2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)

LASAGNE: Locality And Structure Aware Graph Node Embedding

Evgeniy Faerman*, Felix Borutta*, Kimon Fountoulakis† and Michael W. Mahoney‡
*Ludwig-Maximilians-Universität München
faerman@dbs.ifi.lmu.de, borutta@dbs.ifi.lmu.de
†University of Waterloo
kimon.fountoulakis@uwaterloo.ca
‡University of California at Berkeley
mmahoney@stat.berkeley.edu

***Abstract*—In this work we propose LASAGNE, a methodology to learn locality and structure aware graph node embeddings in an unsupervised way. In particular, we show that the performance of existing random-walk based approaches depends strongly on the structural properties of the graph, e.g., the size of the graph, whether the graph has a flat or upward-sloping Network Community Profile (NCP), whether the graph is expander-like, whether the classes of interest are more k-core-like or more peripheral, etc. For larger graphs with flat NCPs that are strongly expander-like, existing methods lead to random walks that expand rapidly, touching many dissimilar nodes, thereby leading to lower-quality vector representations that are less useful for downstream tasks. Rather than relying on global random walks or neighbors within fixed hop distances, LASAGNE exploits strongly local *Approximate Personalized PageRank* stationary distributions to more precisely engineer local information into node embeddings. This leads, in particular, to more meaningful and more useful vector representations of nodes in poorly-structured graphs. We show that LASAGNE leads to significant improvement in downstream multi-label classification for larger graphs with flat NCPs and that it is comparable for smaller graphs with upward-sloping NCPs.**

I. Introduction

Graphs are a common way to describe interactions between entities. The entities are modeled as nodes, and the interactions between pairs of entities are represented by edges between nodes. Describing nodes of a graph as low dimensional vectors has the advantage that many popular machine learning algorithms can be automatically applied, and it is applicable in many areas like visualization, link prediction, classification, etc. Motivated by this, representation learning methods for graph vertices, e.g., [24], [28], [12], [8], [17], [1], [30], [6], focus on learning vectors to represent information in neighborhoods around a node, e.g., nodes within a short geodesic distance or nodes encountered in random walks starting at a given node.

Somewhat more formally, let $G = (V, E)$ be a graph, with $V = \{v_1, \dots v_N\}$ being the set of nodes and $E = \{e \mid e \in V \times V\}$ being the set of (undirected) edges. The general goal is to find a vector embedding or latent representation for each node v_i such that the resulting set of embedded nodes $\mathcal{E} = \{f(v_i) | v_i \in V\}$ in the d-dimensional vector space $\mathbb{R}^d$ still reflects structural properties of G. For instance, such structural properties could be the similarity of the neighborhoods of two nodes v_i and v_j. The neighborhood $\mathcal{N}(v)$ of a node v is defined as the set of nodes having the highest probabilities to be visited by a random walk starting from node v, a geodesic walk starting from v, or some other related process. This means if $\mathcal{N}(v_i) \approx \mathcal{N}(v_j)$ holds in the original graph, it should also hold that $f(v_i) \approx f(v_j)$ in $\mathbb{R}^d$.

The intuition behind these representation learning methods is that nodes having similar neighborhoods are similar to each other, and thus one can use information in the neighbors of a node to make predictions for a given node. Defining the right neighborhood for each node, however, is a challenging task. For example, in unsupervised multi-label classification, the labels of the nodes define the underlying local structure for a particular class, but often this does not necessarily overlap significantly with the local structure defined by the edge connectivity of the graph. Alternatively, realistic graphs typically have large-scale properties that are very poorly structured with respect to the behavior of random walks [20], [22], [21], [15], [2], [3].

The basic assumption of random walk based methods and, of course, the large body of very related methods based on spectral graph theory is that nodes visited more often than others by random walks starting from a particular node are also more useful to describe that node in terms of downstream prediction tasks. However, the problem with random walks is that typically most of the graph can be reached within a few steps, and thus information about where the random walk began (which is the node for which these methods are computing the embedding) is quickly lost.

This issue is particularly problematic for extremely sparse graphs with upward-sloping Network Community Profiles (NCPs) [20], [22], [21] and for flat NCPs [15] (expander-like graphs) or deep k-cores [2], [3]. These properties are ubiquitous among realistic social and information networks. This suggests that, unless carefully engineered, embedding methods based on random walks will perform sub-optimally, since the random walks will mix rapidly, thereby degrading the local information that one hopes they identify.

We explore these issues, and we present a method which takes into account the local neighborhood structure of each node in the graph individually. This leads to insight into how to better exploit graph topology in poorly structured graphs, and it can result in improved embedding vectors. In opposite to random walk based methods, which are likely to get stuck in dense areas of a graph, our approach furthermore better ex-

978-1-5386-7325-6/18/$31.00 ©2018 IEEE
DOI 10.1109/WI.2018.00-83

ploits the local neighborhood structure for peripherical nodes and finally the subsequent learning procedure is not biased towards concentrating on optimizing the representations with preference to nodes within deep k-cores.

Our method, LASAGNE, is an unsupervised algorithm for learning locality and structure aware graph node embeddings. It uses an *Approximate Personalized PageRank* vector [4] to adapt and improve state-of-the-art methods for determining the importance of the nodes in a graph from a specific node's point of view. The proposed methodology is easily parallelizable, even on distributed environments, and it has even been shown that the methods we adapt were applied to graphs with more than billions of nodes on a single machine [27].

We evaluate our algorithm with multi-label classification on several real-world datasets from different domains under real-life conditions. Our evaluations show that our algorithm achieves better results – especially for downstream machine learning tasks whose objectives are sensitive to local information – in terms of prediction accuracy than the state-of-the-art methods. As has been described previously [22], [15], [2], [3], and as we review in Section IV, graphs with flat NCPs and many deep k-core nodes have local structure that is particularly difficult to identify and exploit. Importantly, our empirical results for this class of graphs are substantially improved, relative to previous methods. This illustrates that, by carefully engineering locally-biased information into node embeddings, one can obtain improved results even for this class of graphs, without sacrificing quality on other less poorly-structured graphs.

We also illustrate several reasons why random walk based methods do not perform as expected in practice, justifying our interpretation that our method leads to improved results due to the manner in which we engineer locality.

II. PRELIMINARIES

A. Related Work on Node Embedding

Most recent representation learning methods for graph vertices construct vectors on the basis of local neighborhood information [24], [28], [8], [12], [17], [1], [30], [6]. The unsupervised *DeepWalk* algorithm learns latent representations for graph vertices by using multiple random walks [24]; and it then applies the *Skip-gram* model, originating from natural language processing, to the sequences of nodes given by the random walks. Grover et al. presented the so-called *node2vec* method [12]. Instead of using a random search strategy, *node2vec* introduces two hyperparameters to use second order random walks in order to bias the random walks towards a particular search strategy. The *GraRep* algorithm takes this a step further and computes a sequence of matrices, random walk transition matrix taken to powers ranging from 1 to k, and then applies SVD to them [8]. Abu-El-Haija et al. propose matrix factorization of random-walk occurrences matrix with different approaches to determine the context window size distribution [1]. The *LINE* algorithm learns two different representations [28], the first of which encourages two nodes to have close embeddings when they are directly connected, and the second of which encourages two nodes to be close when they share the same direct neighbors. The SDNE method has similar objectives and additionally uses deep architectures to capture non-linearities within graphs [30]. Variational Graph Auto-Encoders also force directly connected vertices to have similar representations but use a different architecture [17]. The method was originally proposed for attributed graphs, but can also be applied to learn embeddings of non-attributed graphs. Graph2Gauss goes one step further and ranks neighbors according to hop distance [6]. Representations of one-hop neighbors are more similar to the node's own representation than the representations of two-hop neighbors, and two-hop representations are more similar than three-hop representations and so on. The struc2vec method attempts to learn similar representations for nodes having similar role in the graph. Nodes with similar roles do not have to be related locally [25]. Yang et al. proposed a semi-supervised learning technique which uses labels for embedding learning [31]. Further related approaches relying on supervised learning methods, e.g., convolutional approaches like [16], have been presented recently. Most of them can be summarized in message passing neural network approaches as shown in [10]. However, these approaches are supervised and hence not further focussed in this work.

B. Embedding Words with Word2vec

Word2vec [23], [19] is a framework for learning word representations in some vector space by simultaneously preserving the words' semantic meaning. The representations are learned based on some contexts so that embeddings sharing similar contexts end up close to each other in the learned space. The embeddings are learned by maximizing the prediction probability of the contexts given the input embeddings, i.e., *Skip-gram* model. Note, that the model assumes independence of different contexts from each other for the same input. *Negative sampling* is used to estimate the prediction probability during the training. It maximizes the log probability of the input's context by simultaneously minimizing the prediction probability for k randomly selected contexts. Furthermore logistic regression is used to estimate the prediction probability:

$$\log \sigma(v_I^T v_{c_i}) + \sum_{j=1}^{k} \mathbb{E}_{w_{j \sim P_n(w)}} \log \sigma(-v_I^T v_j).$$

For each word the model maintains two representations, embedding and context representation. The vector v_I denotes the embedding representation of the input, v_{c_i} is the context representation and v_j are representations of randomly selected contexts. The stochastic gradient descent algorithm is used for model optimization. An analysis of this *word2vec* method has been provided by [11], reflecting a perspective similar to ours.

C. Approximate Personalized PageRank

The *PageRank* algorithm computes an "importance" score for every node in some graph. Each of the scores corresponds to the probability of a "random surfer" to visit a node given

Algorithm 1 LASAGNE ApproximatePPR

Input: Node s, teleportation parameter α, Probability significance threshold δ
Output: APPR vector p

```
 1: p = 0⃗, r = 0⃗, heap=heap()
 2: r(s) = 1
 3: heap.push((s,1))
 4: sumProbUpdates = 0
 5: lastDistrUpdate = 1
 6: while lastDistrUpdate > δ do
 7:     u = heap.pop()
 8:     probUpdate = (2α/(1 + α))r(u)
 9:     if u ≠ s then
10:         sumProbUpdates += probUpdate
11:         lastDistrUpdate = probUpdate / sumProbUpdates
12:     end if
13:     p(u) = p(u) + probUpdate
14:     neighResUpdate = ((1 − α)/(1 + α))r(u)/d(u)
15:     for v with (u,v)∈ E do
16:         r(v) = r(v) + neighResUpdate
17:         heap.update((v, r(v)/size(v.neighbours)))
18:     end for
19:     r(u) = 0
20: end while
21: p(s) = 0
22: p(s) = max(p)
23: return p
```

some start distribution. The *PageRank* vector is the solution of the linear system:

$$pr(s) = \alpha s + (1 - \alpha) pr(s) W, \tag{1}$$

with $W = D^{-1}A$ being the random walk transition matrix. A is the adjacency matrix, D is the degree matrix having the node degrees on the diagonal. The constant α is the *teleportation* probability. The starting nodes or more specifically the probability for each node to be the starting point of a random walk are given by the vector s. A variant of *PageRank* is the *Personalized PageRank* (PPR) whose result corresponds to the result of the *PageRank* algorithm, where the probabilities in the starting vector s are biased towards some set of nodes. The *push* algorithm described in [14], [5], [4] is used to compute an *Approximate Personalized PageRank* (APPR) vector in a more efficient way if the start distribution vector s is sparse, i.e., has probability mass on only a few nodes. The idea behind the *push* algorithm is to propagate a node's probability locally and only if there is a sufficient amount of probability to update. This leads to a sparse solution which means that only relatively few nodes of the underlying graph are contained in the resulting APPR vector.[1]

III. LASAGNE: LOCALITY AND STRUCTURE AWARE GRAPH NODE EMBEDDING

The LASAGNE algorithm consists of two steps: a preprocessing step, which computes the APPR vectors for each node; and the learning step, which uses the APPR vectors to generate training examples batchwise to learn the final embeddings.

A. Approximate Personalized PageRank for Node Embeddings

The computation of the APPR vectors for the node embeddings is described in Algorithm 1. There are two main modifications compared to the original method in [4]. The first is the assignment of probability mass to the *seed node* in its own APPR vector, and the second is a modification of the stopping criterion.[2]

The first modification allows the seed node to be considered as its own neighbor during sampling the training examples. Consequently, the seed node is considered to be similar to other nodes that have the seed node among their neighbors, which in turn leads to higher proximity of such nodes in the embedded space. To avoid each node being considered to be the most important member of its own neighborhood (and thus being overrepresented during the training phase), we replace the node's own entry in its APPR vector with the second highest probability, c.f., line 21 - 22.

The second modification is an adaptation of the stopping criterion since our main motivation is not to approximate the PPR vector but instead to keep only the *relevant* neighbors that represent a meaningful context for the seed node. The idea is to avoid considering neighbors which are visited relatively rarely by the random walk.[3] Thus, our algorithm stops when the new node, which can be added to the APPR vector during the next iteration has a low chance to be visited by the random walk compared to the overall probability of previously added nodes.

The running time for the algorithm depends on the probability significance threshold δ. The number of updates of the APPR vector is at most $\frac{1}{\delta}$. Given that the amount of probability moved in subsequent steps is always lower, we can assume it to be the same. Therefore it holds that $sumProbUpdates = n \cdot probUpdate$, with n being the number of previous steps. Given that $lastDistrUpdate = probUpdate/sumProbUpdates$, it follows that $lastDistrUpdate \leq \frac{1}{n}$. The overall complexity of this procedure is $O(\frac{N}{\delta})$ for processing entire graphs with N denoting the number of nodes in the graph.

B. Learning of Embeddings From Approximated Personalized PageRank Vector

Each training instance used during the learning step is a pair of nodes. We call one of them *seed node* and the other one *neighbor node*. The embedding is learned for the *seed node* while the *neighbor node* is used as context. The embeddings are learned analogously to the *Skip-gram* model described in Section II-B. For each training pair the probability of the *neighbor node* is maximized given the *seed node*.

To generate the training pairs, we sample the *neighbor nodes* based on the APPR vector of the corresponding *seed node*. This means that for each *seed node*, we consider only those nodes as context which have some probability mass in the *seed node's* APPR vector, i.e., relevant nodes. Each *neighbor node* is sampled with the probability proportionally to its entry in the *seed's* APPR vector. *Neighbor nodes* are sampled with replacement and the probability to be sampled is equal to the relative ratio of probability mass each *neighbor*

[1] We emphasize that this APPR method has been remarkably successful at characterizing the local and global structural properties in large social and information networks [20], [22], [21], [15], suggesting (as we show here) that it can also be used for improved learning on these graphs.

[2] These modifications seem minor, but getting them right is extremely important for obtaining a robust and successful method.

[3] These nodes tend to be "far from" the node of interest; but, in total, they may absorb a significant large amount of the overall probability mass.

node contributes to the entire APPR vector. With this sampling strategy training data can be generated on request and the number of training examples per node can be easily controlled, and it leads to higher quality training data. Using the alias method [18], the sampling setup costs are $O(k)$, where k is the size of the APPR vector and the costs to sample a *neighbor* are $O(1)$. Our approach scales linearly with number of nodes and can easily be parallelized.

IV. Empirical results

We have evaluated the node embeddings produced by the LASAGNE algorithm by performing prediction tasks which aim at inferring node labels in multi-label classification. We have used a variety of real-world graph datasets from various domains, i.e., a biological network, social networks, and a collaboration network. Here, we compare our results against the state-of-the-art techniques *DeepWalk*, *node2vec* and *GraRep*. Note that we omit a comparison with the *LINE* since it is already shown in [12] and [8] that the results produced by *node2vec* and *GraRep* are superior to the ones produced by *LINE*. We have implemented *GraRep* using sparse matrix operations. Despite of this, we were not able to run it for larger graphs due to out of memory errors. We tested on a machine with 387GB RAM.

A. Datasets

We consider the following graph datasets from various domains with different sizes and number of classes.

Protein-Protein Interactions (PPI) [7]: This is a subgraph of the PPI network for Homo Sapiens which is also used in [12]. The nodes represent proteins, edges represent the existence of interactions between the corresponding proteins and the labels represent biological states.

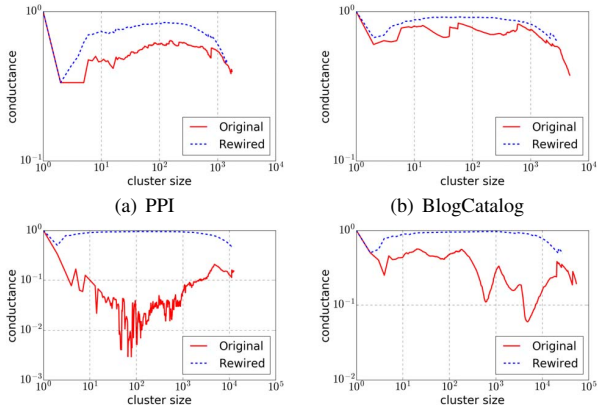
BlogCatalog [29]: This is a social network graph where each node corresponds to a user and edges represent the friendship relationships between bloggers. The interest groups provide the labels. This network is used in both [12] and [24].

IMDb Germany: This dataset is created from the IMDb movie database [13]. Each node represents an actor/actress who played in a german movie. Edges connect actors/actresses that were in a cast together and the node labels represent the genres that the corresponding actor/actress played.

Flickr [29]: The Flickr network is a social network graph with each node describing a user and the links represent friendships. The labels stem from different interest groups. This dataset is also used in [24].

Table I summarizes some statistics of these networks.

The selection of networks captures different structures, and we use Network Community Profile (NCP) plots from [20], [22], [21], [15] to analyze them. The NCP depicts the best "score" for different clusters in the graph as a function of their size. The cluster "score" is defined by *conductance*, i.e., the ratio of edges going out of a cluster to cluster internal edges. As can be seen in Figure 1, the IMDb Germany network has quite clear clusters of about 50 to 100 nodes. For each outgoing edge in the small clusters with near-minimum conductance value, there are about 800 internal edges. The three other datasets are not well separable.[4] The best cluster in the Flickr graph has a size of about 5000 nodes and only about 50 internal edges for each outgoing edge.



(a) PPI (b) BlogCatalog (c) IMDb Germany (d) Flickr

Fig. 1. NCP plots for used datasets. Red, solid lines sketch the community structure of the original graph. (Down represents better cluster quality.) Blue, dashed lines plot the structure of randomly rewired networks.

B. Experimental Setup

Like previous works, we use multi-label classification to evaluate the quality of the node embeddings. We evaluate LASAGNE exactly as in [24], [28], [12]. However, as discussed in the following we think that this evaluation method has a major drawback: it is hardly applicable in real world scenarios. Thus, we also propose a new method for evaluating node embeddings that also relies on multi-label classification but is far closer to a real-life application scenario than the method of [24], [28], [12].

We propose the following modified evaluation method MOREREALISTICMETHOD that reflects better the real world classification scenario where no a priori knowledge is given. Generally, we also train logistic classifiers to predict the labels of the test instances. In contrast to the method in [24], [28], [12], we suggest to use a 10-fold stratified cross-validation for each one-vs-rest classifier. Using such stratified sampling is a common way to split the data into training and test set by coincidently preserving the ratio of subpopulations within the data. In this way, the prediction accuracy does not suffer from classes that may not appear in either the training or the test set due to small numbers of positive examples. Note that the former evaluation method requires to know the number of labels k for each test instance at inference time. Instead of ranking the probabilities and taking the labels corresponding to the top k probabilities, we make the decision of labeling the test instance based on the label probabilities directly, i.e., if the probability of a label l is at least 50% we consider l as positive. This way, we get rid of using prior knowledge to determine the positive predicted labels.

We use micro-F_1 and macro-F_1 as evaluation metrics. Macro-F_1 scores build the unweighted average of F_1 scores for

[4] In particular, the cluster quality is only slightly better than that of a randomly-rewired graph; LASAGNE does particularly well for these graphs.

Network	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{L}\rvert$	$\overline{d}$	$\overline{C}$	D	$\overline{D}$	k_{max}	$P_{k_{max}}$	Description
PPI	3,890	38,739	50	9.959	0.146	8	3.1	30	0.028	biological network
BlogCatalog	10,312	333,983	39	32.388	0.463	5	2.4	115	0.043	social network
IMDb Germany	32,732	1,175,364	27	35.909	0.870	11	3.5	102	0.009	collaboration network
Flickr	80,513	5,899,882	195	73.279	0.165	6	2.9	551	0.018	social network

TABLE I
STATISTICS OF NETWORKS USED FOR MULTI-LABEL CLASSIFICATION: NUMBER OF NODES $|V|$, NUMBER OF EDGES $|E|$, NUMBER OF CLASSES $|\mathcal{L}|$, AVERAGE DEGREE $\overline{d}$, AVERAGE CLUSTERING COEFFICIENT $\overline{C}$, DIAMETER D AND AVERAGE SHORTEST PATH LENGTH $\overline{D}$, MAXIMUM k OF k-CORES k_{max}, FRACTION $P_{k_{max}}$ OF NODES IN k_{max} k-CORE

positive classes over all classifiers. Micro-F_1 scores build the global average based on precision and recall by treating each test example equally. We primarily focus the discussion on the macro-F_1 metric, but we also report the micro-F_1 scores.

A more precise documentation of the experiments, including experiments on link prediction where LASAGNE performs comparable to previous methods can be found in [9].

C. Results of the MoreRealisticMethod

The results reported in this section were obtained by using the parameter settings suggested in [24]. We use $\gamma = 80$ as the length for the random walks performed during the *DeepWalk* and *node2vec* procedures.[5] The number of random walks is $|V|\cdot r$, with $|V|$ being the number of vertices and $r = 10$ being the number of random walks starting from each node in the graph. The size of the window which slides over each random walk sequence extends to at most $w = 10$ in each direction of the currently regarded vertex and the dimensionality of the node embeddings is set to $d = 128$. To get a fair comparison between our method and the random walk based methods, it is crucial to use similarly sized training sets for the learning procedure since larger training sets typically tend to result in higher prediction accuracy for the test phase. Thus we sample

$$|T| = |V| \cdot \left[\gamma \cdot r \cdot 2 \cdot \mathbb{E}(\mathcal{U}(1, w)) - 2 \cdot \sum_{i=1}^{w} \mathbb{E}(\mathcal{U}(1, i))\right]$$

training examples which corresponds to the expected number of training instances generated by the random walk approaches. The notation $\mathbb{E}(\mathcal{U}(x, y))$ denotes the expected value of a uniform distribution $\mathcal{U}$ in the interval $[x, y]$.

For *node2vec* we follow the suggestions of the authors and perform full grid searches over the set $\{0.25, 0.5, 1.0, 2.0, 4.0\}$ for both hyperparameters. The *GraRep* hyperparameter k is ranged from 1 to 6. For LASAGNE we used $\sigma = 1 \times 10^{-4}$ as significance threshold for probability updates in all empirical evaluations. We show results for different values of teleportation parameter α. The learning costs for our and random walk methods are equal. The same applies for sampling or random walks simulations. The only difference is the preprocessing. The LASAGNE preprocessing for the largest dataset we used in our evaluation took few hours on one core without parallelization. In contrast, the node2vec preprocessing took several days. The *DeepWalk* algorithm does not have a preprocessing step. For all datasets and all approaches we demonstrate the results when we used 90% of the data for training and the remaining data as test set for the classification tasks. We adapted the computation of the *Approximated Personalized PageRank* implemented in the *Ligra* framework [26] for our implementation. The learning procedure for the embeddings is implemented in *TensorFlow*.

[5]If diameter $D = 5, 6, 8, 11$, then walk length $\gamma = 80$ is quite long.

For all networks and methods the macro-F_1 scores are reported in Figure 2. For the PPI network, the scores are consistently over 8% for all α values, while the random walk approaches reach scores between 7% and 7.5%. The best *node2vec* setting is $p = 4$ and $q = 1$, which corresponds to a rather low willingness to allow the random walks to return to already visited nodes. This meets the outcomes of LASAGNE, which are best for small α values; the method performed best for $\alpha = 0.2$. The generally low prediction quality for all approaches, and especially the bad score for *GraRep* (best score for $k = 4$), may indicate that the distribution of class labels do not follow any representative, local patterns and hence are hardly graspable within local structures.

The results for BlogCatalog are even more clear. LASAGNE improves the best competitor by approximately 23%. As can be seen in Figure 2(b) the performance of LASAGNE decreases almost monotonically with increasing values for α. The best score is achieved for $\alpha = 0.001$. This means that neighbors which describe a node best are not extremely local. The best *node2vec* setting, i.e., $p = 0.25$ and $q = 0.25$, confirms this result. Recalling Figure 2 from [12], the 2nd order random walks are biased towards leaving the neighborhoods.

For IMDb Germany, the best result of LASAGNE, that is for $\alpha = 0.99$, is only slightly better than the best results achieved with *node2vec*. Since LASAGNE is, as well as *node2vec* with parameter setting $p = 0.25, q = 4$, able to stay extremely local, both approaches reach high prediction scores on this dataset where labels are concentrated in low conductance clusters.

Using the Flickr network, LASAGNE reaches the highest improvement over the other random walk based methods, i.e., more than 33% with $\alpha = 0.001$. The results behave similar to the ones for the BlogCatalog data, but in contrast the scores remain more stable. Indeed, the gap between the smallest and largest selected α values is only 1%. As mentioned previously, we could not run *GraRep* on Flickr, because of its size.

Figure 3 shows the micro-F_1 scores achieved with the same settings as used for the macro-F_1 score evaluation. The results show that the micro scores are higher than the macro scores for all datasets except for IMDb Germany. Also the relative differences between the results for LASAGNE and the best competitor are higher for the macro-F_1 scores than for the micro-F_1 scores. This is due to the micro score metric effectively gives higher weight to larger classes. Since LASAGNE performs better for smaller classes which, except for IMDb Germany, are the vast majority of classes, the macro-F_1 scores

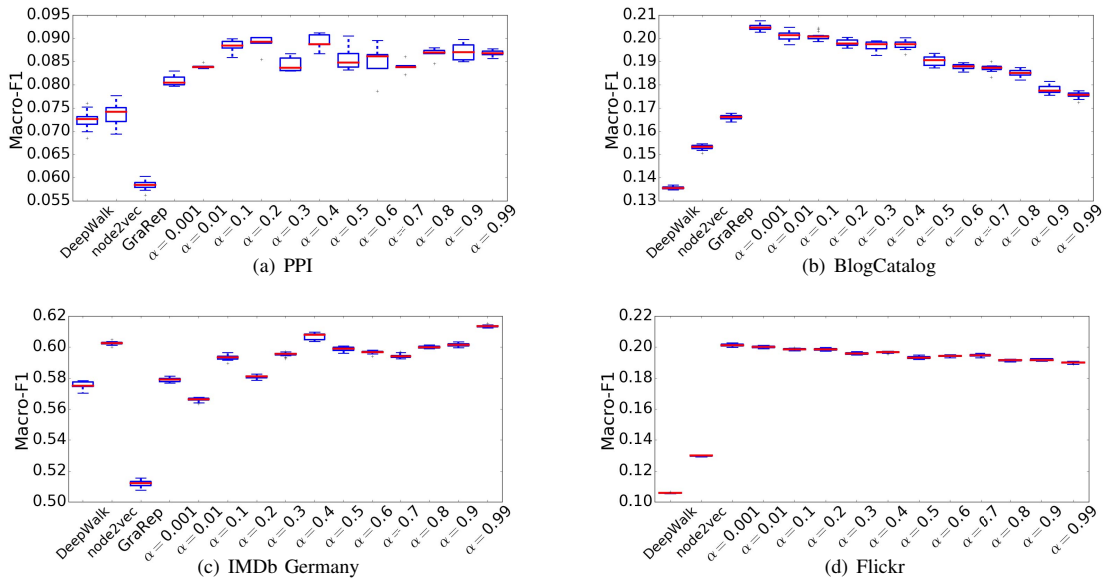


Fig. 2. Macro-F_1 scores achieved by doing multi-label classification as downstream task for the considered representation learning techniques. LASAGNE scores are presented for different values of parameter α.

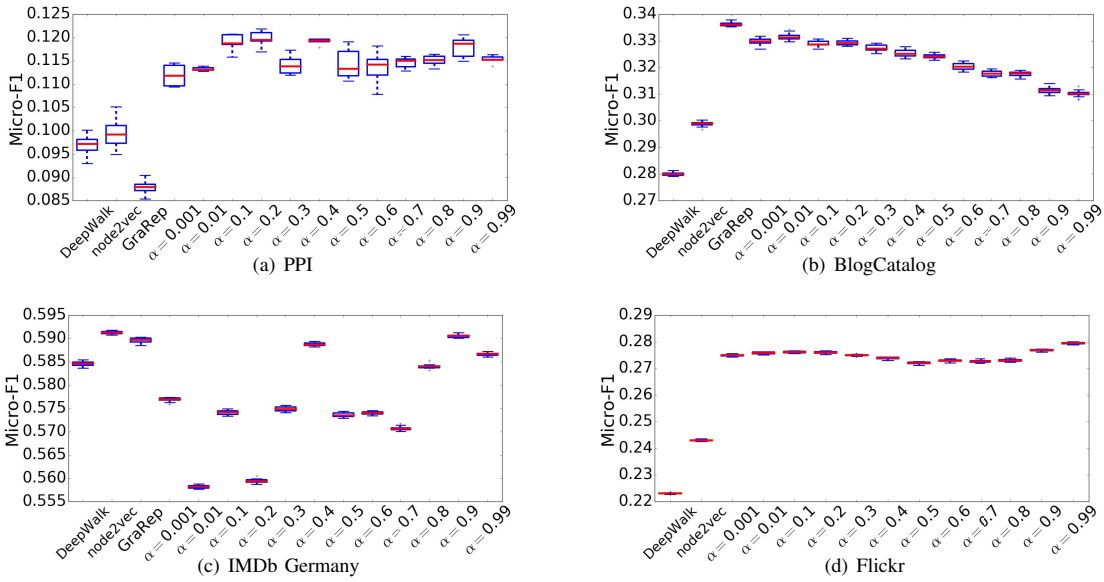


Fig. 3. Micro-F_1 scores achieved by doing multi-label classification as downstream task for the considered representation learning techniques. LASAGNE scores are presented for different values of parameter α.

take benefit due to weighting each class equally independent from the class sizes. Recalling that the micro-F_1 considers the sizes of the classes, the performance improvements for this score are reasoned by the fact that LASAGNE performs better on smaller classes and similarly good to random walk based methodologies on larger classes.

An important summary point from the Figures 2 and 3 is that, in the case of graphs without even small-sized good conductance clusters, the performance of LASAGNE clearly overcomes the performance from random walk based methods. On the other hand, for graphs that have an upward-sloping NCP and thus small-sized good conductance clusters, LASAGNE shows similar prediction quality to random walk based methods. In particular, while we are never worse than previous methods, we observe the weakest improvement for IMDb Germany, which is consistent with Figure 1(c), where the upward-sloping NCP suggests relatively good local structure, and we observe the strongest improvement for the Flickr network, which is consistent with Figures 1(d), which indicate a relatively flat NCP, and many deep k-core nodes (about one tenth of nodes are in k-cores with $k > 150$).

D. Results of the former evaluation method

Tables II and III show the macro-F_1 scores, resp. the micro-F_1 scores when applying the evaluation proposed by [24] and using 90% of the node representations for training. While *GraRep* shows the best results on PPI, the performance of

Algorithm	Dataset			
	PPI	BlogCatalog	IMDb Ger	Flickr
DeepWalk	0.1747±.0133	0.2221±.0119	0.6868±.0159	0.2104±.0049
node2vec	0.1930±.0192	0.2418±.0123	0.6996±.0070	0.2349±.0039
GraRep	**0.1991±.0148**	0.2231±.0127	0.5770±.0177	-
LASAGNE	0.1835±.0116	**0.2843±.0116**	**0.7042±.0252**	**0.2930±.0061**

TABLE II
MACRO-F_1 SCORES FOR MULTI-LABEL CLASSIFICATION WHEN USING FORMER EVALUATION METHOD AND 90% OF INSTANCES FOR TRAINING.

Algorithm	Dataset			
	PPI	BlogCatalog	IMDb Ger	Flickr
DeepWalk	0.2206±.0142	0.3889±.0093	0.7043±.0055	0.3762±.0038
node2vec	0.2293±.0185	0.3963±.0093	**0.7060±.0047**	0.3841±.0037
GraRep	**0.2487±.0152**	0.3913±.0149	0.6648±.0076	-
LASAGNE	0.2216±.0091	**0.4116±.0129**	0.6967±.0045	**0.4078±.0048**

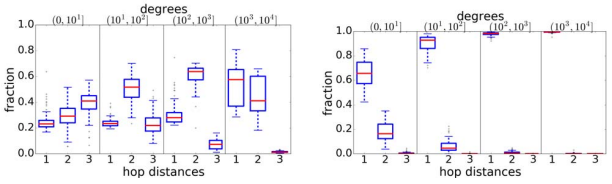
TABLE III
MICRO-F_1 SCORES FOR MULTI-LABEL CLASSIFICATION WHEN USING FORMER EVALUATION METHOD AND 90% OF INSTANCES FOR TRAINING.

the LASAGNE embeddings clearly overcomes the competitors when testing on the considered social networks, similar to the results in our more realistic (and more refined) evaluation.

E. Explaining our improved empirical results

LASAGNE improves previous methods by considering more finely the structure of the graph around each node. In particular, we compute local node neighborhoods by touching only the *relevant* neighbors of each node, which leaves the major part of the graph unconsidered. For the node a we call b its *relevant* neighbor if b has high probability to be visited by a random walk with restart starting from a.

1) Locality for nodes with different degrees: Due to using random walks, existing methods fail to adapt to the local graph structure, even when using biased random walks as in [12]. When random walks are used to obtain neighbors, nodes having very low probability to be visited also appear among the considered neighbors. Nodes having high probabilities to be visited appear more frequently. However, the cumulative probability of low probability nodes may still be significant. The wider the window is, the more far away neighbors end up in it. However, smaller window sizes will not help to tackle the problem with low probability neighbors, since the nodes in sparse graph areas may have distant neighbors with high probability to be visited by random walk. Grover et al. [12] even show, that they achieve better results with larger window sizes. However, since the same window size is used for all nodes in the same graph, the distributions of hop distances of nodes to their neighbors are similar and barely adapt to local node neighborhoods. To confirm this intuition, we computed the hop distances to the nodes considered as context by *node2vec* and *DeepWalk* algorithms for different datasets. For all of them, we observed similar behavior, i.e., the level of locality was barely adapted with increasing node degree, c.f., Figure 4(a). Note that the *node2vec* parameters were set to $p = 0.25$ and $q = 4.0$, which constrains the random walks to capture very local neighborhoods (but in a non-adaptive manner). The distributions of hop distances to the neighbors found by the LASAGNE algorithm are very similar per dataset; an example is depicted in Figure 4(b). In contrast



(a) *node2vec* (b) LASAGNE

Fig. 4. Distributions of hop distances to neighbors from nodes with different degrees. These plots visualize the ability to adjust to differently dense areas in the graph for *node2vec* (left, not well) and LASAGNE (right, very well).

to the previous methods, LASAGNE adapts to the local node environment, i.e., for the high degree nodes only the neighbors with the highest probability to be visited by the random walk are considered as context. Consequently, we observe a clear tendency that the preference to local neighborhoods increases with increasing node degree (which is known to correlate with poor NCP clusters and deep k cores [22], [15], [2]). The *LINE* algorithm considers only one-hop neighbors, and the assumption that only direct neighbors are relevant is very strong, especially for low degree nodes.

2) Locality for more versus less peripheral classes: Large graphs with flat NCP, especially with large and highly connected regions (with large deep *k-cores*) are notably affected by random walk problems. For graphs with flat NCPs, the connectivity among nodes' *relevant* neighbors is not much stronger than to the rest of the graph. Furthermore, the larger and deeper are graphs *k-cores*, the more time random walks will spend in them. This affects the neighborhoods obtained by random walks for most nodes, since most parts of even large graphs can be reached within few steps. Therefore, even if dense parts of the graph have high probabilities to be visited by global random walks, if the probabilities of single nodes in these components are low, then nodes from these components are not considered by LASAGNE as neighbors. Consequently, for the nodes from large deep *k-cores*, the neighborhood will be restricted to the most *relevant* core neighbors. Therefore, our method adapts to the structure of local neighborhood.

To confirm this intuition, we used the Flickr network, a graph with flat NCP. As can be seen in Table I, the graph has large deep *k-cores*, e.g., the largest *k-core* has degree over 550. We expect random walk based methods to perform poorly on such a graph, especially if the similarity to neighbors outside of large deep *k-cores* is important for the downstream task. As multi-label classification is a common downstream task, Figure 5 provides empirical evidence that LASAGNE's embeddings overcome performance issues of previous embeddings. In Figure 5, each line stands for a class, and the color depicts the classification improvement of LASAGNE over the best previous method. Additionally, each plotted line shows the fraction of nodes with the corresponding class label in each *k-core*, relative to the fraction of nodes with that label in the entire graph. When the fraction of class labels is zero, the line breaks. It can be clearly seen from the plot that LASAGNE achieves the best improvement for classes with members outside of large *k-cores* with high *k*, i.e., for classes that are more peripheral.

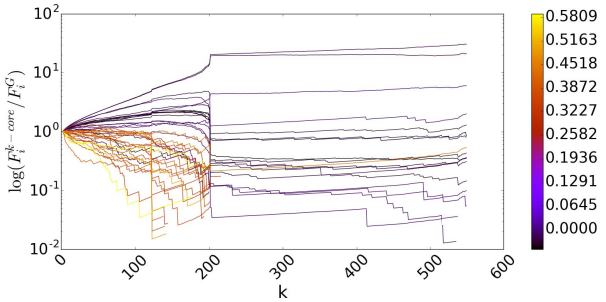


Fig. 5. Each line depicts the class label distribution in k-cores with performance information for one class in the Flickr data. X-axis: k-core; Y-axis: log scaled proportion between fraction of class label i within the k-core, i.e., F_i^{k-core}, and the fraction of this class label within the entire graph G, i.e., F_i^G; color code: absolute difference in F_1 score between LASAGNE and the best random walk based method. For ease of presentation, the plot shows only the 20 classes where LASAGNE reached the highest improvement as well as the 20 classes where the improvement was smallest.

3) Distribution of training examples per node: Another shortcoming of existing random walk based methods is the distribution of training examples per node that they generate. Since high-degree nodes are visited more often by random walks, there are more training examples for them. Since small-degree nodes are visited much less often, they are underrepresented during training. Due to the way in which locality is engineered into LASAGNE, it solves this problem.

V. CONCLUSION

We have proposed LASAGNE, an unsupervised learning algorithm to compute embeddings for the nodes of a graph. The basic idea of LASAGNE is to use an *Approximate Personalized PageRank* algorithm to bias random walks more strongly to the local neighborhood of each node; and, thus, the embedding for a given node is more finely tuned to the local graph structure around that node than the embeddings from previous similar methods. Our method performs particularly well for larger graphs that are not well-structured, e.g., that have flat NCPs and/or have many nodes in deep k-cores. Our empirical evaluation has shown that our embeddings achieve superior prediction accuracy over competitors when used for multi-label classification in several different real-world networks. Our empirical results also provide evidence justifying the reason for this improvement. While LASAGNE is primarily an exploratory tool, if one wants to use it in a more automated manner, then an important question will be how to automate the averaging of the APPR vectors over different values of the locality parameter.

ACKNOWLEDGEMENTS

This work was partially supported by Siemens, the Army Research Office, the Defense Advanced Research Projects Agency, and it was developed in cooperation with the Berkeley Institute of Data Science.

REFERENCES

[1] Abu-El-Haija, S., Perozzi, B., Al-Rfou, R., Alemi, A.: Watch your step: Learning graph embeddings through attention. arXiv preprint arXiv:1710.09599 (2017)
[2] Adcock, A.B., Sullivan, B.D., Mahoney, M.W.: Tree-like structure in large social and information networks. In: Proc. of IEEE ICDM. pp. 1–10 (2013)
[3] Adcock, A.B., Sullivan, B.D., Mahoney, M.W.: Tree decompositions and social graphs. Internet Mathematics **12**(5), 315–361 (2016)
[4] Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: Proc. of IEEE FOCS. pp. 475–486. IEEE (2006)
[5] Berkhin, P.: Bookmark-coloring algorithm for personalized pagerank computing. Internet Mathematics **3**(1), 41–62 (2006)
[6] Bojchevski, A., Günnemann, S.: Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017)
[7] Breitkreutz, B.J., Stark, C., Reguly, T., Boucher, L., Breitkreutz, A., Livstone, M., Oughtred, R., Lackner, D.H., Bähler, J., Wood, V., et al.: The biogrid interaction database: 2008 update. Nucleic acids research **36**(suppl 1), D637–D640 (2008)
[8] Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: Proce. of CIKM. pp. 891–900. ACM (2015)
[9] Faerman, E., Borutta, F., Fountoulakis, K., Mahoney, M.W.: Lasagne: Locality and structure aware graph node embedding. arXiv preprint arXiv:1710.06520 (2017)
[10] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212 (2017)
[11] Gittens, A., Achlioptas, D., Mahoney, M.W.: Skip-gram-zipf+ uniform= vector additivity. In: Proc. of ACL. vol. 1, pp. 69–76 (2017)
[12] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proc. of ACM SIGKDD. pp. 855–864 (2016)
[13] IMDb: The internet movie database. In: http://www.imdb.com/ (2016, last accessed: 2016-11-22)
[14] Jeh, G., Widom, J.: Scaling personalized web search. In: Proc. of the 12th WWW. pp. 271–279. ACM (2003)
[15] Jeub, L.G., Balachandran, P., Porter, M.A., Mucha, P.J., Mahoney, M.W.: Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. Physical Review E **91**(1), 012821 (2015)
[16] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
[17] Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
[18] Knuth, D.: The art of computer programming: Vol 2/seminumerical algorithms, chapter 3: Random numbers (1969)
[19] Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. vol. 14, pp. 1188–1196 (2014)
[20] Leskovec, J., Lang, K., Dasgupta, A., Mahoney, M.W.: Statistical properties of community structure in large social and information networks. In: Proc. of WWW'08. pp. 695–704 (2008)
[21] Leskovec, J., Lang, K., Mahoney, M.W.: Empirical comparison of algorithms for network community detection. In: Proc. of WWW'10. pp. 631–640 (2010)
[22] Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics **6**(1), 29–123 (2009)
[23] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
[24] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proc. of ACM SIGKDD. pp. 701–710 (2014)
[25] Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: Learning node representations from structural identity. In: Proc. of KDD. pp. 385–394. ACM (2017)
[26] Shun, J., Blelloch, G.E.: Ligra: a lightweight graph processing framework for shared memory. In: ACM SIGPLAN Notices. vol. 48, pp. 135–146. ACM (2013)
[27] Shun, J., Roosta-Khorasani, F., Fountoulakis, K., Mahoney, M.W.: Parallel local graph clustering. Proc. VLDB Endow. **9**(12), 1041–1052 (Aug 2016)
[28] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proc. of WWW. pp. 1067–1077. ACM (2015)
[29] Tang, L., Liu, H.: Relational learning via latent social dimensions. In: Proc. of ACM SIGKDD. pp. 817–826. ACM (2009)
[30] Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proc. of ACM SIGKDD. pp. 1225–1234. ACM (2016)
[31] Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proc. of ICDM. pp. 40–48 (2016)