, we used the same datasets used to evaluate Node2vec and DeepWalk in their respective papers. These datasets have not temporal information but this has not been a significant limitation for our evaluation, since we do not aim to solve only a temporal problem but also an incremental one, when a new generic node joins the network and when the entire graph is unknown at training time. Under this hypothesis, for the multi-label and multi-class tasks, we randomly selected the 10% of the nodes in each dataset to simulate their appearance ($I$ set). These nodes are sampled by a non-uniform distribution over the degree that makes nodes with smaller degree more likely to be sampled. This choice is also motivated by the idea that the leaf nodes with few links are often under evaluated by the sampling edges techniques used in the Skip-gram model. However, also Hub nodes can be sampled in the ($I$ set) although is less frequent that a node joins a network directly with an high degree. In order to prove the effectiveness of our methodology, we also treated the moments at which they join the network as totally independent events. In other words, we do not consider the possibly existing edges in the sub-graph induced by the $I$ set. The rest of the network represents the first static snapshot used as input for WalkHubs2Vec. As base for the meta-algorithm we used predominantly DeepWalk and in one case also Node2vec as demonstration that our algorithm and the methodology can deal with any node embedding technique. We also made a comparison with these two algorithms without the use of the meta-algorithm and when the graph is entirely processed with these algorithms. We used the same hyper-parameters proposed in the respective papers of DeepWalk and Node2Vec, without any optimization of these parameters. Each experiment is composed by 10 independent runs where each time nodes for the different sets are sampled.

- Dimension = 128
- Walk lenght = 10
- Number of walk = 80
- P and Q (Node2vec) both equal to zero to compare with DeepWalk

We also used in each case the parameter $\lambda = 20$, in this way the static network is divided into two groups: 20% of hubs and the remaining 80% unconsidered for the embedding.

### *5.3. Multi-label node classification*

In this section we report the results of the multi-label classification on PPI, Blogcatalog and Wikipedia. We experimented our algorithm with the previously introduced methodology by selecting different portions of nodes as training set, in the range between 10% and 90%. However, we report only results for the case of 50%, because we are interested in evaluating how features, computed incrementally on the 10% of the nodes of $I$ set, can affect the prediction results. This condition became more evident when we randomly selected large portions of the network in the 10 runs. Results are presented in Figure 2.

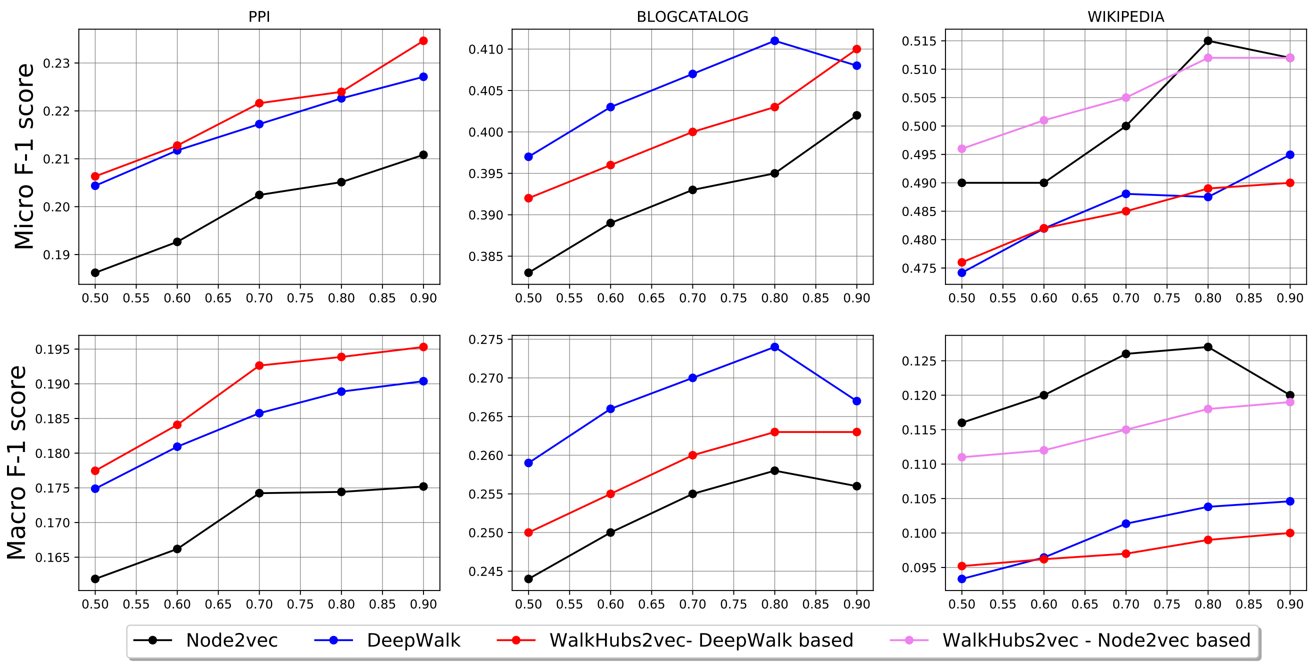



Figure 2: Performance evaluation of different benchmarks on varying amount of labeled data used for training. The x axis denotes the fraction of labeled data, whereas the y axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively.

### *5.3.1. Protein-Protein Interactions*

On this dataset, WalkHubs2Vec performs slightly better than Node2Vec and Deepwalk both with Micro and Macro F-1 score. In particular it is interesting the result when the selected percentage of training is the 90% and probably most of the nodes of $I$ set have been considered in the 10 runs.

### *5.3.2. BlogDatalog*

On this dataset, WalkHubs2Vec performs in a comparable way with DeepWalk with an average divergence of -0.004 and -0.009 in terms of respectively micro and macro F-1 score.

## *5.4. Wikipedia*

Unlike the two previous networks, on Wikipedia Node2Vec performs in general better than DeepWalk. In light of this, we present results obtained with WalkHubs2Vec based on DeepWalk but also based on Node2Vec. In both cases WalkHub2Vec performs in a comparable way than the static techniques. The average divergence between Node2vec and the incremental case is 0.003 and -0.006 for micro and macro F-1 respectively. As expected, the version of WalkHubs2Vec based on Node2Vec performs bettern than the one based on DeepWalk as reflection of the static performance of the algorithms.

### *5.5. Comparative analysis for multi-label classification*

Several features of the selected datasets should be considered, to get an overall evaluation of our algorithm in the multi-label classification task. First of all, each network presents a power-law distribution for node degree, but Wikipedia graph has a more significant cut-off than the others. Practically, on this dataset, the power-law distribution is more acceptable after a certain degree (around 10), while before this value, a log-normal distribution is more evident. In our opinion, this fact is what makes the difference with results we obtained with the PPI dataset since the density of the two graphs is quite similar (PPI: 0.005, Wikipedia: 0.003) and the average node degree (PPI: 19, Wikipedia: 15). Another critical factor is how the topological structure affects the results and how the number of labels that have to be predicted. We analyzed this aspect more by comparing the three networks in terms of modularity for community detection and connected components. In PPI, where we have better results than the other algorithms, the number of labels is 50, modularity is 0.337, and it is possible to find 43 different clusters (or communities) and 35 connected components. That means that nodes tend to arrange in groups with the same labels, and thus, the classification task results to be more simple. In the Wikipedia network, with a Modularity equal to 0.2, we found only 12 clusters, a single connected component with 40 different labels that have to be predicted. Finally, Blogcatalog resulted in a more difficult case: with a Modularity equal to 0.24 we found 7 different clusters, one single connected component when the number of labels is 39. In light of this, we hypothesize that exploiting the power-law properties can make a significant difference when the labels reflect also the topological structure of the network, and on the other hand, have comparable results in the other cases. However, in these last cases, having comparable results by using only the 20 percent of nodes to compute the embedding is an interesting result that should be more analyzed in future works.

### 5.6. Multi-class node classification

In this section, we report the results of the multi-class classification on Cora with WalkHubs2Vec based on DeepWalk with the static case where the entire network is embedded with Node2Vec and DeepWalk. In this case, WalkHubs2Vec outperforms the baseline methods both in micro and macro F-1 score. The average gain of WalkHubs2Vec is about 0.01 in both metrics. Results are presented in Figure 3.

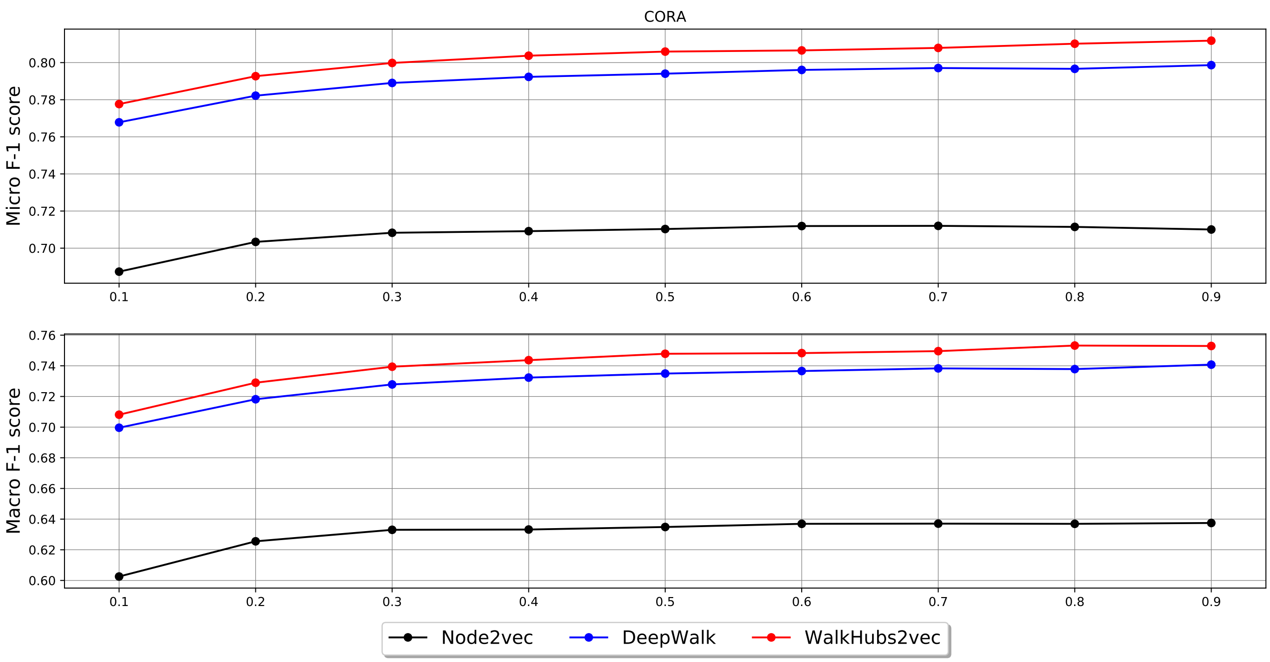



Figure 3: Performance evaluation on Cora of different benchmarks on varying the amount of labeled data used for training. The x axis denotes the fraction of labeled data, whereas the y axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively.

### 5.7. Node classification with temporal graphs

WalkHubs2Vec exploits a continual learning approach since it is designed to learn a representation of nodes and parts of a graph that are not entirely known at training time. For this reason, we experimented with the algorithm in the most common scenario of temporal graphs. Indeed, in this context, the graph structure is dynamic with timestamps associated with each edge

| | Micro f-1 score | Macro f-1 score |
|---|---|---|
| CTDNE | 0.7194 | 0.6397 |
| DySAT | 0.4828 | 0.1312 |
| tNodeEmbed | 0.7740 | 0.7078 |
| **WalkHubs2Vec** | 0.8101 | 0.7532 |

Table 1: Performance results of WalkHubs2Vec vs. baselines for the node classification task over the dynamic Cora dataset.

that determines the moment an edge comes up to be part of the graph. As a consequence, new nodes may join the network.
The Cora dataset exploited in the previous section can be used also to model a temporal scenario by using as timestamps the publication date of the papers, as already presented in [27]. We experimented WalkHubs2Vec on this dataset using as baselines three algorithm designed for temporal graph embedding: tNodeEmbedding [27], CTDNE [21] and DySAT [37]. These three algorithms have been already presented in the Literature review 2.
We used papers from 1900 until 1989 to create the starting graph and then the remaining ten years as new nodes and edges that join the network. In the starting graph, there are 21974 nodes and 64991 edges. Temporal items are 1593 new nodes and 28975 new edges. We computed the embedding representation for each node and then we performed a classification task to predict one of the ten possible labels. Classification has been performed using the One-Vs-Rest Logistic classifier as in the previous experiments except for tNodeEmbed because it learns the embedding while dealing with a classification task performed with an LSTM deep neural network. We evaluated WalkHubs2Vec and the baselines in terms of Micro-F1 and Macro-F1 scores. Table 1 shows the results we achieved on this task. WalkHubs2Vec outperforms the baselines both in micro and macro f1-score. All the experiments have been repeated for 10 runs and the results represent the average score.

CTDNE, DySAT and tNodeEmbedding exploits Node2Vec as node embedding engine for every single snapshot, for this reason, we compared our algorithm using the same embedding technique. Since Node2Vec is the basis of all the algorithms, we selected the same embedding hyper-parameters for all and they are the same of the previous sections and from [20]. We did not optimize the other parameters available within the algorithms and this can represent a reason why in particular DySAT under-performs remarkably with respect of the other algorithms. DySAT and tNodeEmbed train neural network-based models while performing representation learning of nodes and for this reason, the number of epochs represents an important parameter. We decided to train CTDNE, tNodeEmbed, and DySAT for 50 epochs, because over this number we experimented with some computational issues right with DySAT. Another thing that we desire to report is that DySAT is designed to learn the distribution of edges to perform only link prediction although it computes the temporal embedding of each node.

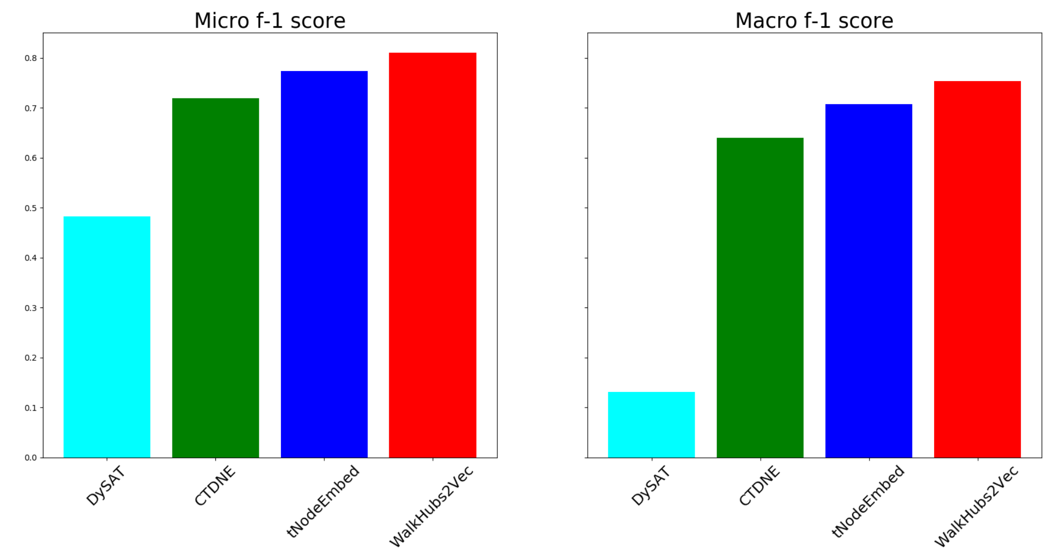



Figure 4: Performance evaluation between WalkHubs2Vec and the baselines on the temporal Cora dataset.

It is interesting the comparison with tNodeEmbed because a common aspect with WalkHubs2Vec is the idea of embedding alignment, that in the first one is computed over the entire graph while in our algorithm is performed considering only the giant component. Moreover, tNodeEmbed has been evaluated with its built-in deep neural network for the classification task and not with the one-vs-rest logistic classifier that is a less powerful model. This last aspect leads us to the idea that considering similarities between the two approaches and the theoretic advantage of the LSTM neural network for tNodeEmbed, one possible reason for our better results can rely exactly upon the importance of making attention to the hubs of the network while learning the new nodes' representation. We plan to further investigate this aspect in our future works in order to quantify this importance.

## 6. Conclusions

In this research work we have addressed the problem of continual feature learning for node embedding in power-law graphs. The work is focused on the idea of taking advantages of the scale-free property of this category of graphs, to compute embedding of new nodes without retraining the learning model in the node classification context. We propose also an implementation of this methodology, WalkHubs2Vec, that can deal with any embedding techniques. Results demonstrate how the combination of partial embeddings based on the hubs component and embedding alignment can solve the problem with good results in terms of features quality for the new nodes. WalkHubs2Vec reaches equal or slightly better results when compared to well-known static methods, as Node2vec and DeepWalk and better results when compared with state-of-the-art techniques for dynamic graph embedding. Future developments are related to performing different tasks with this methodology (e.g. link prediction). Moreover, it would be interesting to use a similar approach also in word embedding and other embedding contexts.

## References

[1] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, et al., A large-scale evaluation of computational protein function prediction, Nature methods 10 (2013) 221.

[2] A. Kocheturov, M. Batsyn, P. M. Pardalos, Dynamics of cluster structures in a financial market network, Physica A: Statistical Mechanics and its Applications 413 (2014) 523–533.

[3] G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, A combined approach for the analysis of support groups on facebook-the case of patients of hidradenitis suppurativa, Multimedia Tools and Applications 78 (2019) 3321–3339.

[4] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (2013) 1798–1828.

[5] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, Journal of machine learning research 3 (2003) 1137–1155.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[7] A.-L. Barabási, M. Pósfai, Network science, Cambridge University Press, 2016. URL: http://barabasi.com/networksciencebook/.

[8] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering 29 (2017) 2724–2743.

[9] W. L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, arXiv preprint arXiv:1709.05584 (2017).

[10] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Transactions on Knowledge and Data Engineering 31 (2018) 833–852.

[11] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes., in: Aaai, volume 14, 2014, pp. 1112–1119.

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in neural information processing systems, 2013, pp. 2787–2795.

[13] A. Bordes, X. Glorot, J. Weston, Y. Bengio, A semantic matching energy function for learning with multi-relational data, Machine Learning 94 (2014) 233–259.

[14] M. Nickel, V. Tresp, H.-P. Kriegel, Factorizing yago: scalable machine learning for linked data, in: Proceedings of the 21st international conference on World Wide Web, 2012, pp. 271–280.

[15] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data., in: Icml, volume 11, 2011, pp. 809–816.

[16] Y. Tay, A. Luu, S. C. Hui, Non-parametric estimation of multiple embeddings for link prediction on dynamic knowledge graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[17] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[18] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 701–710.

[19] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[20] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2016, pp. 855–864.

[21] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, S. Kim, Continuous-time dynamic network embeddings, in: Companion Proceedings of the The Web Conference 2018, International World Wide Web Conferences Steering Committee, 2018, pp. 969–976.

[22] P. Goyal, N. Kamra, X. He, Y. Liu, Dyngem: Deep embedding method for dynamic graphs, arXiv preprint arXiv:1805.11273 (2018).

[23] L. Du, Y. Wang, G. Song, Z. Lu, J. Wang, Dynamic network embedding: An extended approach for skip-gram based network embedding., in: IJCAI, 2018, pp. 2086–2092.

[24] S. Mahdavi, S. Khoshraftar, A. An, dynnode2vec: Scalable dynamic network embedding, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 3762–3765.

[25] P. Goyal, S. R. Chhetri, A. Canedo, dyngraph2vec: Capturing network dynamics using dynamic graph representation learning, Knowledge-Based Systems (2019) 104816.

[26] Y. Han, J. Tang, Q. Chen, Network embedding under partial monitoring for evolving networks., in: IJCAI, 2019, pp. 2463–2469.

[27] U. Singer, I. Guy, K. Radinsky, Node embedding over temporal graphs, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 4605–4612. URL: `https://doi.org/10.24963/ijcai.2019/640`. doi:10.24963/ijcai.2019/640.

[28] J. R. Hurley, R. B. Cattell, The procrustes program: Producing direct rotation to test a hypothesized factor structure, Behavioral science 7 (1962) 258–262.

[29] M. C. Gonzalez, C. A. Hidalgo, A.-L. Barabasi, Understanding individual human mobility patterns, nature 453 (2008) 779.

[30] R. Albert, Scale-free networks in cell biology, Journal of cell science 118 (2005) 4947–4957.

[31] V. Boginski, S. Butenko, P. M. Pardalos, Statistical analysis of financial networks, Computational statistics & data analysis 48 (2005) 431–443.

[32] A.-L. Barabási, E. Bonabeau, Scale-free networks, Scientific american 288 (2003) 60–69.

[33] R. Dunford, Q. Su, E. Tamang, The pareto principle (2014).

[34] I. DRYDEN, Kv: Statistical shape analysis, John Willey, New York (1999).

[35] L. Tang, H. Liu, Relational learning via latent social dimensions, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 817–826.

[36] L. Tang, H. Liu, Scalable learning of collective behavior based on sparse social dimensions, in: Proceedings of the 18th ACM conference on Information and knowledge management, ACM, 2009, pp. 1107–1116.

[37] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, Dysat: Deep neural representation learning on dynamic graphs via self-attention networks,

in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 519–527.

[38] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[39] S. Nandanwar, M. N. Murty, Structural neighborhood based classification of nodes in a network, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 1085–1094.

[40] A. K. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning, Information Retrieval 3 (2000) 127–163.